A Comprehensive Benchmarking Framework (CoBeFra) for Conformance Analysis between Procedural Process Models and Event Logs in ProM

Seppe K.L.M. vanden Broucke*, Jochen De Weerdt†, Jan Vanthienen*, and Bart Baesens*‡
*Department of Decision Sciences and Information Management, KU Leuven
Naamsestraat 69, B-3000 Leuven
Email: seppe.vandenbroucke@kuleuven.be
†Business Process Management Group, Queensland University of Technology
GPO Box 2434, Brisbane QLD 4001, Australia
‡School of Management, University of Southampton
Highfield Southampton, SO17 1BJ, United Kingdom

***Abstract*—Process mining encompasses the research area which is concerned with knowledge discovery from information system event logs. Within the process mining research area, two prominent tasks can be discerned. First of all, process discovery deals with the automatic construction of a process model out of an event log. Secondly, conformance checking focuses on the assessment of the quality of a discovered or designed process model in respect to the actual behavior as captured in event logs. Hereto, multiple techniques and metrics have been developed and described in the literature. However, the process mining domain still lacks a comprehensive framework for assessing the goodness of a process model from a quantitative perspective. In this study, we describe the architecture of an extensible framework within ProM, allowing for the consistent, comparative and repeatable calculation of conformance metrics. For the development and assessment of both process discovery as well as conformance techniques, such a framework is considered greatly valuable.**

I. Introduction

Process mining [1] should be situated in the diagnosis phase of the business process management life cycle, as it bridges the BPM-domain with the domain of Knowledge Discovery in Databases (KDD) [2]. Van der Aalst [3] identifies three main process mining types of tasks: discovery, conformance and extension. Regarding conformance, the need for a comprehensive evaluation framework in the process mining domain was first articulated by Rozinat et al. [4]. To date, the process mining domain still lacks a comprehensive, useable toolset for assessing the goodness of a process model or to easily benchmark the performance of different models against each other using multiple conformance checking metrics. In this paper, we present the architecture of an extensible comprehensive benchmarking framework (abbreviated to *CoBeFra*) integrated within ProM, allowing for the consistent, comparative and repeatable calculation of process conformance metrics. Such an architecture should be considered highly valuable for process mining researchers because it significantly facilitates the development and assessment of process discovery as well as conformance checking techniques. This paper is structured as follows. Section II provides an overview of existing conformance checking metrics. Section III discusses the various requirements which were identified as being important in the construction of a conformance checking framework. Next, Section IV discusses the technical architecture of the developed framework, putting an emphasis on modularity and extendability. Finally, Section V provides a list of current limitations and possibilities for future work and further improvement before closing with some general conclusions. Installation instructions, source files and binaries for the CoBeFra framework are available at http://processmining.be/cobefra. The framework is currently focusing on the control-flow dimension, however, its scope might be broadened to accommodate for other information dimensions.

II. Process Models Quality Metrics

The quality of a procedural process model (designed or discovered) can be judged along different perspectives. Figure 1 details that these perspectives can be categorized into two high-level dimensions: accuracy and comprehensibility. Each of these high-level quality dimensions can be further decomposed. For instance, the accuracy of a process model consists of the following three subdimensions: its recall (or: fitness), indicating the ability of the process model to replay or execute the observed behavior; its precision (or: appropriateness), denoting the model's ability to disallow (i.e. not support the execution of) unwanted, unseen behavior and thus its ability towards preventing underfitting the observed behavior; and last, the model's generalization capability, which indicates the model's ability to allow unseen but nevertheless desired or expected behavior and which can thus be seen as the counterpart of precision (i.e. avoiding overfitting). Next, from the viewpoint of comprehensibility, a distinction can be made between simplicity (or: complexity) and structuredness (or: understandability, entropy), the latter representing the ease of interpretation of the model while simplicity refers to the number of control-flow constructs present in the process model. As such, simplicity represents the principle that "all

other things equal, a simpler explanation is better than a more complex one", a statement famously known as "Occam's razor". Oftentimes, simple counts of model elements are used as simplicity metrics.

