Integration of EPC-related Tools with ProM

Paul Barborka, Lukas Helm, Georg Köldorfer, Jan Mendling, Gustaf Neumann
Vienna University of Economics and Business Administration
Augasse 2-6, A-1090 Wien, Austria
paul@barborka.com, lukas.helm@gmx.at
georg.koeldorfer@benet.at, {jan.mendling|neumann}@wu-wien.ac.at

Boudewijn van Dongen, Eric Verbeek, Wil van der Aalst
Eindhoven University of Technology
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
b.f.v.dongen@tue.nl, h.m.w.verbeek@tue.nl, w.m.p.v.d.aalst@tue.nl

Abstract: The heterogeneity of different formats for EPCs is a major problem for model interchange between specialized tools in practice. In this paper, we compare three different formats for storing EPCs, in particular, the proprietary formats of Microsoft Visio and ARIS Toolset as well as the tool-independent EPML format. Furthermore, we introduce the ProM framework and show using the case of a sales process model expressed in terms of an EPC that ProM is able to serve as a mediator between these formats. Beyond that, we demonstrate that the ProM framework can be used for the analysis of EPCs and to translate EPCs into YAWL models for execution or for further analysis.

1 Introduction

Heterogeneity of formats, tools, and notations is a notorious problem of business process management. While the heterogeneity of notations is extensively discussed in literature (see e.g. [MNN05]), there is only little attention paid to heterogeneity issues related to a single process modeling language. In this context, the availability of XML-based interchange formats plays an important role to facilitate interchange and conversion. EPCs form an example of heterogeneity issues related to a single language: there are several tools for modeling, analyzing, transforming, and managing EPCs using different representations. In order to benefit from specialized functionality, there is a strong need to exchange models between these tools.

Several of these EPC-related business process modeling tools support XML as an open standard for interchange of structured information. Such tools include *ARIS Toolset* of IDS Scheer AG, *ADONIS* of BOC GmbH, *Visio* of Microsoft Corp., *Semtalk* of Semtation GmbH, or *Bonapart* by Pikos GmbH. The heterogeneity problem in this context arises due to the fact that only some of these tools support the tool-independent EPC Markup Language (EPML) interchange format [MN06]. Most of them including e.g. ARIS and

Visio use proprietary formats that stick close to the internal information model of the tool. In order to facilitate the interchange of EPC business process models between these tools there is a strong need for an open and generic transformation platform.

Against this background, this paper aims to demonstrate how such a transformation based integration of specialized tools can be provided in practice. As an example, we present a case study that utilizes ProM for this purpose. Section 2 presents three frequently supported interchange formats for EPCs, i.e., VDX of Microsoft Visio, AML of ARIS Toolset, and EPML as a tool-independent format. In Section 3 we give an overview of ProM. In particular, we introduce the plug-in architecture of ProM and highlight which EPC-related plug-ins are already available. Section 4 showcases a scenario involving multiple specialized tools. In this scenario, ProM will be used as an integration platform. Section 5 summarizes the paper and gives an outlook on future research.

2 EPC Interchange Formats

In this section, we first introduce the interchange formats of Visio (Section 2.1) and ARIS (Section 2.2). We continue with presenting the tool-independent EPML format in Section 2.3. After that we compare the three formats in Section 2.4.

2.1 Visio and VDX

Microsoft Visio is a general drawing and modeling tool (see e.g. [WE03]). It can be customized to support any modeling language by defining a language specific stencil. EPCs are supported by a dedicated stencil that is included in the standard distribution. Visio uses the proprietary VDX XML format to store and read models.

Figure 1 gives an overview of the Visio metamodel that is the basis for the VDX format. `VisioDocument` is the root element of a VDX file. It can contain multiple `Pages` and `Masters`. A `Page` basically represents a Visio model. It is identified by an `ID` and a `NameU` attribute. A page can contain multiple `Shapes` and `Connects`. A `Shape` describes a visual object on a Visio page. It can be a simple object such as a rectangle or a composite object such as a grouping of other shapes. A shape is identified by an `ID` and a `NameU`. Furthermore, it can carry a reference to a `Master`. In Visio a `Master` provides the link between a stencil and a page. The EPC stencil defines a master for each of the EPC element types. Via the master attribute, a shape can be related to a logical object such as e.g. an EPC event or an XOR-connector. `Connect` elements offer a mechanism to logically link two shapes by referencing them in `FromSheet` and `ToSheet` attributes. Since arcs are also shapes in Visio, a control flow sequence between two EPC elements maps to two connects: one from the first object to the arc shape and another from the arc shape to the second object.

