# DISCOVERING MULTI-PERSPECTIVE PROCESS MODELS

Francesco Folino[1], Gianluigi Greco[2], Antonella Guzzo[3] and Luigi Pontieri[1]
[1]*ICAR-CNR, via P. Bucci 41C, I87036 Rende, Italy*
[2]*Dept. of Mathematics, UNICAL, Via P. Bucci 30B, I87036 Rende, Italy*
[3]*DEIS, UNICAL, Via P. Bucci 41C, I87036 Rende, Italy*

Keywords: Business Process Intelligence, Process Mining, Decision Trees.

Abstract: Process Mining techniques exploit the information stored in the executions log of a process in order to extract some high-level process model, which can be used for both analysis and design tasks. Most of these techniques focus on "structural" (control-flow oriented) aspects of the process, in that they only consider what elementary activities were executed and in which ordering. In this way, any other "non-structural" information, usually kept in real log systems (e.g., activity executors, parameter values, and time-stamps), is completely disregarded, yet being a potential source of knowledge. In this paper, we overcome this limitation by proposing a novel approach for discovering process models, where the behavior of a process is characterized from both structural and non-structural viewpoints. In a nutshell, different variants of the process (classes) are recognized through a structural clustering approach, and represented with a collection of specific workflow models. Relevant correlations between these classes and non-structural properties are made explicit through a rule-based classification model, which can be exploited for both explanation and prediction purposes. Results on real-life application scenario evidence that the discovered models are often very accurate and capture important knowledge on the process behavior.

## 1 INTRODUCTION

Process mining techniques have been getting increasing attention in the Business Process Management community because of their ability of characterizing and analyzing process behaviors, which turns out to be particularly useful in the (re)-design of complex systems. In fact, these techniques are aimed at extracting a model for a process, based on the data gathered during its past enactments and stored in suitable log files by workflow (or similar kinds of transactional) systems. In particular, traditional and consolidated approaches —see, e.g., (van der Aalst et al., 2003) for a survey on this topic— are focused on "structural" aspects of the process, i.e., they try to single out mutual dependencies among the activities involved in the process, in terms of relationships of precedence and/or in terms of various routing constructs (such as parallelism, synchronization, exclusive choice, and loops).

Only very recently, process mining techniques have been proposed to deal with important "non-structural" aspects of the process, such as activity executors, parameter values, and performance data. For instance, decision trees have been used in (Ly et al., 2005) to express staff assignment rules, which correlate agent profiles with the execution of process activities; while, a *decision point analysis* has been discussed in (Rozinat and van der Aalst, 2006), where decision trees are used to determine, at each split point of the model, the activities that are most likely to be executed next. Actually, despite the improvements w.r.t. classical approaches, these techniques still exploit a few simplifying assumptions that limit their applicability in some application contexts. In particular, the various approaches share the basic idea of using non-structural data to characterize *which* activities are to be executed, by completely disregarding the specific coordination mechanisms of their enactment, i.e., *how* the flow of execution is influenced by certain data values.

As an example, in a sales order process where different sub-processes are enacted depending on whether the customer is a novel one or has, instead, some fidelity card, current approaches will hardly discover that two different usage scenarios occur and that they can moreover be discriminated by some specific data fields associated with the customer.

The aim of this paper is precisely to enhance current techniques with the capability of characterizing



Folino F., Greco G., Guzzo A. and Pontieri L. (2008).
DISCOVERING MULTI-PERSPECTIVE PROCESS MODELS.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - AIDSS*, pages 70-77
DOI: 10.5220/0001705700700077

how activity executors, parameter values and performance data affect routing constructs and, more generally, how they determine the sub-processes to be executed. To this end, we shall investigate on the mining of a *multi-perspective* process model, where structural aspects and non-structural ones are formally related with each other. In fact, the model consists of:

- a (structure-centric) behavioral schema, where the various sub-processes occurring in the process being analyzed are explicitly and independently described; and,
- a (data-dependent) classification model, assessing in terms of decision rules over process instance attributes when the various sub-processes have to be enacted.

From a technical point of view, on the one hand the behavioral schema will be discovered by resorting to the structural clustering approach presented in (Greco et al., 2006), where log instances are partitioned into a number of clusters according to the sequence of tasks performed, and where each cluster is eventually equipped with a specific workflow schema. On the other hand, the classification model will be built by means of a rule-based classifier, using the clusters previously found as class labels for all their associated traces.

