Using Multi-Level Information in Hierarchical Process Mining: Balancing Behavioural Quality and Model Complexity

Sander J.J. Leemans
Queensland University of Technology Australia
s.j.j.leemans@qut.edu.au

Kanika Goel
Queensland University of Technology Australia

Sebastiaan J. van Zelst
Fraunhofer FIT & RWTH University Germany

***Abstract*—Process mining techniques aim to derive knowledge of the execution of processes, by means of automated analysis of behaviour recorded in event logs. A well-known challenge in process mining is to strike an adequate balance between the behavioural quality of a discovered model compared to the event log and the model's complexity as perceived by stakeholders. At the same time, events typically contain multiple attributes related to parts of the process at different levels of abstraction, which are often ignored by existing process mining techniques, resulting in either highly complex and/or incomprehensible process mining results. This paper addresses this problem by extending process mining to use event-level attributes readily available in event logs. We introduce (1) the concept of multi-level logs and generalise existing hierarchical process models, which support multiple modelling formalisms and notions of activities in a single model, (2) a framework, instantiation and implementation for process discovery of hierarchical models, and (3) a corresponding conformance checking technique. The resulting framework has been implemented as a plug-in of the open-source process mining framework ProM, and has been evaluated qualitatively and quantitatively using multiple real-life event logs.**

***Index Terms*—Process mining, process model complexity, process discovery, hierarchical process models, multi-level event logs**

I. Introduction

Process mining [1] techniques are concerned with deriving process-based insights from historical records of business operations recorded in information system event logs. Two often-used categories of process mining techniques are *process discovery* [5] and *conformance checking* [8]. Process discovery aims to extract understandable and representative process models from event logs in order for stakeholders to uncover the real behaviour of their business processes. Many process discovery algorithms exist; each focusing on generating process models with a certain level of quality, expressed using the dimensions of fitness, precision, simplicity and generalisation [1]. A well-known challenge for any process discovery technique is to strike the right balance between the behavioural quality of a discovered model with respect to the event log and the model's complexity as perceived by stakeholders. Applying most existing process discovery techniques on real-life event logs results either in detailed highly complex and incomprehensible models (due to high numbers of activities and complex behaviour) or abstract and simple but behaviourally inaccurate models (due to leaving out or adding behaviour with respect to the log).

Despite advances made in process discovery, the understandability of the discovered models is often compromised when dealing with complex processes. The reasons for obtaining such complex models are plentiful, e.g., noise is inherently present in logs. Furthermore, often, the abstraction level at which the event attributes are recorded is often much lower than which one ideally models a business process at [39]. At the same time, this problem can be alleviated by using *hierarchical process models* as a primary process representation [28]. A hierarchical process model is characterised by hierarchy of levels, each of which captures a particular granularity of the process. Then, each step in the process constitutes a step in potentially multiple levels of the hierarchy, which inherently leads to the notion of multi-level behaviour. For instance, the following *multi-level trace* represents a hospitalisation: the first four events denote a stay at the intensive care (level 1), during which the patient was intubated (level 2), which consisted of intubation, ventilation and extubation (level 3). Afterwards, the patient was deregistered from the IC and transfered to a ward.

$$\left\langle \begin{pmatrix} \text{IC} \\ \text{intubation} \\ \text{intubate} \end{pmatrix}, \begin{pmatrix} \text{IC} \\ \text{intubation} \\ \text{ventilate} \end{pmatrix}, \begin{pmatrix} \text{IC} \\ \text{intubation} \\ \text{extubate} \end{pmatrix}, \begin{pmatrix} \text{IC} \\ \text{transfer} \\ \text{deregister} \end{pmatrix}, \begin{pmatrix} \text{Ward 1} \\ \text{transfer} \\ \text{register} \end{pmatrix}, .. \right\rangle$$

Existing discovery techniques assume process steps are represented by a single attribute (e.g., `concept:name`), thus logs such as our example (e.g. [17]) would yield either high-level models containing lots of loops (level 1) or overly spaghetti-like complex models (level 3). As we will show in Section VI, real-life event logs readily contain many more such attributes. By considering multiple attributes, "hierarchical decomposition plays a central role for organising processes in an understandable way and for refining coarse-granular towards a fine-granular representation" [26]. Our evaluation illustrates this: Figure 4f illustrates that complex models can be made simpler by splitting them up in hierarchical parts.

The use of hierarchy to improve the understandability of process models is well established [3], [4], [28], [33], [36], and recent techniques have been proposed that abstract attributes in event logs: see [39] for a literature review.

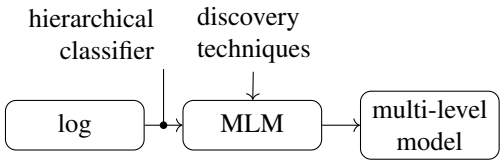


Figure 1: Context of the MLM framework.

In this paper, we introduce the concept of multi-level information in event logs, and propose hierarchical discovery and conformance checking techniques that can use this information. Existing hierarchical discovery techniques either do not consider multi-level information, are limited to 2 layers of hierarchy or are tied to specific combinations of model formalisms [25], [34], [36] (different types of behaviour may require different modelling concepts, e.g. declarative and imperative formalisms [12]). We introduce a framework for hierarchical process discovery that explicitly takes multi-level information into account and is limited in neither layers nor formalisms. This Multi-Level Miner (MLM) framework takes as input a standard event log and a multi-level classifier (which indicates the attributes involved and their order), then recursively abstracts groups of events, applies existing discovery techniques and splits the event log accordingly, to output a hierarchical process model (see Figure 1). *As such, MLM can be regarded as a standardisation effort for event logs that either inherently describe multi-level process information or that contain derived multi-level process information.*

We instantiate an algorithm using MLM and introduce a conformance checking technique to compare multi-level logs and hierarchical models. Then, we evaluate MLM with respect to existing discovery techniques, which shows that MLM can discover models with quality on par with existing techniques, while being arguably simpler to understand for users.

