ProM 6: The Process Mining Toolkit

H.M.W. Verbeek, J.C.A.M. Buijs, B.F. van Dongen, W.M.P. van der Aalst

Department of Mathematics and Computer Science,
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{h.m.w.verbeek,j.c.a.m.buijs,b.f.v.dongen,w.m.p.v.d.aalst}@tue.nl

Abstract. Process mining has been around for a decade, and it has proven to be a very fertile and successful research field. Part of this success can be contributed to the ProM tool, which combines most of the existing process mining techniques as plug-ins in a single tool. ProM 6 removes many limitations that existed in the previous versions, in particular with respect to the tight integration between the tool and the GUI.
ProM 6 has been developed from scratch and uses a completely redesigned architecture. The changes were driven by many real-life applications and new insights into the design of process analysis software. Furthermore, the introduction of XESame in this toolkit allows for the conversion of logs to the ProM native format without programming.

1 Introduction

Process mining allows for the extraction information from event logs [2, 3, 5, 8], and is closely related to BAM (Business Activity Monitoring), BOM (Business Operations Management), BPI (Business Process Intelligence), and data/workflow mining. Unlike classical data mining techniques the focus is on processes and questions that transcend the simple performance-related queries supported by tools such as Business Objects, Cognos BI, and Hyperion.

ProM is a generic open-source framework for implementing process mining algorithms in a standard environment [6, 1]. ProM 5.2 features over 280 plug-ins for process mining, analysis, monitoring and conversion. ProM is available as binary distribution files for Windows, Mac OS X and Unix platforms, and as source code under the terms of the CPL license.

Although ProM (up to version 5.2) has been a huge success, it did have a number of limitations. First of all, the plug-in concept lacked a clean separation between its actual process mining algorithm and its GUI. As a result, one could only use an algorithm in a context that allowed the user to provide necessary parameters through a GUI. When process mining applications become more mature/challenging, this limitation is not acceptable anymore.

A second limitation was the fact that the framework was unaware of the inputs required by the plug-ins and the outputs they delivered (for example, plug-ins were not explicitly annotated with type information to allow chaining of plug-ins). As a result, it was not really possible to define 'macro' plug-ins. The

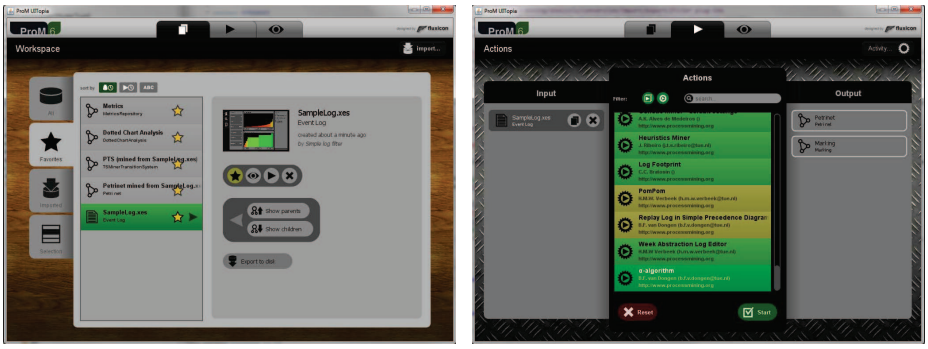

Fig. 1. ProM 6 Look&Feel

framework simply did not know which plug-ins could be 'chained' into a single plug-in. A third shortcoming were problems related to licenses. As an example, the often-used Decision Point Analysis plug-in could not be added to a ProM release because it used an L-GPL licensed library, whereas the ProM framework used the CPL license.

This paper introduces the new version of ProM, ProM 6, which is a complete overhaul of the former ProM versions that alleviates the shortcomings mentioned above. Unfortunately, as a result, plug-ins from versions 5.2 and earlier cannot run in ProM 6. However, as this paper shows, several plug-ins have been re-implemented. Moreover, recent research results are available as ProM 6 plug-ins (and not in ProM 5.2). Along with ProM, the ProM Import Framework has also been completely redesigned with the ease of defining new conversions from legacy log formats to the XES format in mind. The resulting tool called XESame is included in the ProM 6 toolkit.

2 ProM 6

The left-hand side of Fig. 1 shows the workspace of ProM 6. This workspace shows the object pool (containing logs, process models, ...), and the actions the user can take on a selected object. In Fig. 1, a log called "SampleLog.xes" has been selected, which was obtained from another log by filtering it using the "Simple log filter". Apart from other things, this view also offers the possibility to import an object from file, and to export an object to file.