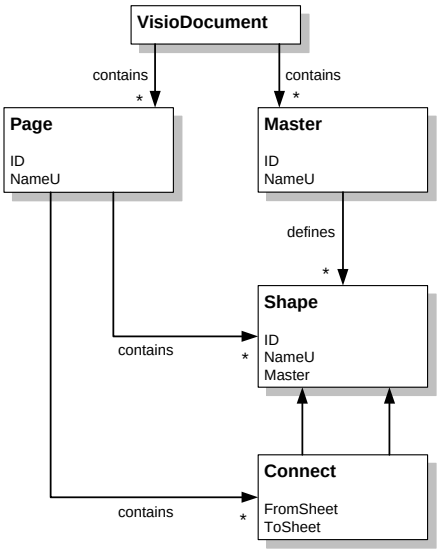

Figure 1: Visio metamodel.

2.2 ARIS and AML

ARIS is a specialized business process management tool that not only supports modeling, but also offers support for simulation and other types of analysis. EPCs are one of the modeling languages offered by ARIS. Individual models and a whole database of models can be written to a file in the proprietary ARIS Markup Language XML format. A complete overview of AML is given in [IDS03].

An AML file starts with an `AML` element as the root element. General information like time of creation, name of the ARIS database, language, and font style is stored in subelements of `AML`. The `Group` element, also a subelement of `AML`, is a container for all model-related information. In ARIS Toolset each `Group` element refers to a directory folder of the ARIS Explorer. A `Group` must have a unique `Group.ID` attribute and it may have multiple `AttrDef`, `ObjDef`, `Model` or further `Group` subelements as children. When the `Group` and its related directory have a name, ARIS Toolset stores it in an `AttrDef` (attribute definition) subelement whose `AttrDef.Type` attribute is set to `AT_NAME`. This is the typical mechanism used by AML to store model information. Every specific information of objects is stored in `AttrDef` or `AttrOcc` subelements of these objects (see Figure 2).

Another principle idea of ARIS Toolset , which is reflected in AML, is the separation between definition and occurrence: each model element is first defined in an abstract way and later referenced as an occurrence in a model. This allows one logical object to be included as multiple occurrences in models. Accordingly, the `Model` element contains `ObjOcc` (object occurrence) elements that refer to `ObjDef` (object definition) elements. The `ObjDef` element provides an abstract definition of an object. It has a unique `ObjDef.ID` attribute

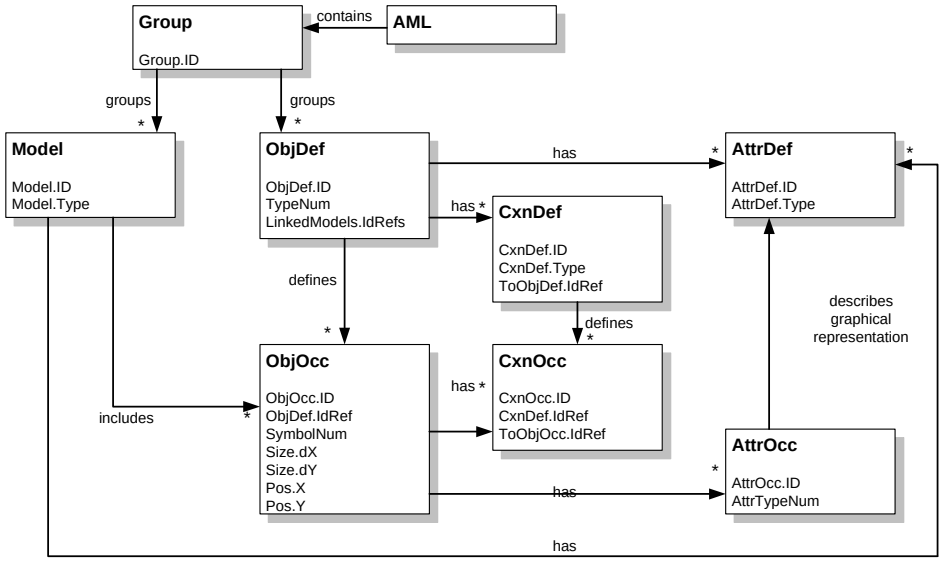

Figure 2: AML metamodel.

and a `TypeNum` attribute that refers to an object type, like e.g. EPC function or EPC event. Its `LinkedModels.IdRefs` attribute provides a list of ID-references to linked models. These can be used e.g. for hierarchical refinement of functions. `ObjDef` elements may have multiple `AttrDef` and multiple `CxnDef` subelements. `CxnDef` elements represent arcs between objects. Each `CxnDef` has a unique `CxnDef.ID` attribute, a `CxnDef.Type` attribute, and a `ToObjDef.IdRef` attribute which represents the target of the arc. Depending on the `CxnDef.Type` attribute the arc may represent control flow, information flow, or different kinds of semantic association between the objects.

