BPAF: A Standard for the Interchange of Process Analytics Data

Michael zur Muehlen[1], Keith D. Swenson[2]

[1]Stevens Institute of Technology, Howe School of Technology Management,
Castle Point on Hudson, Hoboken, NJ 07030 USA
Michael.zurMuehlen@stevens.edu
[2]Fujitsu America Inc., 1250 E. Arques Avenue, Sunnyvale, CA
KSwenson@us.fujitsu.com

Abstract. During the initialization and execution of a process instance, multiple events occur which may be of interest to a business, including events that relate to the instantiation and completion of process activities, internal process engine operations and other system and application functions. Process mining and other analytical techniques often involve extracting this process history data from a process execution environment and submitting the data to the process analytics environment for processing. We present the Business Process Analytics Format, an XML-based interchange format for process audit events that combines an extensible state model with a robust XML representation, is able to accommodate multiple event originators and can map to the popular MXML format used in process mining applications.
Keywords: BPAF, Process Analytics, MXML, Workflow Audit Trail

1 Introduction

Business Process Management Systems support technical and human processes through the coordination of tasks, the governing of data exchanges, and the orchestration application and service invocations. These functions are governed by a formal process model, which in the majority of systems serves as the blueprint for individual process instances.[1] During the initialization and execution of a process instance, multiple events occur which may be of interest to a business, including events that relate to the instantiation and completion of process activities, internal process engine operations and other system and application functions (McLellan, 1996). Process mining and other analytical techniques often involve extracting this process history data from a process execution environment and submitting the data to the process analytics environment for processing. Many commercial BPMS provide analytics environments that are tightly coupled to their internal persistency mechanisms for process history. Such tight coupling makes for the effective design of product-specific dashboards, but limits the integration of history data with third-party analytics tools, such as the ProM process mining suite (Van Dongen et al., 2005). In addition, the integration of process

[1] We acknowledge the existence of systems that do not follow this type-instance dichotomy, but for the purposes of the subject matter discussed in this paper the distinction between design- and run-time does not limit the applicability of the BPAF format.

history data with events that are recorded outside the scope of a BPMS promises richer insights into corporate events, as the analyst is able to extend the scope of his work beyond the boundaries of a BPMS-supported process.

For the purpose of discussion, we will call the process execution environment a "process server" and the analytics environment as an "analytics server". Readers should understand that while this invokes the image of two separate unique machines communicating, other configurations are not excluded. These "servers" might both be on the same machine, they might be themselves distributed services over many machine, they might be simple programs executed under the command of a user, or any combination of these.

The remainder of this paper is structured as follows: In the next section we provide a brief overview of the history and development of process history event formats. In section three we describe the design considerations behind the Business Process Analytics Format (BPAF). In section four we detail the structure of a BPAF event. We conclude this paper with an outlook on potential next steps in the evolution of standardized process history data.

2 A Brief History of Process Audit Event Formats

The availability of process execution logs within BPMS environments can be traced back to the first commercial workflow systems of the late 1980s and early 1990s. Initially thought of as a way to enable troubleshooting and recovery of process instances in case of server failures, the use of process log files for analytics purposes was first highlighted by McLellan in 1996 (McLellan, 1996). But the overhead generated by the constant logging of event information frequently overwhelmed available computing capacity, even in mainframe environments. For example, the IBM FlowMark workflow system was capable of recording up to five different event types for each process activity instance. In scenarios where large-scale throughput was required, such as transactional workflows in the financial industry, operators had to reduce the amount of audit information recorded by switching from a verbose logging format to a condensed format that did not record every event type and omitted certain parameters for each event record.

The first attempt at standardizing process audit events came from the Workflow Management Coalition (WfMC) in 1996, when the Common Workflow Audit Data format was proposed. A revision of this CWAD format was published in 1999 (WfMC, 1999). Members of the WfMC had realized that over time organizations were likely to have more than one BPMS as part of their IT infrastructure and were proposing standards to ease the management and integration of these different systems. The purpose of CWAD was to allow for the aggregation of audit trail information from different data sources (i.e., different workflow systems that might execute parts of an overall process). CWAD specified the data structure for audit events using proprietary data types. Each audit event consisted of a prefix, a suffix, and an interchangeable body, depending on the type of event that was recorded. While some event types were predefined, naming conventions were specified so that different vendors could create extensions to the default events. The events defined in the

CWAD specification go beyond the process- and activity-based events that are the basis for process monitoring, mining and analytics applications. They include the invocation of system API functions, such as the receipt of a request for a process instance attribute value (*WMReceivedRequestGetProcessInstanceAttribute*).