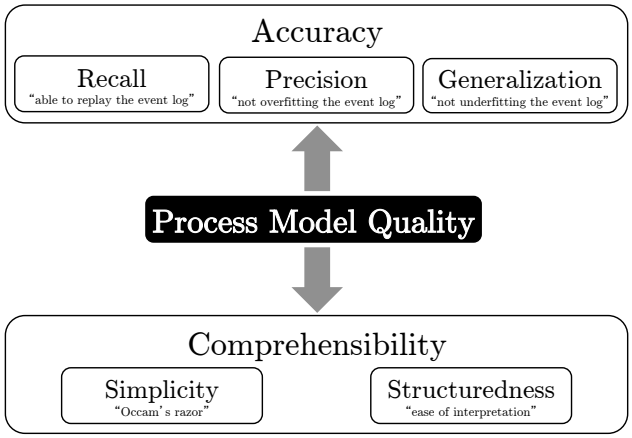

Fig. 1. Quality dimensions for evaluating procedural process models

A. Accuracy Metrics

As indicated above, the overall accuracy of a process model consists of three perspectives: recall, precision and generalization. Cook and Wolf [19] are to be considered the first researchers to quantify the relationship between process models and event logs. Within the process mining domain, Rozinat et al. [8] defined the notions of fitness and appropriateness in a foundational study on conformance checking. Since, the domain has attracted the attention of many other researchers, as demonstrated by Table I, which provides an overview of the most noteworthy conformance metrics and an indication of which metrics are currently already implemented in the proposed benchmarking framework. In the remainder of this section, the metrics included in the CoBeFra framework are discussed in further detail. For detailed information about the other conformance metrics such as CPM, PM, $PF_{complete}$, etc., we refer to our earlier work as well as the original manuscripts describing these metrics.

a) Recall: Recall or fitness can be seen as the primordial accuracy evaluation perspective because it reflects how much behavior present in the event log is captured by the model, which can be regarded as an obvious quality requirement for designed or discovered models before continuing onwards with other analysis tasks. As such, many researchers have proposed metrics that quantify this dimension.

- *Fitness* (f) is a metric that is obtained by evaluating whether each trace in the event log can be reproduced by the generative model, in this case a Petri net. This procedure is called sequence replay [8]. During replay, the transitions in the Petri net will produce and consume tokens to reflect the state transitions. Consequently, the fitness measure punishes for tokens that must be created additionally in the marked Petri net and also for tokens that remain after replay.
- *Proper Completion* (P_{PC}) [8] is a coarse-grained metric which returns the percentage of traces without any missing or remaining tokens after trace replay. Or, put differently, the percentage of traces for which a Fitness (f) value of 1 can be obtained.
- *Behavioral Recall* (r_B^p) as defined by Goedertier et al. [9] is the percentage of correctly classified positive events in the event log. Also by using sequence replay techniques, it is verified whether every positive event can be parsed by the model. Note that this metric differs from Fitness (f) since it denotes the ratio of (positive) events which could be replayed successfully by the process model, without incorporating the exact amount of missing or remaining tokens in the metric definition.
- *(Average) Alignment-Based Trace Fitness* (f_a, f_a^{avg}) is the most recent recall metric in the process mining domain [13]. In contrast to a large majority of recall metrics, this metric is based on aligning a process model and an event log [12], [20], [21], instead of replaying the log in the model. As such, the metric punishes for alignments which require model (log) moves without a corresponding move in the log (model). Other variants base the resulting value on the raw cost given to various kinds of misalignments rather than returning the percentages of non-aligned log-model moves.
- Weidlich et al. [17] have proposed a collection of *Behavioral Profile*-based conformance metrics. Instead of relying on state-based techniques that involve replaying the logs, the authors base their metrics on different types of constraints that a process model can impose for a pair of activities (i.e. the behavioral profile of the process model). We have included the Behavioral Profile Conformance metrics in the CoBeFra framework, both available in the form of the "standard" implementation provided by Weidlich et al. (using the jBPT library) and as an alternative implementation developed by the authors which is able to deal with event logs containing duplicate activities.

b) Precision: Precision entails that process models should prevent the execution of unseen and unwanted behavior (i.e. not underfitting the data). The following metrics are included in CoBeFra.

- *Advanced Behavioral Appropriateness* ($a_B^{'}$) as defined in [8] is a footprint-based, rather coarse-grained precision metric, which in addition requires an exhaustive time consuming state space exploration.
- *Behavioral Specificity* (s_B^n) is the percentage of correctly classified negative events during sequence replay. As such, it is the counterpart of Behavioral Recall (r_B^p). Artificial negative events can be generated with the technique developed by Goedertier et al. [9]. However, the definition of the metric does not exclude the use of natural negative events or negative events stemming from other techniques. Note that, for many process model evaluation tasks, it is not the percentage of correctly