A `Model` has, among others, a unique `Model.ID` and a `Model.Type` attribute. The model type, like e.g. EPC, refers to the allowed set of objects. The `Model` element may contain `AttrDef` elements to store model specific information and `ObjOcc` elements to represent graphical elements in a visual model. An object occurrence has among others a unique `ObjOcc.ID` attribute and a reference to an object definition via the `ObjDef.IdRef` attribute. The `SymbolNum` attribute refers to a graphical icon that is used to represent the object in the visual model. An EPC function would be e.g. represented by a green rectangle with radiused edges. An `ObjOcc` element may have subelements that describe its size and its position in the visual model. Furthermore the `AttrOcc` element defines how information attached via an `AttrDef` is visually represented in a model. It has a unique `AttrOcc.ID` attribute and an `AttrTypeNum` attribute that refers to its type. This type provides a syntactical link between an `AttrOcc` and an `AttrDef` element of two associated `ObjOcc` and `ObjDef` elements. Similar to object definitions `ObjOcc` may also have multiple `CxnOcc` elements. Each of them has a unique `CxnOcc.ID` attribute and a `CxnDef.IdRef` reference to an arc definition and a reference to the target of the arc via an `ObjOcc.IdRef` attribute.

2.3 EPML

EPML is a tool-independent interchange format for EPCs [MN06]. Mainly academic tools use EPML, but there are also commercial tools like Semtalk that provide EPML interfaces. Figure 3 gives an overview of EPML and its syntax elements. In EPML a hierarchy of EPC processes can be organized in directories. A `directory` element has a `name` attribute and it can contain other directories and/or EPC models. Each `epc` element is identified by an `epcId` attribute and has a `name` attribute. The `epcId` can be referenced by hierarchy relations attached to functions or process interfaces. Since an EPC process element might be used in two or more EPC models, there are `definitions` to establish the link between elements that are the same from a logical point of view. The `attributeTypes` element provides a container for the definition of additional structured information. An `attribute` references the type of the attribute and stores its value.

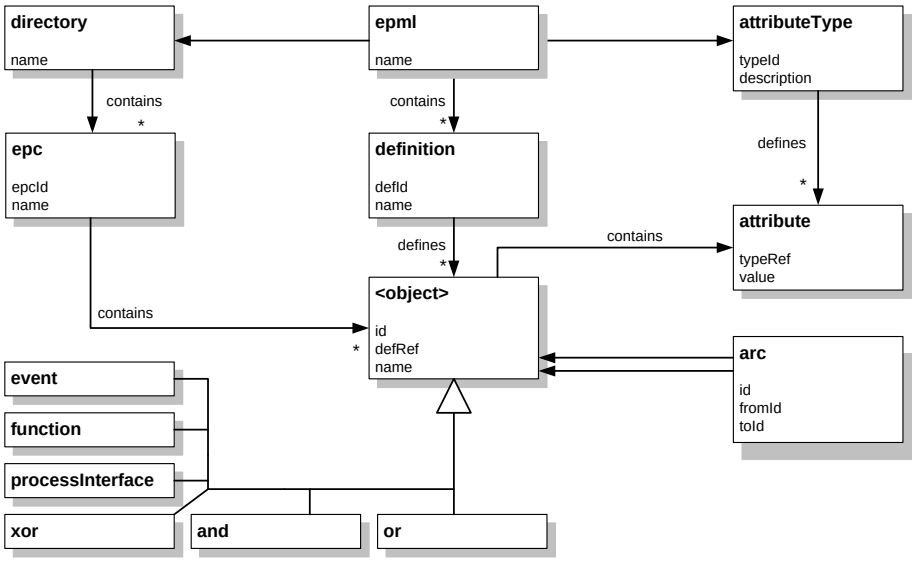

Figure 3: EPML metamodel.

In EPML each `epc` element serves as a container for all elements of the model. For each EPC element type there is a dedicated element. All elements no matter of their type have an `id` and a `name` for identification purposes, and a `defRef` if the logical element is included at different places in the model. EPML supports `event`, `function`, `processInterface`, `xor`, `and`, and `or` element types as well as some elements of extended EPCs (cf. [MN06]). Functions and process interfaces can have a reference to a linked EPC via a `linkToEpcId` attribute. All elements of an `epc` can be connected using `arc` elements with the `fromId` and `toId` attribute defining the direction. For further details on EPML the reader is referred to [MN06].

Table 1: EPC interchange formats compared

	Visio VDX	ARIS AML	EPML
EPC Model	`Page`	`Model[Model.Type]`	`epc`
Model Organization	`Pages`	`Group`	`directory`
Function	`Shape[Master]`	`ObjOcc[SymbolNum]`	`function`
Event	`Shape[Master]`	`ObjOcc[SymbolNum]`	`event`
XOR-connector	`Shape[Master]`	`ObjOcc[SymbolNum]`	`xor`
AND-connector	`Shape[Master]`	`ObjOcc[SymbolNum]`	`and`
OR-connector	`Shape[Master]`	`ObjOcc[SymbolNum]`	`or`
Control flow arc	`Connect`	`CxnOcc`	`arc`
Logical element	-	`ObjDef`	`definition`

2.4 Comparing the EPC Interchange Formats

Table 1 gives a comparison of the three discussed interchange formats Visio VDX, ARIS AML, and EPML. It illustrates that EPML is the only one that directly addresses the meta-model of EPCs. Both Visio and ARIS can store arbitrary graphical models; accordingly the data about whether an element is e.g. an EPC function is not stored in the metadata (as an XML element name), but in element and attribute values. In Visio such information is represented in the master section, in ARIS it is encoded in `TypeNum` and `SymbolNum` attributes. Beyond that, each of the interchange formats utilizes a different mechanism to represent arcs between model elements. In Visio the arc is a graphical element. The logical connection between shapes is established by a connect from the first shape to the arc, and from the arc to the second shape. In ARIS, each arc is a subelement of the source node holding a reference to the target node. In EPML, arcs are first class elements having two references to other elements. EPML is also the most compact format of the three. Since it does not include sophisticated layout and visual information, EPML files are much smaller than the AML and VDX files of the same model.

