Configuring IBM WebSphere Monitor for Process Mining

H.M.W. Verbeek and W.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,w.m.p.v.d.aalst}@tue.nl

Abstract. Process mining has emerged as a way to discover or check the conformance of processes based on event logs. This enables organizations to learn from processes as they really take place. Since web services are distributed over autonomous parties, it is vital to monitor the correct execution of service processes. Fortunately, the "web services stack" assists in collecting structured event logs. These logs can be used to extract new information about service processes (like bottlenecks) and to check the conformance. In this paper, we demonstrate that such an event log can be obtained in the context of the IBMs WebSphere environment. More specifically, we show how to configure the WebSphere Business Monitor in such a way that it collects all the information needed for generating an event log.

1 Introduction

IBM's WebSphere suite is IBM's solution for Service Oriented Architectures (SOAs for short), and includes tools like the WebSphere Business Modeler (Modeler for short), the WebSphere Integration Developer (WID), the WebSphere Process Server (Server), and the WebSphere Business Monitor (Monitor). Fig. 1 shows an overview of these tools, and how they are positioned in the lifecycle of a service. When a service process is running on the Server, the Server will emit all kinds of events related to that service on the system's event bus, called the Common Event Infrastructure (CEI for short). The Monitor can monitor the running process by subscribing to these events.

The events that pass the CEI can potentially be used by process mining techniques. The goal of process mining is to extract information (e.g., process models, social networks, bottlenecks, and predictive models) from event logs. Typically, process mining assumes that it is possible to sequentially record events such that each event refers to an activity (a well-defined step in the process, like "Register complaint") and is related to a particular case (a process instance, like "Complaint of John Doe"). Furthermore, some techniques use additional information such as the resource that performed the activity that generated the event, the timestamp of the event, or data elements recorded with the event (like the name of the complainant).

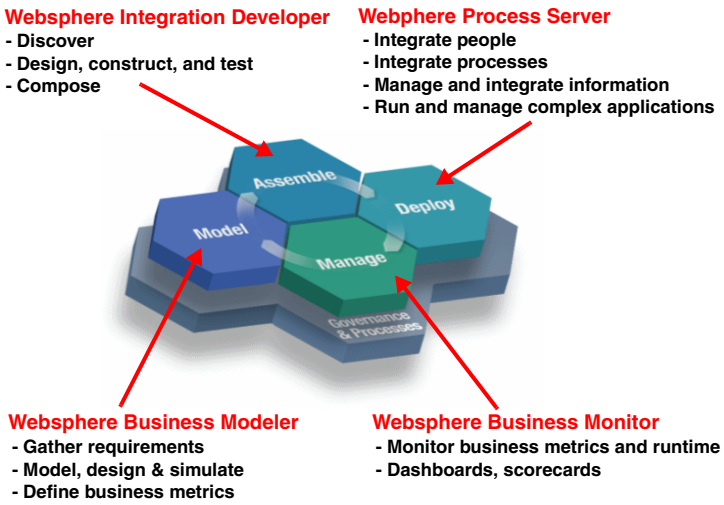

Fig. 1. Service lifecycle supported by the "Modeler", the "WID", the "Server", and the "Monitor" (taken from [7]).

In this paper, we discuss the possibility of doing process mining on the events that pass from the Server to the Monitor through the CEI. It seems straightforward to tap the events from the CEI, store them in a log, and do process mining on that log, but there are some problems with this approach, as we will show. As demonstrated in this paper, these problems can be solved by limiting ourselves to the events that the Monitor requires. Unfortunately, the Monitor only stores aggregated data, and in general this is not sufficient for doing process mining. Finally, we will show a way to have the Monitor store sufficient data for doing process mining, which shows that it is possible to apply process mining techniques in the context of the IBM WebSphere SOA suite.

The remainder of this paper is organized as follows. Section 2 introduces IBM's WebSphere suite. Section 3 briefly introduces process mining. Section 4 shows that an event log can be obtained using the Monitor as the source of events. Finally, Section 5 concludes the paper.

