XES, XESame, and ProM 6

H.M.W. Verbeek, Joos C.A.M. Buijs,
Boudewijn F. van Dongen, and Wil M.P. van der Aalst

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science
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 emerged as a new way to analyze business processes based on event logs. These events logs need to be extracted from operational systems and can subsequently be used to discover or check the conformance of processes. ProM is a widely used tool for process mining. In earlier versions of ProM, MXML was used as an input format. In future releases of ProM, a new logging format will be used: the *eXtensible Event Stream* (XES) format. This format has several advantages over MXML. The paper presents two tools that use this format - *XESame* and *ProM 6* - and highlights the main innovations and the role of XES. *XESame* enables domain experts to specify how the event log should be extracted from existing systems and converted to XES. *ProM 6* is a completely new process mining framework based on XES and enabling innovative process mining functionality.

1 Introduction

Unlike classical process analysis tools which are purely model-based (like simulation models), process mining requires event logs. Fortunately, today's systems provide detailed event logs. Process mining has emerged as a way to analyze systems (and their actual use) based on the event logs they produce [1,2,3,4,6,16]. Note that, unlike classical data mining, the focus of process mining is on concurrent processes and not on static or mainly sequential structures. Also note that commercial Business Intelligence (BI for short) tools are not doing any process mining. They typically look at aggregate data seen from an external perspective (including frequencies, averages, utilization and service levels). Unlike BI tools, process mining looks "inside the process" and allows for insights at a much more refined level.

The omnipresence of event logs is an important enabler of process mining, as analysis of run-time behavior is only possible if events are recorded. Fortunately, all kinds of information systems provide such logs, which include classical workflow management systems like FileNet and Staffware, ERP systems like SAP, case handling systems like BPM|one, PDM systems like Windchill, CRM systems like Microsoft Dynamics CRM, and hospital information systems like Chipsoft. These systems provide very detailed information about the activities that have been executed.

P. Soffer and E. Proper (Eds.): CAiSE Forum 2010, LNBIP 72, pp. 60–75, 2010.

However, also all kinds of embedded systems increasingly log events. An embedded system is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Examples include medical systems like X-ray machines, mobile phones, car entertainment systems, production systems like wafer steppers, copiers, and sensor networks. Software plays an increasingly important role in such systems and, already today, many of these systems log events. An example is the "CUSTOMerCARE Remote Services Network" of Philips Medical Systems (PMS for short), which is a worldwide internet-based private network that links PMS equipment to remote service centers. Any event that occurs within an X-ray machine (like moving the table or setting the deflector) is recorded and can be analyzed remotely by PMS. The logging capabilities of the machines of PMS illustrate the way in which embedded systems produce event logs.

The MXML format [7] has proven its use as a standard event log format in process mining. However, based on practical experiences with applying MXML in about one hundred organizations, several problems and limitations related to the MXML format have been discovered. One of the main problems is the semantics of additional attributes stored in the event log. In MXML, these are all treated as string values with a key and have no generally understood meaning. Another problem is the nomenclature used for different concepts. This is caused by MXML's assumption that strictly structured process would be stored in this format [10].

To solve the problems encountered with MXML and to create a standard that could also be used to store event logs from many different information systems directly, a new event log format is under development. This new event log format is named XES, which stands for eXtensible Event Stream. Please note that this paper is based on XES definition version 1.0, revision 3, last updated on November 28, 2009. This version serves as input for standardization efforts by the IEEE Task Force Process Mining [13]. Minor changes might be made before the final release and publication of the format.

The remainder of this paper is organized as follows. Section 2 introduces the new event log format XES. Of course, we need to be able to extract XES event logs from arbitrary information systems in the field. For this reason, Section 3 introduces the XES tool XESame. This tool can connect to any ODBC database, and allows the domain expert to provide the details of the desired extraction in a straightforward way. After having obtained an XES event log, we should be able to analyze this log in all kinds of ways. For this reason, Section 4 introduces the XES tool ProM 6, which is the upcoming release of the ProM framework [8]. ProM 6 supports the XES event log format, and provides a completely new process mining framework. Finally, Section 5 concludes the paper.