In this section, we introduced and compared three different formats to store EPCs. In the next section, we introduce the ProM framework, which is capable of reading *and* writing all of these formats. Moreover, ProM allows for a wide variety of analysis techniques for EPCs (ranging from verification to process mining) and ProM is able to transform EPCs into various alternative formats and vice versa.

3 The ProM Framework

In this paper, the focus is on interchanging the different EPC formats in the context of the *ProM* (Process Mining) framework [DMV+05]. ProM has been developed as a tool 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

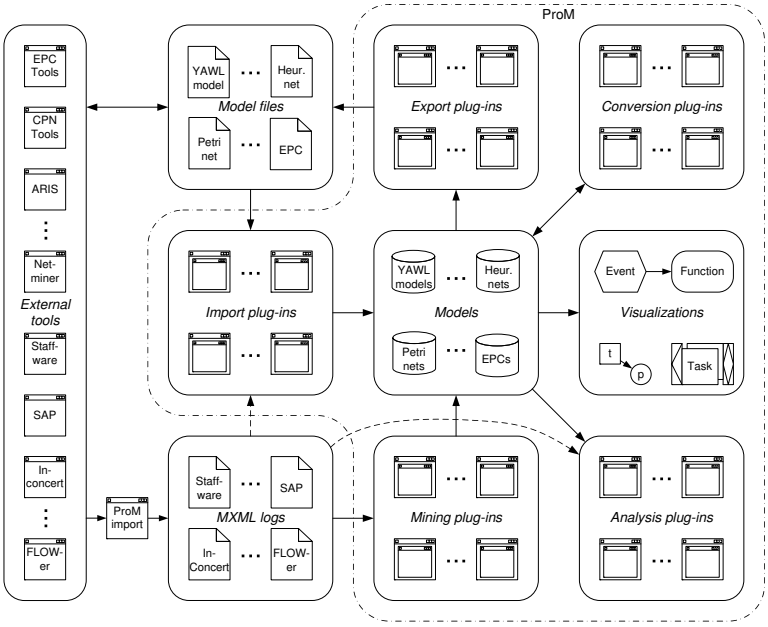

Figure 4: Overview of the ProM framework.

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 were developed [AvDH+03, AWM04, AGL98, CW98, GBG04, GGMS05, GCC+04, Her00] 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 interchange.

Figure 4 shows an overview of the functionality of the ProM framework. The figure shows that ProM can interact with a variety of existing systems, e.g., workflow 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 4 shows there are ways to directly import or export models or to load logs.

ProM is open source and people can change or extend the code. Moreover, ProM offers 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 their own plug-ins with ease. Currently there are more than 130 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 130 plug-ins. Instead we focus on EPCs, using a model for sales price calculation as reported in [BS04, p.427] as a running example (see Figure 5). The process starts with one of alternative start events and leads to a parallel execution to select articles for calculation, to select organization units, and to check the sort of the calculation. After that, the relevant calculation is determined. If the manual calculation is required, the sales price bandwidth is recorded and the sales price is calculated within that. For both automatic calculation and automatic calculation with manual confirmation, the sales price is calculated first, but in the second case it can be still subject to change. After these alternative calculations, the sales price is stored. If there is no listing process required, the price is modified for those retailers that use POS. Finally, the order set is defined to complete the process.

4 A Case Study on Multiple Tool Integration with ProM

In this section, we take the model as shown in Figure 5 in the Visio VDX format and show how we can utilize ProM, to enable the reuse of this model in other tools. Figure 6 shows ProM as a mediator between several formats that we cover in this case.

To start our guided tour of ProM, we take the EPC from Figure 5, as modelled in Visio. Figure 7 shows a screenshot of Visio, where the EPC is being modelled. After saving the EPC to a Visio VDX file, we loaded the EPC in ProM, as shown in Figure 8. Note that ProM loads files without their layout information. Therefore, ProM is able to generate layout information for any imported model. Since ProM was originally designed as a framework for *discovering* models from execution logs where the automatic generation of a layout is of the utmost importance, this feature has been added.

Once the EPC is loaded into the framework, many plugins are available for future processing of the EPC. As an example, we show a tailor-made verification approach, first presented in [DVA05]. This verification approach assumes that the process owner has knowledge about the underlying process and therefore is capable of answering questions about the possible initializations of the process. In our case, the process can be initialized when either one of the initial events occur. Furthermore, we assume that these event cannot

Figure 5: An EPC for sales price calculation taken from [BS04, p.427].