The whole approach has been implemented as a plug-in for the ProM framework (van Dongen et al., 2005), a powerful platform for the analysis of process logs, and has been tested over a challenging real-life application scenario, regarding the trading of containers in an Italian harbor. Results of experimentation evidence that very accurate models can be found with our technique, which moreover provides useful hints for the tuning of the involved logistic operations.

**Organization.** The remainder of the paper is organized as follows. After introducing some notation and basic concepts, Section 2 discusses the proposed approach in a detailed way, and reports some basic implementation issues about the integration within ProM. Then, the real-life case study and results of experimental activity are discussed in Section 3. Finally, a few concluding remarks are reported in Section 4.

## 2 MULTI-PERSPECTIVE PROCESS MODELS

Process logs contain a wide range of information about process executions. Following a standard approach in the literature, we next shall adopt a simple representation of process logs, where each trace storing a single enactment of the process (named, *process instance* or *case*) is just viewed as a sequence of task identifiers, while additional data are represented via attributes associated with traces.

Let $T$ be a set of task labels, and $A = \{A_1, \ldots, A_n\}$ be a set of (names of) process attributes, taking values from the domains $D_1, \ldots, D_n$, respectively. Then, a trace (over $T$ and $A$) is a sequence $s = [s_1, ..., s_k]$ of task labels in $T$, associated with a tuple $data(s)$ of values from $D_1, \ldots, D_n$, i.e., $data(s) \in D_1 \times \ldots \times D_n$. A *process log* (or simply *log*) over $T$ and $A$ is a set of traces over $T$ and $A$.

By inspecting and analyzing a process log, our approach aims at extracting a high-level representation of the (possibly unknown) underlying process, which we call *multi-perspective process model*. Formally, a multi-perspective process model for a log $L$ is quadruple $\mathcal{M} = \langle \mathcal{C}^L, \mathcal{W}^T, \lambda, \delta^A \rangle$ such that:

- $\mathcal{C}^L = \{C_1^L, \ldots C_q^L\}$ is a partition of the traces in $L$, i.e., $\bigcap_{i=1..q} C_i^L = \emptyset$ and $\bigcup_{i=1..q} C_i^L = L$;
- $\mathcal{W}^T = \{W_1^T, \ldots W_q^T\}$ is a set of workflow schemas, one for each cluster in $C^L$, where all tasks are named with labels from $T$;
- $\lambda$ is bijective function mapping the clusters $\mathcal{C}^L$ to their corresponding workflow schemas in $\mathcal{W}^T$, i.e., $\lambda : \mathcal{C}^L \rightarrow \mathcal{W}^T$, where $\lambda(C_i^L)$ is the schema modeling cluster $C_i^L$;
- $\delta^A : D_1 \times \ldots \times D_n \rightarrow \mathcal{C}^L$ is $q$-ary classification function, which discriminates among the clusters in $\mathcal{C}^L$ based on the values of $A$'s attributes.

In words, $\mathcal{W}^T$ is a modular description of the behavior registered in the log, as far as concerns structural aspects only, where each execution cluster is represented with a separate workflow schema, describing the typical way tasks are executed in that cluster. Conversely, $\delta^A$ is a classification function that correlates these structural clusters with (non-structural) trace attributes, by mapping any $n$-ple of values for the attributes $A_1$, ..., $A_n$ into a single cluster of $\mathcal{C}^L$.

For notation convenience, in the rest of the paper the above model will be seen as two sub-models, which represent the process in different and yet complementary ways: a *structural model* $\langle \mathcal{C}^L, \mathcal{W}^T, \lambda \rangle$, which focuses on the different execution flows across the process tasks, and a *data-aware classification model*, which is essentially encoded in the function $\delta^A$ in the form of a decision tree (Buntine, 1992). In fact, yet being strictly connected between each other, both models are relevant in themselves, for they take care of separate aspects of the process.