2 XES: eXtensible Event Stream

To explain the structure of an XES event log, we compare the way a single process containing only a single event is captured in both the MXML and the

Listing 1. Example MXML log

```
<WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
      "http://is.ieis.tue.nl/research/processmining/WorkflowLog.xsd"
    description="Example log">
  <Process id="Order">
    <ProcessInstance id="Order 1" description="instance with Order 1">
      <Data>
        <Attribute name="TotalValue">2142.38<Attribute>
      </Data>
      <AuditTrailEntry>
        <WorkflowModelElement>Create</WorkflowModelElement>
        <EventType>complete</EventType>
        <Originator>Wil<Originator>
        <Timestamp>2009-01-03T15:30:00.000+01:00</Timestamp>
        <Data>
          <Attribute name="currentValue">2142.38</Attribute>
          <Attribute name="requestedBy">Eric</Attribute>
          <Attribute name="supplier">Fluxi Inc.</Attribute>
          <Attribute name="expectedDelivery">
            2009-01-12T12:00:00.000+01:00
          </Attribute>
        </Data>
      </AuditTrailEntry>
    </ProcessInstance>
  </Process>
</WorkflowLog>
```

XES format. The event corresponds to the creation of an order on January 3, 2009 at 15:30 hours CET by the employee called Wil. The total value of the entire order is 2,142.38 Euros, it was requested by a customer called Eric, it is supplied by a company called Fluxi Inc., and delivery is expected on January 12, 2009 at 12:00 hours CET.

Listing 1 shows the way this log is captured in MXML. In MXML, a **ProcessInstance** element captures a single process instance, whereas an **AuditTrailEntry** element captures a single event. Data attributes can be associated to these elements using a **Data** element containing multiple **Attribute** elements. MXML uses a number of predefined MXML attributes:

WorkflowModelElement. This attribute captures the name of the activity that triggered the event.

EventType. This attribute captures the type of the event, like start, complete, suspend, and resume.

Originator. This attribute captures the name of the resource (human or not) who actually executed the activity.

Timestamp. This attribute captures the time at which the event occurred in the system.

Although the meaning of any of these standard attributes is generally well-understood, the meaning of the any of the other, non-standard, attributes is not.

In contrast, Listing 2 shows this log in XES, whereas Fig. 1 shows the XES meta model [11]. Intuitively, the XES **log** element replaces, the MXML **WorfklowLog** element, the **trace** element replaces the **ProcessInstance** element, and the **event** element replaces the **AuditTrailEntry** element. However, there are a number of differences worth mentioning. First of all, in XES the **log**, **trace** and **event** elements only define the structure of the document: they do not contain any information themselves. To store any data in the XES format, attributes are used. Every attribute has a string based key, a known type, and a value of that type. Possible types are **string**, **date**, **integer**, **float** and **boolean**. Note that attributes can have attributes themselves which can be used to provide more specific information.

