IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 April 2023, accepted 16 May 2023, date of publication 29 May 2023, date of current version 7 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3281304

RESEARCH ARTICLE

Imperative vs. Declarative Modeling of Industrial Process. The Case Study of the Longwall Shearer Operation

EDYTA BRZYCHCZY[1], MARCIN SZPYRKA[2], (Senior Member, IEEE), JACEK KORSKI[3], AND GRZEGORZ J. NALEPA[4,5,6], (Member, IEEE)

[1]Faculty of Mechanical Engineering and Robotics, AGH University of Science and Technology, 30-059 Kraków, Poland
[2]Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, AGH University of Science and Technology, 30-059 Kraków, Poland
[3]ITG KOMAG Institute, 44-100 Gliwice, Poland
[4]Faculty of Physics, Astronomy and Applied Computer Science, Institute of Applied Computer Science, Jagiellonian University, 31-007 Kraków, Poland
[5]Jagiellonian Human-Centered AI Laboratory (JAHCAI), Jagiellonian University, 31-007 Kraków, Poland
[6]Mark Kac Center for Complex Systems Research, Jagiellonian University, 31-007 Kraków, Poland

Corresponding author: Edyta Brzychczy (brzych3@agh.edu.pl)

This work was supported in part by the AGH University of Science and Technology (AGH UST) Grant; and in part by the PACMEL Project funded by the National Science Centre, Poland, through the CHIST-ERA Program under Grant NCN 2018/27/Z/ST6/03392.

ABSTRACT Process modeling is an important and necessary step for further analysis and monitoring of industrial processes. In the process modeling two main paradigms exist, namely imperative and declarative ones. In our work, we analyzed information potential of these model paradigms regarding to conformance checking task of real life industrial process – longwall shearer operation carried out in an underground coal mine. The objective of our work was an analysis of selected imperative and declarative models to discover which approach is more appropriate from a practical point of view, taking into consideration criteria formulated by the domain expert. The first novelty of our work rely on real life industrial sensor data analysis and creation of event log with heuristic approach for case ID identification and labeling with expert rules. In parallel, we created prescribed process models. As representatives of imperative and declarative languages, we have selected the Petri nets and Declare models, respectively. We created two Petri nets (with Inductive and Heuristic Miner) and seven declarative models differ in restriction power. Due to the better description of the ideal cycle, to the further analysis and conformance checking task, we selected the Petri net created by Heuristic Miner. After the process model creation, we compared selected Petri net with Declare models using the natural language approach and constraints hierarchy. Based on created similarity measures, we choose one declarative model to conformance checking task and comparison with Petri net due to formulated quantitative and qualitative criteria. As main artifact in the conformance checking task, we used obtained real event log. Evaluation of the created models indicates that in the case of the longwall shearer operation monitoring, the declarative model better captures the necessary information to decision-makers than the Petri net, thus being more appropriate for practical use.

INDEX TERMS Process modeling, petri net, WF-nets, declare model, sensor data, shearer, underground mining.

I. INTRODUCTION

In the digitized organization, there are new opportunities to support analysis and improvement of the executed processes. One of them is process mining, which enables discovery, monitoring, and improvement of real processes by extracting knowledge from data available in IT systems [1]. Nowadays, process mining is a quite well-established standard in automated process model discovery and analysis, e.g., see [2].[1]

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

[1]This paper was funded by the National Science Centre, Poland under CHIST-ERA programme, the CHIST-ERA 2017 BDSI PACMEL Project, NCN 2018/27/Z/ST6/03392.

In industrial practice, an essential issue is process monitoring concerning the real performance of a process, as well as the identification of its deviations. To do so, one should know the prescribed model of the process and validate if its actual execution conforms to the process model. In process mining, this task is known as conformance checking [3].

A process model is always the result of a compromise between full imitation of reality and simplification. It is only an approximation of the real execution of an industrial process, satisfactory from the practical point of view. Process models can be hand-made [4] or can be discovered automatically with the use of process mining based on event logs. In process modeling, the following decisions should be made:

- on the type of model (imperative, declarative, or hybrid),
- the process modeling language used, and
- the level of abstraction, which provide the most valuable information from the point of view of process analysis and improvement for domain users.

Imperative models are more popular in business process modeling (e.g., Business Process Model and Notation models – BPMN) due to clear notation and understandability to the users. Imperative modeling techniques have been implemented in almost every modeling tool [5]. However, declarative models have attracted much attention over the last years and new tools enabling declarative modeling have been developed, e.g., Rule Mining application called RuM [6].

The abstraction level of the model should be related to the needs of the domain user. It is strictly related to the available data on the execution of the process. In many real-world scenarios, events and traces are not readily available, and before process modeling, event logs have to be developed from existing data sources [7]. This is a widespread phenomenon in industrial process modeling using sensor data. In the aspect of process modeling for industrial needs, the following issue also has to be raised: *"domain experts are typically not familiar with business process modeling languages"* [8], therefore, during process modeling collaboration between the process analyst and domain expert is indispensable.

The paper presents the results of research work conducted in the CHIST-ERA Pacmel project. One of the main objectives of the project is to provide new insights into the relationship of low-level sensor data recorded in the monitoring of industrial processes and their high-level models provided to decision makers. Pacmel is a fundamental research project, implemented with close cooperation with several companies. They provide real-life data, models, and expert knowledge to the project. Moreover, they are very interested in the possible exploitation of the project results.

In this setting, our paper provides an application-driven analysis of modeling approaches. We introduce a use case of modeling and analysis of longwall shearer operation industrial process. This process was selected by us as it is quite special. At first sight, it can be described as a very simple sequence of operations. However, what makes it interesting is that its real execution shows that it is highly unstructured. Cycle executions can vary; nevertheless, not all deviations are errors from the mining reality point of view. Thus, a model which is too restrictive can return a lot of unnecessary information to decision makers during the conformance checking task, which would jeopardize its practical usefulness.

Our motivation for the analysis was related to the theoretical process model which existed in the domain literature for the analyzed process. Apparently, it is a relatively simple one and does not represent a variety of behavior and unstructured specificity. We translated this model into a Petri net as an imperative model. We wanted to discover how this process model can support the conformance checking to obtain valuable information about process execution and deviations. Since we observed a high variability in the event log, we believed that in the case of unstructured behavior, a better approach with a better quality of information would be the declarative one (as suggested in [9]).

The specific research problem we are considering is a comparison of imperative and declarative models of real life industrial process to select a better approach for practical application.

Our hypothesis in this paper is as follows: *in the case of analysis of an industrial process whose execution is unstructured, the use of declarative models instead of imperative ones is more useful from a practical point of view*. To perform a practical evaluation, we formulate metrics for the quantitative evaluation and criteria for the qualitative one. We examine a number of different process models (imperative and declarative) aiming at their comparison and selection of the best one for practical application in the company responsible for process monitoring. In our experiments, we decided to use a subclass of Petri nets called workflow nets (WF-nets) and Declare models as the main representatives of the mentioned modeling paradigms.

The remainder of this paper is composed as follows. Section II presents related works on imperative and declarative approaches in process modeling, selected modeling languages, and the process models comparison. Section III describes our case study, i.e., the longwall shearer operation process. Section IV deals with the introduction to WF-nets and the Declare language. Section V gives an overview of our approach. In Section VI, we present the results of evaluation and experiments showing the advantages and disadvantages of the evaluated process models due to the formulated criteria. Finally, Section VII concludes the paper and presents directions for future work.

II. RELATED WORKS

In practice, we have to deal with various types of processes (Fig. 1). Some of them are well structured, repeatable, thus easier to manage and predict. On the other hand, we have unstructured and non-repeatable processes with high flexibility, which can be challenging for their modeling and analysis.

The nature of the process influences its modeling process and the selection of model type. There are two main types of modeling paradigms – imperative and declarative (Fig. 2). The differentiation between the imperative and declarative

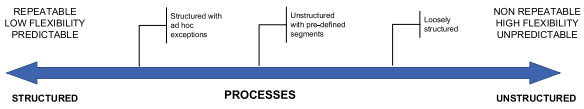

FIGURE 1. Spectrum of processes (based on [10]).

approach has its roots in computer programming [11]: *Imperative programming implies to "say how to do something", whereas declarative programming implies to "say what is required and let the system determine how to achieve it".*

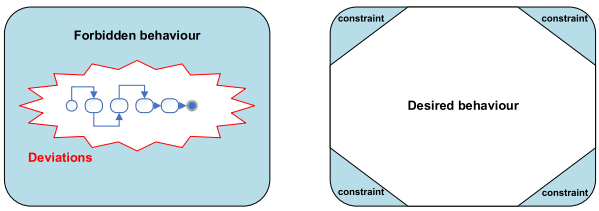

FIGURE 2. Visualization of imperative and declarative approaches (after [12]).