Name	Symbol	Author	Included in CoBeFra	Range	Model input type	Perspective			
						Recall	Precision	Generalization	Comprehensibility
Continuous Parsing Measure	CPM	Weijters et al. [5]		[0,1]	Heuristic net	✓			
Parsing Measure	PM	Weijters et al. [5]		[0,1]	Heuristic net	✓			
Partial Fitness - complete (or: Improved Continuous Semantics)	$PF_{complete}$ (ICS)	Alves de Medeiros et al. [6]		[-∞,1]	Heuristic net	✓			
Completeness		Greco et al. [7]		[0,1]	Workflow schema	✓			
Soundness		Greco et al. [7]		[0,1]	Workflow schema		✓		
Fitness	f	Rozinat and van der Aalst [8]	✓	[0,1]	Petri net	✓			
Proper Completion	p_{PC}	Rozinat and van der Aalst [8]	✓	[0,1]	Petri net	✓			
Behavioral Appropriateness	a_B	Rozinat and van der Aalst [8]		[0,1]	Petri net		✓		
Advanced Behavioral Appropriateness	a'_B	Rozinat and van der Aalst [8]	✓	[0,1]	Petri net		✓		
Structural Appropriateness	a_S	Rozinat and van der Aalst [8]		[0,1]	Petri net				✓
Advanced Structural Appropriateness	a'_S	Rozinat and van der Aalst [8]	✓	[0,1]	Petri net				✓
Behavioral Recall	r_B^p	Goedertier et al. [9]	✓	[0,1]	Petri net	✓			
Behavioral Specificity	s_B^n	Goedertier et al. [9]	✓	[0,1]	Petri net		✓		
Behavioral Precision	p_B	De Weerdt et al. [10]	✓	[0,1]	Petri net		✓		
Weighted Behavioral Precision	p_B^w	vanden Broucke et al. [11]	✓	[0,1]	Petri net		✓		
Behavioral Generalization	g_B	vanden Broucke et al. [11]	✓	[0,1]	Petri net			✓	
Weighted Behavioral Generalization	g_B^w	vanden Broucke et al. [11]	✓	[0,1]	Petri net			✓	
(Average) Alignment-Based Trace Fitness	f_a, f_a^{avg}	van der Aalst et al. [12], [13]	✓	[0,1]	Petri net	✓			
ETC Precision	etc_P	Muñoz-Gama et al. [14]	✓	[0,1]	Petri net		✓		
One Align Precision	a_p^1	Adriansyah et al. [15]	✓	[0,1]	Petri net		✓		
Best Align Precision	a_p	Adriansyah et al. [15]	✓	[0,1]	Petri net		✓		
Alignment Based Precision	$precision(L, M)$	Adriansyah et al. [13]	✓	[0,1]	Petri net		✓		
Alignment Based Probabilistic Generalization	$generalization(L, M)$ and $generalization_s(L, M)$	Adriansyah et al. [13]	✓	[0,1]	Petri net			✓	
Probabilistic Generalization		Buijs et al. [16]		[0,1]	Process tree			✓	
Behavioral Profile Conformance metrics	MCC, CCC, etc.	Weidlich et al. [17]	✓	[0,1]	Petri net	✓			
Various simplicity metrics	$\|nodes\|$, $\frac{\|arcs\|}{\|nodes\|}$, etc.	Mendling et al. [18]	✓	[0,1]	Petri net				✓

TABLE I
OVERVIEW OF PROCESS MINING CONFORMANCE METRICS

classified negative events which is of real interest to the end user (the specificity), but rather the amount of "false positives", i.e. behavior which is allowed by the process model although this behavior is rejected in the given event log in the form of a negative event. This notion better corresponds with the true meaning of precision and is captured in the *Behavioral Precision* metric (p_B) as defined by De Weerdt et al. [10]. The benchmarking framework also includes a weighted variant (p_B^w) of this metric (*Weighted Behavioral Precision*), based on an artificial negative event induction technique described in [11].

- *ETC Precision* (etc_P) [14], [22] is based on the construction of a prefix automaton for the event log at hand. By taking into account the number of so-called escaping edges while mapping the behavior in the event log with the behavior in the model, a state-of-the-art precision metric is defined by comparing the amount of "escaping" behavior the models allows in a given state compared to the log prefix automaton which is brought (using replay) in a comparable state. A very similar metric (*Alignment Based Precision*) based on the concept of log-model alignments is described in [13].
- *One Align Precision* (a_p^1) and *Best Align Precision* (a_p) extend the etc_P metric by first aligning the log and the model [15]. In this way, the main problem of etc_P, i.e. the fact that precision is assessed as long as the trace under investigation can be replayed without error, is solved.

c) Generalization: Although models should be precise, generalizing beyond observed behavior is also a necessity. This is because assuming that all behavior is included in an event log is a far too strong completeness assumption. Metrics quantifying the generalization dimension should punish process models which are overly precise, thus not allowing unseen but very likely (not explicitly forbidden) behavior when taking into account the data in the event log.

- *Behavioral Generalization* (g_B) and *Weighted Behavioral*

Generalization (g_B^w): based on the work published in [10] and [11], it is possible to create a mirrored counterpart for the Behavioral Precision metric, where not only the model's classification of given positive and negative events is assessed, but where it is also investigated whether the model is able to correctly execute other possible behavior which is not explicitly forbidden by the presence of a negative event at the current state. We have added these metrics to the benchmarking framework in order to evaluate generalization.

- The *Alignment Based Probabilistic Generalization* ($generalization(L, M)$ and $generalization_S(L, M)$) metric also starts from the principle of log-model alignment as described by Adriansyah et al. [13]. The authors propose a probabilistic (Bayesian) estimator which tries to assess the chance whether events will occur which exhibit behavior that was not seen before.