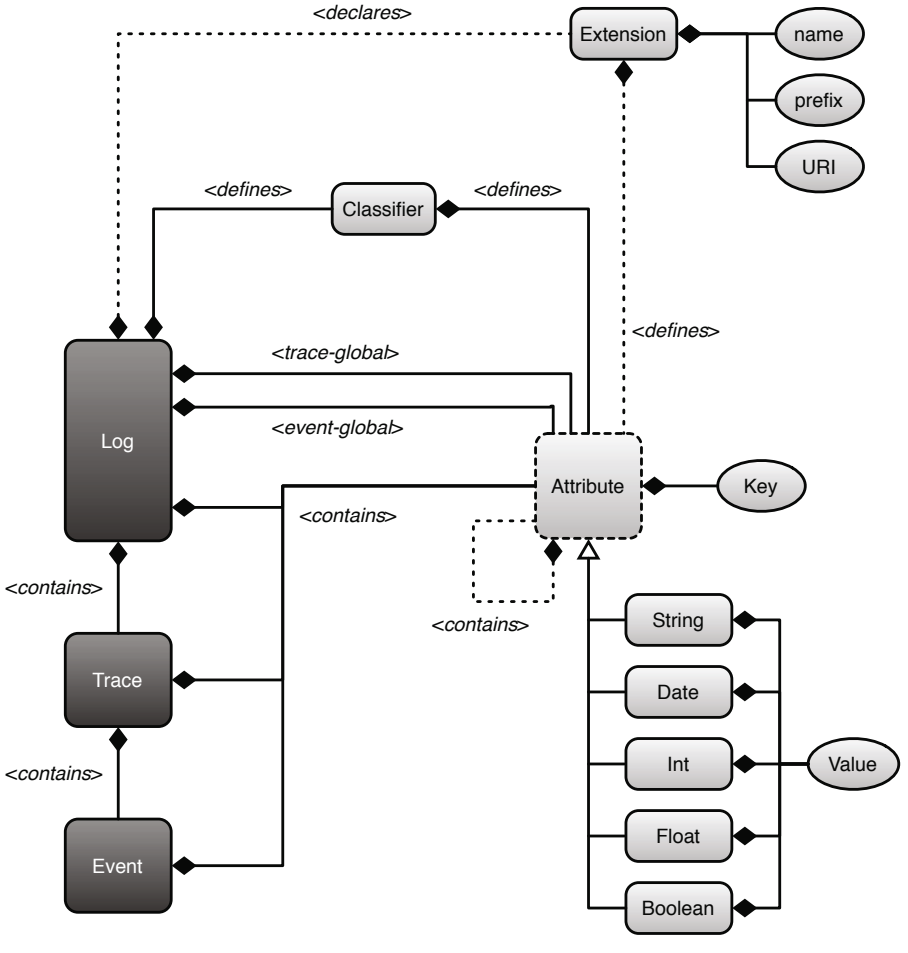

Fig. 1. XES Meta Model

Listing 2. Example XES log

```
<log>
  <extension name="Lifecycle" prefix="lifecycle"
      uri="http://www.xes-standard.org/lifecycle.xesext" />
  <extension name="Time" prefix="time"
      uri="http:///www.xes-standard.org/time.xesext" />
  <extension name="Concept" prefix="concept"
      uri="http:///www.xes-standard.org/concept.xesext" />
  <extension name="Semantic" prefix="semantic"
      uri="http:///www.xes-standard.org/semantic.xesext" />
  <extension name="Organizational" prefix="org"
      uri="http:///www.xes-standard.org/org.xesext" />
  <extension name="Order" prefix="order"
      uri="http://my.company.com/xes/order.xesext" />
  <global scope="trace">
    <string key="concept:name" value="unknown" />
  </global>
  <global scope="event">
    <string key="concept:name" value="unknown" />
    <string key="lifecycle:transition" value="unknown" />
    <string key="org:resource" value="unknown" />
  </global>
  <classifier name="Activity classifier" keys="concept:name lifecycle:transition" />
  <string key="concept:name" value="Example log" />
  <trace>
    <string key="concept:name" value="Order 1" />
    <float key="order:totalValue" value="2142.38" />
    <event>
      <string key="concept:name" value="Create" />
      <string key="lifecycle:transition" value="complete" />
      <string key="org:resource" value="Wil" />
      <date key="time:timestamp" value="2009-01-03T15:30:00.000+01:00" />
      <float key="order:currentValue" value="2142.38" />
      <string key="details" value="Order creation details">
        <string key="requestedBy" value="Eric" />
        <string key="supplier" value="Fluxi Inc." />
        <date key="expectedDelivery" value="2009-01-12T12:00:00.000+01:00" />
      </string>
    </event>
  </trace>
</log>
```

Table 1. List of XES extensions and their attribute keys

Extension	Key	Level	Type	Description
concept	name	log, trace, event	string	Generally understood name.
	instance	event	string	Identifier of the activity whose execution generated the event.
lifecycle	model	log	string	The transactional model used for the lifecycle transition for all events in the log.
	transition	event	string	The lifecycle transition represented by each event (e.g. start, complete, etc.).
organizational	resource	event	string	The name, or identifier, of the resource having triggered the event.
	role	event	string	The role of the resource having triggered the event, within the organizational structure.
	group	event	string	The group within the organizational structure, of which the resource having triggered the event is a member.
time	timestamp	event	date	The date and time, at which the event has occurred.
semantic	modelReference	all	string	Reference to model concepts in an ontology.

The precise semantics of an attribute is defined by its extension, which could be either a standard extension or some user-defined extension. Standard extensions include the *concept* extension, the *lifecycle* extension, the *organizational* extension, the *time* extension, and the *semantic* extension. Table 1 shows an overview of these extensions together with a list of possible keys, the level on which these keys may occur, the value type, and a short description, whereas Listing 3 shows the definition of the concept extension. Note that the semantic extension is inspired by SA-MXML (Semantically Annotated MXML) [15]. Note that in the example of Listing 2 some attributes are defined by an order extension (totalValue and currentValue), where other attributes are not defined by any extension (including details and supplier).