In short, the contributions of this paper are:

- A conceptual model and definition of multi-level logs, and definitions of generalised hierarchical models;
- An implemented, instantiated, extensible model discovery framework (MLM) that considers multi-level information;
- A corresponding conformance checking technique.

The remainder of the paper is organised as follows. Section II discusses related work. Section III introduces our multi-level languages and hierarchical process models, while Section IV presents how we discover them and Section V discusses how to check their conformance. Section VI presents key findings from our evaluations with multiple real-life event logs, and we discuss limitations in Section VII. Section VIII concludes the paper.

II. Related Work & Background

In this section, we discuss related work and concepts of process modelling formalisms, process discovery techniques, single- and multi-formalism hierarchical models, and techniques that identify multi-level structures in logs.

From an abstract point of view, a process model describes behaviour as a possibly infinite set of traces, a *language*, over the *transitions* present in the model. For instance, directly follows models/transition systems use arcs to express which transitions can be executed after each transition [24]. Process trees are a hierarchy of nodes (with transitions as leaves), each combining the languages of its sub-nodes [20]. BPMN models express complex behaviour using advanced gateways and exception handling events [9]. Declarative models explicitly express constraints over pairs of transitions [35]. Finally, a Petri net describes behaviour through occurrences of transitions that consume tokens from states (places) and produce tokens on places [29]. We assume the reader to be familiar with the basic principles of the most commonly used process mining formalisms; see [1] for an introduction.

Process modelling formalisms that support hierarchy include process trees [20], BPMN [9], declarative languages [33] and fuzzy models [15]. Our definition (Section III) generalises over these formalisms by providing a multi-formalism hierarchy, and establishes their multi-level semantics.

A plethora of non-hierarchical discovery techniques has been proposed. For a detailed overview of these techniques, please refer to [1], [5]. The approach of this paper can use any discovery technique to improve on the simplicity and understandability of the discovered models at any level, by using multi-level information from the event log.

Techniques that identify or create levels in event logs include clustering activities based on correlations [16], identifying stages [30], [9], or constructing taxonomies [14]. For a recent literature review, see [39]. Using these techniques, multi-level information can be added to event logs, which can be used as input (levels) for the MLM framework. No technique that explicitly uses this information has been proposed.

Next, we discuss approaches that aim to discover hierarchical models. In [20], process trees are discovered by splitting the event log recursively into smaller parts; the hierarchy bears no meaning (as in [37]). This is combined with other techniques in [22], where a hierarchy is obtained by combining discovery techniques: where one discovery technique gives up, another is started in a hierarchical way. In [19], hierarchy is used to express recursion, derived from inherently hierarchical software stack traces, using process trees. In [9], [10], first events are clustered, after which for each cluster BPMN models are discovered in a hierarchical way. Similarly, in [33], text mining is applied to identify related activity labels, which are consecutively collapsed in the event log and used to discover a hierarchical declarative process model. Finally, the Fuzzy Miner [15] dynamically groups activities based on their already-discovered relation with other activities, and thereby implicitly induces a hierarchy.

None of the aforementioned techniques explicitly takes multi-level information into account, i.e., even though the techniques allow us to derive a model comprising a hierarchy, the primary input is still of a "flat" nature. Moreover, none of the techniques is able to combine multiple process modelling formalisms, i.e., allow us to pick the most suitable modelling formalism for a specific level within the process.

Hybrid models combine multiple modelling formalisms to reduce complexity, in particular they use declarative and

imperative parts. In hybrid models, a transition can contain a sub-model of a different formalism. To define the semantics of hybrid models, the open-world assumption, trace termination and concurrency in atomic languages (such as Petri net) need particular attention [35]. While in hybrid models only a lowest-level transition generates an event, the formalism introduced in this paper generalises over this approach by adding support for multi-level events. Discovery techniques for hybrid models were proposed in [25], [34], [36]. Neither of these approaches consider multi-level information in event logs and the last two are limited to 2 layers, as they specifically aim to combine declarative models as sub-models of imperative models. This paper generalises these techniques by combining multiple formalisms without limiting the number of layers. Furthermore, we specifically use multi-level information in the event log to control the creation of hierarchical levels whereas previous works derive this information using e.g. clustering.

Finally, artefact-centric process mining considers the relation between multiple process/case notions [31], [13], such as orders, products and packages. This notion differs from multi-level as we assume that one level is fully contained in another, where artefact-centric assumes no such relation. Similarly, multi-dimensional process mining [38] does not consider hierarchies in a single process model.

III. Multi-Level Event Logs and Hierarchical Models

In an information system, an event e represents the occurrence of an activity. We generalise this to each event having several activities, at different levels of abstraction.