## 2.1 Algorithmic Issues

**Input:** A log $L$ over tasks $T$ and attributes $A$.

**Output:** A *multi-perspective process model* for $L$.

**Method:** Perform the following steps:

1 $\mathcal{C}^L$ :=`structuralClustering` $(L)$;
2 $\delta^A$:= `induceClassifier` $(\mathcal{C}^L)$;
3 $\mathcal{C}^L$:= `classifyTraces` $(L,\delta)$;
4 **for each** $C_i \in \mathcal{C}^L$ **do**
5 $W_i'$ :=`WFDiscovery` $(C_i)$;
6 $\mathcal{W}^T := \mathcal{W}^T - \{\lambda(C_i)\} \cup \{W_i'\}$;
7 $\lambda(C_i) := \{W_i'\}$;
8 **end**
9 **return** $\langle \mathcal{C}^L, \mathcal{W}^T, \lambda, \delta^A \rangle$

Figure 1: **Algorithm** `ProcessDiscovery`.

As discussed in the introduction, the problem of discovering process models based on structural information has been widely investigated in the literature. In Figure 1 an algorithm (named `ProcessDiscovery` ) is reported that addresses instead the more general problem of discovering a high-level process representation, by computing a *multi-perspective process model* for a given log $L$.

To this aim, the recursive clustering scheme recently proposed in (Greco et al., 2006) is first exploited as a sub-routine (function `structuralClustering` ) to recognize behaviorally homogenous groups of traces. Actually, log traces are partitioned by applying the well-known $k$-means method (Mitchell, 1997) to a vectorial representation of the traces, obtained by using a special kind of sequential patterns over the tasks —further details can be found in (Greco et al., 2006).

Afterwards, the algorithm takes advantage of all available data attributes by invocating the function `induceClassifier` , which derives a classification function (see Step 2), expressing the mapping from data attributes to the clusters discovered previously. Note that, by using the clusters produced by `structuralCLustering` as class labels for all their associated traces, we may learn a classification model over the attributes in $A$. Indeed, a wide variety of models and algorithms might be exploited to this end. Among them, we selected the popular formalism of decision trees, mainly for the following two reasons. First, decision trees can easily be understood and do not require any prior assumptions on data distribution; and, second, a number of algorithms are available that compute them in an efficient way, which cope quite well with noise and overfitting as well as with missing values.

The remainder of the `ProcessDiscovery` algorithm is meant to refine the structural model computed in the first phase, by making it more consistent with the classification function $\delta$ just discovered. Indeed, all the log traces are first remapped to the clusters, in Step 3, by using the model in a predictive way. A new workflow model is then discovered for each cluster, in Steps 4-8, which should model more precisely the structure of the traces that the cluster has been made contain—note that the function `mineWF` basically stands for any standard process mining algorithm in the literature, such as those discussed in the survey (van der Aalst et al., 2003). Eventually, the algorithm concludes by returning a *multi-perspective process model* integrating the different kinds of knowledge discovered.

## 2.2 Implementation Issues

The above approach has been implemented into a prototype system, which has been integrated as a plug-in within the ProM framework (van Dongen et al., 2005), a powerful platform for the analysis of process logs, quite popular in the Process Mining community.

The logical architecture of the system is sketched Figure 2, where solid arrow lines stand for information exchange. Note that the whole mining process is driven by the module *Process Mining Handler*, while the other modules roughly replicate the computation scheme in Figure 1. By *Log Repository* we denote a collection of existing process logs, represented in MXML (van Dongen and van der Aalst, 2005), a format shared by many process mining tools (including, in particular, the ProM framework).

The *Structural Mining* module, together with the ones connected with it, is responsible for the construction of structural models. In particular, the discovery of different trace clusters is carried out by the module *Structural Clustering*, which exploits some functionalities of the plug-in *DWS* implementing the clustering scheme in (Greco et al., 2006). The module *WFDiscovery* is used, instead, to derive a workflow schema for each discovered cluster, by exploiting the ProM implementation of the Heuristic miner algorithm (Weijters and van der Aalst, 2003).

Discovered clusters and schemas are stored in the *Trace Clusters* repository and in the *Workflow Repository*, respectively, for inspecting and further analysis.

