ProMetheuS: A Suite for Process Mining Applications

Lucantonio Ghionna[1], Luigi Granata[2], Gianluigi Greco[1], and Massimo Guarascio[3]

[1]Dipartimento di Matematica, Università della Calabria, I-87036 Rende, ITALY
{l.ghionna,ggreco}@mat.unical.it
[2]Exeura SRL, Via Pedro Alvares Cabral - C.da Lecco, I-87036, Rende, ITALY
{luigi.granata@exeura.com}
[3]ICAR-CNR, Via P.Bucci 41C, I-87036, Rende, ITALY
{guarascio@icar.cnr.it}

Abstract. Process mining is an established approach for analyzing and modeling complex business processes. In this paper we showcase ProMetheuS, a flexible and scalable suite for process mining natively designed for industrial applications. Moving from the experience of the *ProM* framework, the state-of-art process mining tool, ProMetheuS introduces three innovative designing elements. Firstly, ProMetheuS defines the concept of flow of mining, which is aimed at supporting the design of complex mining applications, where various mining tasks can be combined and automatically orchestrated at run-time. Secondly, ProMetheuS exports a rich set of facilities to help developers in building interactive applications providing on-the-fly feedback during analysis. Finally, behind the scenes, a powerful stream-based log-handling subsystem ensures scalability in data-intensive applications.

1 Introduction

In the context of enterprise automation, *process mining* is an established approach to support the analysis and the design of complex business processes [1]. In a typical process mining scenario, the goal is to derive a model for a transactional process capable of explaining all activities registered in some log given at hand. Eventually, the “mined” model can be used to design a detailed process schema possibly supporting forthcoming enactments, or to describe its actual behavior.

The *ProM* framework [2] is an open and extendable tool for process mining, which enables users to write and import their own mining algorithms as plug-ins. *ProM* currently supports a wide range of process mining applications (e.g., control-flow mining, decision tree induction, or clustering, to cite a few) and analysis tasks (e.g., validation of process models, performance analysis, or statistical evaluations). Thanks to this valuable packaging, *ProM* represents the state-of-art tool for process mining, and many real-world scenarios exploiting its mining capabilities have been discussed in literature (see e.g., [3]). Despite its success, however, certain issues of flexibility and scalability might arise with the use of the framework, which limit its effectiveness in handling complex industrial applications [2].

In this paper, we describe ProMetheuS[1], a novel suite for process mining introducing innovative designing elements, which are aimed at facing some limitations of *ProM* and at providing new insights on the development of process analysis software.

First, ProMetheuS has been specifically conceived to support *the definition of complex mining applications, where various mining tasks can be combined and automatically orchestrated at run-time*: Process mining applications may involve dozens of different tasks, ranging from data acquisition, to data manipulation, information extraction based on different mining algorithms, recombination of mining results, and visualization. These different kinds of task can be managed in *ProM*, but at the price of requiring human intervention in their coordination. Indeed, constructing complex mining applications requires manually invoking the various tasks by collecting and storing each intermediate result and by reusing them as the input for some further tasks. *ProM* 6.0 has simplified the chaining of intermediate results by letting tasks be aware of the kinds of inputs/outputs they are supporting [2]. In order to automatize and easily deploy mining applications involving different tasks, ProMetheuS introduces instead the concept of "flow of mining", a very natural and manageable way of designing complex mining processes. Indeed, ProMetheuS supports the deployment of mining applications in their entity, by allowing to design mining processes as complex flows of elementary bricks. Each brick produces an output that may be used as input for other bricks in the flow. Consequently, users may incrementally build the desired flow, by connecting existing blocks or adding new ones to manipulate produced outputs. In fact, ProMetheuS comes equipped with a run-time engine that supports and monitors the execution of the mining flow and that orchestrates the compositions of the various elementary bricks. Notably, ProMetheuS allows the definition and the organization of bricks in *workspaces*, grouping resources according their application domain (e.g., text mining, rules learning, etc.). Resources belonging to different workspaces can be transparently connected together to build mixed flows.