Definition 1 (Multi-level language, log)**.** *Let $\Sigma_1 ... \Sigma_m$ be alphabets of activities. Then, a* multi-level event *is a vector $(a_1, \ldots, a_m)$ with $\forall_{1 \leqslant i \leqslant m} a_i \in \Sigma_i$ of values describing a process step. A* multi-level trace *is a finite sequence of multi-level events, a* multi-level language *is a collection of multi-level traces and a* multi-level log *is a multi-level language with a finite number of traces.*

An example of a multi-level event is (Intensive Care, Intubation, Extubate), which indicates that an activity "extubate" (removing an air pipe from a patient's throat) was performed, which is the final stage of a higher-level activity/procedure "intubation", which in turn is a step in an intensive care stay. A multi-level trace in this example would be a patient being admitted to the intensive care, and the corresponding multi-level event log would be all intensive care stays recorded.

Every (standard) event log that contains multiple event attributes can be transformed into a multi-level log by deciding on the attributes that indicate the levels, and their order, using a *multi-level classifier*, which is a list of classifiers, which are lists of attributes (i.e. columns in a CSV file).

A *hierarchical model* combines process models of several formalisms to hierarchically express a multi-level language. We define hierarchical models recursively using a process model M of any formalism:

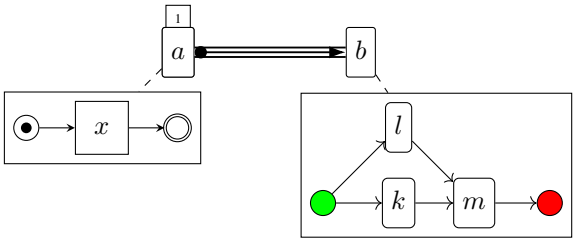


Figure 2: A multi-level process model, consisting of a Declare model (top), a Petri net and a Directly Follows model.

Definition 2 (Hierarchical model - syntax)**.** *Let Σ be an alphabet of activities. Then, a* hierarchical model *M is a process model $M(T_1, \ldots T_n)$ expressed over transitions $T_1 \ldots T_n$, where a* transition *T_i is either: (1) an activity $a \in \Sigma$; (2) a silent step $\tau \notin \Sigma$; or (3) an annotated activity $a_{M'}$ in which $a \in \Sigma$ and M' is a hierarchical model.*

Figure 2 shows an example, consisting of a Declare model, of which activity a is annotated with a sub-model (a Petri net), and activity b is annotated with a directly follows model.

In the following definition, Y replaces a single execution of a transition with the sub-model language of the transition, while X exhaustively calls Y until all transitions have been replaced:

Definition 3 (Hierarchical model - semantics)**.** *Let M be a model over transitions $T_1 \ldots T_n$. Let $\varphi(M(T_1, \ldots T_n))$ be a function that returns the language of M in terms of starts and completes of its transitions, that is, over $T_{1,s}, T_{1,c}, \ldots T_{n,s}, T_{n,c}$. Then, the multi-level language of M is:*

$$
\begin{aligned}
\mathcal{L}(M) &= X(\varphi(M)) \\
X(L) &= \begin{cases} X(Y(L)) & \text{if } \exists_{t \in L} \exists_i T_{i,s} \in t \\ L & \text{otherwise} \end{cases} \\
Y(L) &= \{t' \mid t \in L \\
&\quad \wedge\, t = t_1 \cdot \langle T_{i,s} \rangle \cdot t_2 \cdot \langle T_{i,c} \rangle \cdot t_3 \wedge T_{i,c} \notin t_2 \\
&\quad \wedge\, t' \in \{t_1\} \bullet (\mathcal{L}(T_i) \parallel \{t_2\}) \bullet \{t_3\}\} \\
&\quad \cup \{t' \in L \mid \neg \exists_{t \in L} \exists_i T_{i,s} \in t\} \\
\mathcal{L}(a) &= \{\langle (a) \rangle\} \\
\mathcal{L}(\tau) &= \{\langle\rangle\} \\
\mathcal{L}(a_{M'}) &= \{\langle \binom{a}{e_1}, \ldots \binom{a}{e_m} \rangle \mid \langle e_1, \ldots e_m \rangle \in \mathcal{L}(M')\}
\end{aligned}
$$

Where $\bullet$ is the sequential cross product and $\parallel$ is the interleaving cross product [21].

For instance, the multi-level language of the model shown in Figure 1 is $\{\langle \binom{a}{x}, \binom{b}{l}, \binom{b}{m} \rangle, \langle \binom{a}{x}, \binom{b}{k}, \binom{b}{m} \rangle\}$.

Hierarchical models are defined for any formalism for which φ can be defined, which entails the introduction of true concurrency between transition executions (which can be simulated in atomic Petri nets, process trees and directly follow models; see Section V). The only requirement to this φ is that for every trace generated, there is a bijective mapping of every

$T_{i,s}$ to/from a later following $T_{i,c}$. However, for declarative models, other factors (e.g. the open-world assumption) need to be considered [35].

In the remainder of this paper, for simplicity, we assume that the hierarchy of a hierarchical model is of a consistent depth, even though this is not required by our definitions (that is, such models simply emit multi-level events of uneven sizes), and we may omit the "multi-level" prefix where appropriate.

IV. Discovering Multi-Level Models

In this section, we introduce the *Multi-Level Miner (MLM)* framework. We first give its pseudocode and explain its key steps, after which we give a specific algorithm implementing it, and we sketch strategies in case extra information is available.

A. The Multi-Level Miner Framework

As input, MLM takes a multi-level language (which can be obtained from an event log using a multi-level classifier) and a list of discovery techniques $D_1 \dots D_m$ to be applied over m levels of attributes.