CWAD never found much acceptance in commercial systems. One reason was the still significant performance degradation that workflow management systems of the late 1990s experienced through the logging of audit data. The other reason was that by the time the CWAD specification was ratified XML had become the predominant format for data exchange between Information Systems, and vendors showed little interest in implementing a specification that was not XML-based. Because CWAD was not based on XML it contained a large number of proprietary type definitions (e.g. for timestamps) that an XML-based standard would simply inherit from the default XML types. Since CWAD did not specify how audit events should be serialized it is theoretically possible to create an XML representation of CWAD events, but the technical advantages of XML such as built-in common datatypes would be lost in the process.

When the interest in process mining grew in the academic community, a shared format for audit trail information for these types of applications was proposed by van der Aalst et al. (van der Aalst et al., 2005)(Dongen and Aalst, 2005). This Mining XML Format (MXML) became the import format of choice for the popular ProM framework, and several converters for proprietary commercial formats (e.g. Staffware, Pallas Athena) are available. MXML is based on a core data structure for process events, and allows for the addition of arbitrary data structures at the event, process instance, process (model) and log level. The event types supported by MXML are based on the lifecycle of a process (or activity) instance, and focus on the transitions between lifecycle states, e.g., suspend or resume. MXML has demonstrated usefulness as part of the ProM framework but has two shortcomings. One is the limited support for structured extensions. For instance, if a system wants record the business owner of a process instance it can record this information at the instance level or the individual event level, and can use any data structure. The other is the limited number of state changes that are part of the underlying lifecycle model. For instance, the state model does not distinguish between between a graceful abort (where running activities and subprocesses are allowed to finish) and a forced terminate (where any child activities and subprocesses will be terminated as well).

As part of the EU-funded SUPER project, a semantic extension to the MXML format was proposed (Alves de Medeiros et al., 2007). The SA-MXML format links MXML entries with ontologies to allow for semantic reasoning over audit trails, but does not address the two limitations listed above.

More recently, a successor format for MXML has been proposed in OpenXES (Guenther, 2009). The OpenXES format is designed as an open event log standard that can accommodate arbitrary event types and offers defined extension mechanisms, e.g. for different lifecycle models or event log attributes.

MXML, SA-MXML and OpenXES have in common that they are oriented toward the collection of log events in a coherent trace. For this purpose, the underlying XML schemas allow for the roll-up of atomic events into instances, models, and logs. CWAD on the other hand focuses on individual events, and treats the aggregation of these events as out of scope.

3 BPAF Design Considerations

One lesson learned from the standardization of CWAD was that a mere syntactical standardization of the event format is not sufficient for the integration of audit trail data from different systems. A shared state model for processes and activities is necessary to standardize event types that can be recorded. The Wf-XML specification for the loosely coupled integration of independent processes was built on such a state machine, and its applicability for process monitoring has been demonstrated in the *Africa* prototype (zur Muehlen and Klein, 2000). While the Wf-XML state model represents a good starting point for a universal state machine for process and activity events, it is limited due to its focus on machine-to-machine interaction at the process level. In Wf-XML the states of a process instance are defined so that a client can invoke state transitions through predefined messages (Mendling et al., 2005), but it does not include states for activities, and user interaction with these activities. Suitable sources for these states are the state models that are part of the BPEL extensions for human interaction, BPEL4People (OASIS, 2008b) and WS-HumanTask (OASIS, 2008a). In basing parts of the BPAF state model on these two standards we enable BPMS that implement BPEL4People and WS-HumanTask to create a simplified mapping between their internal state machines and the states represented in the BPAF model.

3.1 The BPAF State Model

BPAF is based on one unified state machine for both activities and processes. One reason for this unification is the recognition that a process in one system may correspond to an activity in another system, i.e. there may be more than one level of processes. The other reason is that in some systems elementary activities can spawn subprocesses of their own, e.g. in BPMS that support the dynamic modification of process instances or ad-hoc diversions from a predefined process model.