2 IBM WebSphere

This section describes the tools of the WebSphere suite that are relevant for this paper: the Modeler, the WID, the Server, and the Monitor. As the Monitor is crucial for our process mining goal, the emphasis of this section will be on the Monitor. The reader interested in additional details on the Monitor is referred to [7]. As a running example for this Section, we take the fictional Clips and Tacks company which is also used in [7].

Clips and Tacks is an office supply company. First, some client submits an order. Second, a business rule evaluates whether the order can be automatically approved, which happens if the total order price does not exceed 750 US$. Otherwise, it requires approval from an employee. Third, after the order has been approved, the systems checks whether the customer account is in good standing. If not, an additional review by an employee is required. Fourth, if all is okay, the order is shipped. Otherwise, the order is canceled and the client is notified.

2.1 Modeler

The Modeler is the component used by business analysts to make an abstract model of the business. Apart from a process model and a list of business items, this model also contains list of business measures. These business measures are to be monitored by the Monitor. A business measure can either be a business performance indicator or a monitored value. Business performance indicators include Key Performance Indicators (KPIs for short), instance metrics, and aggregate metrics. A KPI corresponds to a variable with a numerical value (which includes durations), and has a target and an optional time period. The target of the KPI corresponds to the value the business has to achieve at least. Examples of KPIs are "Average Process Duration" with a target of 3 days, or "Percentage of Orders Shipped" with a target of 90% and a rolling time period of 30 days. Note that the Modeler does not provide any information on how to actually compute these KPIs.

2.2 WID

The WID is the component used by developers to make an executable model. Fig. 2 shows the Clips and Tacks process model as it was imported from the Modeler. Like with the abstract model, the executable model contains both a process model and a monitor model. In the end, the executable process model will be deployed to the Server, whereas the monitor model will be used to configure the Monitor (as it needs to know which events to expect and how to interpret them).

Before creating a monitor model, the developer needs to select all relevant process-related events that the Server needs to emit to the Monitor. Relevant events include process events (see Fig. 3), BPEL invoke events, and BPEL flow events. The process events allow the Monitor to compute for example an average process duration, the invoke events allow it to compute for processing times and other activity-based measurements, whereas the flow events allow the Monitor to extract which decisions were taken. With regard to the decisions, note that the Modeler exports an IBM-like BPEL process to the WID, which means that BPEL links are used to model decisions. As these links belong to a BPEL flow, we need to emit the events for this flow.

The monitor model contains information including KPIs, metrics, stopwatches, triggers, events, and dimensions. Basically, the Monitor receives events

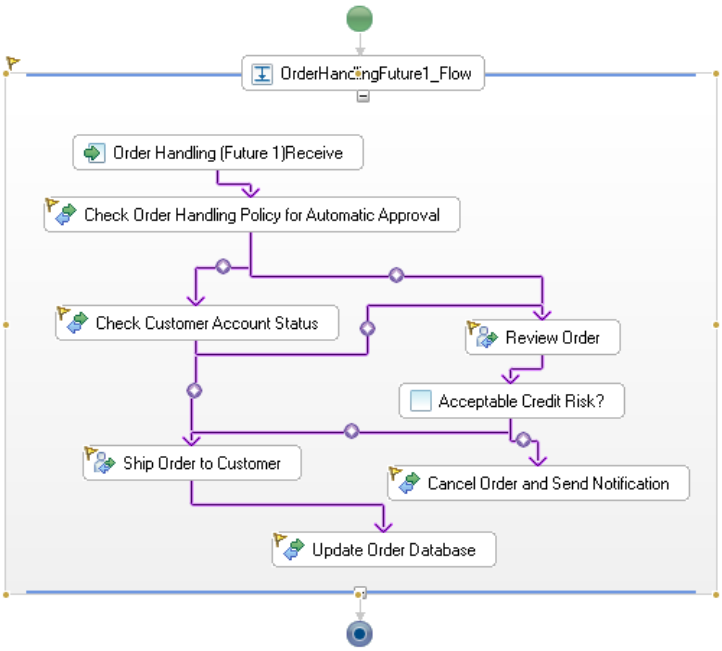

Fig. 2. The BPEL process model.