Furthermore, *event classifiers* can be specified in the **log** element which assign an identity to each event. This makes events comparable to other events via their assigned identity. Classifiers are defined via a set of attributes, from which the class identity of an event is derived. A straightforward example of a classifier is the combination of the event name and the lifecycle transition as used in MXML, which is included in Listing 2.

Finally, the fact that certain attributes have well-defined values for every trace and/or event in the log can be specified by the **global** element. For example, in the example shown in Listing 2 the event attributes concept name and lifecycle

Listing 3. Concept extension

```
<xesextension name="Concept" prefix="concept"
    uri="http:///www.xes−standard.org/concept.xesext">
  <log>
    <string key="name">
      <alias mapping="EN" name="Name"/>
      <alias mapping="DE" name="Name"/>
      <alias mapping="FR" name="Appellation"/>
      <alias mapping="ES" name="Nombre"/>
      <alias mapping="PT" name="Nome"/>
    </string>
  </log>
  <trace>
    <string key="name">
      <alias mapping="EN" name="Name"/>
      <alias mapping="DE" name="Name"/>
      <alias mapping="FR" name="Appellation"/>
      <alias mapping="ES" name="Nombre"/>
      <alias mapping="PT" name="Nome"/>
    </string>
  </trace>
  <event>
    <string key="name">
      <alias mapping="EN" name="Name"/>
      <alias mapping="DE" name="Name"/>
      <alias mapping="FR" name="Appellation"/>
      <alias mapping="ES" name="Nombre"/>
      <alias mapping="PT" name="Nome"/>
    </string>
    <string key="instance">
      <alias mapping="EN" name="Instance"/>
      <alias mapping="DE" name="Instanz"/>
      <alias mapping="FR" name="Entit"/>
      <alias mapping="ES" name="Instancia"/>
      <alias mapping="PT" name="Instncia"/>
    </string>
  </event>
</xesextension>
```

transition have well-defined values for every event, which of course is very nice (though not mandatory) as these attributes are used in an event classifier. In case a trace and/or event does not have the attribute, the value of the corresponding **global** attribute will be used. As a result of these **global** elements, plug-ins that require these attributes to have values for every trace and/or event can quickly check whether these attributes indeed have values for every trace and/or event.

3 XESame

Although many information systems record the information required for process mining, chances are that this information is not readily available in the XES format. Since the information is present in the data storage of the information system, it should be possible to reconstruct an event log that contains this information. However, extracting this information from the data storage is likely to be a time consuming task and requires domain knowledge, knowledge which is usually held by domain experts like business analysts.

For the purpose of extracting am (MXML) event log from an information system, the ProM Import Framework [9] was created. Although there is a collection of plug-ins for various systems and data structures, chances are that a new plug-in needs to be written by the domain expert in Java. The main problem with this approach is that one cannot expect the domain expert to have Java programming skills. Therefore, there is a need for a tool that can extract the event log from the information system at hand without the domain expert having to program. This tool is *XESame* [5][1].

XESame provides a generic way for extracting an event log from some data source, and is designed to be easy to use. A key strength of XESame is that no programming skills are required: The entire conversion from data source to event log can be defined through the GUI. We use a simple example to showcase XESame. Table 2 shows the contents of two tables, from which we will generate an event log. Fig. 2 shows the internal representation of the conversion in XESame. Among other things, these details show that:

- The resulting log will use the concept, lifecycle, organizational, time, and semantic extension.
- The resulting log will originate from the **events.csv** table, will have name 'Testlog', and will use the standard lifecycle model.
- The resulting log will contain the standard event classifier.
- In the resulting log, every trace corresponds to an order.
- The resulting log will contain all traces that correspond to orders with **orderID** less than 100.
- Every trace will contain all events that are related to the corresponding order.
- Every event will use the **eventname** field as concept name, the **eventtype** field as lifecycle transition, and the **timestamp** field as time timestamp.
- Every event will use the **usergroup** field from the **users.csv** table as organizational group, where both tables have been linked on the **userId** field.

To demonstrate the applicability of XESame, we have performed two case studies on data from different systems. The first case study was performed on data from an SAP system. This case study showed that a conversion definition could be defined using different tables and columns from the data source. The other case study showed that data exported from a custom system can also be converted to

[1] Note that in [5] the tool is called the XES Mapper instead of XESame.