Intuitively, MLM first prepares the event log by abstracting groups of events to the current level l, then applies a discovery technique, and finally recurses on the transitions of the discovered model, replacing each transition with a process model of the next level. Thus, the framework uses two key functions that algorithms implementing the framework must provide for each level $l \in \{1 \dots m\}$: event group abstraction P_l and log splitting S_l. The event group abstraction function P_l aims to remove repetition from traces. For instance, a "flat" discovery technique would falsely consider the trace $\langle a, b, b \rangle$ to contain a repetition of b. However, these bs indicate lower level steps, rather than a repetition of b itself. Thus, P_l abstracts this trace into $\langle a, b \rangle$ by removing the repeated b. After a process discovery technique has been applied, the log splitting step S_l splits the (non-abstracted) log into sub-logs: one sub-log for each transition in the discovered process model. Intuitively, each sub-log contains a trace for each execution of the corresponding activity. In our example, the trace would be split into $\langle a \rangle$ and $\langle b, b \rangle$.

To formally define MLM, we use a projection function ϕ that projects a particular level of an event to its l^{th} attribute value, and by extension multi-level traces and multi-level languages onto traces and languages: $\phi((a_1, \dots a_m), j) = a_j$. Then, the MLM framework is defined as follows:

procedure $MLM_{P,D,S}$(multi-level language L, level l)
  if l is the bottom level **then**
    $M \leftarrow D_l(\phi(L, l))$ ▷ discover a process model
  else
    $L' \leftarrow P_l(L)$ ▷ abstract groups in L
    $M \leftarrow D_l(\phi(L', l))$ ▷ discover a process model
    for all transition $t \in M$ **do**
      $L_t \leftarrow S_l(M, L)$ ▷ split L using M
      $M_t \leftarrow$ MLM($L_t, l{+}1$) ▷ recurse
      replace t with t_{M_t} in M ▷ link t to sub-model M_t
    end for
  end if
  return M
end procedure

B. An Example Instantiation

To illustrate the MLM framework, we instantiate it in an actual algorithm as follows. For the log abstraction function P_l, consecutive appearances of the same activity are removed. For instance, the trace $\langle \binom{a}{x}, \binom{a}{y}, \binom{b}{x} \rangle$ at level 1 would be filtered to $\langle \binom{a}{x}, \binom{b}{x} \rangle$. For the log splitting function S_l, (1) the discovered model is transformed to allow for repeated executions of activities by replacing each transition as in Table I; (2) alignments [2] are computed to link events in the log to a transition in the transformed model, and to remove non-fitting events, which makes the instantiation non-deterministic; and (3) for each transition T, the trace is split into sub-trace(s): events that are not linked to T are removed, and a new sub-trace is started depending on the model formalism. [directly follows models] a new sub-trace is started when an event occurs of a transition $T' \neq T$ (as directly-follows models do not support concurrency); [process trees] a new sub-trace is started when an event occurs of a transition $T' \neq T$ such that T' is not concurrent with T in the tree; [Petri nets] a new sub-trace is started when an event occurs of any transition.

For instance, consider the top-level model of Figure 2 and the trace $\langle \binom{a}{x}, \binom{b}{k}, \binom{b}{m} \rangle$ at level 1. This trace would be split into the subtrace $\langle \binom{a}{x} \rangle$ for transition a and the subtrace $\langle \binom{b}{k}, \binom{b}{m} \rangle$ for transition b.

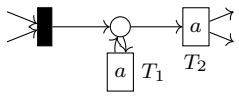
V. Conformance Checking

To compare a hierarchical model with a multi-level log, existing state-space exploring conformance checking techniques could be updated to support multi-level logs. However, we can use existing techniques unchanged by translating a hierarchical model to a workflow net (*flattening*), if all levels are individually translatable to workflow nets. In the following, let M be a process model translated to a workflow net over transitions $(T_1, \dots T_n)$, which are replaced as follows:

a, τ are not replaced;

$a_{M'}$ (1) the sub-model M' is recursively translated to a workflow net having a source place p_{source} and a sink place p_{sink}; (2) all labelled transitions of M' get their labels superpositioned with a, e.g. $(x) \rightarrow \binom{a}{x}$; (3) a silent transition t_{start} is added, which has the same input arcs as $a_{M'}$ and one output arc to p_{source}; and (4) another silent transition t_{end} is added with one input arc from p_{sink} and

Table I: Log splitting: model transformation for several formalisms.

Formalism	transition	transformed
Directly follows model		
Process tree	a	$\circlearrowleft(a, \tau)$
Petri net		

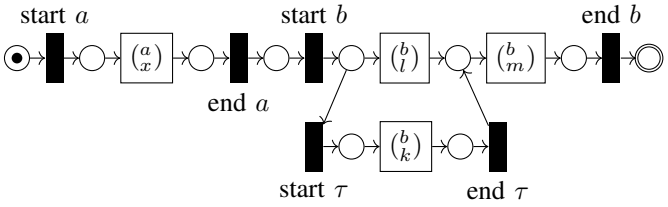


Figure 3: Our multi-level model flattened into a Petri net.

the same output arcs as $a_{M'}$. The remaining net is sound, that is, free of deadlocks and other anomalies [20], if all of the sub-model-workflow nets are sound;

For instance, the hierarchical model shown in Figure 2 would be flattened to Figure 3.

VI. Evaluation

In this section, we evaluate the MLM framework: first, we illustrate its applicability in a qualitative comparison with other discovery techniques. Second, we evaluate the quality of the models discovered by the MLM framework.

Reproducibility. All event logs used in this paper are publicly available, as well as the source code of our technique.