The *Training Set Builder* module mainly labels each trace with the name of the cluster it has been assigned to, as it is registered in the *Trace Clusters* repository, in order to provide the *Classifier induction* module with a training set for learning a classification model. The module *Classifier* applies such a model

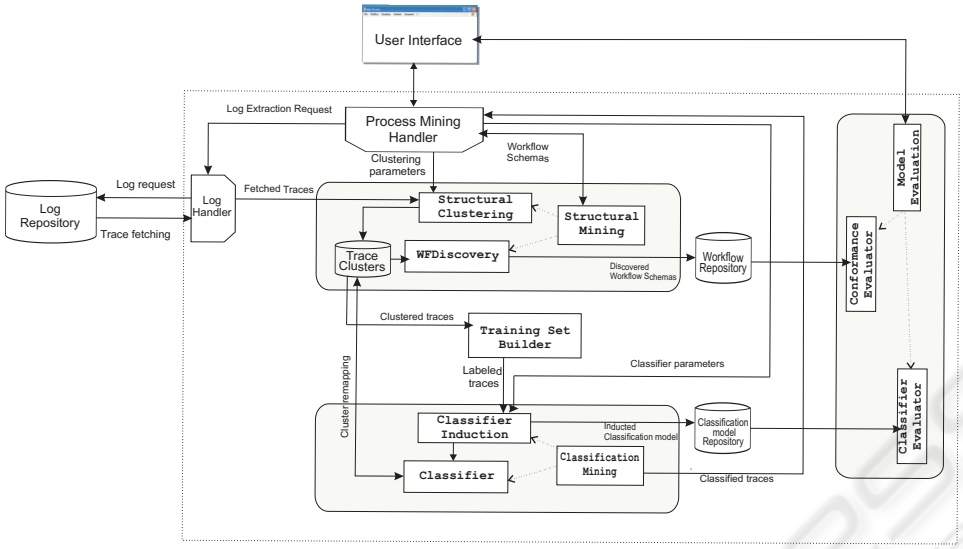



Figure 2: System architecture.

to reassign the original log traces to the clusters. Both these latter modules, coordinated by the module *Classification Mining*, have been implemented by using the Weka library (Frank et al., 2005) and, in particular, the J48 implementation of the well known algorithm C4.5 (Quinlan, 1993). In this regard, we notice that the *Training Set Builder* also acts as a "translator" by encoding the labelled log traces into a tabular form, according to the ARFF format used in Weka. Incidentally, this format requires that proper types (such as nominal, numeric, string) are specified for all attributes. Since this information misses in the original (MXML) log files, a semi-automatic procedure is used to assign such a type to each attribute.

Two modules help the user in evaluating the quality of the discovered models: the *Classifier Evaluator*, which computes standard performance indexes for the classification models, and the *Conformance Evaluator* module which measures what an extent workflow models represents the behaviors actually registered in the log traces they have been derived from. To this end, some metrics have been implemented, which will be briefly described in Section 3.2.

Figure 3 shows two screenshots of the plug-in. Specifically, Figure 3(a) reports the input panel, which allows for setting all the parameters needed in our approach, whereas Figure 3(b) illustrates the outcome of a mining session. For the latter, note, in particular, the decision tree (on the left) and one of the workflow schemas (on the right) found by the tool.

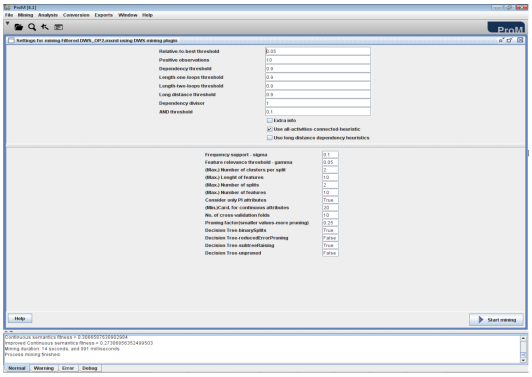
# 3 CASE STUDY

This section is devoted to discuss the application of the proposed approach within a real-life scenario, concerning an Italian maritime container terminal, as a way to give some evidence for the validity of our proposal, as well as to better explain its behavior through a practical example. In Section 3.1, we first illustrate the application scenario, by discussing the kind of data involved in it. Then, in Section 3.2, we introduce the setting adopted for evaluating the quality of discovered models, while in Section 3.3, we eventually discuss results of experiments carried out on the scenario.