Imperative models represent an "inside-out" approach; therefore, every possible execution sequence must be explicitly modeled. Thus, imperative modeling may lead to over-specification and lack of flexibility in real process modeling. In contrast to imperative approaches, declarative models take an "outside-in" approach. Instead of exactly specifying how the process should be executed, only the essential characteristics are described in the form of constraints. Constraints are represented by rules that restrict the possible execution of activities [5]. Probably the most notable difference between imperative and declarative modeling is how a given behavior can be classified as satisfying or not satisfying the model [9].

While imperative approaches are a strong concept regarding well-defined processes, they lack clarity once an observed behavior allows for flexible execution. In this case, models following a declarative approach can describe the behavior in a more compact way [13]. The description of circumstantial dependencies may be easier on the basis of a declarative process model than an imperative process model [9].

The most known and useful (from the process mining point of view) imperative process modeling languages are Petri nets. Since their beginning in the 1960s, many classes and extensions were proposed, ranging from low-level classes [14], [15], to high-level ones [16], [17], [18]. The former are used to build models at a high level of abstraction. The latter combine the capabilities of Petri nets with the capabilities of a high-level programming language that provides primitives for the definition of data types, variables, expressions for describing data manipulation, etc., and thus are used to build models that are closer to real systems [19]. From the conformance checking point of view, WF-nets are very popular [1]. This is because they are suitable for automatic model generation from event logs. WF-nets are used for building imperative models in this paper. Imperative models are more popular in business process modeling (especially in the form of Event-Driven Proces Chain (EPC) or BPMN models) and have been implemented in many tools (e.g., ARIS, Bizagi, Signavio).

In recent years, there has been a growing interest in declarative modeling, both from the scientific and practical point of view [5]. In declarative modeling, we build a model as a set of constraints that are true for the model. A well-known example of declarative languages is Declare [20]. The language consists of activities and constraints [21] and is equipped with a graphical representation of these elements. An activity is marked as a rectangle and constraints are expressed as arcs. An example of a model built using Declare is shown in Fig. 6. The flexibility of declarative models can be burdensome in modeling processes of a rather procedural nature. If the desired execution of the process is well defined, it is easier to express in the form of an imperative model how the process is to run than to introduce constraints that are to be met. Furthermore, when we introduce too many (also interrelated) constraints, the declarative model may become hardly understandable to the user [20]. There are other declarative modeling languages, e.g. PENELOPE [22], Business Process Constraint Network [23] or DCR Graphs [24]. A review of selected declarative languages is presented in [25].

Currently, one of the development directions for process modeling languages is the so-called hybrid approach that enables the construction of process models, using both elements of imperative and declarative languages [26], [27]. It is a response to the actual internal heterogeneity of processes, which consist of certain structured and unstructured fragments. However, there is a challenge in choosing which parts would be shown better in either way [13]. It is also worth mentioning that an extension of the BPMN language with declarative elements (BPMN-D) exists. This proposal is described in detail in [28].

In modeling reality, it is important to answer the question if there is a possibility to represent the same behavior regardless of the notation. As an initial response, the authors in the paper [13] presented an approach for the transformation of a declarative model (Declare) into a behaviorally equivalent imperative model in the form of a Petri net. The proposed transformation consists of three main steps: (i) translation of declarative constraints to regular expressions, (ii) translation of regular expressions to Finite State Automaton (FSA), and (iii) transformation of FSA to PN, based on the regions theory. Applying the mentioned steps enables building an imperative model from declarative constraints. When a transformation of declarative models into imperative models is possible, the comparison of their ability to model the reality similarly is an important question, especially for process model users.

The similarity of business process models (mainly in the form of imperative models) was investigated in many papers (e.g., [29], [30], [31], [32]), often in the context of a model repository. Analysis of model similarity can be useful before adding a new process model to the model repository to avoid process model duplication. Various similarity metrics have been proposed, e.g. [29]: for node matching similarity,

structural similarity, and behavior similarity. At the same time, research on similarity matching for declarative models has not been highly developed [33].

Comparison of declarative and imperative modeling approaches was discussed considering a number of important directions, such as: understability [9], [11], maintainability [34], flow dependencies [33] or, more recently, the quality of models [35].

The first interesting approach, related to our work [33] comprises a proposal of extracting the flow dependencies from both imperative and declarative process models. After that, based on the hierarchy of dependencies, process models can be compared considering a subsumption relation, contradictory, or their non-comparability. The cases inconsistencies can be located. The flow dependencies are investigated only locally (avoiding a potentially irresolvable problem with all possible executions in the model). The approach is based on dependency matrices that consider the dependencies between activities and the dependency hierarchy. However, none of the specific similarity measures is proposed.

The second paper [36] worth mentioning in the context of our work empirically studies the consequences of the selection of the modeling approach (imperative / declarative) on the quality of the output model. For model quality evaluation, the authors used notation-agnostic metrics for precision and generalization. The developed framework (''qmpm'') enables building Petri nets (with Inductive Miner) and a declarative model (with MINERful) with the calculation of the mentioned metrics. However, recently, the framework requires that all traces are replayable on the model (perfect fitness), and thus a comparison of other input models than default (built with Inductive Miner and MINERful) is currently not available.

For the sake of clarity, it is worth mentioning that conformance checking can detect two types of deviations between an imperative process model and a log [37]: 1) behavior observed in the log but disallowed by the model (unfitting behavior, known also as *move in the log only*), and 2) behavior allowed by the model but not observed in the log (additional behavior, known also as *move in the model only*). In the case of declarative models, conformance checking shows the activation of the constraints, their fulfillment, or violation. In practice, the trace may not activate a constraint at all, than vacuously satisfies the constraint [38]. A comprehensive review of the conformance checking techniques is presented in [3]. The most recognized include token-based replay [39], casual footprints [1], behavior profiles [40], and the most popular alignment techniques [41]. Our work is also related to the well-established trend in the literature regarding the declarative models in the form of Petri nets to the analysis of industrial systems, e.g. see [17].

In our approach, as a reference, we assume similar representatives of declarative and imperative models as presented in [36]. However, due to the requirement of perfect fitness of input models in the *qmpm* framework, we defined our own metrics to compare the similarity of the developed models.

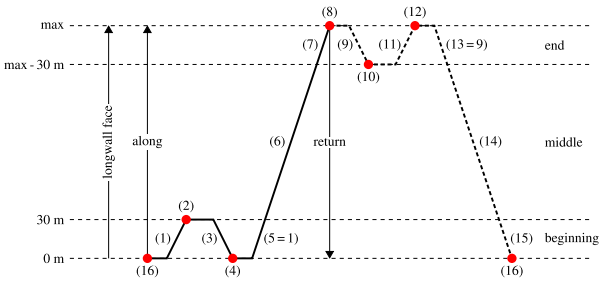

FIGURE 3. Model of the shearer cycle [4].

In the work at hand, we present our approach for the comparison of imperative and declarative models using a translation of a WF-net model to the regular expressions and calculation of similarity between the created models considering the constraints hierarchy, presented in [42]. After comparison, the selected imperative and declarative models are conformed to real process traces to evaluate their usability with respect to the formulated criteria (see Section VI). In the following section, we introduce the specific industrial case study, which is later used for the evaluation.

III. CASE STUDY: THE CHARACTERISTICS OF THE LONGWALL SHEARER INDUSTRIAL PROCESS

The longwall mining process can be defined as a cyclical implementation of a set of operations (activities) repeated in a specific order and time in a longwall face. The set of these activities depends on technology, equipment, and work organization. In our work, we focus on the process modeling of longwall shearer as the main machinery in the longwall face in the underground mine. Most often (i.e., in the two-way mining technology) the shearer operation process consists of the following activities (Fig. 3): 1) Cutting beginning (along), 2) Stoppage beginning (along), 3) Return to drive (along), 4) Stoppage beginning (along), 5) Cutting beginning (along), 6) Cutting middle (along), 7) Cutting end (along), 8) Stoppage end (along), 9) Cutting end (return), 10) Stoppage end (return), 11) Return to drive (return), 12) Stoppage end (return), 13) Cutting end (return), 14) Cutting middle (return), 15) Cutting beginning (return), and 16) Stoppage beginning (return).

The main operations of the shearer are related to the cutting of coal. The *Cutting* can occur in any location in the longwall face. The *Return to the drive* stage can occur only at the beginning or at the end of the longwall face. There are also other characteristic stages at the beginning and end of the longwall face, named the *Stoppage in ON mode* (marked in Fig. 3 with red bullets). They are strictly related to the planned changes in the position of the organs. In the shearer process, some additional operations can occur in relation to unexpected changes of shearer moving direction or organ usage. The first operation is called *Reversion* and can be defined as cutting in the opposite direction than expected in the along and return direction. The prescribed model of the shearer process also does not include the *Moving* stage, because it is undesirable

in a two-way cutting system. However, it can occur when the cleaning of the conveyor path is needed.