A. Implementation

Both the MLM framework instantiation (Section IV-B) and the conformance checking technique (Section V) have been implemented as plug-ins of the ProM framework [11], in package `multi-level miner`. The MLM and its implementation allow for easy extension with other discovery techniques and sub-model formalisms; currently Petri nets, directly follows models, BPMN models and process trees are supported. The resulting hierarchical model can be visualised in two ways: either as a single model (as in Figure 4f), or using a separate visualisation for each level, where a user can click on an annotated node, which opens the corresponding sub-model (Figure 4e).

B. Qualitative Evaluation

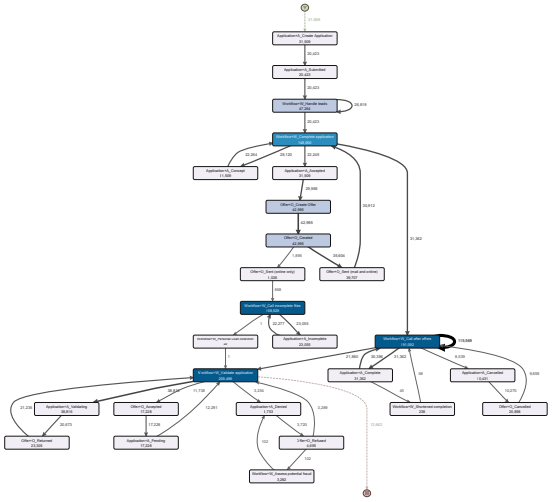
To study qualitative and understandability aspects of the MLM framework, we applied Disco (see http://fluxicon.com), Declare [35], Inductive Miner - infrequent [20] and Split Miner [6] at their default settings to the BPI Challenge 2017 log, which was recorded from a loan application process at a financial institution in the Netherlands. Based on domain knowledge and a small explorative analysis, we chose the attributes `eventOrigin` and `concept:name` as the multi-level classifier, and chose the Directly Follows Miner [24] and the Inductive Miner as discovery techniques for the MLM framework. The discovered models are shown in Figure 4. A visual inspection of these models immediately reveals that the multi-level model is much simpler and understandable than any of the other models, especially on more abstract levels. Hierarchical models enable an understanding of the information in the logs in a systematic manner, by displaying the behaviour in layers. For instance, the model obtained using Disco may not be complex on first sight (even though it is highly filtered as per default), although, understanding the flow of activities under a particular event origin is not easy, unlike in multi-level models. Other models might be more structured (Inductive Miner), more flexible (Declare) or free of concurrency (Split Miner), however the MLM model is arguably simpler.

In [28], seven Process Modelling Guidelines for understandability were identified, of which four are relevant for this experiment: (G1) use as few model elements as possible: compared to the other process models, the MLM model decomposes information in layers which contain fewer elements. This enables users to focus on a particular section of the model rather than on all the elements at the same time, even though the total number of elements might be higher. (G2) minimise the routing paths per element: the experiment did not show large differences between the techniques. (G4) model as structured as possible. MLM introduces structure by its layers, which for some models increases the structuredness of the entire model compared to other techniques. (G7) decompose a model with more than 30 elements: our technique enables decomposition of the model increasing the understandability of the discovered model. Overall, we can see that our model improves on most of the relevant guidelines, enabling better understanding of the behaviour in the logs.

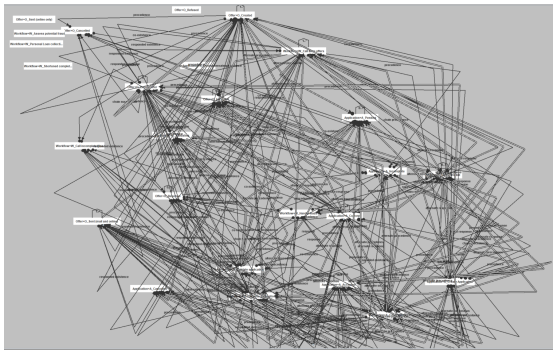
C. Quantitative Evaluation

To compare the MLM framework with existing discovery techniques, we applied several techniques to six publicly available real-life event logs. These logs are publicly available from https://data.4tu.nl/repository/collection:event_logs_real, and were chosen as they contain multiple event attributes that might serve as activities. As a first step, for each log we manually chose a multi-level classifier, based on domain knowledge on the mentioned web page. The event logs are summarised in Table II, which also shows the chosen classifiers. The included existing discovery techniques are Split Miner (SM) [6], Inductive Miner - infrequent (IM) [20] and Directly Follows Model Miner (DM) [24]). Furthermore, we included two baseline techniques: the trace model (T) that represents all traces of the event log, and the flower model (F) that represents all possible behaviour given the activities in the log. Finally, we included several instantiations of the MLM framework: (a) three instantiations that use the same existing discovery technique for each of its levels, using Split Miner (MLM-SM), Inductive Miner - infrequent (MLM-IM) and Directly Follows Model Miner (MLM-DM); (b) an instantiation that uses the flower model for each of its levels (MLM-F); a trace-MLM would not involve multiple layers and was not included; (c) an instantiation that combines Directly Follows Model Miner and Inductive Miner - infrequent (MLM-DM-IM).

