Interoperability in the ProM Framework

H.M.W. Verbeek[1], B.F. van Dongen[1], J. Mendling[2], and W.M.P. van der Aalst[1]

[1] Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{h.m.w.verbeek,b.f.v.dongen,w.m.p.v.d.aalst}@tm.tue.nl
[2] Vienna University of Economics and Business Administration
Augasse 2-6, 1090 Vienna, Austria
jan.mendling@wu-wien.ac.at

Abstract. Originally the ProM framework was developed as a design artifact for the *process mining* domain, i.e., extracting process models from event logs. However, in recent years the scope of the framework has become broader and now includes process verification, social network analysis, conformance checking, verification based on temporal logic, etc. Moreover, the framework supports a wide variety of process models, e.g., Petri nets, Event-driven Process Chains (EPCs), Heuristics nets, YAWL models, and is plug-able, i.e., people can add plug-ins without changing the framework itself. This makes the ProM framework an interesting environment for *model interoperability*. For example, people can take transaction log from IBM's WebSphere, discover a process model in terms of a heuristics net, convert the heuristics net to a Petri net for analysis, load an EPC defined using the ARIS toolset, verify the EPC and convert it to a Petri net, determine the fitness of the ARIS model given the transaction log from WebSphere, and finally convert both models to a YAWL specification that is exported. Such application scenarios are supported by ProM and demonstrate true model interoperability. In this paper, we present ProM's interoperability capabilities using a running example.

1 Introduction

Information technology has changed business processes within and between enterprises. More and more, work processes are being conducted under the supervision of information systems that are driven by process models [10]. Examples are: workflow management systems such as Staffware, enterprise resource planning systems such as SAP and Baan, and recently also web services composition languages such as BPEL4WS and BPML. Unfortunately, there is little consensus on the language to be used. Existing languages are typically vendor or tool specific and do not have formal and/or executable semantics. This has resulted in the "Tower of Babel of process languages": A plethora of similar but subtly different languages inhibiting effective process support. Despite the many results in concurrency theory, it is not realistic to assume that the situation will improve

in the near future [16]. Hence there is a need to be able to convert models from one notation to another.

Moreover, even within one organization there may be many models in different languages. For example, an organization may have process models developed using ARIS, simulation models developed using Arena, and Staffware models to configure the workflow system. Even if these models describe the same process, they focus on different aspects and use different notations. Therefore, it is useful to convert models from one notation into the other.

Given the existence of a wide variety of process modeling languages and the fact that within organizations models in different languages (e.g., for simulation, for decision making, for enactment, etc.) are being made for the same process, (process) *model interoperability* is a relevant topic.

In this paper, the focus is on interoperability in the context of the *ProM* (Process Mining) framework [7]. ProM has been developed as a design artifact [15] for the *process mining* domain. Process mining aims at extracting information from event logs to capture the business process as it is being executed. Process mining is particularly useful in situations where events are recorded but there is no system enforcing people to work in a particular way. Consider for example a hospital where the diagnosis and treatment activities are recorded in the hospital information system, but where health-care professionals determine the "careflow". Many process mining algorithms have developed [3–6, 11–14] and currently a variety of these techniques are supported by ProM.

Although the initial focus of ProM was on process mining, over time the functionality of ProM was extended to include other types of analysis, model conversions, model comparison, etc. This was enabled by the plug-able architecture of ProM (it is possible to add new functionality without changing the framework itself) and the fact that ProM supported multiple modeling formalisms right from the start. By applying ProM in several case studies, we got a lot of practical experiences with model interoperability. This paper reports on these experiences using the running example depicted in Figure 1. This example will be used to provide a guided tour of the ProM framework.

Figure 1 shows an EPC (Event-driven Process Chain) [18, 20] describing a review process. In principle each paper should be reviewed by three people. However, reviewers may be tardy resulting in time-outs. After a while the reviews are collected and based on the result: a paper is rejected, a paper is accepted, or an additional reviewer is invited. In the EPC each activity is represented by a function (shown as a rectangle), states in-between activities are events (shown as hexagons), and to model the splitting and joining of flows connectors are used (shown as circles). Events and functions alternate (even in the presence of connectors). Connectors may be split or join connectors and we distinguish between XOR, OR, and AND connectors. For example, in Figure 1 the connector following function "Invite reviewers" is an OR-split connector. The last connector joining two flows after "accept paper" and "reject paper" is an XOR-join connector.