B. Comprehensibility Metrics

Because structuredness (ease of interpretation) is a difficult dimension to measure, many researchers focus on simplicity for quantifying the comprehensibility of process models. In [18], around 20 different metrics are defined to assess a model's compressibility. We have opted to include just a few count-based simplicity metrics by default in the CoBeFra framework: the number of arcs, nodes, places, transitions or cut vertices; the average node arc degree and the weighted place/transition node arc degree. Next to these metrics, the Advanced Structural Appropriateness ($a_S^{'}$) [8] is also included. This metric evaluates two specific design guidelines for expressing behavioral patterns, namely the occurrence of alternative duplicate tasks and redundant invisible tasks.

C. Combining Metrics

Another aspect that is becoming increasingly important in the conformance checking domain is the question on how to combine evaluation dimensions. In earlier work, we have proposed the use of the F-score to combine recall and precision metrics [10]. In [23], Buijs et al. propose to weight four evaluation dimensions, i.e. recall, precision, generalization and comprehensibility, in one metric for steering their genetic programming inspired Evolutionary Tree Miner (ETM) algorithm. As such, it was decided to add the F-score technique as well as a so-called "free weigher" to CoBeFra. The F-score allows to combine a fitness and precision metric with a configurable β value (F_1 being the harmonic mean of fitness and precision), whereas the free weigher allows to configure a linear combination of various metrics with configurable coefficients (weights).

III. Requirements for the Architecture of a Benchmarking Framework

The following principles summarize the basic design requirements that were considered when developing the benchmarking framework in ProM.

a) Ease of Use: The first design requirement puts an emphasis on user friendliness. With this benchmarking framework, we aim to offer a straightforward interface for importing event logs and process models, for mapping each event's class (i.e. its activity name and life cycle transition) to one of the activities (tasks, transitions) in the process model, for configuring the various metrics and, finally, for inspecting and exporting the obtained results. Furthermore, although the ProM framework allows for a clear differentiation between the end-user oriented graphical interface and internal logic, we have found that many plugins still rely on the presence of the user interface (UI), preventing an easy implementation in external (headless) scripts or tools. Therefore, in order to simplify experimental setups requiring a great deal of scripting and batch processing, we have strengthened the decoupling of the user interface and programming logic in CoBeFra, allowing each step (log and model setup, metric configuration and result processing) to be executed in a pure "headless" manner.

b) Reproducibility of Experiments: A second important design requirement consists of providing the functionality to reproduce results. By allowing to store input and metric configurations, it is very straightforward to repeat experiments over time.

c) Comparative Consistency: Consistency is safeguarded by a number of elements. First of all, because the same initially configured model-log mapping is used across all metrics, no obvious mistakes are made to this regard. Furthermore, by streamlining the metric configuration step, users will have less trouble in configuring the often huge amount of configurations across different experiments, which promotes consistency of results.

d) Computation Management: A limitation of the ProM framework is that it is not straightforward to set up an environment in which multiple conformance checks can be executed at the same time. Therefore, our approach is to first allow the user to configure all model-log inputs together with the list of desired metrics to run and their configuration. Afterwards, the calculation of the metrics itself is started; we have implemented a computation manager which allows to run multiple metric calculations in parallel. Section IV provides more technical details. Finally, since some metrics can consume a large amount of time before finishing, an option to both manually and automatically cancel a metric's calculation procedure was added as well. This allows researchers to easily impose time-based bounds while running experiments.

e) Extensibility: Finally, the framework is designed so to be easily extended with other or future conformance metrics. A large number of metrics have already been implemented, but we invite scholars and authors to implement their work in CoBeFra as well. Making a ProM conformance checking plugin available in CoBeFra is quite straightforward, especially if best practises were followed during the development of the original ProM plugin, since CoBeFra also uses and allows to easily tie in with the ProM architecture.

IV. Framework Architecture

This section describes the technical architecture of the framework in more detail. Particular attention is paid to the topics of model-log mapping, separation between domain logic and user interface, legacy support and parallelism.

A. General Architecture

The CoBeFra framework is integrated in ProM 6 and reuses various existing libraries and components available in ProM. Figure 2 provides a schematic overview of the developed architecture. At the root of the architecture is a global application controller which is responsible for managing the general user interface, the flow between the initial input setup, metric configuration and result handling, and also provides an application programming interface (API) to perform all steps programmatically. CoBeFra can be started as any other ProM plugin, and allows to optionally specify a previously saved project to resume or restart an experiment. Importing and exporting of input objects is done through the standard ProM provided architecture as well, and visualization of obtained metric results was decoupled into a separate visualization plugin. Once CoBeFra is started, the framework discards all ProM-specific dependencies so that it can easily be ran as a stand-alone application as well.