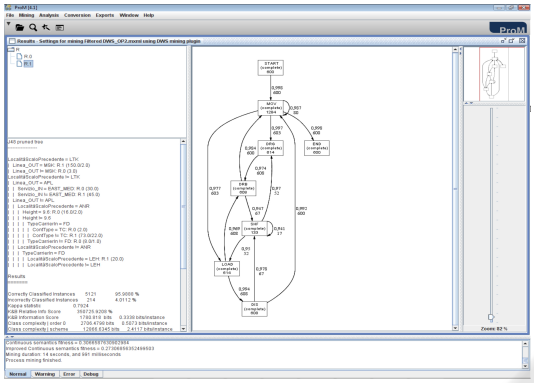
## 3.1 Application Scenario

A series of logistic activities are registered for each of the containers that pass through the harbor, which actually amount to nearly 4 millions per year. Massive volumes of data are hence generated continually, which can profitably be exploited to analyze and improve the enactment of logistics processes. For the sake of simplicity, in the remainder of the paper we only consider the handling of containers which both arrive and depart by sea, and focus on the different kinds of moves they undergo over the "yard", i.e., the main area used in the harbor for storage purposes. This area is logically partitioned into a finite number of bi-dimensional *slots*, which are the units of storage space used for containers, and are organized in a fixed number of *sectors*.

The life cycle of any container is roughly sum-



(a) Parameter setting



(b) Result view

Figure 3: Screenshots of the plug-in within ProM.

marized as follows. The container is unloaded from the ship and temporarily placed near to the dock, until it is carried to some suitable yard slot for being stocked. Symmetrically, at boarding time, the container is first placed in a yard area close to the dock, and then loaded on the cargo. Different kinds of vehicles can be used for moving a container, including, e.g., cranes, straddle-carriers (a vehicle capable of picking and carrying a container, by possibly lifting it up), and multi-trailers (a sort of train-like vehicle that can transport many containers).

This basic life cycle may be extended with additional transfers—classified as "house-keeping"—which are meant to make the container approach its final embark point or to leave room for other containers. More precisely, the following basic operations are registered for any container $c$:

- `MOV`, when $c$ is moved from a yard position to another by a straddle carrier;
- `DRB`, when $c$ is moved from a yard position to another by a multi-trailer;
- `DRG`, when a multi-trailer moves to get $c$;
- `LOAD`, when $c$ is charged on a multi-trailer;
- `DIS`, when $c$ is discharged off a multi-trailer;
- `SHF`, when $c$ is moved upward or downward, possibly to switch its position with another container.
- `OUT`, when a dock crane embarks $c$ on a ship.

Critical performance aspects in this scenario are the latency time elapsed when serving a ship (where, typically, a number of containers are both discharged off and charged on), and the overall costs of moving the containers around the yard. Both these measures are heavily affected by the number of extra, "housekeeping", moves applied to the containers. Reducing such operations is a major goal of the allocation policies, established to decide where to place a newly arrived container, based on different features of it, such as, e.g., the kind of conveyed goods, its origin and next destination, as well as some future events (e.g. which ships are going to take the container and when this is going to happen). Unfortunately, this information can partly be missing or even incorrect, while diverse types of delays or malfunctioning are likely to occur. Therefore, some ex-post analysis of yard operation logs via data mining and process mining technique is expected to provide precious feedback to the planning of logistic processes, and to help in decision making tasks.

## 3.2 Evaluation Setting

In order to provide a quantitative evaluation of the multi-perspective process models mined with our approach, two aspects have been measured in the experimentation: *(i)* the predictive power of the classification model in discriminating the structural clusters found, and *(ii)* the level of conformance of each workflow schema w.r.t. the clusters that it models.

As for point *(i)*, a number of well-known measures are available in the literature to evaluate a classification model. Here, we simply resort to the *Accuracy* measure, which roughly expresses the percentage of correct predictions that the classifier would make over all possible traces of the process. In particular, we estimate this measure via the popular cross-validation method (with ten folders) (Mitchell, 1997). In addition, standard *Precision* and *Recall* measures will be used to provide a "local" evaluation of the classifier, w.r.t. each single cluster; and, for each cluster, the $\mathcal{F}$ measure will be reported as well, which is defined as $\mathcal{F}_c = ((\beta^2+1)P_c \times R_c)/(P_c + \beta^2 R_c)$—note that for $\beta = 1$, it coincides with the harmonic mean of the precision and recall values.