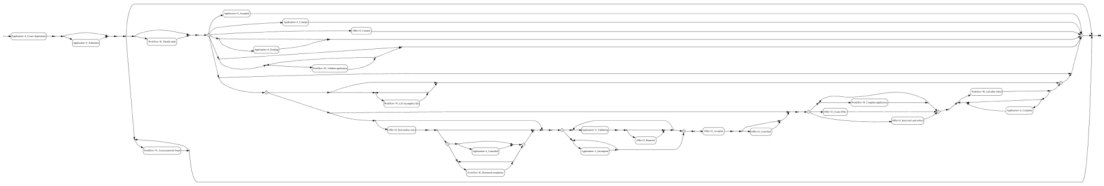
To these logs and discovery techniques, we applied 3-fold cross validation: the log was split into 3 buckets, and for each bucket a model was discovered, which was validated against the remaining buckets. To minimise the influence of randomness in the cross validation, this validation was repeated 10 times. Thus, for each log and discovery technique, 30 models



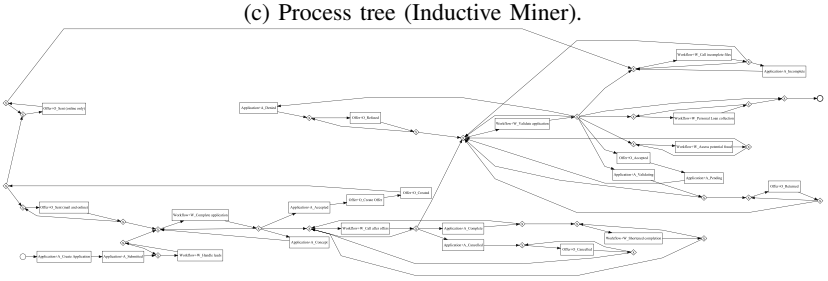
(a) Directly follows model (Fluxicon Disco).



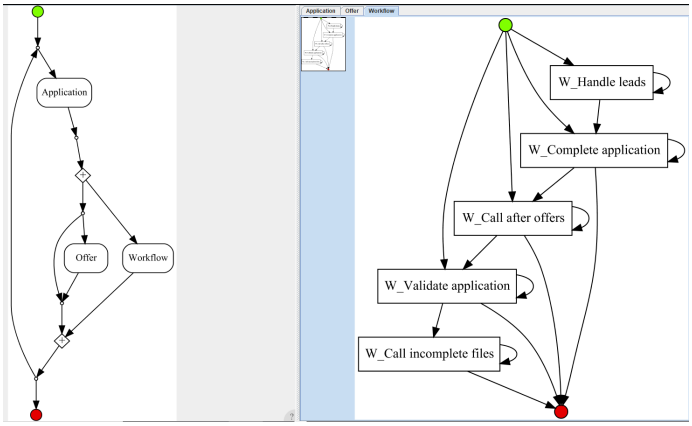
(b) Declarative model (Declare).



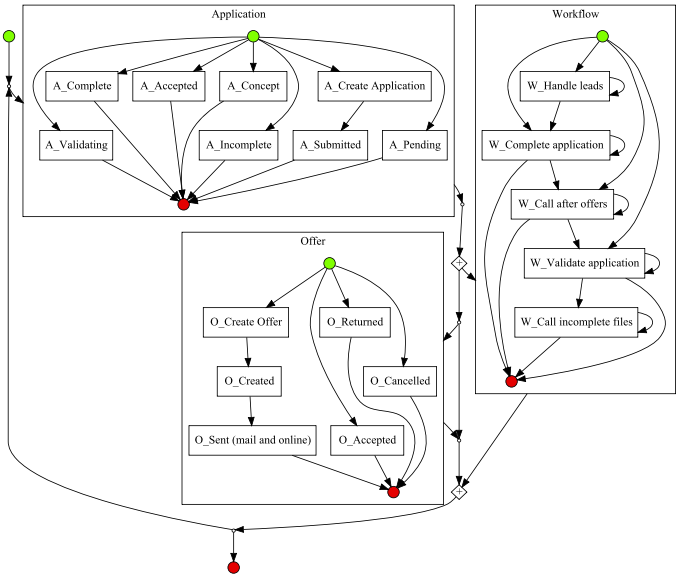
(c) Process tree (Inductive Miner).



(d) BPMN model (Split Miner).



(e) This paper - MLM-DM-IM (tabbed).



(f) This paper - MLM-DM-IM (hierarchical).

Figure 4: Impressions of mined process models for our qualitative evaluation.

Table II: Event logs used in the quantitative evaluation.

log	multi-level classifier	traces	events	acts
bpic 2012	concept:name, lifecycle:transition	13087	262200	36
bpic 2013 closed problems	concept:name, lifecycle:transition	1487	6660	7
bpic 2015 - 1	monitoringResource, activityNameEN	1199	52217	2302
bpic 2017	eventOrigin, concept:name	31509	1202267	26
bpic 2018 payment applications	subprocess, concept:name, lifecyle:transition	43809	984613	52
sepsis	org:group, concept:name	1050	15214	42

were discovered and evaluated. The reported measures are the averages over these 30 measures.

For each log and discovered model, we measured projected fitness and precision [23] on a Petri-net representation of the model; the MLM-models were flattened using the method described in Section V. Furthermore, we recorded the total number of nodes and edges, and the average connector degree. Together, these measures have been shown to be significantly negatively correlated with understandability [27], [32]. Both the total number of nodes and edges and the average connector degree were measured on a flattened Petri net (as a machine would) and on the native graph visualisation of the model (as a user would), where for MLM-models we averaged the values over the sub-models in the MLM-models. We refer to these measures as automated size and user size.