In the mine, the operations performed in the longwall face are monitored by various IT systems. One of them is the monitoring system of machinery, usually directly used as a data supplier for the dispatcher system. In the case of a longwall face, the monitoring system most often provides the following inputs: a) the main information from compact stations and converters (e.g., connectors statuses, protection activation, currents, voltages, resistances), b) data from a longwall machinery (e.g. oil temperature, element's pressures, moving of elements, location), c) input and output of industrial PLC drivers. Based on the domain knowledge, the following variables are useful for shearer operations description: 1) shearer main status, 2) currents on shearer drums, 3) currents on shearer haulages, 4) shearer arms move event, 5) shearer move direction, 6) shearer speed, 7) shearer location, and 8) general direction of shearer operation.

Based on the domain knowledge from the company in the form of expert rules, we are able to label the sensor data as shearer operations and model them with process modeling languages in the form of a higher-level process model. This model is built using the methods introduced in the next section.

IV. LANGUAGES AND TOOLS

Workflow nets (WF-nets) are a subclass of low-level Petri nets with a well-defined net structure [1]. A WF-net is equipped with a dedicated start place and a dedicated end place. Moreover, all nodes are on a path from start to end. The formal definitions are as follows:

Definition 1: A Petri net *is a tuple* $\mathcal{N} = (P, T, F, M_0)$, *where*

- P *and* T *are finite sets of* places *and* transitions *such that* $P \cap T = \emptyset$,
- $F \subseteq P \times T \cup T \times P$ *is the* flow relation *(a set of arcs),*
- $M_0 \colon P \to \mathbb{N}$ *is the initial marking, where* $\mathbb{N} = \{0, 1, 2, \dots\}$.

The *preset* of a transition t is the set ${}^\bullet t = \{p \in P \colon (p, t) \in F\}$; the *postset* of t is the set $t^\bullet = \{p \in P \colon (t, p) \in F\}$. Presets and postsets for places are defined in the same way.

Definition 2: Let $\mathcal{N} = (P, T, F, M_0)$ *be a Petri net.* $\mathcal{N}$ *is a* workflow net *(WF-net) if and only if*

- P *contains an input place* i *(start place) such that* ${}^\bullet i = \emptyset$;
- P *contains an output place* o *(end place) such that* $o^\bullet = \emptyset$;
- $\mathcal{N}' = (P, T \cup \{\bar{\tau}\}, F \cup \{(o, \bar{\tau}), (\bar{\tau}, i)\}, M_0)$ *is strongly connected, i.e., there is a directed path between any pair of nodes in* $\mathcal{N}'$ *– referred to as the short-circuited net.*

Usually, labeled WF-nets are used. Let A be a set of activity labels. We use a labeling function $l \colon T \to A$. A special τ label is used to indicate *unobservable* (silent) transitions. A WF-net model for the longwall mining process is presented in Figs. 4,5: $i = P_0$, $o = P_{20}$, transitions without labels are silent transitions. We use the standard definition of transitions activity.

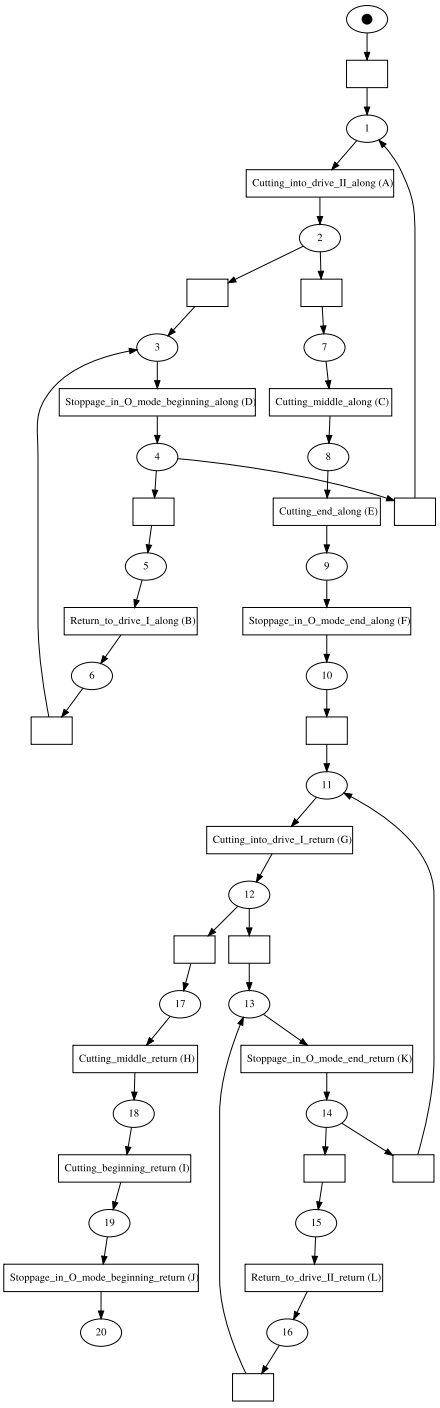

FIGURE 4. Resulting WF-net.

Definition 3: A marking M *is a mapping* $M \colon P \to \mathbb{N}$, *i.e. it is a distribution of tokens in places of the net.*

A transition t *is enabled in a marking* M *iff* $\forall p \in {}^\bullet t \colon M(p) \geq 1$.

If a transition t *is enabled in a marking* M *it may fire, changing the marking* M *to a marking* M' ($M \xrightarrow{t} M'$), *such*

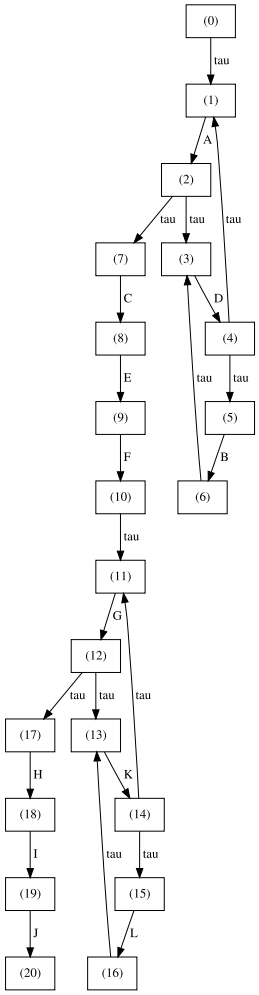

FIGURE 5. Resulting reachability graph.

that for any $p \in P$

$$M'(p) = \begin{cases} M(p) + 1 & \text{for } p \in t^\bullet \setminus {}^\bullet t \\ M(p) - 1 & \text{for } p \in {}^\bullet t \setminus t^\bullet \\ M(p) & \text{otherwise} \end{cases} \quad (1)$$

A *firing sequence starting at marking* M is a sequence transitions $t_1, \ldots, t_n$ such that there exists a sequence of markings $M_1, \ldots, M_n$ such that $M \xrightarrow{t_1} M_1 \ldots \xrightarrow{t_n} M_n$. In such a case we say that $t_1, \ldots, t_n$ *leads* from M to M_n and that M_n is *reachable* from M. Am *empty* firing sequence leads from M to M. The set of all markings that are reachable from M is denoted by $Reach(M)$.

Definition 4: A Labeled Transition System *(LTS) is a tuple* $LTS = (S, E, L, S_0)$*, where* S *is non-empty, countable set of* states, L *is countable set of labels,* $E \subseteq S \times L \times S$ *is a set of arcs, and* S_0 *is the* initial state.

A reachability graph *of a Petri net* $\mathcal{N} = (P, T, F, M_0)$ *is the transition system* $LTS = (S, E, L, S_0)$*, such that*

- $S = Reach(M_0)$,
- $L = T$,
- $(M, t, M') \in E$ *iff* $M \xrightarrow{t} M'$,
- $S_0 = M_0$.

For labeled Petri nets, transition labels are usually used instead of transition names. Figure 4 presents a WF-net and Fig. 5 its reachability graph. The number inside a node indicates the only place that contains a token for the given state (marking).

Imperative process modeling approaches understand a process as an LTS as shown in Fig. 5, where control has a clearly defined begin, end, and flow between begin and end. In the case of a declarative approach, models focus on specifying constraints as rules that have to be followed during process execution. In this context, the definition of declarative process models can be formulated as follows [43].

Definition 5: Let Σ *be an alphabet, and* $\tau \in \Sigma$ *is a special symbol denoting a silent action.* Declarative Process Model *(DPM) is a tuple* $\mathcal{DP} = (R, T, A, l, K)$*, where*:

- R *is repertoire of templates, i.e. predicates* $R(x_1, \ldots, x_n) \in R$ *on variables* $x_1, \ldots, x_n$ *(we say* $n \in \mathbb{N}$ *is the* arity *of* R*),*
- T *is a finite non-empty set of transitions,*
- $A = \Sigma \cup \{\tau\}$,
- l *is labeling function* $l : T \to A$*, and*
- $K \ni \kappa$ *is a set of constraints, namely templates of arity* n *whose variables are assigned by a mapping with labeled transitions* $x_i \overset{\kappa}{\leftarrow} t_i$ *with* $t_i \in T$, $1 \leq i \leq n$.
- *A constraint* $\kappa \in K$ *can be denoted as* $R(t_1, \ldots, t_n)$.