At the highest level BPAF distinguishes between the two states *Open* and *Closed*. A process (or activity) is in the state *Open* if it can traverse though the state model through internal or external impulses. A process (or activity) is *Closed* if it has reached a terminal state that it will not exit on its own. It is theoretically possible that an administrator might take a terminated process instance and reopen it, but this would constitute a manual intervention. Each state is divided into a number of substates. The design consideration behind the sub-states was that different BPMS may expose a different level of fidelity with regard to the events that can be observed. It is possible that one system records events that represent state changes in another system without access to the internal workings of this system. Take for instance a web service that is invoked by a service activity within a BPMS. The BPMS might record the service as *Open* when the service is invoked, and *Closed* when the service returns a result to the calling activity. It might even record whether the service invocation was successful from a business perspective (*Closed.Completed.Success*) or whether it did not deliver the desired results (*Closed.Completed.Failed)*.

Figure 1 shows the BPAF state model. Note that the transitions shown in figure 1 are the most typical transitions in a BPMS context, but manual interventions and different system implementations may lead to additional transitions not depicted in the model.

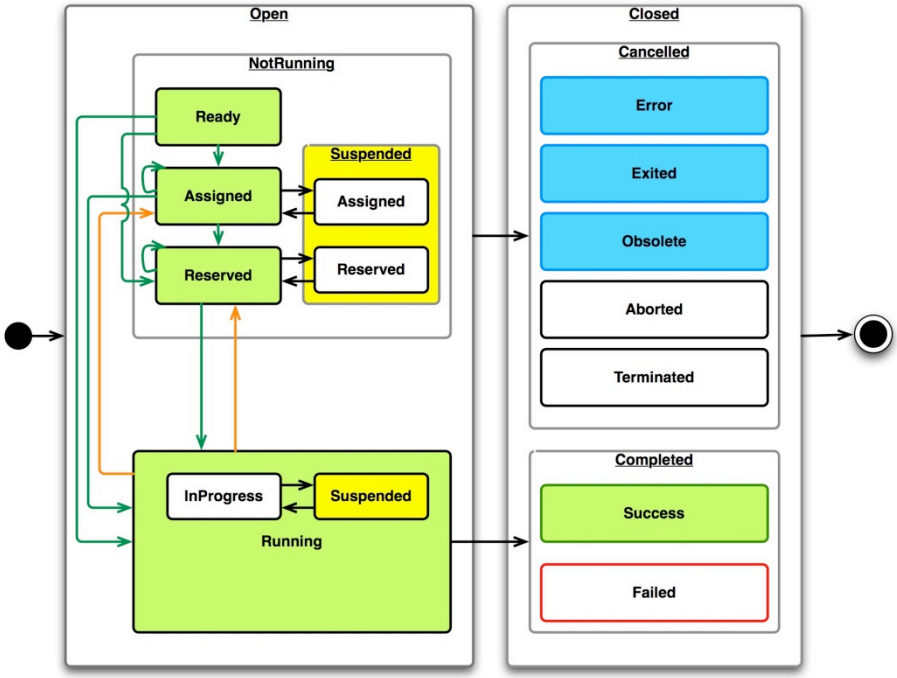

Figure 1: Business Process Analytics Format State Model (from (WfMC, 2009))

The *Open* state is divided into the two sub-states *Running* and *NotRunning*. A process in the state *Running* is actively progressing toward its objective and is consuming resources, while a process in the state *NotRunning* can be scheduled for execution, but does not progress toward its objectives. The state *NotRunning* is further divided into the sub-states *Ready*, *Assigned*, and *Reserved,* as well as *Suspended (*and its sub-states *Assigned* and *Reserved)*. These states accommodate the behavior of a BPMS with human activities that are handled through a worklist. An activity that is ready for execution may be placed on the worklists of suitable performers (*Open.NotRunning.Assigned*), and one of these performers chooses to work on the activity instance (*Open.NotRunning.Reserved*). During this time the activity instance may be barred from execution, i.e. it is moved to the *Open.NotRunning.Suspended* sub-state. Transitions between these states accommodate events such as the reassignment of activity instances (*Assigned* to *Assigned*) or the delegation of an activity from one performer to another (*Reserved* to *Reserved*).

The *Closed* state is divided into the sub-states *Cancelled* and *Completed*. The *Completed* state represents that natural end of processing for a process or activity instance. It is further divided into successful and unsuccessful completions (*Success*

and *Failed*). For example, a sales process may not lead to the signing of a contract. While the process was executed successfully, it was not successful from a business perspective. The distinction between *Success* and *Failed* allows for the quick aggregation of activity and process instances based on the different exit points that may be defined at the model level.