Second, ProMetheuS allows users to *build interactive applications providing on-the-fly feedback during analysis*: A plug-in based architecture is a crucial factor to provide flexibility for real-world applications. However, each plug-in is current viewed in the *ProM* framework as a monolithic box, where interaction is limited to the startup phase in which users configure the execution environment of each algorithm by setting all parameters. ProMetheuS extends the flexibility of each plug-in by introducing an "interactive execution" mode (in addition to the standard "batch" one), i.e., it supports an approach to process mining where users may continually interact with the mining algorithms and provide feedbacks trough the graphical user interface.

Finally, ProMetheuS *ensures scalability over large volumes of data*: In real industrial environments, enormous volumes of data are available for mining analysis. Yet, few efforts (see e.g., [4]), have been spent to provide an adequate support for data-intensive applications. *ProM* imports the whole log into the main memory or, if this is not feasible, loads only a batch of data per time and stores remaining batches in disk-

[1]The system has been released by Exeura S.r.l.---under the name of "OKT Process Mining Suite"---as an open source software for the OpenKnowTech project founded by the Italian Ministry of University and Research (MIUR).

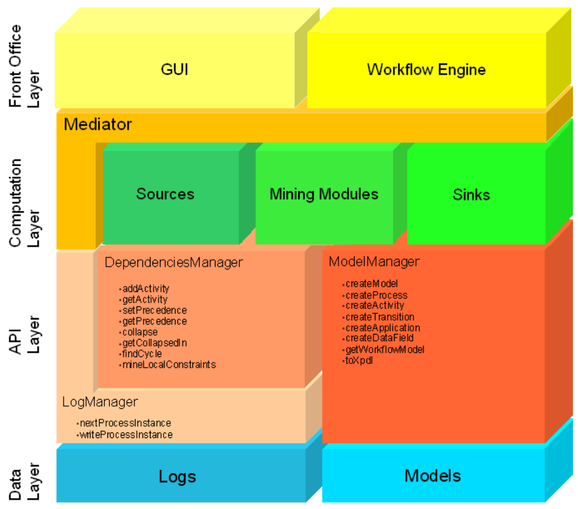


Fig. 1. ProMetheuS's Architecture

resident swap files, at the price of slower access time[2]. To face scalability issues, ProMetheuS adopts instead a data management subsystem based on a stream handling model for data acquisition. Thus, rather than building a complete in-memory representation of data, this model stores statistical sketches only, while supporting on-demand streaming access to the original log (no additional paging files are required).

2 ProMetheuS Architecture

As shown in Fig. **1**, ProMetheuS is implemented over four distinct logic layers. The *data* layer manages physical low-level operations for acquiring and storing elementary data types. The layer defines primitives for the input/output and for the modification of *Log* data, representing log files, of *Model* data, representing the abstraction of a process model, and of *Custom* data, representing user defined data-types. Regarding log representation, ProMetheuS supports the MXML data model [5], while a subset of the standard XPDL 2.0 [6] is used for model representation.

The *API* layer is responsible of two basic functionalities. Firstly, it supports the efficient internal storage of log files. In particular, it handles a main-memory repository storing *dependencies graphs*, i.e., graphs whose nodes represent the activities in the process log and whose edges represent the relationship of precedence among them. In fact, these structures are internally built by scanning once the input log, while further I/O operations may be executed when additional information is needed[3]. Secondly, it allows a transparent access to the data layer through dedicated *managers*. In details, a

[2]*ProM* 6.0 uses the *OpenXES* library---see http://www.xes-standard.org/openxes/start.
[3]*SAX* libraries are used for reading XML streams---see http://www.saxproject.org.

LogManager, a *DependenciesManager*, and a *ModelManager* provide primitives to manipulate logs, dependencies graphs, and models respectively (see Fig. **1**).