Table III shows an excerpt of the results, with the full results available in a technical report at https://eprints.qut.edu.au/134958/. While run time of MLM was considerably longer than for the other discovery techniques, due to the repeated application of alignments, in this experiment run time never exceeded two hours, and was generally lower than 20 minutes. Thus, while feasible for the real-life logs considered here, larger or more complex logs with more traces, events per trace or activity might prove challenging. For some models, fitness and precision could not be measured. Remarkably, on bpic17, several variants of MLM got the same precision as the flower model, even though the corresponding models (Figure 4f) clearly limit behaviour considerably, and this was not the case for the other logs. Comparing an algorithm and the same algorithm in MLM, we see slight differences in both directions for fitness and precision, but consistently smaller models. For all event logs, F was pareto optimal (over fitness, precision and user-average-size) once, T 5 times, DF 4 times, IM 3 times, SM 3 times, MLM-F 5 times, MLM-DM 5 times, MLM-IM once, MLM-SM once, and MLM-DM-IM 4 times. MLM techniques got the highest fitness for 3 logs (disregarding the baselines MLM-F and F), and the highest precision for 2. This shows that MLM scores comparable or better on the key metrics that are related to the understandability of process models, and comparable on fitness and precision, compared to existing techniques.

We conclude that even though the hierarchical models are slightly more complex for automated analysis (as elements are introduced with flattening), these models are much simpler for user analysis (considering the average size per level/sub-model), and have a similar quality as existing techniques.

VII. Discussion & Limitations

While the positive effects of hierarchy on understandability are well-established in process modelling literature [3], [4], [28], [33], [36], the extent to which multiple formalisms might increase complexity requires further research. Thus, more research is necessary into simplicity measures that compare hierarchical and "flat" process models fairly, and into users' perceptions of these measures.

In our evaluation, we selected multi-level classifiers based on domain knowledge and on a quick analysis of short loops for each attribute. We argue that this adds little complexity over manually choosing a single classifier, as it requires choosing multiple columns of a CSV file rather than a single one. Obviously, the outcomes of multi-level process mining highly depend on this choice, it may introduce a bias of the results, and it requires that multiple suitable attributes are present in the log. Users might need to iterate to find suitable multi-level classifiers, discovery techniques and formalisms. In future work, it would be interesting to explore methodologies, scoring and best-practices to recommend these (e.g. as in [7]), and to study their influence in detail.

The abstraction step (5) of MLM suffers from a chicken-and-egg problem when regarding concurrency: in order to abstract, knowledge of the process model would be beneficial, however that is not available at that step. This manifests in our instantiation in two situations. First, our event-group abstraction step cannot be aware of concurrency: if executions of two activities a and b overlap by having alternating events, then our instantiation will split these executions whenever an alternating event occurs. For instance, $\langle \binom{a}{x}, \binom{b}{y}, \binom{a}{y}, \binom{b}{x} \rangle$ would be split into $\langle \binom{a}{x} \rangle$ and $\langle \binom{a}{y} \rangle$ for a, and $\langle \binom{b}{y} \rangle$ and $\langle \binom{b}{x} \rangle$ for b, rather than $\langle \binom{a}{x}, \binom{a}{y} \rangle$ and $\langle \binom{b}{y}, \binom{b}{x} \rangle$. However, if the discovery technique (step 6) nevertheless discovers the concurrency, the log splitting (step 8) will split the log correctly. Second, repeated execution of an activity (length-1-loops) challenges both abstraction and log splitting: given a repeated execution of an activity (for instance, $\langle a, a, a, a \rangle$), without further information, both steps cannot decide how many executions of a are present, nor where one execution ends and the next one starts. In future work, both could be addressed using attributes such as `concept:instance` or `lifecycle:transition`.

Finally, our conformance checking approach could be extended to support declarative formalisms [35].

VIII. Conclusion

Process mining is challenged by the need to strike a good balance between model complexity and behavioural quality of process models. In this paper, we introduced multi-level logs and generalised hierarchical models, which enable the use of multi-level data in existing logs to discover simpler models, and generalise over existing approaches by combining multiple formalisms with multiple levels of abstraction. We introduced a corresponding discovery framework and conformance checking technique, working on standard event logs,

Table III: Quantitative experiment results, with pareto-optimal results in **bold**. For all 6 logs, see https://eprints.qut.edu.au/134958/.

log	algorithm	fitness	precision	automated		user		time
				size	avg con deg	avg size	avg conn deg	(ms)
bpic17	F	1.00	0.60	78.50	3.85	58.67	1.97	1175.00
	T	**1.00**	**0.73**	2121196.70	2.02	**1088537.93**	2.00	816.97
	DM	**0.99**	**0.71**	213.23	2.90	**213.23**	2.90	4605.50
	SM	0.91	0.69	218.00	2.54	142.00	2.90	43399.23
	IM	0.95	0.69	178.77	2.67	147.13	1.99	370274.87
	MLM-F	**1.00**	**0.60**	111.50	3.33	**34.63**	2.83	1115102.67
	MLM-DM	**0.95**	**0.69**	237.00	3.10	**10.00**	1.06	31809.00
	MLM-IM	1.00	0.60	214.17	2.73	46.10	1.95	147507.60
⋮	MLM-SM	1.00	0.60	235.67	2.73	45.08	2.80	68784.10
	MLM-DM-IM	1.00	0.60	437.80	2.84	42.10	1.97	136822.30

where a user selects several attributes. The proposed approach has been evaluated in two ways: a qualitative evaluation on a real-life event log demonstrated that discovered hierarchical models can be much simpler and more understandable than flat models. Second, a quantitative evaluation on six real-life event logs showed that our approach scores better on the key metrics that are related to the understandability of process models, and comparable on fitness and precision to existing techniques.