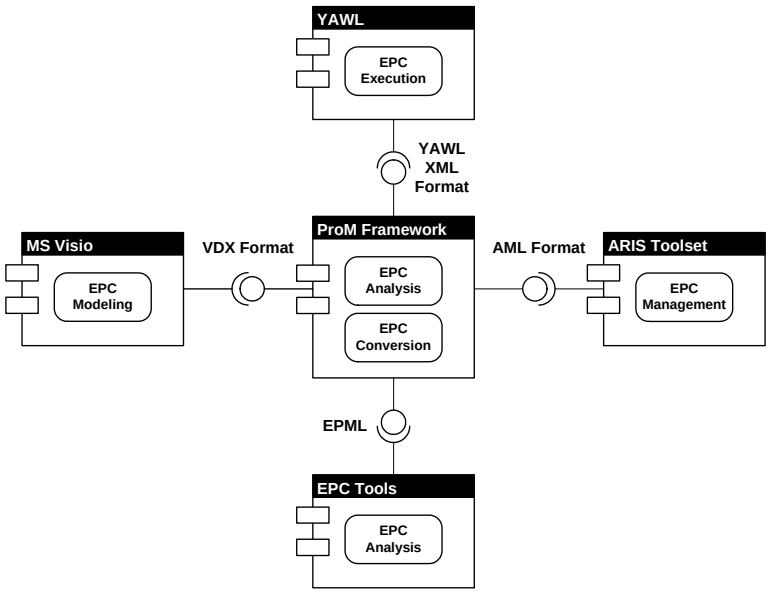

Figure 6: ProM as a mediator between different EPC formats and tools.

occur simultaneously, since that would obviously lead to a problem. The result of the EPC verification in ProM under this assumption is shown in Figure 9, where ProM indicates that there are structural errors in the EPC and the erroneous part is highlighted.

Another option is to export the EPC to an EPML file and to do the verification with EPC Tools [CK04]. EPC Tools implements EPC semantics based on the framework of Kindler [Kin03, Kin04, Kin06] which was defined to fix formal problems of the semantics reported in [NR02]. It provides a soundness check and interactive simulation capabilities. In right part of Figure 10, there is a red light indicating that the EPC is not sound. The modeler can then propagate process folders in the EPC to detect that it will not complete properly whenever automatic calculation or automatic calculation with manual confirmation is chosen. The error in this model, which was indicated by ProM and EPC Tools, now has to be repaired. Since ProM is not a modelling tool (although it does allow the user to change the type of each connector on the fly), we can now export the EPC to Visio again, update the model there and load it back into ProM. In fact, repairing the model is not so difficult, since it only requires us to change the type of one connector: if the AND-split after sales price calculated becomes and XOR, ProM now no longer indicates an error in the EPC.

Let us assume that a process designer at this point is satisfied with the model and therefore wants to upload it into the ARIS toolset. The process for doing so is simple. First, in ProM, we export the EPC to the AML format. Then, in the ARIS Toolset, we import the EPC to the database. The result of this process is shown in Figure 11. Note that again the layout is slightly different since ARIS provides its own layout algorithms as well.

So far, we have shown that EPCs can be imported and exported in several formats. Further-

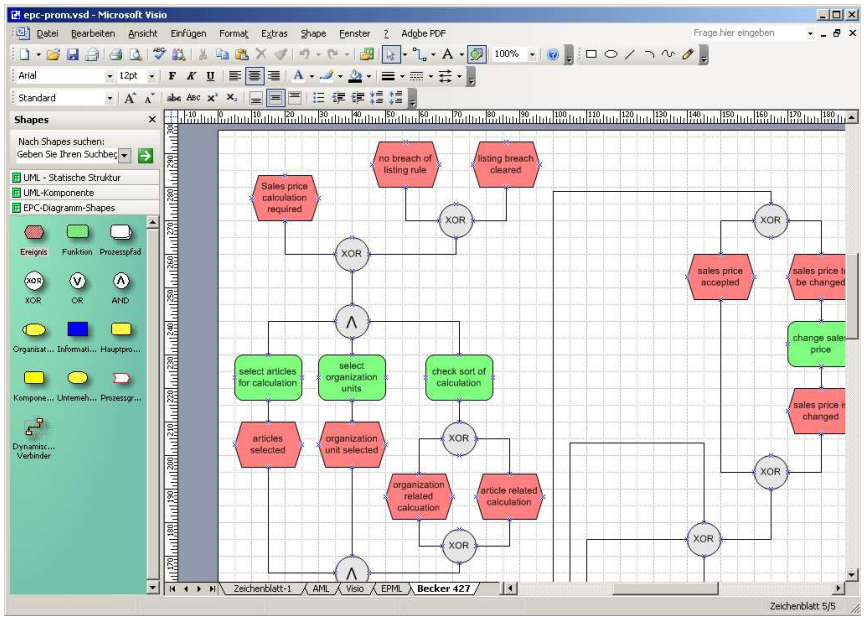

Figure 7: The example EPC modelled in Visio.