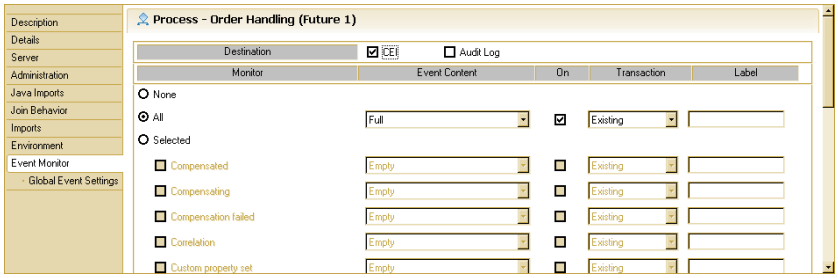

Fig. 3. Emitting events to the Monitor.

from the Server, and the monitor model allows the Monitor to generate the KPIs from these events through stopwatches, triggers, and metrics. The dimensions can be used to drill down in the data by the business user who is viewing the Monitor dashboards.

For a correct functioning of the Monitor, the concept of a *monitor context* is of utmost importance. Inbound events will be grouped together by the Monitor into these monitor contexts, and the computation of KPIs depends heavily on these contexts. As an example, to compute the process duration KPI we need the first and last event that correspond to a running instance. These events should be grouped into one context, where the first event should create this context and the last event should terminate it (through a trigger). In the mean time, every

Property	Value
Name	Average Review Time
Operator	Average
Measure	Review Order Elapsed Duration for Measure
Target	4 minutes
Ranges	Good (up to 3 minutes)
	Passable (from 3 to 5 minutes)
	Bad (from 5 to 10 minutes)

Table 1. Properties for the Average Review Time KPI.

other event that can be correlated to both events just adds information to this context.

The WID can deploy the executable process model to the Server, and can export the executable monitor model to the Monitor. While running the deployed process model, the Server will emit events that will be caught and interpreted according to the monitor model by the Monitor.

2.3 Server

The Server runs the deployed models, and emits the prescribed events that are needed by the Monitor to interpret the stream of events correctly. All configuration for the Server was done in the WID, except for some 'business situations'. These situations need to be translated into alerts, which will show up on the Alert view in the Monitor dashboard. This configuration is done using the administrative console of the Server.

2.4 Monitor

The Monitor receives the events emitted by the Server. It interprets these events according to the monitor model, and stores the aggregated event data in its own database. Using the Monitor dashboards, a business user can access the stored data through a number of predefined views, which include KPI views, Instance views, and Human Task Views.

Apart from creating views on the stored data, The Monitor also allows the business user to create a new KPI. As an example, the business user can create a new KPI for the completion time of the "Review Order task" in the Clips and Tacks example, with the properties as shown by Table 1. Fig. 4 shows this new KPI in the Monitor.

3 Process Mining

This section introduces the concept of process mining. Unlike classical process analysis tools which are purely model-based (like simulation models), process