Above the API layer, it is placed the *computational* layer. In ProMetheuS, a computational resource is a plug-in component, which performs a specific task in a flow of mining. ProMetheuS provides three main templates for computational resource. A *source* is a template conceived to access the input data on which the mining analysis has to be performed. In particular, a *Log Source*, a *Model Source*, and a *Custom Source* are provided for handling logs, models, and custom data sources respectively. *Mining modules* are responsible of performing mining algorithms and statistical evaluations on the input provided by source modules. In particular, a *Log Miner* template manages a *Log* as input, and produces as output one or more instances of *Log*. A *Model Miner* template works on a *Log* input, and produces a *Model*. ProMetheuS comes equipped with various mining modules, with the default one being the α-miner [1]. A *Custom Module* template is conceived to work on *Custom* types. Finally, *sinks* templates are intended to manage the outputs of the mining process. These templates are useful for visualization, statistical analysis and storage of the computed results.

At the computational layer a *Mediator* manages communications between plug-ins and the *Front Office* layer. In particular, during the batch execution mode, the mediator automatically checks for the dependencies among the involved plug-ins, by tracing the state of the various executions and the execute availability of their input. Indeed, a plug-in may be executed only when all its inputs are available. Importantly, during the interactive mode, the mediator manages the interaction between the various graphical component associated with the different plug-ins. Basically, when the state of a component is modified, an event is generated and sent to the mediator, being then in charge of dispatching it to the other components, which can react accordingly.

The Front Office Layer exports GUI functionalities for the creation of a process mining flow, for the configuration of environment parameters, and for the visualization of results. A workflow engine is provided for the batch execution of the analysis.

3 ProMetheuS in action

We now overview the functionalities of ProMetheuS, by showcasing the complete deployment of a sample flow of mining. We also discuss some scalability results obtained by executing the flow in different configuration scenarios.

3.1 Designing and executing a flow of mining

To design a flow of mining, ProMetheuS provides the user with an intuitive GUI consisting of several graphical elements and facilities. The *Workspace Explorer* shows all available workspaces as navigable entries, in which plug-ins are organized according their type (i.e., source, mining modules, or sinks). The *Workarea* is the design panel on which users can freely customize mining flow properties. Users can quickly add/remove concrete instances of plug-in definitions (by dragging them from