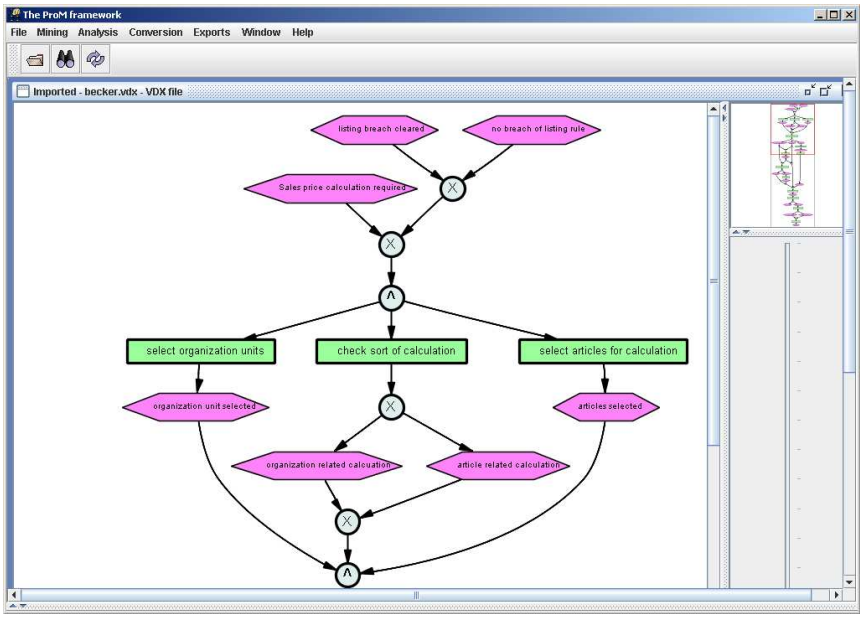

Figure 8: The example EPC imported in ProM.

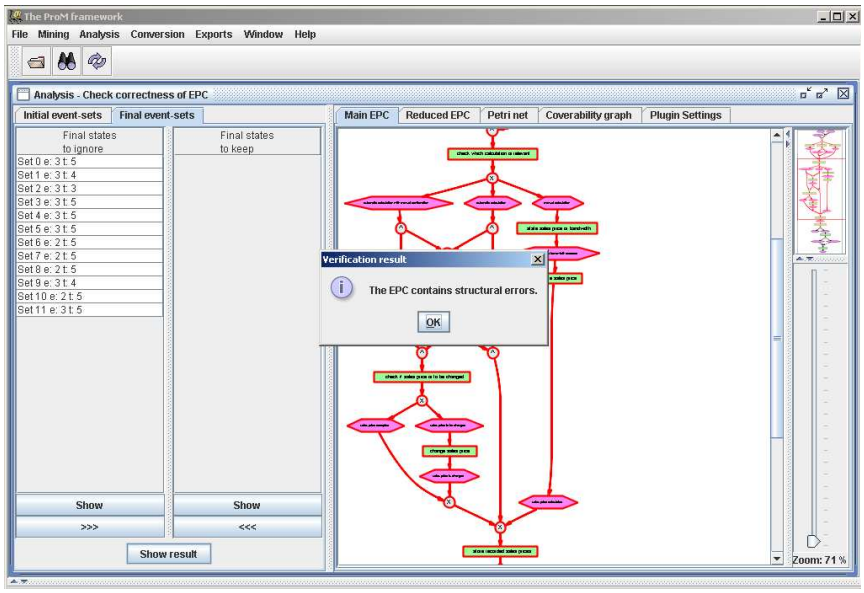

Figure 9: The result of EPC verification in ProM.

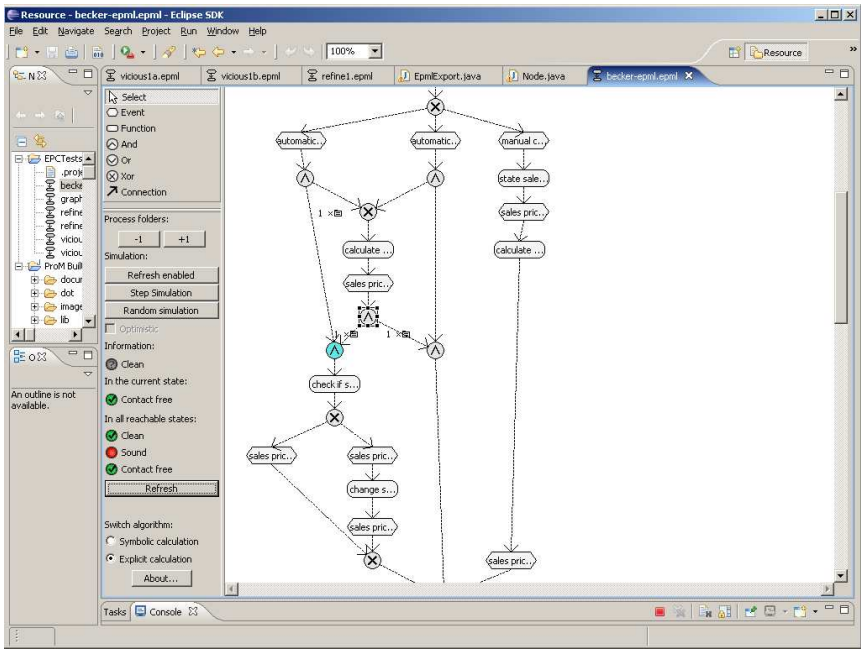

Figure 10: The result of EPC verification in EPC Tools.