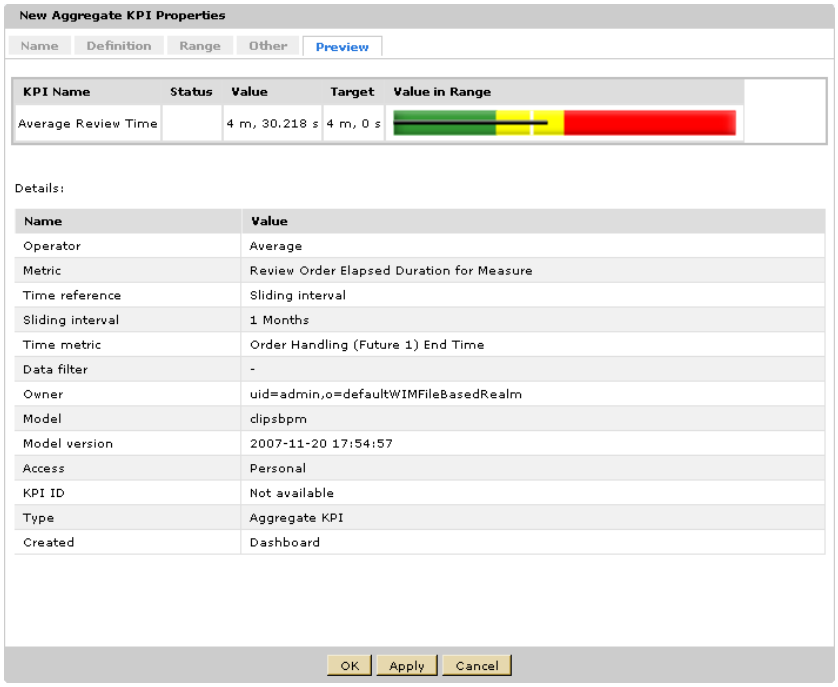

Fig. 4. Result for the Average Review Time KPI.

mining is based on the actual behavior captured in 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 [2, 3, 4, 5, 8, 10, 11, 13, 14]. 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 (like the Monitor) 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 at a much more refined level. In the context of a hospital, BI tools focus on KPIs such as the number of knee operations, the length of waiting lists, and the success rate of surgery. Process mining is more concerned with the paths followed by individual patients and whether certain procedures were followed or not.

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 Staffware and FileNet, ERP systems like SAP and Oracle, 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.

Traditionally, process mining has been focusing on discovery, that is, deriving information about the original process model, the organizational context, and execution properties from enactment logs. An example of a technique addressing the control flow perspective is the α-algorithm, which constructs a Petri net model [9, 12] describing the behavior observed in the event log. However, process mining is not limited to process models, and recent process mining techniques are more and more focusing on other perspectives, like the organizational perspective or the data perspective. For example, there are approaches to extract social networks from event logs and analyze them using social network analysis [1]. This allows organizations to monitor how people, groups, or software/system components are working together.

Note that the Monitor in IBM WebSphere does not (yet) support process mining. The focus is on showing KPIs rather than finding out what the actual process is. For example, the Monitor does *not* support the discovery of process models, social networks, predictive model, etc. Given the large installed based of WebSphere and the need for more refined run-time analysis, it is interesting to add process mining to IBM's suite.

4 WebSphere Event Logs

This section investigates whether the IBM WebSphere suite allows for the collection and extraction of event logs that can be used as input for process mining. At first sight, it seems logical to tap the event log directly from the CEI, as all events in the suite pass this bus and this format seems well-suited for process mining. However, there are a number of problems with this approach (see also [6]). First, we would need to specify which events should be collected into the event log. If we're interested in a certain process, we should only collect events that correspond to that process. Adding events from other processes will make process mining hard. Second, the events related to a particular process may correspond to different levels of abstraction. Some events correspond to a very high-level event, like starting some activity, whereas some other events correspond to a very low-level event, like updating the value for some variable. Having events on multiple levels of abstractions also makes process mining hard. Third, we would need to be able to cluster the collected events into process instances, that is, we would need to be able to retrieve for some event to which process instance it actually belongs. Unfortunately, this information is not always easy to obtain. As a result, it is certainly not straightforward to collect the events directly from the CEI.

The use of a monitor model and the Monitor can help solving the problems mentioned above. Creating a monitor model in the WID serves two purposes. First, it extends the process model in such a way that the Server will emit a proper set of events (all related to the process and all at the same level of abstraction) once it is running. Second, it creates a matching monitor model

that enables the Monitor to catch and interpret these incoming events, and to divide these events up into *monitor contexts*. By default, the monitor model will be such that a monitor context corresponds to the notion of a process instance, hence the Monitor will collect all events to a single process instance into a single monitor context.