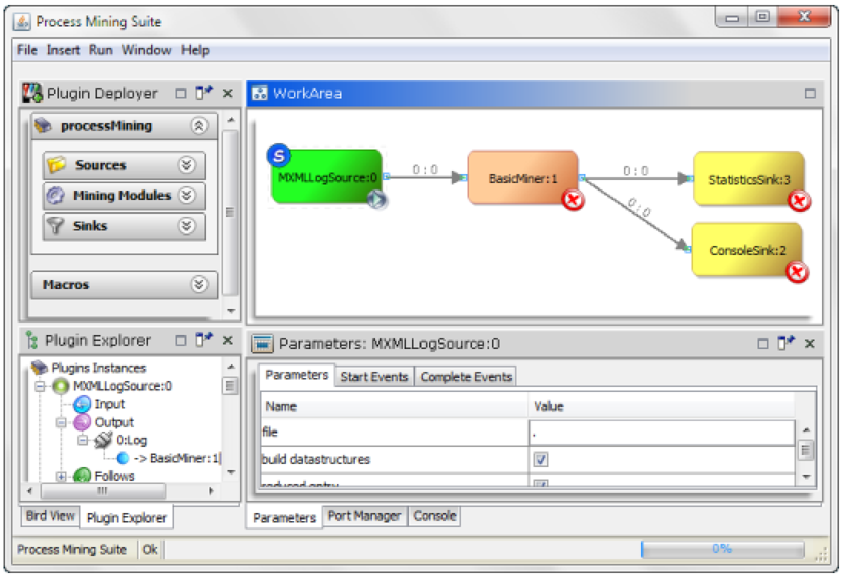
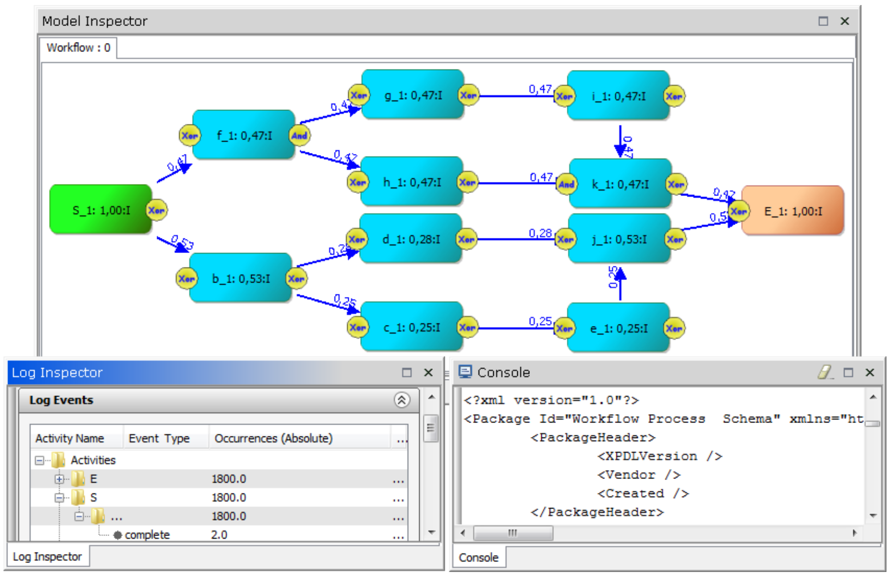


Fig. 2. Flow of mining: A *MXML Log* is connected to a *Basic α-Miner* redirecting computed model to sinks for visualization in XPDL format and statistics evaluations. Plug-ins's parameters can be configured by double-clicking on the plug-in instance.

the workspace explorer), edit connections between plug-ins, combine input/outputs, and control execution flow. Once a plug-in instance is placed, users can configure its execution environment in two steps: *Parameters Configuration*, where if the selected plug-in requires some input parameters, then users can proceed to their configuration, and *Edge Configuration*, where users can insert a new connection between modules, can rearrange a defined connection, or can remove it from the mining flow.

A sample flow of mining ready for the execution is depicted in Fig. **2**. Given a flow of mining, users can run all executable plug-ins at once, or execute only selected ones. Interestingly, users may define different flows in the same work area and run unrelated plug-ins in parallel, reducing the overall execution time of the analysis. After the computation, users can visualize the actual value of input/output data by using ProMetheuS's default *inspectors* graphical components. An inspector is a very generic data explorer, which is able to produce a suitable representation of a specific data type of the flow (see Fig. **3**). Notably, users can program their own inspectors for custom data types or can create multiple views on the same data set, each one depicting some portion of the data information of interest.

Plug-ins can be graphically composed in high-level blocks of components performing user-defined operations. In many occasions, it might be necessary to perform the same operation many times in the same mining flow or in different flows as well. In order to suite this need, ProMetheuS supports the grouping of connected plug-ins into *macros* that can be used as bricks with their own input and outputs (see Fig. **4**).



Activity Name	Event Type	Occurrences (Absolute)
Activities		
E		1800.0
S		1800.0
...		1800.0
	complete	2.0

```
<?xml version="1.0"?>
<Package Id="Workflow Process  Schema" xmlns="ht
        <PackageHeader>
                <XPDLVersion />
                <Vendor />
                <Created />
        </PackageHeader>
```

Fig. 3. Inspecting the flow: The input/output of executed plug-ins can be inspected by clicking on flow arrows or on connection ports. The log inspector shows relevant statistics (e.g., number of activities) on the log, the model inspector draws the process workflow reporting summary information (e.g., frequency of a transition), the output inspector provides the resulting XPDL.