more, we have shown that ProM provides plugins for the analysis of EPCs. However, there is one essential aspect of EPCs that we have not addressed yet, namely the execution of EPCs. Due to the OR-join in an EPC, the execution of EPCs is not straightforward. However, EPCs can directly be translated to YAWL models, which are executable in YAWL. ProM again provides a translation of EPCs to YAWL. These YAWL models can then be loaded in the YAWL engine (see [AADH04]). Figure 13 shows an activity of the sales price calculation enabled in the YAWL engine for execution. Beyond that, YAWL models can also be analyzed using a verification tool called WofYAWL regarding the relaxed soundness criterion [DR01]. The corrected example EPC is shown as a YAWL model in Figure 13. This approach has been used to verify all 604 EPCs in the SAP reference model, the results of which can be found in [MMN+06].

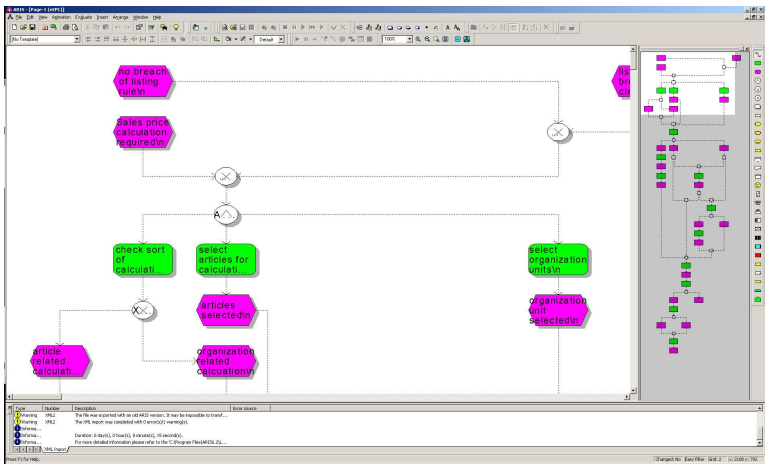

Figure 11: The corrected EPC uploaded into the ARIS Toolset

5 Conclusion and Future Work

The heterogeneity of different formats for EPCs is a major hinderance for model interchange between specialized tools in practice. In this paper, we compared three different formats for storing EPCs, in particular, the proprietary formats of Microsoft Visio and ARIS Toolset as well as the tool-independent EPML format. Furthermore, we introduced the ProM framework and showed using a realistic example (sales price calculation) that ProM is able to serve as a mediator between these formats. Beyond that, we demonstrated that the ProM framework can be used for the analysis of EPCs and to translate EPCs into YAWL models for execution or for further analysis. In future research, we aim to provide further EPC plug-ins for the ProM framework. First, there are several other popular business process modeling tools that offer EPCs, but whose proprietary interchange formats are not yet supported. Second, we aim to provide additional EPC analysis plug-ins, e.g. for deriving EPCs from a configured C-EPC.

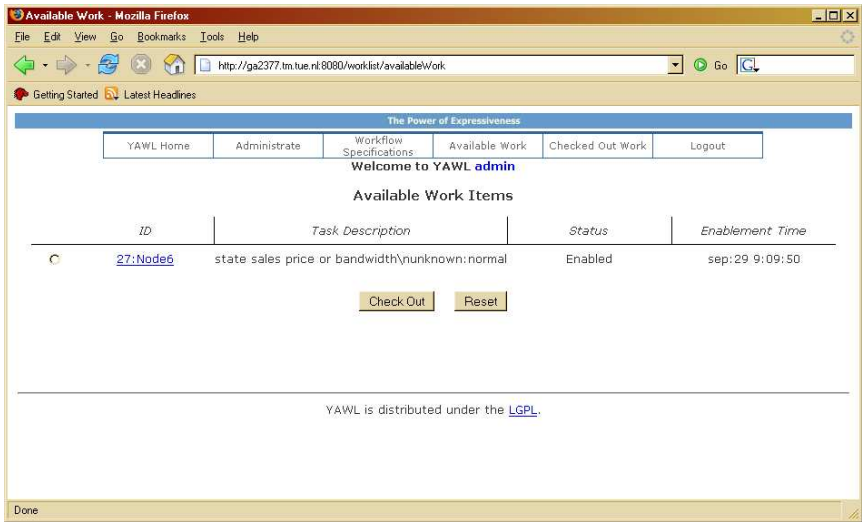

Figure 12: The state sales price or bandwidth activity enabled in the YAWL engine.