There is, however, a downside related to using the Monitor as the source of event logs, as it only stores aggregated data. By default, this aggregated event data contains all kinds of metrics on the process level, which are required to compute the necessary KPIs. Thus, if the sojourn time of a case happens to be a KPI, then the timestamps of the first and last events will be stored as metrics. Clearly, for process mining the event log should contain more detailed information on these events (like the name of the corresponding activity), and it should also contain this information for the other events. By default, this information is not stored by the Monitor.

Fortunately, the monitor model can be easily adapted to our needs. Until now, we more or less have assumed that a monitor context would contain all events of an entire process instance. The first event of a process instance typically creates the monitor context, the last typically terminates it, while the other events just add information to the running monitor context. However, it is not mandatory to have monitor context correspond to a process instance. For example, it is also possible to have a monitor context for a single event, and to store the necessary event-related information required for process mining into that monitor context. This would fit our purposes nicely, as we can simply store all information we need for a certain event into its corresponding monitor context.

For the process instance id, which is vital to process mining, it is important to realize that process-instance-related events will have this id as the current id (that is, ECSCurrentID), whereas the activity-related (BPEL invoke) events will have this as the parent id (ECSParentID). For this reason, it is important to distinguish these two kinds of events. As a result, in our monitor context, we will have two types of inbound events. Fig. 5 shows how the process instance id can be defined for the Clips and Tacks example, where the AnyParentEvent corresponds to a process-instance-related event, and the AnyChildEvent to an activity-related event.

Fig. 5 also shows a number of metrics for the event context, which include the activity name, the resource name, and the creation time. This information can be retrieved from the standard event in a straightforward way, as Fig. 6 shows for the creation time metric. For sake of completeness, we mention here how the other metrics can be retrieved as well:

activityKind AnyChildEvent/BPELGenericData/bpc:activityKind
activityName AnyChildEvent/BPELGenericData/bpc:activityTemplateName
resourceName AnyChildEvent/BPELGenericData/bpc:principal

After having added this monitor context to our monitor model, we can redeploy our running Clips and Tacks example and extract the events. Each event refers to a process instance id and has an activity name, activity type, timestamp, and selected data elements. Based on this we can apply a wide variety

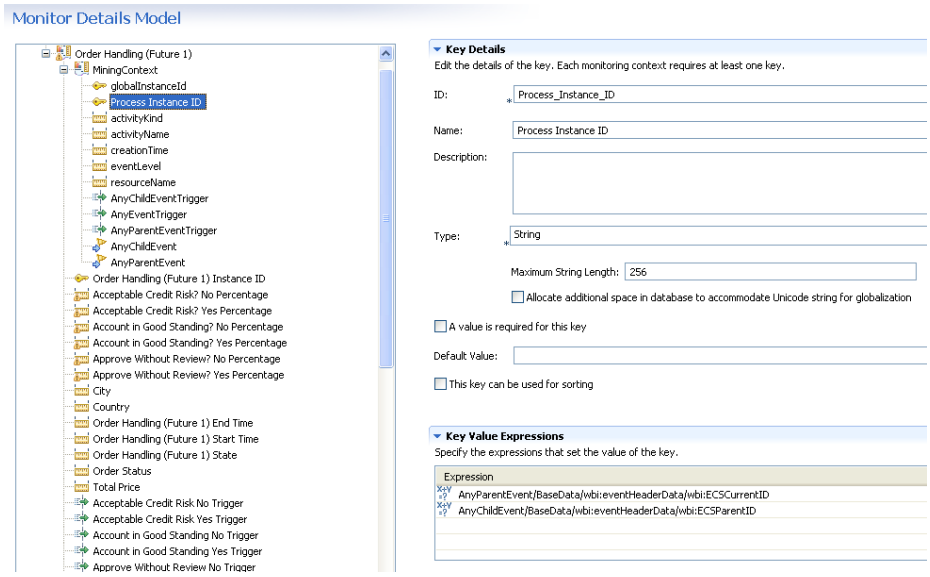

Fig. 5. Process instance id.