The right-hand side of Fig. 1 shows the action view for the log object mentioned earlier. This view shows a list of possible actions (plug-ins) together with the required inputs and expected outputs of the selected action. The color of the action indicates whether all (green), some (yellow), or none (red) of the required inputs are present and meet the conditions. In Fig. 1, the "α-algorithm" has been selected on the "ExampleLog.xes" log, which requires an "Event log" as input and is expected to output a "Petri net" and a "Marking" (the initial

marking of the Petri net). As all inputs of the "α-algorithm" are present (it is green), it can be run immediately by selecting the "Start" button.

2.1 Contexts

In earlier versions of ProM, the actual process mining algorithms implemented by plug-ins assumed the presence of a GUI. Most algorithms require parameters, and the plug-in would ask the user for these parameters using some GUI-based dialog. Furthermore, some plug-ins displayed status information using progress bars and such. Thus, the actual process mining algorithm and the use of the GUI were intertwined. As a result, the algorithm could only be run in a GUI-aware context, say on a local workstation. This way, it was impossible to effectively run process mining experiments using a distributed infrastructure and/or in batch.

In ProM 6, the process mining algorithm and the GUI have been carefully separated, and the concepts of *contexts* has been introduced. For a plug-in, the context is the proxy for its environment, and the context determines what the plug-in can do in its environment. A plug-in can only display a dialog or a progress bar on the display if the context is GUI-aware. Typically, in ProM 6 a plug-in is split into a number of plug-ins: A plug-in for every context. The actual process mining algorithm will be implemented in a generic way, such that it can run in a general (GUI-unaware) context. This allows the algorithm to be run in any context, even in a distributed context. The dialog for setting the required parameters is typically implemented in a GUI-aware variant of the plug-in. Typically, this GUI-aware plug-in first displays the parameter dialog, and when the user has provided the parameters and has closed the dialog, it will simply run the generic plug-in using the provided parameters.

The major advantage of this is that the ProM framework may decide to have the generic plug-in run on a different computer than the local workstation. Some plug-ins may require lots of resources (time, memory, disk, ...), like for example the genetic miner. Basically, the genetic miner takes a model and a log, and then generates a number of alternative models for the given log. The best of these alternatives models are then taken as new starting points for the genetic miner. The genetic miner repeats this until some stop criterium has been reached, after which it returns the best model found so far. Clearly, this miner might take considerable time (it may take hundreds of iterations before it stops and the fitness calculation is very time-consuming for large logs), and it may take considerable memory (the number of alternative models may grow rapidly). For such an algorithm, it might be preferable to have it run on a server which is more powerful than the local workstation. Moreover, genetic mining can be distributed in several ways. For example, the population can be partitioned over various nodes. Each subpopulation on a node evolves independently for some time after which the nodes exchange individuals. Similarly, the event logs may be portioned over nodes thus speeding up the fitness calculations.

2.2 Chaining

In earlier versions of ProM, the framework did not know the required inputs and/or the expected outputs of a plug-in. As a result, it was impossible to refer to a required input or an expected output. Hence, it was not possible to combine a number of plug-ins into a 'macro' plug-in.

In ProM 6, the framework can tell the what the required inputs for a given plug-in are and what the expected outputs are. See also the action view (right-hand side of Fig. 1), which shows the inputs and outputs for the "α-algorithm" plug-in. By selecting a plug-in, the action view will show the required inputs and the expected outputs for the selected plug-in. By selecting appropriate objects as inputs, the plug-in can then be run. Furthermore, the framework can also tell which plug-ins fit with given a set of input types and a given set of output types. Suppose we have an event log, and we want to create a Petri net model from this log. Then we can simply select the "Event log" type as input for the plug-in and the "Petri net" type as expected output, and the action view will show us the list of plug-ins that actually take an event log as input and produce a Petri net as output.

Another advantage of this feature is that we can now delegate the construction of an object of a given type out of a collection of existing objects, without specifying which plug-in should be used, to the framework. The framework simply takes the existing objects as inputs, the given object type as the output type, and searches for the plug-ins that fit these inputs and outputs. Out of these fitting plug-ins one is selected and run to obtain the object of the given type. This way, a plug-in can run another plug-in without knowing that other plug-in. Furthermore, if a third plug-in becomes available that is superior to the other plug-in, we can simply replace the other plug-in by the superior plug-in.