The constraints are formulated with the use of specific templates. The following basic templates can be used [44]:

- existence - a unary cardinality constraint describing the number of possible activity executions;
- relation - a binary constraint that forces the presence of an action in conjunction with another activity (e.g., response, precedence or succession);
- choice - n-ary constraints expressing a choice between activities;
- negation - negative version of the relation constraints.

Selected templates are presented in Table 1. An example of a declarative model is shown in Fig. 6.

There is a plethora of algorithms enabling automated process models discovery in the form of declarative or imperative models. An exhaustive review of existing methods is presented in [45]. In our work, we have implemented the Inductive Miner [46] and Heuristic Miner [47] algorithms for Petri net creation. All experiments on Petri nets were performed using the *ProM* [48] tool. In declarative modeling, we used Declare Miner [49]. All experiments on Declare models we have performed using the *RuM* [50] software.

V. MODELS COMPARISON METHOD

The overview of our approach to the comparison of imperative and declarative models presented in this paper is given in Fig. 7. The approach consists of three main stages: 1) event log creation, 2) process modeling, and 3) process mining.

Firstly, we transform sensor data into a raw event log enabling process mining. Sensor data are raw data without case ID and event labels. Then, after cleaning the sensor

TABLE 1. Selected templates of Declare language – regular Expression semantics.

No.	Constraint	Semantics
1	$Existence(n, a)$	Activity a must be executed at least n times
2	$Absence(n+1, a)$	Activity a can be executed at most n times
3	$Exactly(n, a)$	Activity a must be executed exactly n times
4	$Init(a)$	Activity a must be the first executed activity
5	$End(a)$	Activity a must be the last executed activity
6	$Response(a, b)$	If a is executed, then b must be executed at any time after a
7	$ChainResponse(a, b)$	If a is executed, then b has to be executed immediately after a
8	$AlternateResponse(a, b)$	If a is executed, then b has to be executed at any time after a and before a is again executed
9	$RespondedExistence(a, b)$	If a is executed, then b also has to be executed (any order)
10	$CoExistence(a, b)$	If a is executed, then b also has to be executed and vice versa
11	$Precedence(a, b)$	If b is executed, then a has to be executed at any time before b
12	$ChainPrecedence(a, b)$	If b is executed, then a has to be executed immediately before b
13	$AlternatePrecedence(a, b)$	If b is executed, then a must be executed at any time before b and no other B can be executed in between
14	$Succession(a, b)$	$Response(a, b)$ and $Precedence(a, b)$
15	$ChainSuccession(a, b)$	$ChainResponse(a, b)$ and $ChainPrecedence(a, b)$
16	$AlternateSuccession(a, b)$	$AlternateResponse(a, b)$ and $AlternatePrecedence(a, b)$
17	$NotCoExistance(a, b)$	If a is executed, then b cannot be executed and vice versa
18	$NotSuccession(a, b)$	a is never executed before b
19	$NotChainSuccession(a, b)$	a i b are both executed if and only if the latter does not immediately follows the former

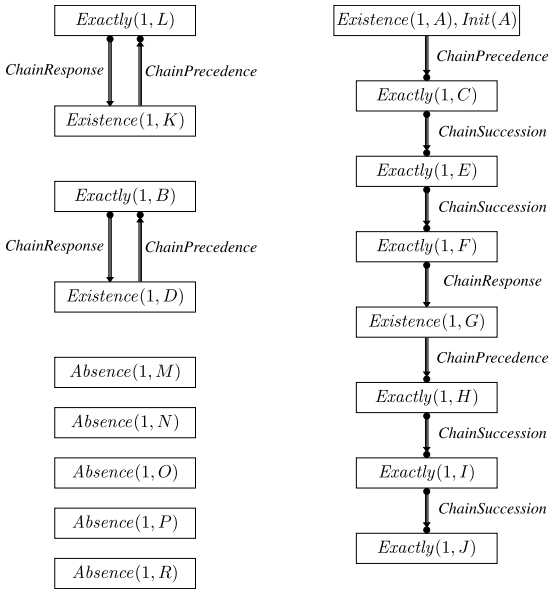

FIGURE 6. Example of declarative model.

data (due to outliers, missing data), we perform case labeling using the detection technique presented in [51]. After case identification, we label the records with rules provided by domain experts (described in [4]). Since case ID and event labeling are very sensitive to data quality, the raw event log is additionally inspected to detect unreliable traces in the log. The cleaned event log is our first input into the conformance checking task (CC).

On the other hand, we prepare an ideal event log that includes perfect cases. It enables us to create process models in an automatic manner. In our work, we use a Petri net model (as a representative of the imperative approach) and Declare (as a declarative representation). In the case of declarative models, we create them (i) automatically, (ii) as hand-made models, and (iii) as enhanced models discovered automatically. After model creation, we compare Petri nets and Declare models using natural language approach and the constraints hierarchy. The next step is selection of the most similar process models (imperative-declarative) and introduction to the CC task.

In the CC task, we use real event logs and replay their content on the created models. We compare the quantity and quality of information provided by each model to the user. The quantity of information we defined as the sum of deviations revealed by each model during the CC task. In the quality assessment, we adopt the following criteria formulated by the domain expert: do the CC results show us directly (with available software): a) the *Reversion* or *Moving* events, b) the *Stoppages* event in the middle of the longwall and their number, or c) how many *stoppages* events are at the beginning and at the end of the longwall. In our approach, we want to evaluate whether declarative models are appropriate from a practical point of view to model and monitor the longwall shearer operation process. The results of our experiments and evaluation are presented in the next section.

VI. EVALUATION RESULTS FOR THE CASE STUDY

To provide a practical evaluation on real data, for our case study, we used an event log file collected from a coal mine in Poland. The event log obtained during the event log creation process (described in the previous section) contains 222 cycles (Fig. 8). The event log comprises 222 variants of trace execution; thus variability of the process is unquestionable.

We also prepared an ideal event log containing 10 perfect cases to generate ideal process models, imitating the ideal cycle as shown in Fig. 3. With the implementation of

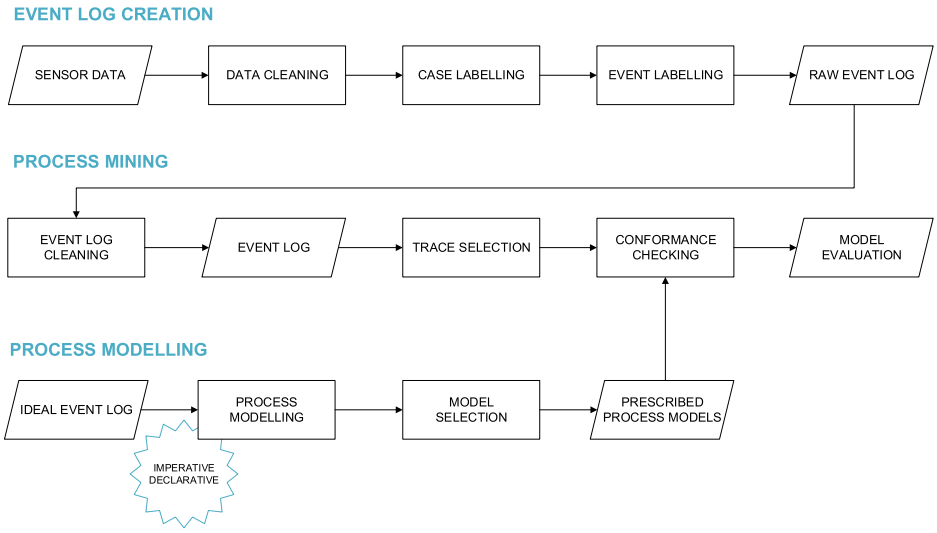

FIGURE 7. Stages of our experiments.

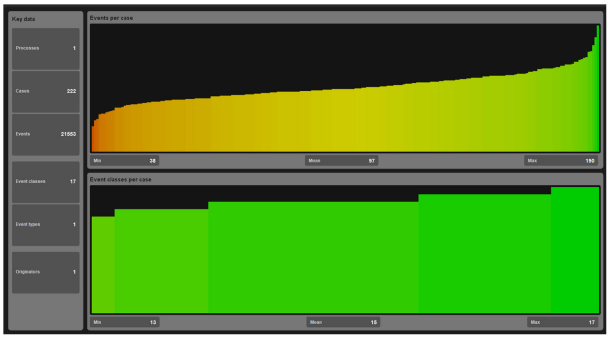

FIGURE 8. Characteristic of the event log we used.

two process model discovery algorithms (with perfect fitness settings), we have obtained two Petri net models (slightly different, especially in the modeling of the beginning part of the cycle) – with Heuristic Miner shown in Fig. 4 and with Inductive Miner shown in Fig. 9. Due to the better description of the ideal cycle (clear indication of an initial activity), as the representative of the imperative process model for the further stages, we selected the Petri net created by Heuristic Miner.

As a comparative base, we created seven declarative process models, described in the form of used templates, shown in Table 2. In our attempts, we tried to create various variants of the process, aiming at finding the declarative model most similar to the selected Petri net.