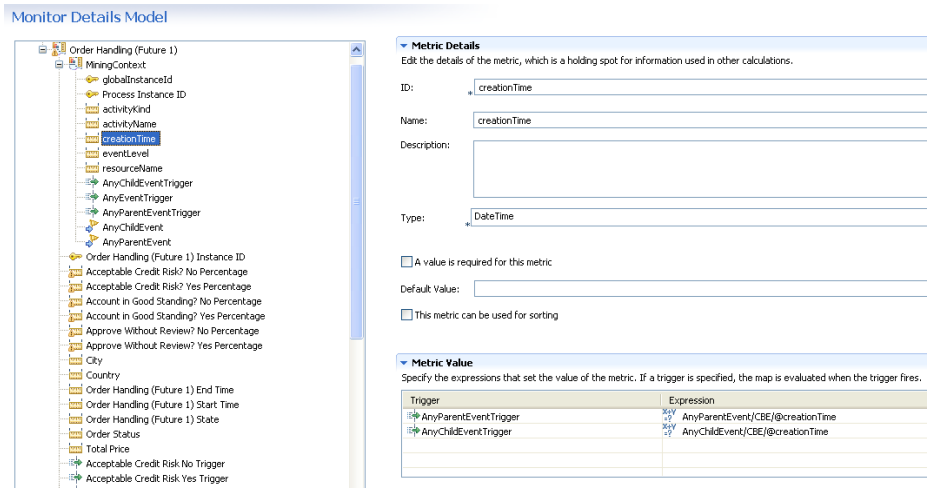

Fig. 6. Creation time.