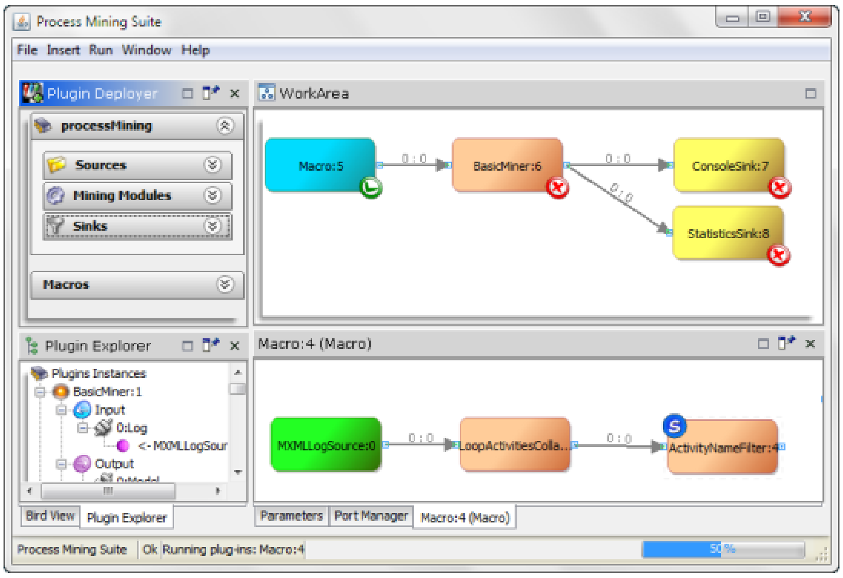


Fig. 4. Defining a macro: The source log is processed by collapsing looping activities, and by filtering activities names. Then, the output of the macro is used as input in the original flow.

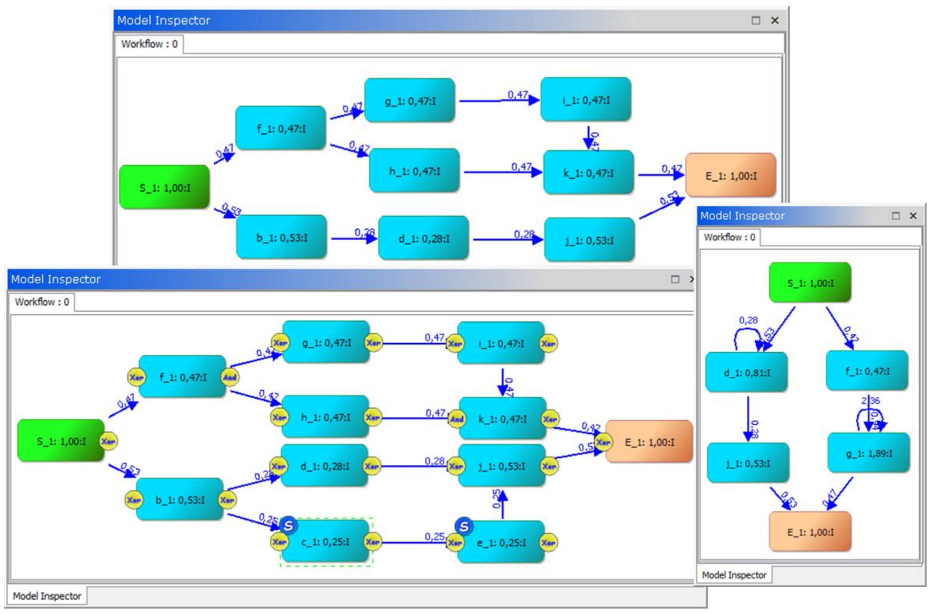


Fig. 5. Model refinement: The user runs the α-miner to compute a preliminary model. Based on summary information, (s)he sets a cut parameter for removing infrequent activities, an runs the α-miner again to compute a pruned model. Finally, (s)he identifies the desired workflow by interactively collapsing some of the intermediate model's activities through the interface.