B. Particular Items

a) Mapping Process Models with Event Logs: Although it appears straightforward to link process model activities (Petri net transitions) with events in a process model after executing a process discovery algorithm or during model execution, this link or "mapping" is lost once both objects are saved separately, or are modified asynchronously. Mapping a model to a log is thus a crucial step in each conformance checking analysis task, as there needs to be a clear and unambiguously defined relation between process model activity elements and log events. A model-log mapping is established as follows. First, an event class alphabet is constructed from the event log by deriving the class of each event using a given classifier[1]. The most basic event classifier just extracts the activity name from each event. Next, a process model activity alphabet is constructed, which contains all process model elements whose semantical definition corresponds with a task executing or firing element. For Petri nets, this alphabet simply contains each transition. Finally, a mapping function between these two sets is established, so that for a Petri net, each transition is either mapped to an event class (a "visible" transition), is unmapped and denoted as a "silent" or "invisible" transition (meaning that this transition can be executed whenever it is enabled without corresponding with a logged event), or is unmapped and denoted as a "blocked" transition, effectively preventing (or hence, blocking) this transition from being executed at all. Note that multiple transition can be mapped to the same event class. Another requirement is that every event class must be the mapped target of at least one transition in the event log, otherwise, such "alien" events will block further trace execution when they are encountered (possible ways to deal with this include either adding never enabled dummy transitions to the Petri net mapped to the alien event class, or first filtering the corresponding events out of the log). We have found that many conformance checking plugins in ProM provide alternative implementations of the above described mappings; to be exact, six different methods to describe the link between event log activities and Petri net transitions were discovered. Furthermore, many of them do not deal with the possibility of "blocked" transitions, and have different ways to specify "invisible" transitions. Finally, many existing mappers come with UI components that allow users to construct a mapping, but many of them are cumbersome to use, especially when having to deal with large models containing many transitions or event classes, or when having to execute repeated experiments. Since even a single incorrectly mapped transition may lead to largely skewed quality results, the mapping process should try to prevent errors as best as possible. Therefore, we have implemented an unified mapping system to deal with the above issues. A mapping must be performed by the user between each model-event log pair, but the provided user interface allows to configure mappings for multiple models towards the same log file simultaneously. Furthermore, an intelligent string matching routine pre-maps transitions to event classes whenever possible, providing an option to the user to immediately map the non-automatically matched transitions as being invisible or blocked. Finally, autocompletion-enabled UI controls allow users to rapidly assign the remaining transitions to an event class. Our mapping object serves as the basis to save and load model-event log combinations, preventing having to re-enter the mapping between an event log and process model every time an experiment is ran. Finally, our collection of utility classes provides support to convert our mapping to the different other mapping representations used by various metrics in a transparent manner, so that the step of constructing a mapping is completely decoupled from the configuration and execution of the conformance checking metrics.

b) User Interface – Program Logic Separation: A second architectural aspect we wish to emphasize is the great deal of attention given to separating user interface components with the actual domain model and program logic (e.g. calculation of metric values). To do so, the conformance checking metrics themselves have been implemented as classes which fulfill their metric contract by implementing a "Metric" interface. In general, this contract imposes the constraints on an implementing metric that it must be able to return a numeric result value, and should be configurable via a standard means, i.e. by getting and setting key-value attributes. The latter also allows for easy serialization of a metric configuration, which is used to save and load experimental setups. These metric calculating classes are completely UI-agnostic. Next, to provide an easy means to configure each metric, UI components can be defined which take a metric class as an input and allow to modify a

[1]The notion of event classes and event classifier is part of the XES standard, used to store and manage event log data. See: http://www.xes-standard.org/openxes/start