As for point *(ii)*, the conformance of a workflow model $W$ w.r.t. a set $L$ of traces can be evaluated through the following metrics, all defined in (Rozi-

Table 1: Summary of results obtained against real log data.

| Test | Clusters | Structural model | | | Data-aware classification model | | |
|---|---|---|---|---|---|---|---|
| | | *Fitness* | *BehAppr* | *StrAppr* | *Accuracy* | *Tree Size* | *Top-4 Discriminant Attributes* |
| `Test A` | 2 | 0.8725 | 0.9024 | 1 | 96.01% | 69 | PrevHarbor, NavLine_OUT, ContHeight, ShipType_IN |
| `Test A-bis` | 4 | 0.9324 | 0.9351 | 1 | 94.78% | 77 | PrevHarbor, NavLine_OUT, ContHeight, ShipType_IN |
| `Test B` | 5 | 0.8558 | 0.9140 | 1 | 91.64% | 105 | ShipSize_IN, NavLine_OUT, PrevHarbor, ContHeight |

nat and van der Aalst, 2008) and ranging over the real interval [0,1]:

- *Fitness*, which essentially evaluates the ability of $W$ to parse all the traces $L$, by indicating how much the events in $L$ comply with $W$.
- *Advanced Behavioral Appropriateness* (denoted by *BehAppr*, for short), which estimates the level of flexibility allowed in $W$ (i.e., alternative/parallel behavior) really used to produce $L$.
- *Advanced Structural Appropriateness* (or *StrAppr*, for short), which assesses the capability of $W$ to describe $L$ in a maximally concise way.

These measures have been defined for a workflow schema and do not apply directly to the multi-schema structural model discovered by our approach. In order to show a single overall score for such a model, we simply average the values computed by each of these measures against all of its workflow schemas. More precisely, the conformance values of these schemas are added up in a weighted way, where the weight of each schema is the fraction of original log traces that constitute the cluster it was mined from.

## 3.3 Experimental Results

For all the experiments described next, we selected only a subset of containers that completed their entire life cycle in the hub along the first two months of year 2006, and which were exchanged with four given ports around the Mediterranean sea—this yielded about 50Mb log data concerning 5336 containers. In order to apply our analysis approach, the transit of any container through the hub has been regarded as a single enactment case of a (unknown) logistic process, for which a suitable model is to be discovered.

We next discuss three tests, which were carried out on these data according to different perspectives:

- *Test A* ("operation-centric"), where we focus on the sequence of basic logistic operations applied to the containers. In more detail, for each container a distinct log trace is built that records the sequence of basic operations (i.e., `MOV`, `DRB`, `DRG`, `LOAD`, `DIS`, `SHF`, `OUT`) it underwent.
- *Test A-bis*, where we still consider the sequence of operations performed on each container, while distinguishing different cost-dependent variants for each of them. In fact, each operation is annotated with a suffix denoting the cost spent for performing it.
- *Test B* ("position-centric"), where we focus on the flow of containers across the yard. Specifically, original data are transformed into a set of log trace, each of them encoding the sequence of yard sectors occupied by a single container during its stay.

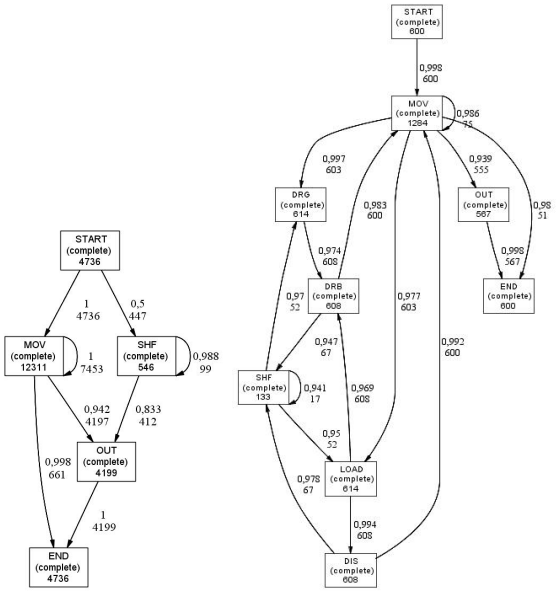
In all cases, two dummy activities (denoted by `START` and `END`, respectively) are used to univocally mark the beginning and the end of each log trace. Further, various data attributes have been considered for each container (i.e., for each process instance), including, e.g., its origin and final destination ports, its previous and next calls, diverse characteristics of the ship that unloaded it, its physical features (e.g., size, weight), and a series of categorical attributes concerning its contents (e.g., the presence of dangerous or perishable goods).