3.2 Interactive refinement of intermediate results

ProMetheuS allows users to modify at the run-time the parameters of mining algorithms and to manipulate their execution logic on the basis of feedbacks they provide during the current execution. To support interaction, plug-ins can be equipped with customizable graphical components. In particular, a plug-in can be associated with a *menu*, with a *toolbar*, with a *main pane* (i.e., a graphical area for controlling execution), with a *bottom pane* (i.e., a panel for setting parameters), and with the *quick view* (i.e., a panel providing an overview of the plug-in status). Fig. **5** depicts a possible interactive refinement of a model computed by the α-miner of the flow of Fig. **2**.

3.3 Example execution

The execution time of the flow shown in Fig. **2** has been measured in both ProMetheuS and *ProM* considering different log sizes[4] (see Fig. **6**). Notably, the time required by ProMetheuS to complete the whole flow is much lower than the time *ProM* needs to just import the log. Moreover, as shown in Table **1**, the performances of the *ProM* buffered importer deteriorate quickly at the growing of the log size, because of the overhead of writing swap files.

[4] We used a Xeon 4 quad-core, with 8 Gb of Ram and running Ubuntu 11.04 Server.

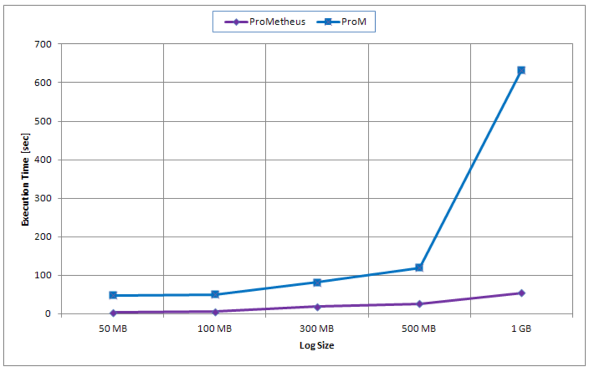


Fig. 6. Analysis Execution Times

Log size	**ProMetheuS**	***ProM***
1 Gb	55	1750
3 Gb	180	1800
5 Gb	225	1800
7 Gb	420	1800
10 Gb	580	1800

Table 1 Log importing time (timeout: 1800 sec)

4 Conclusions

In this paper we presented ProMetheuS, a suite for process mining applications [7] providing novel design elements. ProMetheuS is based on the concept of flow of mining, which enables user to design complex mining processes in which different mining tasks can be combined. Each task can be controlled interactively, and users can exploit run-time feedbacks to improve the quality of their analysis. In the case of mining large logs, the stream-based log handling may also help in achieving good scalability performances by just loading information needed for the analysis.

References

1. W. van der Aalst, A. Weijters and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128-1142, 2004.
2. H. Verbeek, J. Buijs, B. van Dongen and W. van der Aalst, "ProM 6: The Process Mining Toolkit," in *Proceedings of BPM Demonstration Track*, 2010.
3. N. van Beest and L. Maruster, "A Process Mining Approach to Redesign Business Processes - A Case Study in Gas Industry," in *Proceedings of SYNASC '07*, 2007.
4. J. Ingvaldsen and J. Gulla, "Preprocessing Support for Large Scale Process Mining of SAP transactions," in *Proceedings of Business Process Management Workshops*, 2008.
5. W. van der Aalst and B. van Dongen, "A meta model for process mining data," in *Proceedings of the CAiSE 05 Workshops*, 2005.
6. J. Ping, Q. Mair, and J. Newman, "Using UML to design distributed collaborative workflows: from UML to XPDL," in *Proceedings of IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2003.
7. W. van der Aalst, B. van Dongen, and J. Herbs, "Workflow mining: a survey of issues and approaches," *Data Knowledge Engineering*, vol. 47, no. 2, pp. 237-267, 2003.