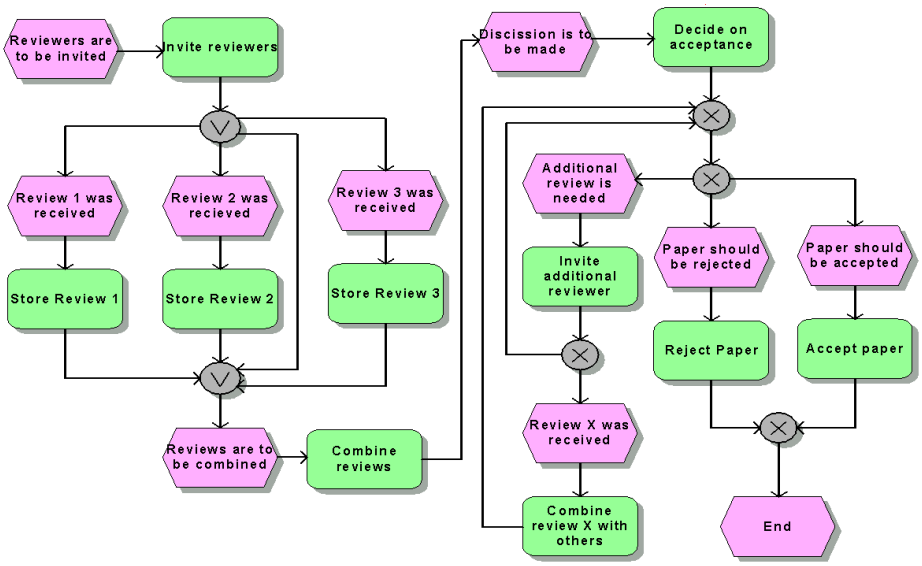

Fig. 1. The example review process model.

The EPC shown in Figure 1 could have been imported into ProM from ARIS [23], ARIS PPM [17], or EPC Tools [19]. (Note that each of these tools uses a different format.) Moreover, the EPC could have been discovered using some process mining plug-in or be the result of some conversion (e.g., translating Petri nets into EPCs). Once a model such as the EPC shown in Figure 1 is in the ProM framework, it can be used as a starting point for analysis and model conversion. For example, the EPC could be translated to a Petri net for analysis or to a YAWL diagram for enactment. In this paper, we show that such model interoperability is possible. Clearly, information can be lost in the conversions. However, it is definitely possible to support mature forms of interoperability by following the rather pragmatic approach used in ProM.

The remainder of this paper is organized as follows. Section 2 briefly introduces the ProM framework. For a more detailed introduction we refer to [7]. Section 3 shows an example of a process discovery, i.e., based on a log file a Petri net model is constructed. Section 4 takes this Petri net, and analyses to what extent *another* log corresponds to it. Section 5 converts the Petri net to both an EPC and a YAWL model. Section 6 exports the resulting YAWL model to a YAWL engine files, and shows that we can upload this file into a running YAWL engine where the process can be enacted. Section 7 concludes the paper.

2 The ProM Framework

Figure 2 shows an overview of the functionality the ProM framework. The figure shows that ProM can interact with a variety of existing systems, e.g., work-

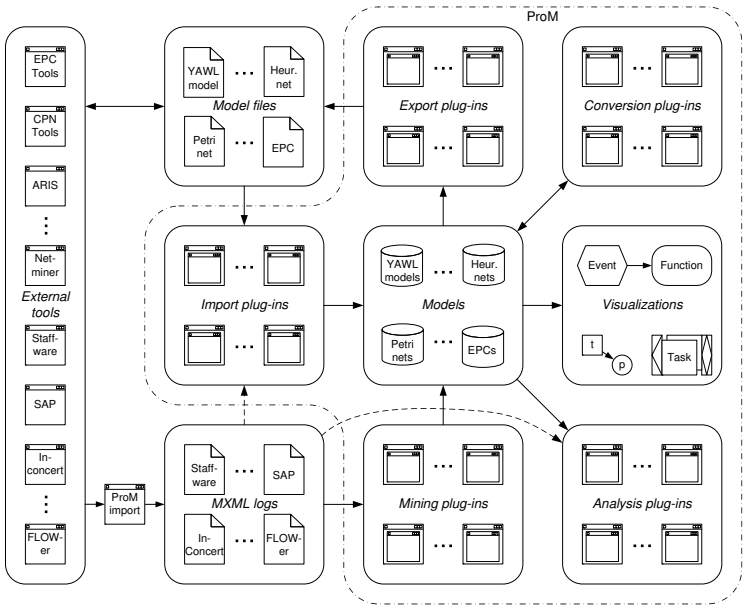

Fig. 2. Overview of the ProM framework.