The first two declarative models were prepared manually using the *RuM* software and validated by the domain expert, checking whether the templates used properly describe behavior in each part of the model. Model 1 is more restrictive than Model 2 due to the implementation of *ChainPrecedence* and *ChainResponse* constraints. Moreover, the hand-made models include the *Absence* constraint, which can not be discovered from the ideal event log automatically. Model 3 was discovered automatically with Declare Miner (we used the following settings: pruning type – all reductions, min. constraint support = 100%) concerning the following templates: $Init(a)$, $Existence(a)$, $ChainPrecedence(a, b)$, $ChainResponse(a, b)$ and $ChainSuccession(a, b)$. Model 4 includes Model 3 templates enhanced with *Absence* constraints. Model 5 is a transformation of Model 4 – *Existence* templates were replaced by more restrictive ones: $Exactly(1, a)$ and $Exactly(2, a)$. Model 6 is a relaxed version of Model 5 without $Exactly(2, a)$ and limitation of the *Exactly* templates. Finally, Model 7 is an enhanced version of Model 1 with the *Exactly* templates.

To address the question whether the declarative approach is better than the imperative one, for the longwall mining case study, we had to choose the declarative model that is the most similar to the WF-net. Therefore, to facilitate further considerations, we mapped the original labels to single letters as presented in Table 3.

Thus, using the alphabet $\Sigma = \{A, B, \ldots, R\}$ any legal trace can be treated as word over Σ. To find the legal traces, from the WF-net model point of view, we must analyze the reachability graph presented in Fig. 4 (the right graph). For example, *ACEFGHIJ* is such a trace (τ transitions are omitted). The set of legal traces is infinite, but can be represented as a single regular expression:

$$L \equiv (AD(BD)^*)^*ACEF(GK(LK)^*)^*GHIJ \qquad (2)$$

where * denotes zero or more copies of the subexpression inside the braces.

The main idea here is to count the words in the language (2) that are accepted by the considered declarative model. Unfortunately, the language contains infinitely many words because some subexpressions can be repeated infinitely many times. However, since we apply a theoretical

TABLE 2. Declarative models.

Constraints	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model7
$ChainPrecedence(E, F)$	•						•
$Existence(1, K)$	•	•	•	•		•	•
$Existence(1, I)$	•	•	•	•			
$ChainResponse(F, G)$	•		•	•	•	•	•
$Existence(1, E)$	•	•	•	•			
$Existence(1, B)$	•	•	•	•		•	•
$Existence(1, F)$	•	•	•	•			
$Existence(1, D)$	•	•	•	•		•	•
$Existence(1, C)$	•	•	•	•			
$Existence(1, L)$	•	•	•	•		•	•
$Existence(1, G)$	•	•	•	•		•	•
$Existence(1, H)$	•	•	•	•			
$ChainPrecedence(C, E)$	•						•
$Existence(1, J)$	•	•	•	•			
$Init(A)$	•	•	•	•	•	•	•
$Response(A, C)$	•	•					•
$Response(G, H)$	•	•					•
$Response(D, A)$	•	•					•
$Response(K, G)$	•	•					•
$ChainPrecedence(H, I)$	•						•
$ChainPrecedence(I, J)$	•						•
$Absence(1, M)$	•	•		•	•	•	•
$Absence(1, N)$	•	•		•	•	•	•
$Absence(1, O)$	•	•		•	•	•	•
$Absence(1, P)$	•	•		•	•	•	•
$Absence(1, R)$	•	•		•	•	•	•
$CoExistence(A, D)$	•	•					•
$CoExistence(D, B)$	•	•					•
$CoExistence(G, K)$	•	•					•
$CoExistence(K, L)$	•	•					•
$Precedence(E, F)$		•					
$Response(F, G)$		•					
$Precedence(C, E)$		•					
$Precedence(H, I)$		•					
$Precedence(I, J)$		•					
$Existence(1, A)$	•	•	•	•		•	•
$ChainSuccession(E, F)$			•	•	•	•	
$ChainSuccession(I, J)$			•	•	•	•	
$ChainSuccession(C, E)$			•	•	•	•	
$ChainSuccession(H, I)$			•	•	•	•	
$ChainPrecedence(G, H)$			•	•	•	•	
$ChainPrecedence(K, L)$			•	•	•	•	
$ChainPrecedence(D, B)$			•	•	•	•	
$ChainPrecedence(A, C)$			•	•	•	•	
$ChainResponse(L, K)$			•	•	•	•	
$ChainResponse(B, D)$			•	•	•	•	
$Exactly(1, H)$					•	•	•
$Exactly(2, A)$					•		
$Exactly(2, K)$					•		
$Exactly(2, G)$					•		
$Exactly(2, D)$					•		
$Exactly(1, I)$					•	•	•
$Exactly(1, E)$					•	•	•
$Exactly(1, B)$					•		
$Exactly(1, C)$					•	•	•
$Exactly(1, F)$					•	•	•
$Exactly(1, L)$					•		
$Exactly(1, J)$					•	•	•

model to a real system where the number of repetitions must be finite, we can use an upper bound k for *, i.e. we assume that the given subexpression can be repeated at most k times.

Let $k = 1$ for the considered language L. Then, L reduces to the language L' that contains the words presented in Table 4.

Let c denote the number of constraints, and n denote the number of words. For any given constraint c_i, the *support* of c_i is defined as follows:

$$sup(c_i) = \frac{a_i}{n} \tag{3}$$

where a_i is the number of words accepted by c_i.

We formulate two similarity measures to compare a declarative model M_i with the basic Petri net model:

$$sim_1(M_i) = \frac{\sum_{i=1}^{c} sup(c_i)}{c} \tag{4}$$

$$sim_2(M_i) = \frac{\sum_{i=1}^{c} w_i sup(c_i)}{c} \tag{5}$$

where $w_i \in (0, 1]$ is the weight of c_i.

According to the templates hierarchy, we assumed the following weights (w_i):

- 1 - for $Init(a)$, $Absence$, $Exactly(1, a)$, $Exactly(2, a)$ and $ChainSuccession(a, b)$ constraints,

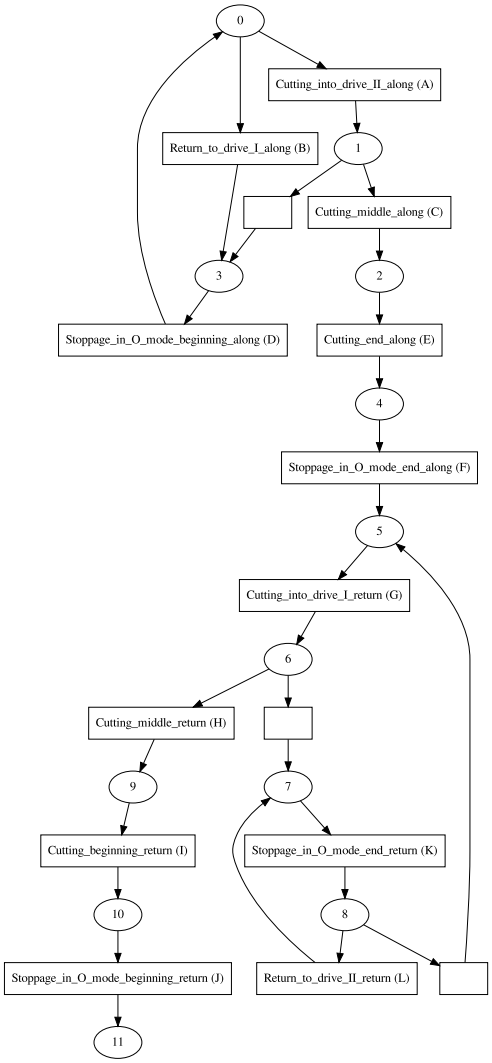

FIGURE 9. Process model in form of Petri net from Inductive Miner.

TABLE 3. Mapping labels into letters.

Label	Letter
Cutting_into_drive_II_along	A
Return_to_drive_I_along	B
Cutting_middle_along	C
Stoppage_in_O_mode_beginning_along	D
Cutting_end_along	E
Stoppage_in_O_mode_end_along	F
Cutting_into_drive_I_return	G
Cutting_middle_return	H
Cutting_beginning_return	I
Stoppage_in_O_mode_beginning_return	J
Stoppage_in_O_mode_end_return	K
Return_to_drive_II_return	L
Moving	M
Reversion_along	N
Reversion_return	O
Stoppage_in_O_mode_middle_along	P
Stoppage_in_O_mode_middle_return	R

- 0.9 - for *Existence*, *ChainPrecedence* and *ChainResponse* constraints,
- 0.8 - for *Response* and *Precedence* constraints,
- 0.7 - for *CoExistence* constraint.

TABLE 4. Language L'.

No	Word
1	ACEFGHIJ
2	ACEFGKGHIJ
3	ACEFGKLKGHIJ
4	ADACEFGHIJ
5	ADACEFGKGHIJ
6	ADACEFGKLKGHIJ
7	ADBDACEFGHIJ
8	ADBDACEFGKGHIJ
9	ADBDACEFGKLKGHIJ

The similarity measures between the created declarative models and the Petri net are presented in Tab. 5. The highest value of the similarity measure was obtained by Model 6. In our experiments, we conformed event logs to the selected models (Petri net model – PN and Model 6 – DM) due to the criteria presented in Tab. 6.