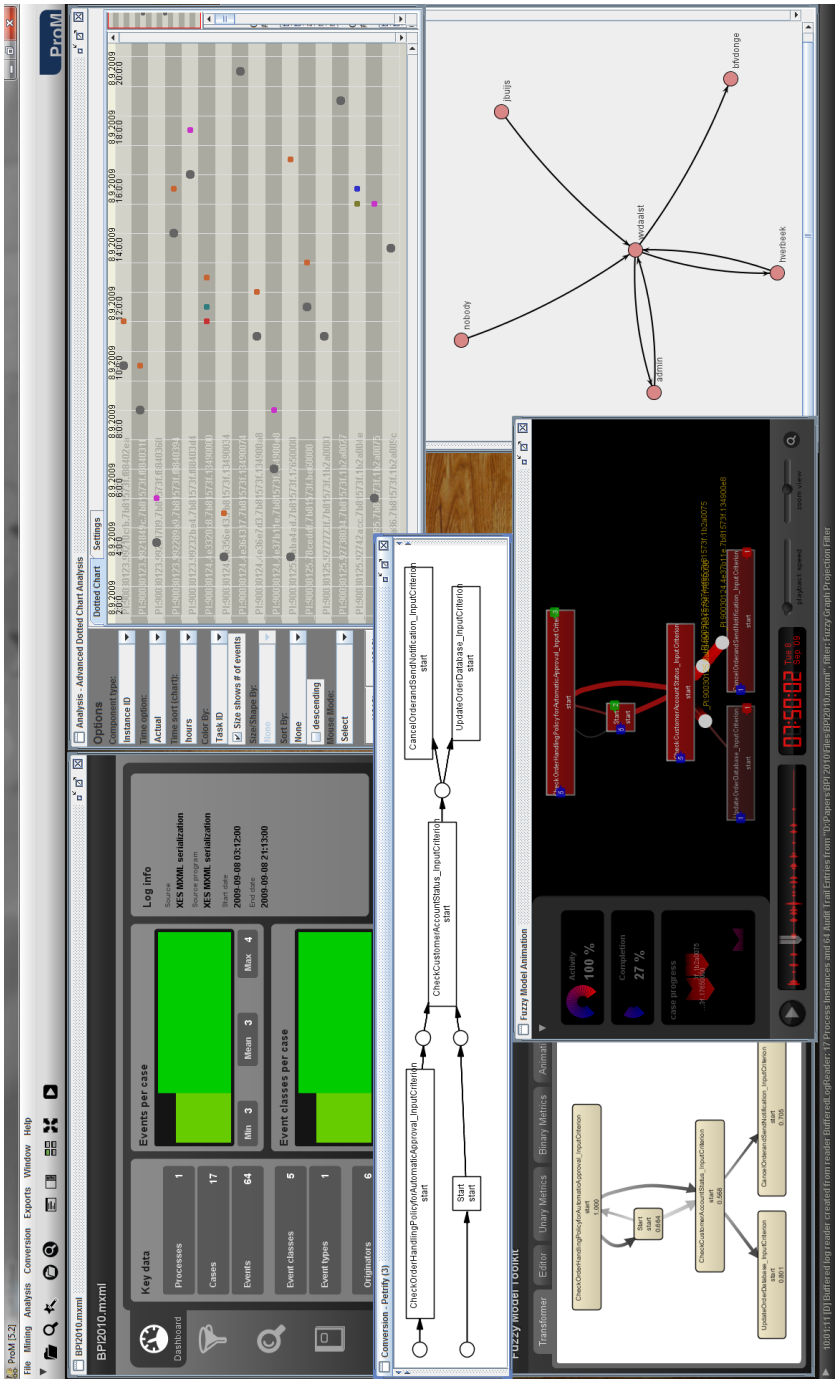

Fig. 7. Some process mining results using ProM 5.2

of process mining techniques. Figure 7 shows a screenshot of ProM 5.2 while analyzing the Clips and Tacks log. The top-left window shows some information about the event log loaded. The top-right window shows the events using a so-called "dotted chart". This visualization provides a helicopter view of the processes and the resources involved. The bottom-left part of Figure 7 shows two process models; one is discovered by the "fuzzy miner" and the other by the "FSM miner". The bottom-middle window in shows an animation of the event logs on top of the discovered model. Such an animation can be used to visualize bottlenecks, hotspots, transient behavior, etc. The lower-right window shows a social network extracted from the log. This model shows the cooperation among the resources involved.

Note that Figure 7 merely serves as an illustration of possible process mining results based on WebSphere logs. It demonstrates that through our approach process mining comes into reach for WebSphere users. Due to space constraints we cannot elaborate on the many types of analysis supported by ProM 5.2 nor illustrate our approach using a more extensive example. For more information on ProM and the various process mining techniques we refer to `www.processmining.org`.

5 Conclusions

The main conclusion of this paper is that process mining is possible in the context of the IBM WebSphere suite, as it is quite easy to configure a monitor model in such a way that the WebSphere Business Monitor (Monitor for short) will collect the information required for an event log. This configuration is achieved through addition of a new monitor context. This context corresponds to a single event instead of to a single process instance. This single-event monitor context will contain all the information of an event that is relevant to process mining.

We have tried various alternative approaches, but all failed because of the lack of a clear instance concept that is consistent throughout the whole WebSphere suite. Our findings will help developers and end-users interested in analyzing processes supported by WebSphere. The monitor based approach has been tested on various systems and example processes. Next we plan to conduct several in-depth case studies to evaluate the usability of our approach.

6 Acknowledgements

The research was supported by IBM Research & Development GmbH. The authors would like to thank Dave Enyeart, Eric Wayne, and Gerhard Pfau for their help and feedback.

References

1. 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.
2. W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32(5):713–732, 2007.
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. G.A. Chipaila. Mining Data in the Context of Semi-Structured Business Processes: Applying Process Mining in IBM WebSphere. Master's thesis, Eindhoven University of Technology, 2007.
7. IBM Corporation. Business Process Management: Modeling through Monitoring Using WebSphere V6.0.2 Products Redbook. www.redbooks.ibm.com, 2006.
8. A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.
9. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
10. B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *International Conference on Conceptual Modeling (ER 2004)*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.
11. 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.
12. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
13. 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 (Workshop on Business Process Intelligence)*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2006.
14. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.