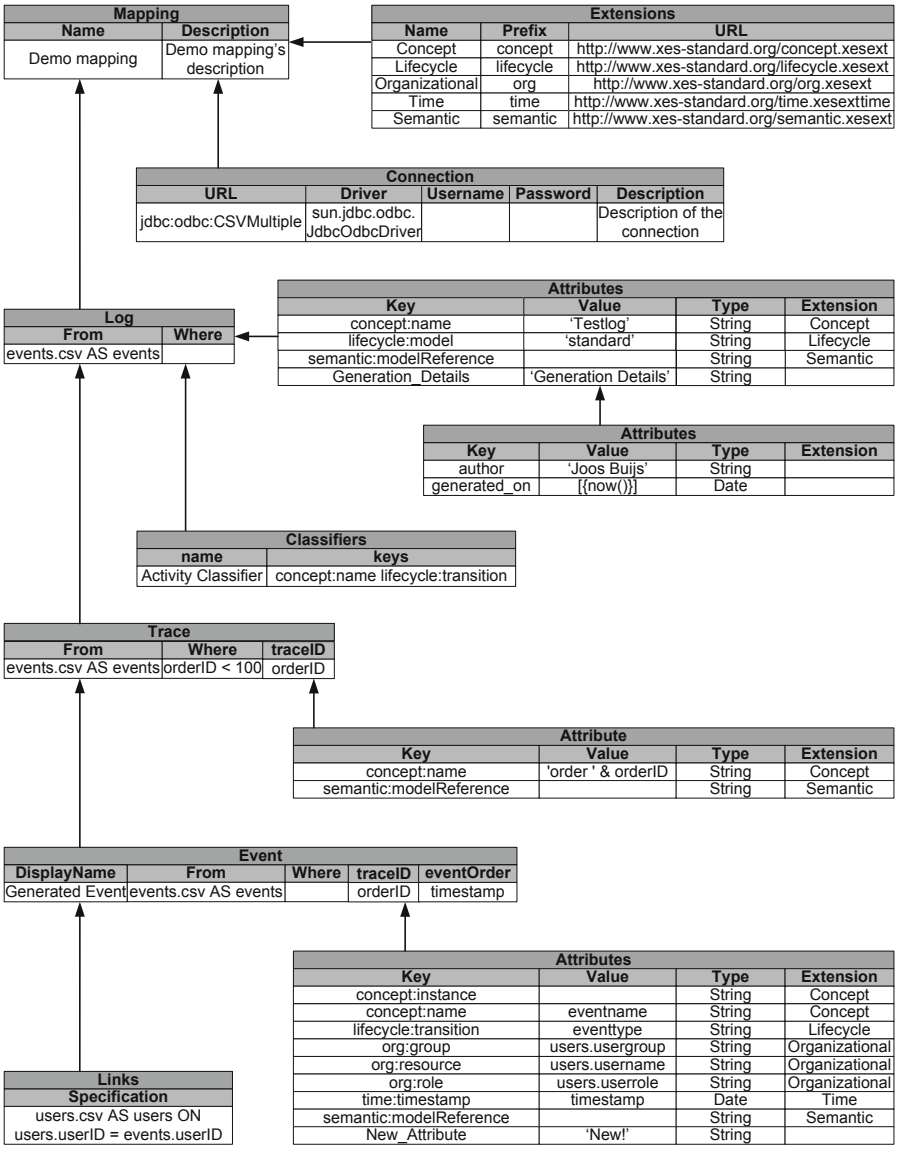

Fig. 2. An example mapping instance

Table 2. Example source data

events.csv

orderID	eventName	timestamp	eventType	userID
1	Create	1-1-2009 10:00	Start	1
1	Create	1-1-2009 11:00	Complete	1
2	Create	1-1-2009 11:00	Start	1
2	Create	1-1-2009 12:00	Complete	1
1	Send	2-1-2009 10:00	Start	2
3	Create	2-1-2009 10:01	Start	1
1	Send	2-1-2009 10:10	Complete	2
3	Create	2-1-2009 11:00	Complete	1
2	Send	2-1-2009 12:00	Start	2
2	Send	2-1-2009 13:00	Complete	2
1	Receive	3-1-2009 10:00	Start	3
1	Receive	3-1-2009 10:05	Complete	3
1	Pay	3-1-2009 15:00	Start	4
2	Pay	3-1-2009 15:00	Start	4
1	Pay	3-1-2009 15:30	Complete	4
2	Pay	3-1-2009 15:30	Complete	4
2	Receive	3-1-2009 17:00	Start	3
2	Receive	3-1-2009 17:05	Complete	3
123	Create	14-2-2009 9:00	Start	1
123	Create	14-2-2009 9:10	Complete	1

users.csv

userID	userName	userGroup	userRole
1	George	Purchase	OrderCreator
2	Ine	Support	Secretary
3	Eric	Warehouse	Reciever
4	Wil	Finance	Payer

a log by XESame. For both case studies, the performance was also investigated and shown to be linear in time with the size of the resulting event log.