TABLE 5. Similarity measures between declarative models and Petri net.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
sim_1	0.828	0.828	0.917	0.931	0.885	0.931	0.828
sim_2	0.741	0.725	0.846	0.872	0.861	0.893	0.760

TABLE 6. Comparison of selected models.

Criterion	PN	DM
1) Quantity of information – summary	20979	17361
2) Direct presentation of Reversion or Moving events	-	+
3) Direct presentation of stoppages event in the middle of the longwall	-	+
4) Number of stoppages events at the beginning or at the end of the longwall	-/+	+

Firstly, we counted the number of deviations between the model and the event log. During the replay of a real event log on PN, we took into consideration the difference between the model and the log expressed by the numbers of moves only in the log and moves only in the model. In the case of DM, we summarized a number of constraints' violations.

The initial results show a general advantage of DM over PN. However, we performed a detailed analysis for each trace. On the basis of deviation distributions for each model, we tested the following hypotheses:

- H0: *PN and DM models return the comparable number of deviations for analyzed event log*
- H1: *PN model returns more deviations than DM model for analyzed event log*

Due to the fact that the distribution of the analyzed variable is not the normal one, we have used the two-samples Wilcoxon test. Obtained results ($W = 35899$, $p - value \sim 0.0000$) confirmed that at significance level $p = 0.05$ we can reject H0 and adopt the alternative hypothesis that PN returns more deviations than DM for the analyzed event log. We calculated difference (*Diff*) between the number of deviations revealed in CC with PN and the number of deviations in CC

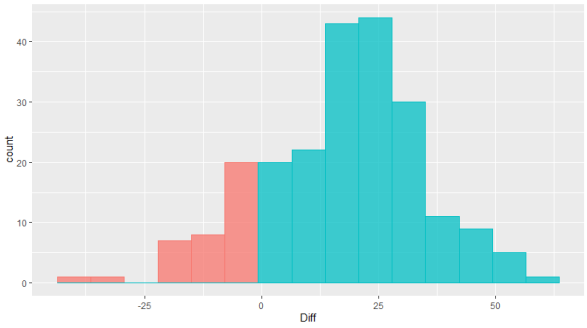

FIGURE 10. Histogram of diff variable.

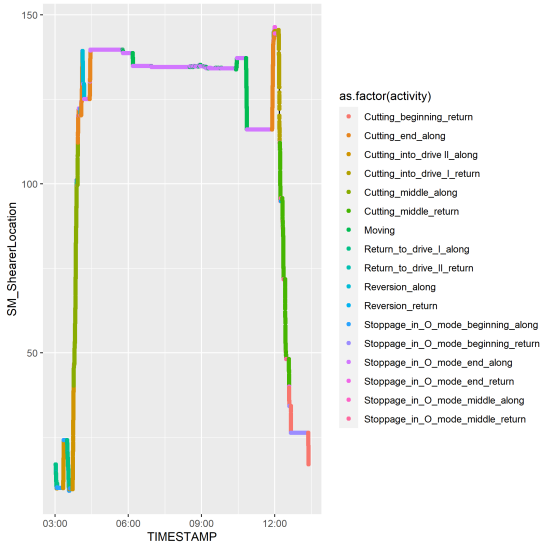

FIGURE 11. Trace (no. 216) with the lowest Diff value (PN over DM).

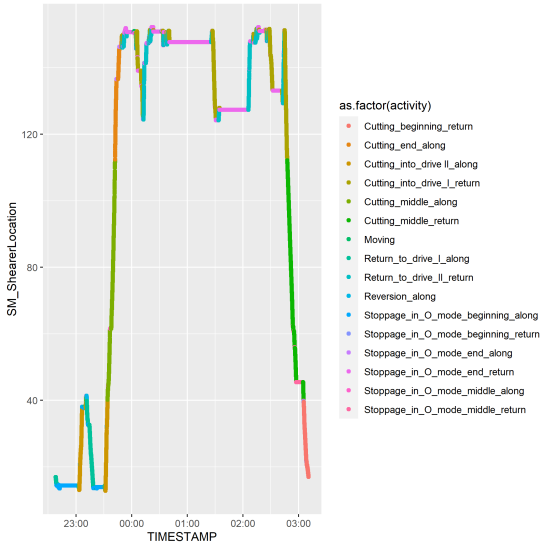

FIGURE 12. Trace (no. 120) with the highest Diff (DM over PN).

with DM. The histogram of the obtained values is presented in Fig. 10.

There are 37 traces out of 222, for which fewer deviations indicated CC with PN. To analyze how the number of

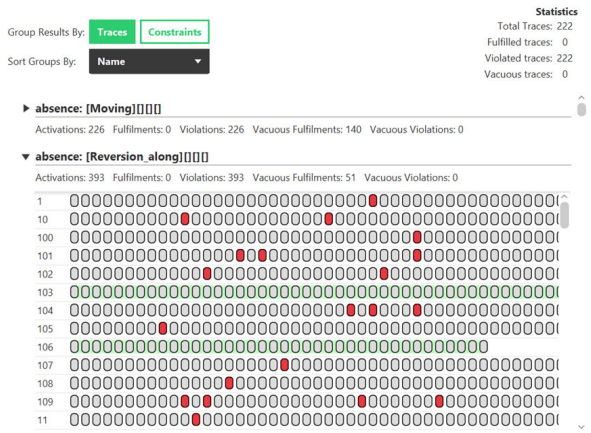

FIGURE 13. Visualisation of information about deviations in event log in RuM.

deviations depends on the trace length, we calculated the Kendall rank correlation between these variables. The obtained results (0.94 for PN and 0.54 for DM – with statistical significance $p = 0.05$) showed a clear dependence between the length of the trace and the number of revealed deviations, clearly, higher for PN models. However, the traces with the highest number of deviations for each model give an interesting view. The visualization of the trace (no. 216), in which PN outperforms DM the most, is presented in Fig. 11. The opposite example (DM outperforms PN) is presented in Fig. 12.

For the trace (no. 216) with the lowest Diff value (−41), CC with PN model returned 96 deviations (with DM – 137 deviations). It is a trace with 97 events (equals to the mean trace length in the event log). The number of deviations from the PN (according to the existing correlation between the trace length and the number of deviations) is not a surprise. Interestingly, for this trace, CC with DM returned much more deviations. The justification of such a situation comes from the specific process execution in this trace. There were multiple executions of activities at the beginning and at the end of the longwall which are enabled by the Petri net, however, highly violate the chain precedence constraints (*Cutting beginning along*, *Cutting middle along*) and (*Stoppage end return*, *Return to drive return*). For the trace (no 120) with the highest Diff value (59), CC with DM models returned 65 deviations in comparison to 124 with PN. The trace is longer (125 events) and more complex; thus CC with PN showed more deviations.

The comparison of the deviations revealed by CC with each model leads to the conclusion that the length of the trace (reflecting in some way the complexity of process execution) is not a predictor of the predominance of any of the models. Differences in model performance are derived from the execution of the process.

Before considering qualitative criteria for evaluation, it can be observed that in the case of declarative models, it is easier to formulate clear rules that should be met during process

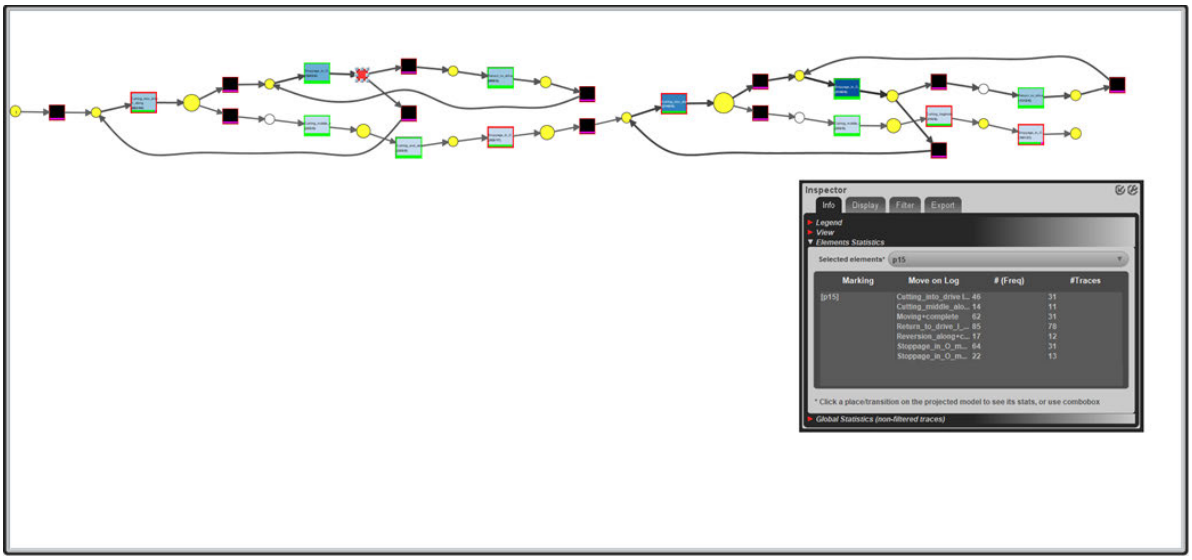

FIGURE 14. Visualization of information about deviations in event log in ProM – selected place statistics.

execution and are of most importance to users who monitor the execution of the process. It is a matter of the general idea of the declarative approach – we define the desired behavior that does not have to be executed in full range, as is in the case of the imperative approach.