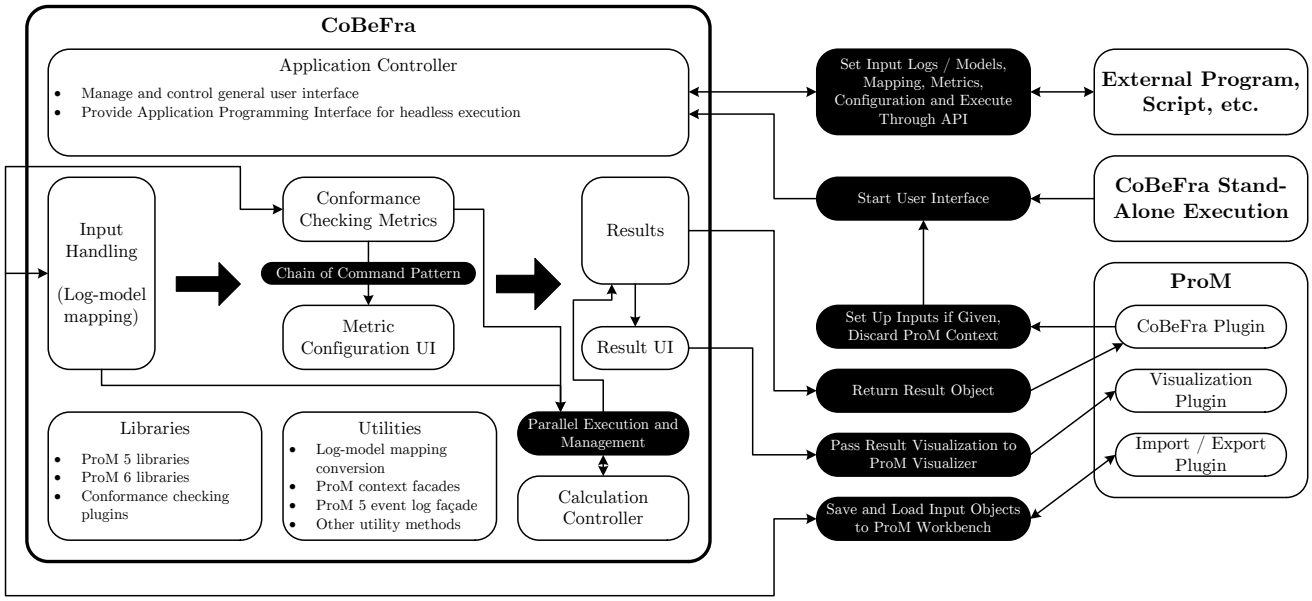

Fig. 2. Schematic overview of the CoBeFra architecture

metric's configuration via a visual interface. The UI components are annotated (using standard Java Annotations) with information to specify which metric classes can be configured through their provided interface. When starting CoBeFra, an internal repository of all UI providing classes is constructed. When a user wishes to configure a specific metric, all matching UI classes are iterated through (using the chain of command programming pattern) and shown. Users accessing CoBeFra using the API can completely bypass the user interface and directly configure the metrics themselves using the key-value store discussed above. Finally, it should be noted that some existing ProM plugins rely heavily on (or assume) the presence of a user interface, for example by hooking logical segments directly into UI actions (such as pressing a button). To cleanly implement these metrics in CoBeFra, some code was copied and refactored to remove the UI-relying segments. Another issue often encountered when inspecting ProM plugins is that authors assume the presence of the ProM-provided "plugin context" object (for example to report progress back to the ProM framework), but are not able to continue when this context is absent or – worse – when this context is not specifically defined as being a UI-enabled (ProM allows to specify a command line interface context, but some plugins specifically check for the UI-based context, even when this is not required). Therefore, we have also defined a utility class which wraps around the ProM context object definition by overloading UI-specific functions, which can be passed directly to a conformance checking metric's methods.

c) Support for ProM 5 Metrics: An important contribution of our conformance checking framework is that it includes some well-known conformance checking metrics which are only implemented in ProM 5, namely Fitness (f), Advanced Behavioral Appropriateness ($a_B^{'}$) and Advanced Structural Appropriateness ($a_S^{'}$). The ProM 5 architecture greatly differs from ProM 6. Not only is the notion of plugin contexts missing (instead, progress listener objects are used), but the internal way event logs are represented is dissimilar as well (ProM 5 does not use the XES standard). Since it would be infeasible to redundantly rewrite large parts of ProM 5 conformance checking metrics (possibly leading to new errors as well), we have defined facade classes wrapped around ProM 5's progress and event log related definitions. Not only does this allow to use ProM 5-only conformance checking metrics, but also to directly load in XES-files and use them in combination with these metrics, which is not possible when using ProM 5 itself (which only accepts the MXML-format). Since the facade objects overload the event log reading methods on the fly, this process is transparent to the end-user.

d) Computation Management and Parallelism: The CoBeFra benchmarking framework allows to run multiple metric calculations in parallel, thus speeding up experiments, especially on multi-core systems. Next to parallelization, the ability to (automatically) cancel a metric's calculation routine is also an important requirement, as this allows users to impose time constraints on the executed experiments. To implement these features, we have opted towards using a multi-process architecture (rather than multi-threaded). This approach entails some useful advantages. Not only does this avoid having to wrap each metric's calculation routine in a threaded worker unit (possibly leading to synchronization issues or other multi-threading related problems), but also prevents that a fatal error in one metric would halt the complete experiment and discard its results. Management of processes is a simple matter. Users

can specify a time limit after which the metric's process is killed. To communicate back and forth from the CoBeFra host process and the metric executing processes, we make use of standard interprocess communication practices by redirecting standard POSIX input, output and error streams. The CoBeFra host process first sends the metric's configuration, event log and model to the process using a compressed stream, after which the host process waits until a result is sent back on the metric's process output stream, or an error is thrown on the metric's process error stream, which is captured and shown to the end user in the final result overview as well. Although currently unimplemented, it can indeed be remarked that this setup theoretically allows to parallelize conformance checking not only on the same host machine, but also across multiple machines over a networked architecture. Exploring the possibilities towards enabling conformance checking to be executed on a computing grid is acknowledged to be an interesting path for further work. Other possibilities towards future work and current limitations are listed in the following section.