4 ProM

After having extracted the event log from the information system, we can analyze the event log using ProM [8], the plugable generic open-source process mining framework. The ProM toolkit has been around for about six years. During this period, the ProM framework has matured to a professional level, which has allowed dozens of developers in different countries to contribute their research in the form of plug-ins. In the end, this resulted in ProM 5.2, which contains 286 plug-ins, and which has been used in over one hundred case studies. Some examples include:

For a provincial office of *Rijkswaterstaat* (the Dutch National Public Works Department), we have analyzed [1] its invoice process and have shown that its bad performance was mainly due to employees working at remote sites. Furthermore, we showed that it is worthwhile to combine different mining perspectives to reach a richer understanding of the process. In this case, for example, the process model revealed the problems (loops), but it took an organizational model to identify the key players, and a case-oriented analysis to understand the impact of these loops on the process performance.

For *ASML* (the leading manufacturer of wafer scanners in the world), we have investigated [17] its test process, and have made concrete suggestions for process improvement, which included reordering of tasks to prevent feedback loops and using idle time for scheduling. However, we also showed that further research is needed to develop process mining techniques that are particularly suitable for analyzing processes like the highly dynamic test process of ASML.

For the Dutch *AMC hospital*, we have shown [14] that we were able to derive understandable models for large groups of patients, which was confirmed by people of the hospital. Nevertheless, we also showed that traditional process mining approaches have problems dealing with unstructured processes as, for example, can be found in a hospital environment.

As XES is a new log format that is still under development, the older versions of ProM do not handle XES logs. Fortunately, the upcoming version of ProM, ProM 6, will be able to handle XES logs.

The fact that ProM 6 can handle XES logs where earlier versions of ProM cannot is not the only difference between ProM 6 and its predecessors (ProM 5.2 and earlier). Although these predecessors have been a huge success in the process mining field, they limited future work for a number of reasons. First and foremost, the earlier versions of ProM did not separate the functionality of a plug-in and its GUI. As a result, a plug-in like the α-miner [3] could not be run without having it popping up dialogs. As a result, it was impossible to run the plug-in on some remote machine, unless there would be somebody at the remote display to deal with these dialogs. Since we are using a dedicated process grid for process mining, this is highly relevant. Second, the distinction between the different kind of plug-ins (mining plug-ins, analysis plug-in, conversion plug-ins, import plug-ins, and export plug-ins) has disappeared; leaving only the concept of a generic plug-in. Third, the concept of an object pool has been introduced: plug-ins take a number of objects from this pool as input, and produce new objects for this pool. Fourth, ProM 6 allows the user to first select a plug-in, and then select the necessary input objects from the pool. As some plug-in can handle different configurations of objects as input, ProM 6 also introduces the concept of plug-in variants. The basic functionality of variants of some plug-in will be identical, but every variant will be able to take a different set of objects as input.

Fig. 3. ProM 6 view on the log

We use the XES event log obtained from XESame, as described in the previous section, to showcase ProM 6. Fig. 3 shows shows some basic characteristics of the log, which includes:

- the number of case (traces) (3),
- the number of events (18),
- the number of event classes (8),
- the number of event types (2),
- the number of originators (4),
- a graphical representation of the distribution of events per case,
- a graphical representation of the distribution of event classes per case, and
- some general information on the log.

Fig. 4 shows the action view of ProM 6, which includes a view on the filtered list of installed plug-ins. For every plug-in this list includes:

- the name of the plug-in (like "Simple Log Filter"),
- the name of the author of the plug-in (like "H.M.W. Verbeek"), and
- a URL where additional information on this plug-in may be found (like "http://www.processmining.org").

Fig. 4. ProM 6 action view

ProM 6 is aware of all required inputs and all output types for every plug-in. As a result, because the user has selected the event log named "Testlog.xes" to be a required input, the list of installed plug-ins show only those plug-ins that can actually take an event log as input. Plug-ins for which all required inputs are satisfied, that is, plug-ins that only need an event as input, are colored green in the plug-in list (like "Simple Log Filter"), whereas plug-ins that require additional inputs are colored yellow (like "Fitness", which also requires a Petri net as input). Note that it is also possible to filter the list of plug-ins on the output types, which allows the user to get quickly to those plug-ins that can produce an object of a type s/he needs.