We consider the *qualitative criteria* based on the information available for users provided by the used software. Results of CC in *RuM* are presented in the form of a clear visualization of the violated constraints, thus interesting information for the user is presented in a direct manner in Fig. 13. In *ProM* detailed information has to be additionally filtered out from the conformance analysis report. In the case of additional events occurring in the event log, information about their existence is hidden in the statistics of various places of PN, see the screenshot in Fig. 14. In the context of the last criterion, both used software tools present a number of errors. However, in *ProM* the detailed information is hidden in the activities and places statistics. In *RuM* the number of event occurrences is provided directly in the constraint's result box as existence constraint fulfillment.

In Tab. 6 we summarize the criteria and the summary of the evaluation of the models we selected. The results show that the declarative model is superior to the imperative model, based on the criteria we used. This supports the hypothesis formulated in the introduction of the paper. In our case of a complex, unstructured industrial process, the use of the declarative model for process modeling and further monitoring is more appropriate from the practical point of view.

VII. SUMMARY

In the paper, we focus on the comparison of modeling approaches for industrial process. Our analysis is application-driven, based on a specific use case of the longwall shearer operation process in the domain of underground mining. Although the model of this process seems conceptually simple, its practical execution in the industrial environment shows that it is highly unstructured. Using this case, we demonstrate how declarative models instead of imperative ones can be more useful from the practical point of view. In the paper, we examined several different process models (imperative and declarative) and formulated criteria for their comparison.

We conducted practical experiments using Petri WF-nets and the Declare models as the main representatives of the two modeling paradigms. We analyzed real-life event logs from a coal mine in Poland. The paper was prepared in cooperation with Famur S.A. which is the producer of the longwall shearers we considered in the paper. The company also provided us with data for the experiments. In addition, domain experts from the company participated in the analysis of the results.

Our finding consists in demonstrating the advantages of the declarative approach in the modeling of a complex industrial process of longwall shearer operation. In fact, our approach extends the idea of comparison of different paradigms [33] with proposal of a similarity measure based on natural language approach and a constraints hierarchy. It also enables the comparison of models without perfect fitness to an event log assumed in [36], thus comparison of a hand-made model can be carried out. Thus, our approach can be a valuable aid for industrial experts that analyze industrial processes.

Our work has certain limitations. One of them is the actual presence of the theoretical process model, mostly in the literature, design, and manuals, which is not always the case. Another one is the transformation process, which is manually conducted by an expert at this stage. Furthermore, the challenge of the uncertainty in the definition of similarity of the selected models (i.e., finite number of compared traces, assumed weights for constraints) is important. It is not a trivial task due to the fundamental difference in the presented

approaches and the general issue in the art of modeling itself, thus building models that equivalently describe reality.

Our future work will be conducted in several directions. The first is devoted to the identification, based on the revealed deviations between process instances and compared models, of potential places of hybrid approach application, since there exist traces when Petri Net returns fewer deviations than a Declare model. Another one consists in developing a semisupervised method for model transformation.

ACKNOWLEDGMENT

The authors would like to thank F. M. Maggi and A. Alman for support with RuM Software, and also would like to thank C. Olling Back and T. Slaats for support with qmpm framework.

REFERENCES

[1] W. van der Aalst, *Process Mining: Data Science in Action*. Heidelberg, Germany: Springer, 2016.

[2] H. Ariouat, A. H. Cairns, K. Barkaoui, J. Akoka, and N. Khelifa, "A two-step clustering approach for improving educational process model discovery," in *Proc. IEEE 25th Int. Conf. Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2016, pp. 38–43.

[3] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking: Relating Processes and Models*. Cham, Switzerland: Springer, 2018.

[4] M. Szpyrka, E. Brzychczy, A. Napieraj, J. Korski, and G. Nalepa, "Conformance checking of a longwall shearer operation based on low-level events," *Energies*, vol. 13, no. 24, pp. 1–10, 2020.

[5] H. Reijers, T. Slaats, and C. Stahl, "Declarative modeling—An academic dream or the future for BPM?" in *Proc. 11th Int. Conf. Bus. Process Manage.*, in Lecture Notes in Computer Science, vol. 8094, F. Daniel, J. Wang, and B. Weber, Eds. Cham, Switzerland: Springer, 2013, pp. 307–322.

[6] A. Alman, C. D. Ciccio, F. M. Maggi, M. Montali, and H. van der Aa, "RuM: Declarative process mining, distilled," in *Proc. 19th Int. Conf. Bus. Process Manage.*, in Lecture Notes in Computer Science, vol. 12875, A. Polyvyanyy, M. T. Wynn, A. V. Looy, and M. Reichert, Eds. Cham, Switzerland: Springer, 2021, pp. 23–29, doi: 10.1007/978-3-030-85469-0_3.

[7] K. Diba, K. Batoulis, M. Weidlich, and M. Weske, "Extraction, correlation, and abstraction of event data for process mining," *WIREs Data Mining Knowl. Discovery*, vol. 10, no. 3, pp. 1–17, May 2020.

[8] M. Dumas, M. La Rosa, J. Mendling, and H. Reijers, *Fundamentals of Business Process Management*, 2nd ed. Cham, Switzerland: Springer, 2018.

[9] D. Fahland, D. Lübke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, and S. Zugal, "Declarative versus imperative process modeling languages: The issue of understandability," in *Proc. 10th Int. Workshop Enterprise, Bus.-Process Inf. Syst. Modeling*, vol. 29, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, and R. Ukor, Eds. Cham, Switzerland: Springer, 2009, pp. 353–366.

[10] C. D. Ciccio and M. Mecella, "On the discovery of declarative control flows for artful processes," *ACM Trans. Manag. Inf. Syst.*, vol. 5, no. 4, pp. 24:1–24:37, 2015.

[11] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, and H. Reijers, "Imperative versus declarative process modeling languages: An empirical investigation," in *Proc. Int. Workshops Bus. Process Manage. Workshops*, vol. 99, F. Daniel, K. Barkaoui, and S. Dustdar, Eds. Cham, Switzerland: Springer, 2011, pp. 383–394.

[12] B. Weber, J. Pinggera, S. Zugal, and W. Wild, "Handling events during business process execution: An empirical test," in *Proc. 1st Int. Workshop Empirical Res. Process-Oriented Inf. Syst.*, in CEUR Workshop Proceedings, vol. 603, B. Mutschler, J. Recker, R. Wieringa, J. Ralyté, and P. Plebani, Eds., 2010, pp. 19–30.

[13] J. Prescher, C. Di Ciccio, and J. Mendling, "From declarative processes to imperative models," in *Proc. 4th Int. Symp. Data-driven Process Discovery Anal.*, in CEUR Workshop Proceedings, vol. 1293, R. Accorsi, P. Ceravolo, and B. Russo, Eds., 2014, pp. 162–173.

[14] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[15] W. Reisig, *Understanding Petri Nets. Modeling Techniques, Analysis Method, Case Studies*. Heidelberg, Germany: Springer, 2013.

[16] K. Jensen and L. Kristensen, *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Heidelberg, Germany: Springer, 2009.

[17] Y. Dong, Z. Li, and N. Wu, "Symbolic verification of current-state opacity of discrete event systems using Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 12, pp. 7628–7641, Dec. 2022.

[18] M. Szpyrka, J. Biernacki, and A. Biernacka, "Tools and methods for RTCP-nets modeling and verification," *Arch. Control Sci.*, vol. 26, no. 3, pp. 339–365, Sep. 2016.

[19] B. Jasiul, M. Szpyrka, and J. Sliwa, "Malware behavior modelling with colored Petri nets," in *Proc. 13th Comput. Inf. Syst. Ind. Manag.*, vol. 8838. Cham, Switzerland: Springer, 2014, pp. 667–679.

[20] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," *Comput. Sci.-Res. Develop.*, vol. 23, no. 2, pp. 99–113, May 2009.

[21] M. Westergaard and F. Maggi, "Declare: A tool suite for declarative workflow modeling and enactment," in *Proc. Demo Track 9th Conf. Bus. Process Manage.*, Clermont-Ferrand, France, vol. 820, Aug. 2011, pp. 1–5.

[22] S. Goedertier and J. Vanthienen, "Designing compliant business processes with obligations and permissions," in *Proc. Bus. Process Manage. Workshops*, J. Eder and S. Dustdar, Eds. Berlin, Germany: Springer, 2006, pp. 5–14.