flow management systems such as Staffware, Oracle BPEL, Eastman Workflow, WebSphere, InConcert, FLOWer, Caramba, and YAWL, simulation tools such as ARIS, EPC Tools, Yasper, and CPN Tools, ERP systems like PeopleSoft and SAP, analysis tools such as AGNA, NetMiner, Viscovery, AlphaMiner, and ARIS PPM. We have used more than 20 systems to exchange process models and/or event logs with ProM. As Figure 2 shows there are ways to directly import or export models or to load logs.

Although ProM is open source and people can change or extend the code, in addition we offer the so-called "plug-in" concept. Plug-ins allow for the addition of new functionality by adding a plug-in rather than modifying the source code. Without knowing all details of the framework, external parties can create (and have created) their own plug-ins with ease. Currently there are more than 70 plug-ins. ProM supports five kinds of plug-ins:

Mining plug-ins typically take a log and produce a model,
Import plug-ins typically import a model from file, and possibly use a log to identify the relevant objects in the model,
Export plug-ins typically export a model to file,
Conversion plug-ins typically convert one model into another, and
Analysis plug-ins typically analyse a model, eventually in combination with a log.

In the paper, we cannot show each of the more than 70 plug-ins in detail. Instead we focus on our running example of the review process and related mining, analysis, conversion, import and export plug-ins.

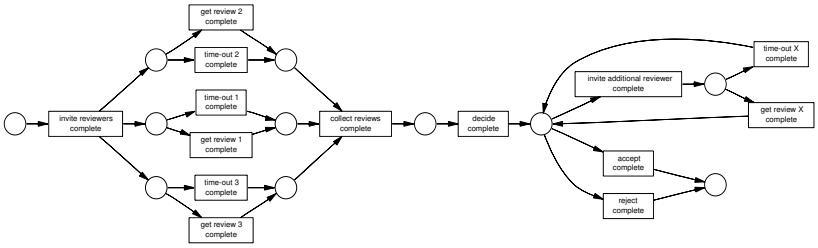

Fig. 3. The Petri net resulting from applying the α-algorithm on some mxml log.

3 Mining

Mining plug-ins like the alpha algorithm [4] and social network analyzer [2] extract models from even logs. Extracting these event-logs from different operational systems is an interoperability issue in itself, which has been dealt with in [8], where the mapping from these systems to our MXML format is described.

Most mining plug-ins discover process models represented in terms of Petri nets, EPCs, etc. However, some mining plug-ins also address other perspectives such as the data or organizational perspective.

Starting point for our running example is a log containing events related to the reviewing of papers. Based on such events we can automatically create a process model as shown in Figure 3. This model has been created using the α-algorithm [4]. Using the same log, we can also construct and analyze a social network as shown in Figure 4.

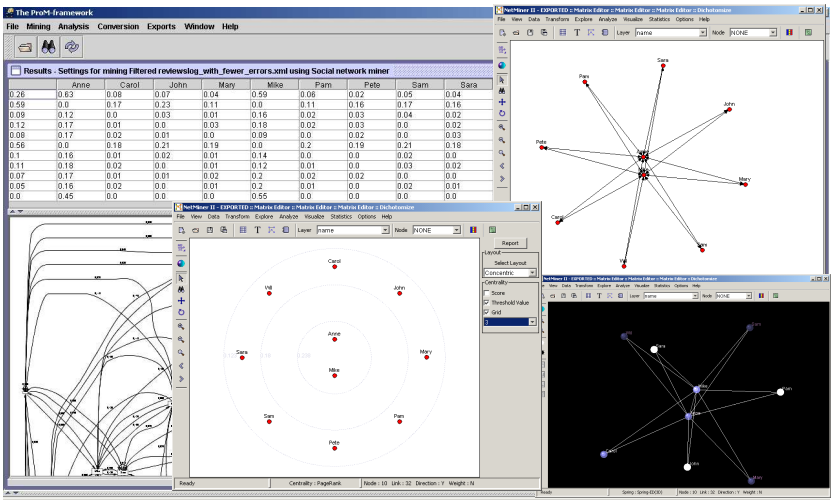

Fig. 4. A social network derived by ProM (smaller windows export into NetMiner)