This discussion on architecture concludes the presentation of our comprehensive benchmarking framework (CoBeFra). Figure 3 depicts a screen capture of the benchmarking tool. We invite peers and researchers studying conformance checking to contribute and improve upon the framework.

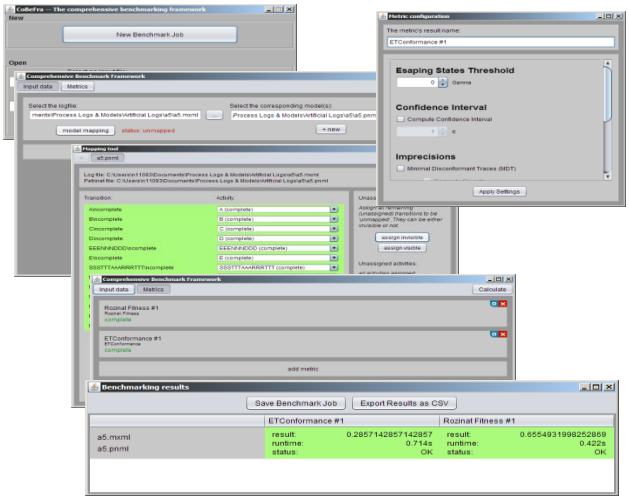

Fig. 3. Screen captures of the CoBeFra user interface

V. Possibilities for Future Work

Although the proposed CoBeFra framework satisfies the desired design requirements and already includes a great deal of conformance checking metrics, different potential improvements exist, summarized in the following overview.

a) Other Process Model Representations: The currently-available implementation of CoBeFra only includes Petri net oriented metrics. However, many other process representations exist for which an extensive conformance checking framework would be beneficial. For instance, de Leoni et al. [24] apply the principle of model-log alignment on declarative process models (Declare models). Accordingly, we plan to investigate how other representations can be incorporated in CoBeFra.

b) Graphical Output: The current version of CoBeFra presents the metric results as a table, exportable as a CSV-file. Graphical outputs such as charts, graphs, pareto-maps, etc. will make the resulting output more user friendly. This will also be explored in future work.

c) Root Cause Analysis: It is important to note that while conformance checking metric values give a good initial indication regarding the quality of a process model (or the level of conformance of an event log), it is also important to know where errors occurred in the process model (or perhaps time frame in the event log). Some conformance checking metrics already allow to do so. Standardizing this feature over multiple metrics is a challenging task which is perhaps out of scope for a benchmarking oriented framework.

d) Automatizing Process Discovery: CoBeFra allows to automatize a great deal of experimental setup, configuration and management tasks when executing a quality assessment study. Nevertheless, the presence of process models and event logs is assumed, which may require that users first run several process discovery algorithms to obtain the set of desired process models to be checked. Automatizing the task of process discovery (perhaps in a separate tool) is a possible path for future consideration, but was not included in the current scope of the project.

e) Fine Tuning Event Classification and Non-Control Flow Conformance Checking: As mentioned in the introductory section, the benchmarking framework proposed in this paper mainly focussed on the control-flow perspective of process models. It is possible to incorporate other perspectives (e.g. data or resource-based views) in the task of conformance checking as well. To do so, the current mapping system has to be modified to allow for more configurability regarding the classification of events. In addition, conformance checking metrics have to be incorporated which are able to check on these other dimensions. We note, however, that much of the conformance checking literature so far has focussed on the control-flow perspective.

f) Standard Validation Event Log Set: Especially for researchers developing a new process discovery algorithm, or conformance checking technique, it is necessary to compare ones own work with the efforts of other peers. The question then becomes which input data set(s) (event logs) should be used to do so. Ideally, a "standard" data set should be constructed, including logs of different sizes (trace variant count, trace instances count, activity type count, etc.) and of different known quality level (e.g. flower model). Additionally, a standard data set should also include some robustness checks, for example by including very noisy (purely random) event logs. Constructing such an ideal event log set is an interesting challenge which would be a fitting complementary item for the proposed benchmarking framework.

g) Fine Tuning Computation Management: Although the current version of CoBeFra allows to parallelize the calcu-

lation of conformance checking metrics, it should be noted that executing too many tasks at once may bias run time results of the metric. The current recommendation is thus to keep the number of parallel executions below the amount of available processing cores (the host operating system will then attempt to distribute each metric as best as possible over the possible cores), but it is also possible to further fine tune the computational management by strictly binding a process to a specific processing core and only allowing one metric execution per core. Furthermore, we have indicated that an interesting possibility for future work is to leverage the multi-process execution architecture so that conformance checking metrics can be executed on a computing grid over a network. Finally, while the current computation manager allows to set a time bound after which a metric is automatically cancelled, a second option could be added which sets a bound on the amount of memory which may be consumed by a conformance checking metric.

h) Cross-Validation: Cross-validation-based evaluation is not frequently applied in the area of process mining, since process mining tasks are typically applied for descriptive, rather than predictive analysis purposes. Furthermore, sensibly splitting a given event log is not easy. Still, it should be possible to implement cross-validation (or train/test splitting, jackknifing) methods in a conformance checking setting by making guided decisions (instead of choosing randomly) when splitting the event log.