[23] R. Lu, S. Sadiq, and G. Governatori, "On managing business processes variants," *Data Knowl. Eng.*, vol. 68, no. 7, pp. 642–664, Jul. 2009.

[24] T. Hildebrandt and R. Mukkamala, "Declarative event-based workflow as distributed dynamic condition response graphs," in *Proc. 3rd Workshop Program. Lang. Approaches Concurrency Commun.-Entric Softw.*, vol. 69, K. Honda and A. Mycroft, Eds., 2010, pp. 59–73.

[25] S. Schönig and S. Jablonski, "Comparing declarative process modelling languages from the organisational perspective," in *Proc. Bus. Process Manage. Workshops*, M. Reichert and H. Reijers, Eds. Cham, Switzerland: Springer, 2016, pp. 17–29.

[26] T. Slaats, D. Schunselaar, F. Maggi, and H. Reijers, "The semantics of hybrid process models," in *Proc. OTM Confederated Int. Conf. 'Move Meaningful Internet Syst.'*, in Lecture Notes in Computer Science, vol. 10033, C. Debruyne, H. Panetto, R. Meersman, T. Dillon, E. Kühn, D. O'Sullivan, and C. Ardagna, Eds., 2016, pp. 531–551.

[27] T. Slaats, "Declarative and hybrid process discovery: Recent advances and open challenges," *J. Data Semantics*, vol. 9, no. 1, pp. 3–20, Mar. 2020, doi: 10.1007/s13740-020-00112-9.

[28] G. De Giacomo, M. Dumas, F. Maggi, and M. Montali, "Declarative process modeling in BPMN," in *Proc. 27th Int. Conf. Adv. Inf. Syst. Eng.*, in Lecture Notes in Computer Science, vol. 9097, J. Zdravkovic, M. Kirikova, and P. Johannesson, Eds. Cham, Switzerland: Springer, 2015, pp. 84–100.

[29] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Inf. Syst.*, vol. 36, no. 2, pp. 498–516, Apr. 2011.

[30] C. Ekanayake, M. Dumas, L. García-Bañuelos, M. La Rosa, and A. ter Hofstede, "Approximate clone detection in repositories of business process models," in *Business Process Management*, A. Barros, A. Gal, and E. Kindler, Eds. Berlin, Germany: Springer, 2012, pp. 302–318.

[31] M. Baumann, M. Baumann, S. Schönig, and S. Jablonski, "Towards multi-perspective process model similarity matching," in *Proc. Enterprise Organizational Modeling Simulation*, J. Barjis and R. Pergl, Eds. Berlin, Germany: Springer, 2014, pp. 21–37.

[32] M. Baumann, M. H. Baumann, S. Schönig, and S. Jablonski, "Resource-aware process model similarity matching," in *Proc. Service-Oriented Comput.-ICSOC Workshops*, F. Toumani, B. Pernici, D. Grigori, D. Benslimane, J. Mendling, N. B. Hadj-Alouane, B. Blake, O. Perrin, I. Saleh Moustafa, and S. Bhiri, Eds. Cham, Switzerland: Springer, 2015, pp. 96–107.

[33] M. Baumann, "Comparing imperative and declarative process models with flow dependencies," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Mar. 2018, pp. 63–68.

[34] D. Fahland, J. Mendling, H. Reijers, B. Weber, M. Weidlich, and S. Zugal, "Declarative versus imperative process modeling languages: The issue of maintainability," in *Proc. Int. Workshops Bus. Process Manage. Workshops*, in Lecture Notes in Business Information Processing, vol. 43, S. Rinderle-Ma, S. Sadiq, and F. Leymann, Eds. Cham, Switzerland: Springer, 2009, pp. 477–488.
[35] C. Back, S. Debois, and T. Slaats, "Towards an empirical evaluation of imperative and declarative process mining," in *Proc. Adv. Conceptual Modeling*, C. Woo, J. Lu, Z. Li, T. Ling, G. Li, and M. Lee, Eds. Cham, Switzerland: Springer, 2018, pp. 191–198.
[36] C. Back, S. Debois, and T. Slaats, "Imperative versus declarative process mining: An empirical comparison," Dept. Comput. Sci., Univ. Copenhagen, Copenhagen, Denmark, Tech. Rep., 2020.
[37] D. Reißner, R. Conforti, M. Dumas, M. La Rosa, and A. Armas-Cervantes, "Scalable conformance checking of business processes," in *Proc. OTM Confederated Int. Conf. 'Move Meaningful Internet Syst.'*, H. Panetto, Eds. Cham, Switzerland: Springer, 2017, pp. 607–627.
[38] F. Maggi, M. Montali, C. Di Ciccio, and J. Mendling, "Semantical vacuity detection in declarative process mining," in *Business Process Management*, M. La Rosa, P. Loos, and O. Pastor, Eds. Cham, Switzerland: Springer, 2016, pp. 158–175.
[39] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, Mar. 2008.
[40] M. Weidlich, "Behavioural profiles: A relational approach to behaviour consistency," Doctoral thesis, Hasso Plattner Inst., Universität Potsdam, Potsdam, Germany, 2011.
[41] A. Adriansyah, "Aligning observed and modeled behavior," Ph.D. dissertation, Dept. Math. Comput. Sci., Technische Univ. Eindhoven, 2014.
[42] D. Schunselaar, F. Maggi, and N. Sidorova, "Patterns for a log-based strengthening of declarative compliance models," in *Integrated Formal Methods*, J. Derrick, S. Gnesi, D. Latella, and H. Treharne, Eds. Berlin, Germany: Springer, 2012, pp. 327–342.
[43] B. van Dongen, J. De Smedt, C. Di Ciccio, and J. Mendling, "Conformance checking of mixed-paradigm process models," 2020, *arXiv:2011.11551*.
[44] S. Mertens, F. Gailly, and G. Poels, "Enhancing declarative process models with DMN decision logic," in *Proc. Enterprise, Bus.-Process Inf. Syst. Modeling*, vol. 214, K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, and Q. Ma, Eds. Cham, Switzerland: Springer, 2015, pp. 151–165.
[45] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 686–705, Apr. 2019.
[46] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Proc. Bus. Process Manage. Workshops*, N. Lohmann, M. Song, and P. Wohed, Eds. Cham, Switzerland: Springer, 2014, pp. 66–78.
[47] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Data-driven process discovery—Revealing conditional infrequent behavior from event logs," in *Advanced Information Systems Engineering*, E. Dubois and K. Pohl, Eds. Cham, Switzerland: Springer, 2017, pp. 545–560.
[48] H. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst, "XES, XESame, and ProM 6," in *Information Systems Evolution* (Lecture Notes in Business Information Processing), vol. 72, P. Soffer and E. Proper, Eds. Cham, Switzerland: Springer, 2010, pp. 60–75.
[49] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst, "Efficient discovery of understandable declarative process models from event logs," in *Advanced Information Systems Engineering*, J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza, Eds. Berlin, Germany: Springer, 2012, pp. 270–285.
[50] A. Alman, C. D. Ciccio, D. Haas, F. M. Maggi, and A. Nolte, "Rule mining with RuM," in *Proc. 2nd Int. Conf. Process Mining (ICPM)*, Oct. 2020, pp. 121–128.
[51] E. Brzychczy and A. Trzcionkowska, "Creation of an event log from a low-level machinery monitoring system for process mining purposes," in *Intelligent Data Engineering and Automated Learning—IDEAL 2018*, H. Yin, D. Camacho, P. Novais, and A. Tallón-Ballesteros, Eds. Cham, Switzerland: Springer, 2018, pp. 54–63.

EDYTA BRZYCHCZY is currently an Associate Professor with the Faculty of Mechanical Engineering and Robotics, AGH University of Science and Technology, Kraków, Poland. She has coordinated projects related to data mining and expert systems in the mining domain. Her research interests include the modeling and optimization of industrial processes, stochastic networks, natural computing algorithms, process re-engineering based on sensor data, and process mining.

MARCIN SZPYRKA (Senior Member, IEEE) is currently a Full Professor with the Department of Applied Computer Science, AGH University of Science and Technology, Kraków, Poland. He has authored over 140 publications in the domains of formal methods, software engineering, and knowledge engineering. His research interests include the theory of concurrency, systems security, functional programming, and data science.

JACEK KORSKI is currently pursuing the Ph.D. (Eng.) degree with the ITG KOMAG Institute, Gliwice, Poland. He has authored over 130 publications in the domains of mining technology and process management and optimization in the mining industry. For more than 40 years, he was the manager in mining industry and expert and a Board Advisor with FAMUR SA. He is a fellow of the ITG KOMAG Institute.

GRZEGORZ J. NALEPA (Member, IEEE) is currently a Full Professor with Jagiellonian University, Kraków, Poland. He has coauthored over 200 research papers in international conferences and journals. He was involved in tens of projects, including research and development projects with a number of companies. His current research interests include the applications of AI in industry 4.0 and business, explainable AI, affective computing, context awareness, and the intersection of AI with law.