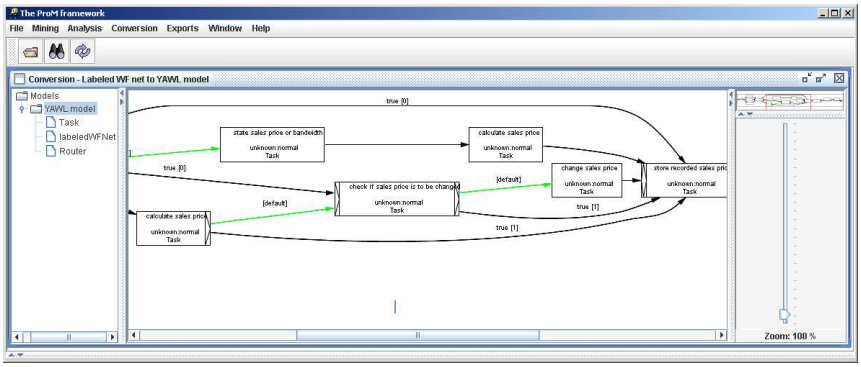

Figure 13: The corrected EPC converted to YAWL for execution.

References

[AADH04] W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and Implementation of the YAWL System. In A. Persson and J. Stirna, editors, *Advanced Information Systems Engineering, Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE'04)*, volume 3084 of *Lecture Notes in Computer Science*, pages 142–159. Springer-Verlag, Berlin, 2004.

[AGL98] 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.

[AvDH+03] 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.

[AWM04] 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.

[BS04] Jörg Becker and Reinhard Schütte. *Handelsinformationssysteme*. Moderne Industrie, Landsberg/Lech, 2nd edition, 2004.

[CK04] N. Cuntz and E. Kindler. On the semantics of EPCs: Efficient calculation and simulation. In *Proceedings of the 3rd GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2004)*, pages 7–26, 2004.

[CW98] 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.

[DMV+05] 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.

[DR01] Juliane Dehnert and Peter Rittgen. Relaxed Soundness of Business Processes. In K. R. Dittrick, A. Geppert, and M. C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, volume 2068 of *Lecture Notes in Computer Science*, pages 151–170, Interlaken, 2001. Springer.

[DVA05] B.F. van Dongen, H.M.W. Verbeek, and W.M.P. van der Aalst. Verification of EPCs: Using reduction rules and Petri nets. In *Conference on Advanced Information Systems Engineering (CAiSE 2005)*, volume 3520 of *Lecture Notes in Computer Science*, pages 372–386. Springer-Verlag, Berlin, 2005.

[GBG04] 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.

[GCC+04] 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.

[GGMS05] 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.

[Her00] 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.

[IDS03] IDS Scheer AG. *XML-Export und -Import (ARIS 6 Collaborative Suite Version 6.2 Schnittstellenbeschreibung)*. ftp://ftp.ids-scheer.de/pub/ARIS/HELPDESK/EXPORT/, Juni 2003.

[Kin03] E. Kindler. On the semantics of EPCs: A framework for resolving the vicious circle (Extended Abstract). In M. Nüttgens, F. J. Rump, editor, *Proc. of the 2nd GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2003), Bamberg, Germany*, pages 7–18, 2003.

[Kin04] E. Kindler. On the semantics of EPCs: A Framework for resolving the vicious circle. In J. Desel and B. Pernici and M. Weske, editor, *Business Process Management, 2nd International Conference, BPM 2004*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97, 2004.

[Kin06] Ekkart Kindler. On the semantics of EPCs: Resolving the vicious circle. *Data Knowl. Eng.*, 56(1):23–40, 2006.

[MMN+06] J. Mendling, M. Moser, G. Neumann, H.M.W. Verbeek, B.F. van Dongen, and W.M.P. van der Aalst. Faulty EPCs in the SAP Reference Model. In J.L. Fiadeiro S. Dustdar and A. Sheth, editors, *Proceedings of BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, page 451457, Vienna, Austria, 2006. Springer-Verlag.

[MN06] Jan Mendling and Markus Nüttgens. EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management*, 4(3):245 – 263, 2006.

[MNN05] Jan Mendling, Gustaf Neumann, and Markus Nüttgens. *Workflow Handbook 2005*, chapter A Comparison of XML Interchange Formats for Business Process Modelling, pages 185–198. Future Strategies Inc., Lighthouse Point, FL, USA, 2005.

[NR02] M. Nüttgens and F. J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel and M. Weske, editor, *Proceedings of Promise 2002, Potsdam, Germany*, volume 21 of *Lecture Notes in Informatics*, pages 64–77, 2002.

[WE03] Mark H. Walker and Nanette J. Eaton. *Microsoft Office Visio 2003 Inside Out*. Microsoft Press, October 2003.