4 Analysis

After obtaining a process model using process mining or by simply loading the model from another tool, we can analyse it using one of the available analysis plug-ins for this model type. Because the process model is a Petri net, we can only start a Petri-net analysis plug-in. The framework is capable of determining at runtime which plug-ins can handle the current model, and it will only offer plug-ins that can handle the current model to the user. In addition to classical analysis tools such as a verification tool, ProM also offer a conformance checker and an LTL checker as described below.

4.1 Conformance Checker

As an example, and to show how versatile ProM is, we can analyze to what extent another log fits the mined review process model. For this reason, we open another log, and start a conformance checker [22] plug-in with the combination of the process model and the log as input (note that ProM automatically offers this combination to the conformance plug-in in the analysis menu). Figure 5 shows a snippet of the results. From these results, we learn that (for example):

- The log does not fit the model entirely, as the fitness ≈ 0.89 (if the log would fit the model, the fitness would be 1).
- In 65 out of 100 cases, the process ended just before the "decide" task.
- In 29 out of the remaining 35 cases, the "decide" task was executed successfully.
- In the remaining 6 cases, an execution of the "decide" task had to be inserted to allow logged successors (like "accept" and "reject") to execute.

4.2 LTL Checker

Another interesting analysis plug-in is the LTL-checker [1], that can check logical expressions that involve time on a log. Using this plug-in, we can for example check whether in all cases the 'four-eyes principle' was satisfied, using the following LTL expressions:

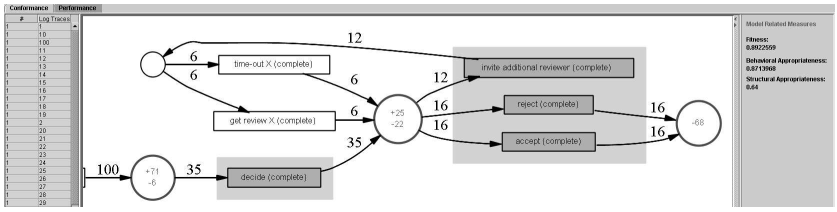

Fig. 5. A snippet of the results of the conformance checker.

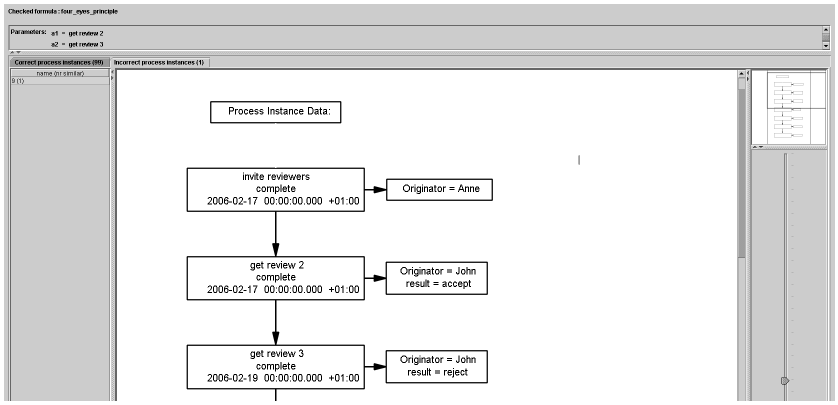

Fig. 6. A violation of four-eyes principle is discovered using the ProM LTL checker.

```
subformula execute( p : person, a : activity ) :=
{Is a specific activity executed by a specific person?}
<> ( (activity == a  /\  person == p ) ) ;

formula four_eyes_principle(a1:activity,a2:activity) :=
{Two specific activities should not be executed by the same person.}
forall[p:person |(!(execute(p,a1)) \/ !(execute(p,a2)))];
```

Figure 6 shows that this is not the case for the tasks "get review 2" and "get review 3": "John" has done both reviews.