Table 1 summarizes a few relevant figures for these tests, which concern both the clustering structure and the two models associated with it. More specifically, for each test, we report the number of clusters found and the different conformance measures for the structural model (i.e., the set of workflow schemas), as well as the accuracy and size of the decision tree. Moreover, to give some intuition on the semantics value of this latter model, we report four of the most discriminant attributes, actually appearing in its top levels[1]. In general, we note that surprisingly high effectiveness results have been achieved, as concerns both the structural model and the classification model. Notably, such a precision in modelling both structural and non-structural aspects of the logged events does not come with a verbose (and possibly overfitting) representation. Indeed, for all the tests, the number of clusters and the size of the tree are quite restrained, while the workflow models collectively attain a maximal score with the *StrAppr* metric.

In order to give more insight on the behavior of the approach, we next illustrate some detailed results for two of these tests (namely *Test A* and *Test B*).

**Test A.** In this case, the approach has discovered

[1] Decision trees are not shown in detail for both space and privacy reasons—many attribute express, indeed, sensible information about the hub society and its partners.




(a) Cluster 0 (b) Cluster 1

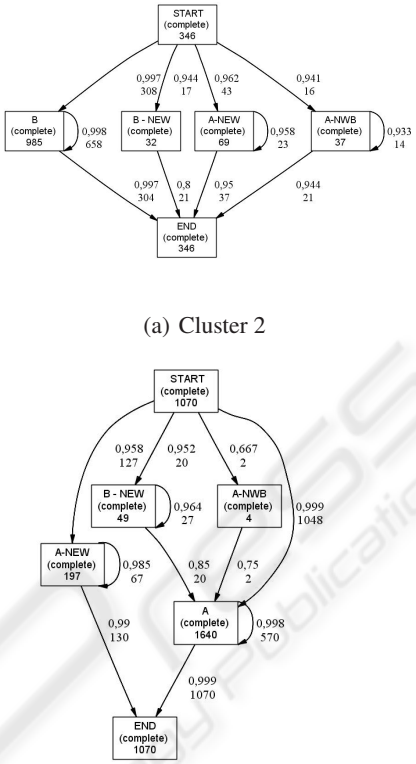
Figure 4: Test A - the two workflow schemas discovered.

Table 2: Test A - details on the discovered clusters (sizes and classification metrics).

| *Cluster* | *Size* | *P* | *R* | $\mathcal{F}(\beta=1)$ |
|---|---|---|---|---|
| 0 | 4736 | 97.28% | 98.25% | 97.76% |
| 1 | 600 | 84.99% | 78.33% | 81.53% |

two distinct usage scenarios, whose structural aspects are described by the workflow schemas in Figure 4. These schemas substantially differ for the presence of operations performed with multi-trailer vehicles: the schema of Figure 4.(a) does not feature any of these operations, which are instead contained in the other schema. Notably, the former schema captures the vast majority of handling cases (4736 containers of the original 5336 ones). This reflects a major aim of yard allocation strategies: to keep each container as near as possible to its positions of disembarkation/embark, by performing short transfers via straddle-carriers.

Interestingly enough, these two markedly different structural models appear to strongly depend—an astonishing 96% accuracy score is achieved (cf. Table 1)—on some features that go beyond the mere occurrence of yard operations. Among these features, the following container attributes stand out: the provenance port (`PrevHarbor` ) of a container, the navigation line that is going to take it away (`NavLine _OUT`), the height of a container (`ContHeight` ), and the kind of ship that delivered it to the hub (`ShipType _IN`). A finer grain analysis performed with the help of Table 2 (where individual pre-




(a) Cluster 2

(b) Cluster 3

Figure 5: Test B - two of the workflow schemas found.