2.3 Packages

In earlier versions of ProM, the collection of supported plug-ins was fixed. As a result, plug-ins with conflicting licenses could not be added to any ProM distribution, and the distribution most likely contained a number of plug-ins that were of no use to the user. As a result of the former, useful plug-ins like the "Decision Point Analysis" had to be left out of the standard ProM distribution, as ProM itself used the CPL license whereas the plug-in used an L-GPL licensed library. Besides licence issues there were other reasons for changing the management of plug-ins. For example, some users were overwhelmed by the many plug-ins (over 200) signaling the need for versions with smaller subsets of plug-ins.

In ProM 6, plug-ins can be installed in a dynamic way using the *ProM Package Manager*. The ProM Package Manager is a separate tool that allows the user to add and/or remove so-called packages to the installed ProM distribution. Every package contains a collection of plug-ins, and these plug-ins are installed if the package has been installed. As a result, ProM 6 can be distributed with only a core set of plug-ins, and users can add relevant plug-ins themselves. Furthermore, plug-in developers can create their own packages using only the ProM

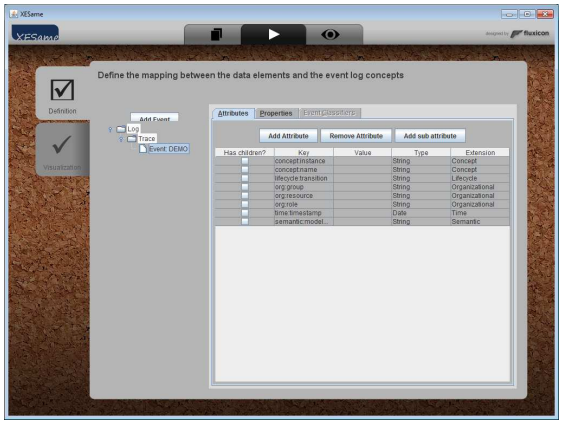

Fig. 2. XESame screenshot DUMMY

distribution: No ProM sources are required to create packages. Finally, plug-in developers can create their own package repositories, and users can install these packages through the Package Manager.

3 XESame

Although many information systems record the information required for process mining, chances are that this information is not readily available in an event log format. In ProM 5.2, the ProM Import Framework [7] was included in the ProM 5.2 for converting logs to MXML. Although there is a collection of plug-ins for various systems and data structures, new plug-ins often needed to be written in Java for a new process mining project. The main problem with this approach is that one cannot expect a domain expert to have Java programming skills. Therefore, *XESame* [4][1] has been developed and is included in the ProM 6 toolkit. XESame allows a domain expert to extract the event log from the information system at hand without having to program.

Fig. 2 shows the main conversion definition screen of XESame. Here the user can set the attribute values for different elements of the XES event log as shown in the tree at the left hand side. In this example attributes for an event are defined as displayed at the right hand side. Most of the attribute values are extracted from the data source. The name of the event, which is recorded in the *concept:name* attribute, for instance is extracted from the *eventname* field in the data source. The name of the user that executed the event is recorded in the *org:resource* attribute and is extracted from the *username* field of the related record in the *users* table. Furthermore, a custom attribute *New_Attribute* was defined which has the fixed value *New!* for each event in the resulting event log.

[1] Note that in [4] the tool is called the XES Mapper instead of XESame

4 Conclusions

ProM 6 is not just a upgrade from ProM 5.2. The tool has been completely re-implemented using an enhanced architecture and user interface. It offers many features that were not possible in earlier versions of ProM:

- Plug-ins can run in distributed environments,
- Plug-ins can be chained into 'macro' plug-ins,
- Plug-ins can be installed and updated at run-time,
- Logs can be extracted from information systems without the need for programming.

Because of the new architecture, the plug-ins from the earlier versions do not run under ProM 6, which makes it necessary to re-implement them. We believe that the new features offered by ProM 6 are well worth the effort. Moreover, we also use the opportunity to improve the functionality of plug-ins based on lessons learned for many real-life applications.

References

1. W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer-Verlag, Berlin, 2007.
2. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
3. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
4. J.C.A.M. Buijs. Mapping Data Sources to XES in a Generic Way. Master's thesis, Eindhoven University of Technology, 2010.
5. A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.
6. B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.
7. C. Günther and W.M.P. van der Aalst. A Generic Import Framework for Process Event Logs. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops, Workshop on Business Process Intelligence (BPI 2006)*, volume 4103 of *Lecture Notes in Computer Science*, pages 81–92. Springer-Verlag, Berlin, 2006.
8. J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.