5 Conversion

After we have analyzed the process model (a Petri net), we can convert it into other process models. For example, we can convert it into an EPC or a YAWL model. However, before doing so, we declare the four "time-out" transitions in Figure 3 to be invisible. Figure 7 shows the result. The four "time-out" transitions did not correspond to any real activities in the process, i.e., they were only there for routing purposes (to bypass the "get review" tasks). When converting one model to another we can use such information.

5.1 From a Petri Net to an EPC

First, we convert the Petri net shown in Figure 7 into an EPC. The general idea of this conversion is to map transitions to EPC functions, to derive connectors from splits and joins in the Petri Net, and to add events in order to conform with the EPC definition. The resulting EPC has the same structure as the one in Figure 1. Of course, after converting the Petri net to an EPC, different plug-ins may be applied to the process model. For example, we could check the correctness of the resulting EPC using the plug-ins described in [9]. Figure 8 shows the result: The EPC is trivially correct.

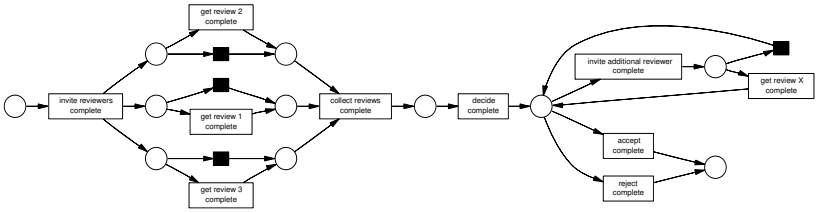

Fig. 7. The Petri net with the "time-out" transition made invisible.

5.2 From a Petri Net to a YAWL Model

Figure 9 shows the result from converting the Petri net into a YAWL model. Note that, in this case, the conversion plug-in is able to remove all routers (i.e., the invisible transitions in Figure 7) from the resulting process model. Removing the invisible transitions introduces an OR-join and an OR-split, moreover conditions (corresponding to Petri net places) are only introduced when needed. Clearly, such a "smart" translation is far from trivial, since the splits and joins have to be derived from blocks of several places and transitions in the Petri net. Similarly, there are innovative conversions from EPCs to YAWL and conversions from heuristics nets (used for genetic mining) to Petri nets.

6 Export

Of course, we can also export any model to file. For example, we can export the converted YAWL model to a YAWL engine file, which can be uploaded right-away by a YAWL engine. Figure 10 shows the result after we've uploaded the file: a YAWL model with ID "WFNet28922354" has been uploaded. Note that most fields (specification ID, specification name, documentation, ...) are generated by ProM. Figure 10 also shows a work list for the uploaded process. Currently, three work items are available in the work list: One for the task "invite reviewers", one for "decide", and one for "collect reviews".

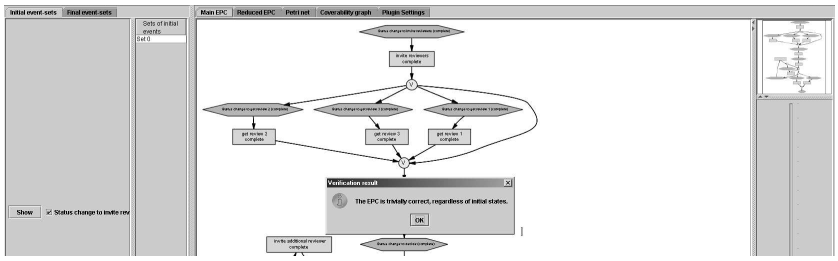

Fig. 8. A snippet of the verification result of the EPC of Figure 1

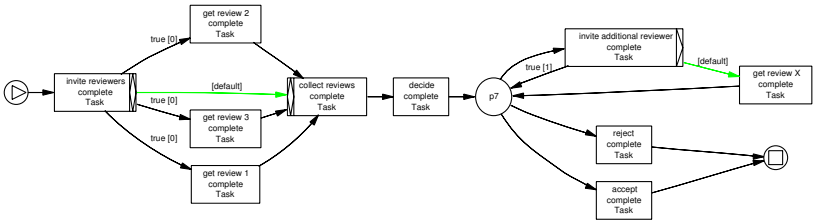

Fig. 9. The mined review process model converted to a YAWL model.