Table 3: Test B - details on the discovered clusters (sizes and classification metrics).

| *Cluster* | *Size* | *P* | *R* | $\mathcal{F}(\beta=1)$ |
|---|---|---|---|---|
| 0 | 3664 | 93,76% | 98,39% | 96,02% |
| 1 | 188 | 66,67% | 53,19% | 59,17% |
| 2 | 346 | 90,81% | 74,28% | 81,72% |
| 3 | 1070 | 87,25% | 80,56% | 83,77% |
| 4 | 68 | 94,29% | 97,06% | 95,65% |

cision/recall measures for the two clusters are shown), confirms that the model guarantees a high rate of correct predictions for either cluster.

**Test B.** The "position-centric" approach is addressed to analyze the different ways of displacing the containers around the yard, yet comparing the usage of different storage areas. In principle, due to the high number of sectors and moving patterns that come to play in such analysis perspective, any flat representation of container flows, just consisting of a single workflow schema, risks being either inaccurate or difficult to interpret. Conversely, by separating different behavioral classes our approach ensures a modular representation, which can better support explorative analyses. In fact, the five clusters found in this test have been equipped with clear and compact workflow schemas, which yet guarantee a high level of confor-

mance (see Table 1). As an instance, in Figure 5, we report two of these schemas, which differ both in the usage of sectors and in some of the paths followed by the containers across these sectors.

Good quality results are achieved again both for the structural model and for the decision tree. In actual fact, by comparing these results with those obtained in the other two tests, we notice some slight decrease in the accuracy and a larger tree size, mainly due to the higher level of complexity that distinguish the position-centric analysis from the operation-centric one. Incidentally, Table 3 reveals that such worsening is mainly to blame on the inability of the decision tree to recognize well the second cluster, which is, in fact, slightly confused with the third one —further details are omitted here for lack of space. Almost the same attributes as in *Test A* have been employed to discriminate the clusters, except for the usage of `ShipSize _IN` (i.e., the size category of the ship that delivered the container) in place of `ShipType _IN`.

# 4 CONCLUSIONS

In this paper we have proposed a novel process mining approach which allows to discover multi-perspective process models, i.e., process models where structural and non-structural aspects are formally related with each other. In a nutshell, the approach recognizes a number of homogeneous execution clusters (by way of a clustering method), while providing each of them with a specific workflow model. These classes of behavior are then correlated with non-structural data (such as activity executors, parameter values and performance metrics) by means of a classification model, which can be used to both explain and predict them.

The whole approach has been implemented, integrated as a plug-in in the ProM framework, and validated on a real test case. Results of experimentation evidenced that the discovered classification models provide a valuable help for interpreting and discriminating different ways of executing the process, because of their ability of making explicit the link between these variants and other process properties.

As future work, we will investigate the extension of the proposed approach with outlier detection techniques, in order to provide it with the capability of spotting anomalous executions, which may mislead the learning of process models, and could profitably be analyzed in a separate way. Moreover, we are exploring the integration of multi-perspective models in an existing process management platform, as a basic means for providing prediction and simulation features, supporting both the design of new processes and the enactment of future cases.

# REFERENCES


Buntine, W. (1992). Learning classification trees. *Statistics and Computation*, 2:63–73.

Frank, E., Hall, M. A., Holmes, G., Kirkby, R., and Pfahringer, B. (2005). Weka - a machine learning workbench for data mining. In *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314.

Greco, G., Guzzo, A., Pontieri, L., and Saccà, D. (2006). Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1010–1027.

Ly, L. T., Rinderle, S., Dadam, P., and Reichert, M. (2005). Mining staff assignment rules from event-based data. In *Business Process Management Workshops*, pages 177–190.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rozinat, A. and van der Aalst, W. M. P. (2006). Decision mining in ProM. In *Proc. of 4th Intl. Conf. on Business Process Management (BPM'06)*, pages 420–425.

Rozinat, A. and van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95.

van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. J. M. M. (2003). Workflow mining: A survey of issues and approaches. *Data Knowledge Engineering*, 47(2):237–267.

van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., and van der Aalst, W. M. P. (2005). The ProM framework: A new era in process mining tool support. In *Proc. of 26th International Conference on Applications and Theory of Petri Nets (ICATPN '05)*, pages 444–454.

van Dongen, B. F. and van der Aalst, W. M. P. (2005). A meta model for process mining data. In *Proc. of EMOI-INTEROP*, pages 309–320.

Weijters, A. J. M. M. and van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162.