Fig. 5 shows a dotted chart [18] on the log. This chart shows a graphical view on the log, where all events of a single case are plotted on a horizontal line, where the position on this line corresponds to the timestamp of the event, and where the color of the dot corresponds to the name of the event. For example, the two selected green dots in the middle correspond to two events for the case named "order 2", the first event corresponds to a send activity started by "Ine" on January 2, 2009 at 12:00 PM, whereas the second event corresponds to the matching complete event that occurred an hour later.

Fig. 6 shows the fuzzy model that result from running the "Fuzzy miner" [12] on the example log. This fuzzy model clearly shows that no strict ordering exists between the "Receive" and "Pay" activities.

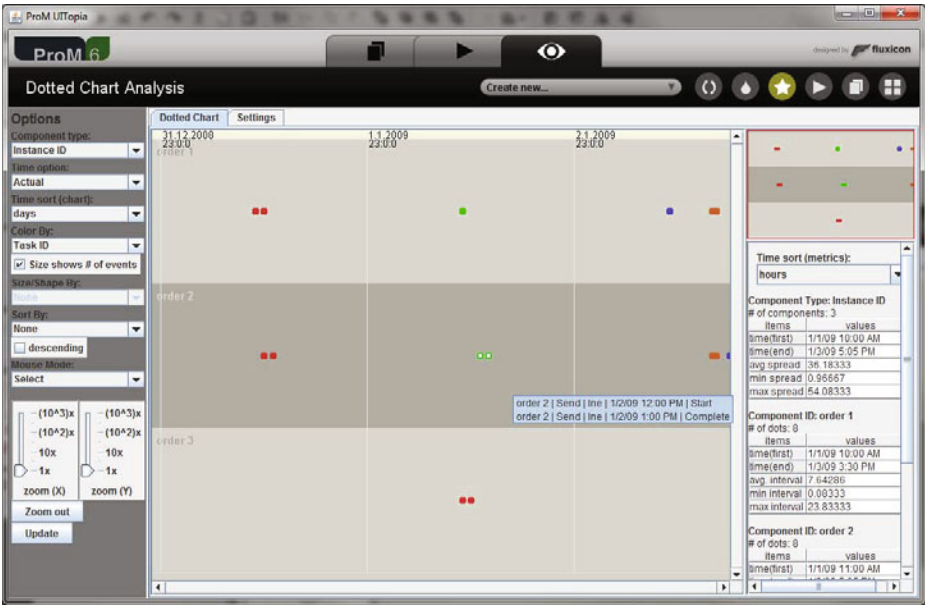

Fig. 5. ProM 6 dotted chart

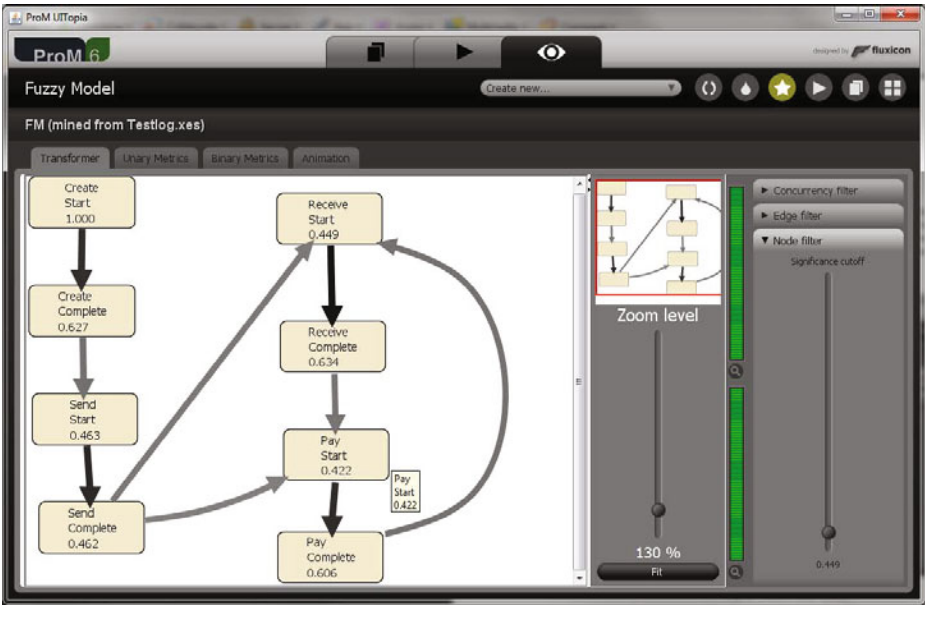

Fig. 6. ProM 6 fuzzy model

5 Conclusions

This paper has introduced the new event log format XES. The XES format enhances the existing MXML [7] in many ways, as is shown in this paper. XES is used as input for standardization efforts within the IEEE Task Force on Process Mining [13].

This paper also introduced a tool that allows the domain expert to extract an XES event log from some existing system. This tool, XESame [5], improves on the ProM Import framework [9] in the way that it is generic, and that it does not require the domain expert to create a Java plug-in for specifying (and executing) the extraction. Instead, XESame allows the domain expert to simply specify from which fields in the database which attributes in the event log should be extracted.

Finally, this paper has introduced a new version of the ProM framework [8], ProM 6. In contrast to earlier versions of ProM, ProM 6 can handle XES event logs, can be executed on remote machines, and can guide the user into selecting the appropriate inputs for a certain plug-in. As a result, it better supports the analysis of event logs than any of the earlier releases did.

ProM 6 will be released in the Summer of 2010, but interested readers may already obtain a prerelease (which also contains XESame) or so-called 'nightly builds' through the Process Mining website (`www.processmining.org`).

Acknowledgements. The authors would like to thank Christian Günther for his work on the XES standard and the new UI of ProM 6.

References

1. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business Process Mining: An Industrial Application. Information Systems 32(5), 713–732 (2007)
2. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering 47(2), 237–267 (2003)
3. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
4. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
5. Buijs, J.C.A.M.: Mapping Data Sources to XES in a Generic Way. Master's thesis, Eindhoven University of Technology (2010)
6. Datta, A.: Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. Information Systems Research 9(3), 275–301 (1998)
7. van Dongen, B.F., van der Aalst, W.M.P.: A Meta Model for Process Mining Data. In: Casto, J., Teniente, E. (eds): Proceedings of the CAiSE 2005 Workshops (EMOI-INTEROP Workshop), vol. 2, pp. 309–320. FEUP, Porto, Portugal (2005)

8. van Dongen, B.F., Alves de Medeiros, A.K., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
9. Günther, C.W., van der Aalst, W.M.P.: A Generic Import Framework for Process Event Logs. In: Eder, J., Dustdar, S. (eds.) BPI 2006. LNCS, vol. 4103, pp. 81–92. Springer, Heidelberg (2006)
10. Günther, C.W.: Process Mining in Flexible Environments. PhD thesis, Eindhoven University of Technology, Eindhoven (2009)
11. Günther, C.W.: XES Standard Definition. Fluxicon Process Laboratories (November 2009)
12. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
13. IEEE Task Force on Process Mining, `http://www.win.tue.nl/ieeetfpm`
14. Mans, R.S., Schonenberg, M.H., Song, M., van der Aalst, W.M.P., Bakker, P.J.M.: Application of process mining in healthcare - a case study in a Dutch hospital. In: Fred, A., Filipe, J., Gamboa, H. (eds.) Biomedical Engineering Systems and Technologies. Communications in Computer and Information Science, vol. 25, pp. 425–438. Springer, Heidelberg (2009)
15. Alves de Medeiros, A.K., Pedrinaci, C., van der Aalst, W.M.P., Domingue, J., Song, M., Rozinat, A., Norton, B., Cabral, L.: An Outlook on Semantic Business Process Mining and Monitoring. In: Meersman, R., Tari, Z., Herrero, P. (eds.) SWWS 2007. LNCS, vol. 4806, pp. 1244–1255. Springer, Heidelberg (2007)
16. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33(1), 64–95 (2008)
17. Rozinat, A., de Jong, I.S.M., Günther, C.W., van der Aalst, W.M.P.: Process mining applied to the test process of wafer steppers in ASML. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews (2009) (to appear)
18. Song, M., van der Aalst, W.M.P.: Supporting Process Mining by Showing Events at a Glance. In: Chari, K., Kumar, A. (eds.) Proceedings of 17th Annual Workshop on Information Technologies and Systems (WITS 2007), Montreal, Canada, pp. 139–145 (December 2007)