Note that sometimes a model type in ProM (e.g., Petri net or EPC) can have multiple export and import formats. For example, ProM supports three EPC formats: the ARIS Markup Language (AML) used by the ARIS toolset, the ARIS graph format used by ARIS PPM, and the EPC Markup Language (EPML) used by EPC Tools. For a detailed analysis of the heterogeneities and the different scope of these EPC formats refer to [21]. For Petri nets four different formats are supported: PNML, TPN, PNK, and CPN Tools.

7 Conclusions

This paper described the many models types and associated plug-ins that exist inside the ProM framework. Although the initial focus of ProM was on process mining, the current functionality of the tool makes ProM also interesting from a model interoperability point of view. To demonstrate this, we have used a running example.

Figure 11 provides an overview of the different ways we have used ProM regarding this example. The numbers on the edges refer to the sections where the edges were used. Prior to the paper, we used CPN Tools to generate both logs (the one we used for the mining and the one we used for the analysis), and we

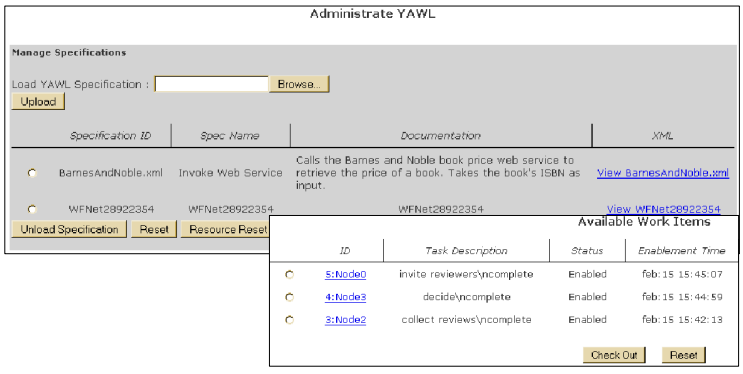

Fig. 10. The YAWL model uploaded to a YAWL server, and a worklist thereof.

used ProM*import* to convert the generated logs to the common MXML format. After having mined one log for the review process model and its social network (see Section 3), we analyzed the mined process in combination with the second log (see Section 4) to check (i) to what extent the process model and the other log fit (conformance checker) and (ii) whether the log adheres to some additional properties one would want to hold for the review process (LTL checker). Next, we converted the discovered Petri net into an EPC (which was used in Section 1) and a YAWL model (see Section 5). Finally, we exported the YAWL model (see Section 6) and uploaded the resulting YAWL engine file into a running YAWL engine.

It is important to note that in the process described Figure 11 we only partially used the broad functionality of ProM. At the moment, ProM contains 10 import plug-ins, 13 mining plug-ins, 19 analysis plug-ins, 9 conversion plug-ins, and 19 export plug-ins. Although we could only show a fraction of the model interoperability offered by ProM, Figure 11 nicely demonstrates how versatile the ProM framework is, and how it can link different external tools together.

The development and practical applications of ProM and experiences in the BABEL project [16] helped us to get a deeper understanding of model interoperability. One of the important lessons is that it is fairly easy to convert one model into another model if one is willing to accept some loss of information or precision. For example, there exist many interpretations of the semantics of EPCs (cf. the "Vicious Circle" discussion in [20]). Nevertheless, rough translations from EPCs to YAWL and Petri nets can be very useful because they