The *Cancelled* state summarizes all completions of process and activity instances that were premature or forced. It is divided into sub-states that detail the cause for the cancellation. These might be a forcible abort or termination, the obsolescence of a process or activity (e.g. in case of a timeout), an error condition or the manual exit from an activity instance.

BPAF assumes at the most basic level that a system will record *Open* and *Closed* states, but a system can choose to implement any number of sub-states, and may choose to extend the state model with sub-states of its own. If an analytics system is presented with a BPAF event that is based on an extended state model it can reduce the extended state until it arrives at a state it recognizes. For example, the extended state *Closed.Completed.Failure* would be reduced to *Closed.Completed* by an analytics system that does not understand the extended state model.

3.2 Mapping to MXML

MXML Event	BPAF source state	BPAF target state
schedule	NULL	Open.NotRunning.Ready
assign	Open.NotRunning.Ready	Open.NotRunning.Assigned
	Open.NotRunning.Assigned	Open.NotRunning.Reserved
	Open.NotRunning.Ready	Open.NotRunning.Reserved
withdraw	Open.NotRunning.Assigned	Open.NotRunning.Ready
	Open.NotRunning.Reserved	Open.NotRunning.Ready
reassign	Open.NotRunning.Ready	Open.NotRunning.Assigned
reassign	Open.NotRunning.Assigned	Open.NotRunning.Assigned
start	Open.NotRunning	Open.Running
suspend	Open.Running	Open.Running.Suspended
resume	Open.Running.Suspended	Open.Running
pi_abort	Open	Closed.Cancelled.Aborted
	Open	Closed.Cancelled.Terminated
ate_abort	Open	Closed.Cancelled.Aborted
	Open	Closed.Cancelled.Terminated
complete	Open	Closed.Completed
autoskip	Open	Closed.Cancelled
manualskip	Open	Closed.Cancelled.Exited
unknown	N/A	N/A

Table 1: Mapping between MXML transitions and BPAF transitions

Contrary to MXML the BPAF event model is based on the state that a process or activity instance enters at a particular point in time, not on the transition in the lifecycle model. To construct the transition information recorded by MXML an analytics system must look at the current event and the immediately preceding event that relates to the same object (i.e. activity instance or process instance). As a result, there is a 1:N mapping of MXML transitions to BPAF transitions, because the BPAF state model allows different transitions into the same state. For example, a manual activity may transition through the states *Ready* → *Assigned* → *Reserved* → *Running*, while an automated activity would move directly from *Ready* → *Running*. This means that the transition Reserved to Running and Ready to Running would both map to the MXML event start. Table 1 illustrates the mapping of BPAF state transitions to MXML events. It is evident that audit events in the BPAF format can easily be transformed into the MXML format.

4 BPAF Event Format

The BPAF event format is described as an XML Schema. Each event has a unique identifier, as there could be concurrent events with the same timestamp. In particular when events from different sources are to be integrated the ability to distinguish between the different events is critical. Each event contains references to a process definition (i.e. the underlying process model) and a process instance, as well as the state that the process or activity instance entered. Additionally, the names of process and activities can be recorded. This element is useful when audit events are aggregated over longer periods of time. Since modifications to process and activity definitions will typically result in different process and activity IDs, the availability of names simplifies the correlation of similar processes and activities, even if the underlying models have changed.

Optional elements of the BPAF schema are elements related to activities (if the system distinguishes between activities and processes), the preceding state of the process or activity instance (in order to ease the identification of transitions) and an extension mechanism for arbitrary data elements. This extension mechanism will typically be used to record key attributes of a process instance in order to provide a link to business data, or it can be used to link an event to an originator, such as the performer of the current activity. While the availability of an activity performer ID is a common requirement for the analysis of process logs (e.g. for the analysis of social networks based on process log data), privacy laws and/or union agreements may prohibit the recording of personally identifiable information in an automated log file, thus the BPAF standard recommends how this information can be recorded as part of the extended elements, but does not mandate it.