There are several avenues for future work. Better ways to visualise hierarchical process models and evaluate the impact of visualisation on the understandability of these models could be explored. Also, automatic suggestions for suitable attributes and corresponding discovery techniques could be developed [7], and our approach could be combined with clustering-based hierarchical discovery techniques [39], [10]. Furthermore, performance measures for hierarchical models can be investigated, similar to [18].

Acknowledgement. We thank Dirk Fahland and Moe Wynn for their contributions to earlier versions of this paper.

REFERENCES

[1] van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer (2016)
[2] van der Aalst, W.M.P., et al.: Replaying history on process models for conformance checking and performance analysis. DMKD **2**(2), 182–192 (2012)
[3] Andaloussi, A.A., et al.: Evaluating the understandability of hybrid process model representations using eye tracking: First insights. In: BPM workshops. pp. 475–481 (2018)
[4] Andaloussi, A.A., et al.: Exploring the understandability of a hybrid process design artifact based on DCR graphs. In: CAiSE workshops. pp. 69–84 (2019)
[5] Augusto, A., et al.: Automated discovery of process models from event logs: Review and benchmark. TKDE **31**(4), 686–705 (2019)
[6] Augusto, A., et al.: Split miner: automated discovery of accurate and simple business process models from event logs. KaIS **59**(2), 251–284 (2019)
[7] Back, C.O., et al.: Towards an empirical evaluation of imperative and declarative process mining. In: ER workshops. pp. 191–198 (2018)
[8] Carmona, J., et al.: Conformance Checking - Relating Processes and Models. Springer (2018)
[9] Conforti, R., et al.: Beyond tasks and gateways: Discovering bpmn models with subprocesses, boundary events and activity markers. In: BPM. pp. 101–117 (2014)
[10] Conforti, R., et al.: BPMN miner: Automated discovery of BPMN process models with hierarchical structure. IS **56**, 284–303 (2016)
[11] van Dongen, B.F., et al.: The ProM framework: A new era in process mining tool support. In: Petri Nets. pp. 444–454 (2005)
[12] Fahland, D., et al.: Declarative versus imperative process modeling languages: The issue of understandability. In: CAiSE workshops. pp. 353–366 (2009)
[13] Fahland, D., et al.: Conformance checking of interacting processes with overlapping instances. In: BPM. pp. 345–361 (2011)
[14] Greco, G., et al.: Mining taxonomies of process models. DKE **67**(1), 74–102 (2008)
[15] Günther, C.W., van der Aalst, W.M.: Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In: BPM. pp. 328–343 (2007)
[16] Günther, C.W., Rozinat, A., van der Aalst, W.M.: Activity mining by global trace segmentation. In: BPM. pp. 128–139 (2009)
[17] Johnson, A.E., et al.: Mimic-iii, a freely accessible critical care database. Scientific data **3**, 160035 (2016)
[18] Leemans, M., et al.: Hierarchical performance analysis for process mining. In: ICSSP. pp. 96–105 (2018)
[19] Leemans, M., et al.: Recursion aware modeling and discovery for hierarchical software event log analysis. In: SANER. pp. 185–196 (2018)
[20] Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: BPM workshops. pp. 66–78 (2013)
[21] Leemans, S.J.J., et al.: Discovering block-structured process models from event logs - A constructive approach. In: Petri Nets. pp. 311–329 (2013)
[22] Leemans, S.J.J., et al.: Indulpet miner: Combining discovery algorithms. In: CoopIS. vol. 11229, pp. 97–115 (2018)
[23] Leemans, S.J.J., et al.: Scalable process discovery and conformance checking. SoSyM **17**(2), 599–631 (2018)
[24] Leemans, S.J.J., et al.: Directly follows-based process mining: Exploration & a case study. In: ICPM. pp. 25–32 (2019)
[25] Maggi, F.M., et al.: The automated discovery of hybrid processes. In: BPM. pp. 392–399 (2014)
[26] Malinova, M., et al.: An empirical investigation on the design of process architectures. In: ICW. pp. 1197–1211 (2013)
[27] Mendling, J., et al.: What makes process models understandable? In: BPM. pp. 48–63 (2007)
[28] Mendling, J., et al.: Seven process modeling guidelines (7PMG). IST **52**(2), 127–136 (2010)
[29] Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
[30] Nguyen, H., et al.: Stage-based discovery of business process models from event logs. IS **84**, 214–237 (2019)
[31] Popova, V., et al.: Artifact lifecycle discovery. IJCIS **24**(01), 1550001 (2015)
[32] Reijers, H.A., Mendling, J.: A study into the factors that influence the understandability of business process models. TSMCP **41**(3), 449–462 (2011)
[33] Richetti, P.H.P., et al.: Declarative process mining: Reducing discovered models complexity by pre-processing event logs. In: BPM. vol. 8659, pp. 400–407 (2014)
[34] Schunselaar, D.M.M., et al.: Mining hybrid business process models: A quest for better precision. In: BIS. pp. 190–205 (2018)
[35] Slaats, T., et al.: The semantics of hybrid process models. In: CoopIS. pp. 531–551 (2016)
[36] Smedt, J.D., et al.: Fusion miner: Process discovery for mixed-paradigm models. DSS **77**, 123–136 (2015)
[37] Verbeek, E.: Decomposed process mining with divideandconquer. In: BPM (Demos). p. 86 (2014)
[38] Vogelgesang, T., et al.: Multidimensional process mining: Questions, requirements, and limitations. In: CAiSE. vol. 1612, pp. 169–176 (2016)
[39] van Zelst, S., et al.: Event abstraction in process mining -literature review and taxonomy. Granular Computing (04 2020)