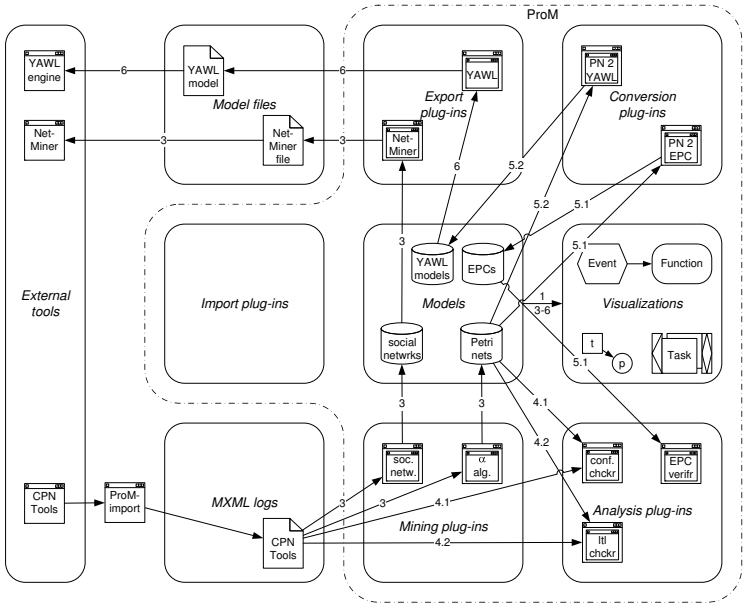

Fig. 11. An overview of the way we used ProM in this paper.

are correct in most practical cases. Moreover, operations such as EPC reduction and verification can be applied without selecting one particular semantical interpretation [9]. Therefore, we advocate a pragmatic approach which is based on simply testing model interoperability by implementing this in an environment like the ProM framework and by applying it to a wide variety of real-life models. For example, at this point in time we are converting all EPCs in the SAP R/3 reference model (approximately 600 process models) to YAWL for the purpose of verification.

Acknowledgements and relation to INTEROP

We thank INTEROP for supporting this work that has been conducted in the context of the INTEROP work package "Domain Ontologies for Interoperability" and the INTEROP-SIG "Contract and Webservices Execution Monitoring through Conformance Testing". We also thank EIT, STW, and NWO for supporting the development of the ProM framework, cf. www.processmining.org. The authors would also like to thank Ton Weijters, Ana Karla Alves de Medeiros, Anne Rozinat, Christian Günter, Minseok Song, Lijie Wen, Laura Maruster, Huub de Beer, Peter van den Brand, Andriy Nikolov, et al. for developing parts of ProM.

References

1. W.M.P. van der Aalst, H.T. de Beer, and B.F. van Dongen. Process Mining and Verification of Properties: An Approach based on Temporal Logic. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer-Verlag, Berlin, 2005.
2. W.M.P. van der Aalst, H.A. Reijers, and M. Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative work*, 14(6):549–593, 2005.
3. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
4. 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.
5. 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.
6. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
7. B. 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.

8. B.F. van Dongen and W.M.P. van der Aalst. A Meta Model for Process Mining Data. In J. Casto and E. Teniente, editors, *Proceedings of the CAiSE'05 Workshops (EMOI-INTEROP Workshop)*, volume 2, pages 309–320. FEUP, Porto, Portugal, 2005.
9. B.F. van Dongen, W.M.P. van der Aalst, and H.M.W. Verbeek. Verification of EPCs: Using Reduction Rules and Petri Nets. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *Lecture Notes in Computer Science*, pages 372–386. Springer-Verlag, Berlin, 2005.
10. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
11. W. Gaaloul, S. Bhiri, and C. Godart. Discovering Workflow Transactional Behavior from Event-Based Log. In R. Meersman, Z. Tari, W.M.P. van der Aalst, C. Bussler, and A. Gal et al., editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004*, volume 3290 of *Lecture Notes in Computer Science*, pages 3–18, 2004.
12. G. Greco, A. Guzzo, G. Manco, and D. Saccà. Mining and Reasoning on Workflows. *IEEE Transaction on Knowledge and Data Engineering*, 17(4):519–534, 2005.
13. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
14. 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.
15. A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.
16. A.H.M. ter Hofstede, M. Dumas, and W.M.P. van der Aalst. Unraveling the Babel of Process Support: On the expressiveness and exchange of business process execution languages (BABEL). Project Proposal ARC Discovery, http://www.bpm.fit.qut.edu.au/projects/babel/, 2003.
17. IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.
18. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
19. E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97. Springer-Verlag, Berlin, 2004.
20. E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. *Data and Knowledge Engineering*, 56(1):23–40, 2006.
21. J. Mendling and M. Nüttgens. Transformation of ARIS Markup Language to EPML. In *Proceedings of the 3rd GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2004)*, pages 27–38, 2004.
22. A. Rozinat and W.M.P. van der Aalst. Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In C. Bussler et al., editor, *BPM 2005 Workshops (BPI Workshop)*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2006.
23. A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.