VI. Conclusion

This study proposes a new comprehensive benchmarking framework in ProM, called CoBeFra. The current release of the framework already includes a great amount of well-known conformance checking metrics and focuses on the consistent, comparative and repeatable calculation of conformance metrics. For the development and assessment of both process discovery as well as conformance techniques, we conclude that the proposed CoBeFra framework offers a valid contribution for both practitioners and scholars in the field.

Acknowledgment

The authors would thank the Flemish Research Council for financial support under Odysseus grant B.0915.09 and KU Leuven for financial support under grant OT/10/010. This work is also supported by an ARC Discovery grant number: DP120101624. The authors also express their gratitude towards Niels Lambrigts for the help provided in designing and implementing the CoBeFra tool.

References

[1] W. M. P. van der Aalst, *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[2] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Knowledge discovery and data mining: Towards a unifying framework," in *KDD*, 1996, pp. 82–88.

[3] W. M. P. van der Aalst and A. J. M. M. Weijters, "Process mining: a research agenda," *Computers in Industry*, vol. 53, no. 3, pp. 231–244, 2004.

[4] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The need for a process mining evaluation framework in research and practice," in *Business Process Management Workshops*, 2007, pp. 84–89.

[5] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. Alves de Medeiros, "Process mining with the heuristicsminer algorithm," TU Eindhoven, BETA Working Paper Series 166, 2006.

[6] A. K. Alves de Medeiros, "Genetic process mining," Ph.D. dissertation, TU Eindhoven, 2006.

[7] G. Greco, A. Guzzo, L. Pontieri, and D. Saccà, "Discovering expressive process models by clustering log traces," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1010–1027, 2006.

[8] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[9] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens, "Robust process discovery with artificial negative events," *Journal of Machine Learning Research*, vol. 10, pp. 1305–1340, 2009.

[10] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A robust f-measure for evaluating discovered process models," in *CIDM*, 2011, pp. 148–155.

[11] S. vanden Broucke, J. De Weerdt, B. Baesens, and J. Vanthienen, "Improved artificial negative event generation to enhance process event logs," in *CAiSE*, ser. Lecture Notes in Computer Science, J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza, Eds., vol. 7328. Springer, 2012, pp. 254–269.

[12] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *EDOC*. IEEE Computer Society, 2011, pp. 55–64.

[13] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[14] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *Business Process Management*, ser. Lecture Notes in Computer Science, R. Hull, J. Mendling, and S. Tai, Eds., vol. 6336. Springer, 2010, pp. 211–226.

[15] A. Adriansyah, J. Muñoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Alignment based precision checking," BPMcenter.org, BPMcenter.org, BPM Center Report BPM-12-10, 2012.

[16] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "A genetic algorithm for discovering process trees," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, june 2012, pp. 1–8.

[17] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske, "Process compliance analysis based on behavioural profiles," *Inf. Syst.*, vol. 36, no. 7, pp. 1009–1025, 2011.

[18] J. Mendling, G. Neumann, and W. M. P. van der Aalst, "Understanding the occurrence of errors in process models based on metrics," in *OTM Conferences (1)*, 2007, pp. 113–130.

[19] J. E. Cook and A. L. Wolf, "Software process validation: Quantitatively measuring the correspondence of a process to a model," *ACM Trans. Softw. Eng. Methodol.*, vol. 8, no. 2, pp. 147–176, 1999.

[20] A. Adriansyah, N. Sidorova, and B. F. van Dongen, "Cost-based fitness in conformance checking," in *ACSD*, B. Caillaud, J. Carmona, and K. Hiraishi, Eds. IEEE, 2011, pp. 57–66.

[21] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Towards robust conformance checking," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, M. zur Muehlen and J. Su, Eds., vol. 66. Springer, 2010, pp. 122–133.

[22] J. Muñoz-Gama and J. Carmona, "Enhancing precision in process conformance: Stability, confidence and severity," in *CIDM*, 2011, pp. 184–191.

[23] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery (to appear)," in *20th International Conference on Cooperative Information Systems (CoopIS 2012)*, ser. lncs, 2012.

[24] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking," in *BPM*, ser. Lecture Notes in Computer Science, A. P. Barros, A. Gal, and E. Kindler, Eds., vol. 7481. Springer, 2012, pp. 82–97.