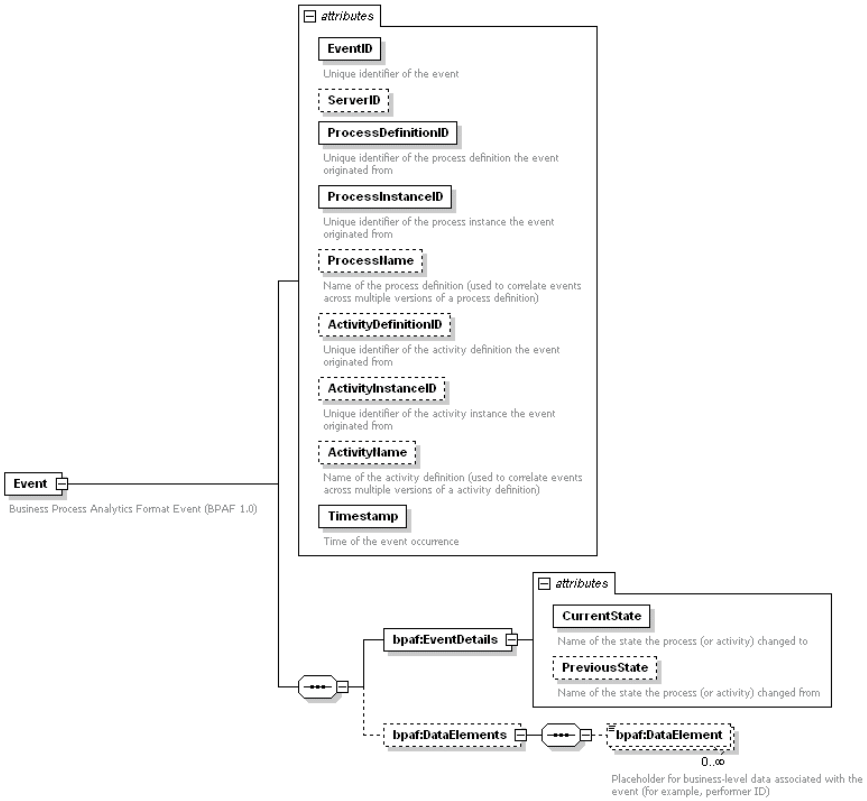

Figure 2: BPAF XML Schema (from (WfMC, 2009))

5 Summary and Future Directions

This paper introduced the Business Process Analytics Format, illustrated the design rationale behind its features, and discussed its relationship to the popular MXML format. BPAF was built on the lessons learned from the WfMC CWAD format and takes into account the developments of Wf-XML, BPEL4People and WS-HumanTask.

The purpose of BPAF is to enable data interoperability between the events generated by BPMS and different process analytics platforms. These platforms range from Process Mining applications to monitoring dashboards, Business Intelligence systems, simulation platforms and Complex Event Processing systems. We do not assume a single process server and a single analytics server. In practice there is often a many-to-many relationship: Analytics servers need to consolidate events from multiple process servers, while a process server may need to fan-out events to multiple analytics servers. Full interoperability among these tools requires a single common format for the representation of events. A standard format is important for long term archival

of case histories, and for making those histories readily accessible. We need interoperability not only at a given instant of time, but over long spans of time as well.

Events might be extracted from a process server that includes months or years of historical event data to be processed in a single batch job. Alternately, the process server may be delivering information about an event to the analytics server in near-real-time as a stream. Both batch mode and stream mode should be supportable through the analytics format. Although BPAF focuses on individual events, it is designed to allow for the easy aggregation of information to the process level. However, it does not contain the aggregation structures present in MXML. This, and the proposed OpenXES format are areas of possible convergence between BPAF and the formats in use in the Process Mining community.

References

A.K. Alves de Medeiros, W.M.P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral., "An Outlook on Semantic Business Process Mining and Monitoring," *4806*, 2007, Springer, pp. 1244-1255.

B. F. v. Dongen, and W. M. P. van der Aalst, "A Meta Model for Process Mining Data," *2*, Porto, Portugal, 2005, pp. 309-320.

C. Guenther, "OpenXES Development Guide," *1.0 RC5*, 2009, http://code.deckfour.org/xes/

M. McLellan, "Workflow Metrics - One of the great benefits of workflow management," in *Praxis des Workflow-Management*, Vieweg, Braunschweig, 1996, pp.301-318.

J. Mendling, M. zur Muehlen, and A. Pierce, "Standards for Workflow Definition and Execution," in *Process-Aware Information Systems. Bridging People and Software Through Process Technology*, John Wiley & Sons, Inc, Hoboken, NJ, 2005, pp.281-316.

OASIS, "Web Services - Human Task (WS-HumanTask) Specification," *1.1 Working Draft*, 2008a, oasis-open.org.

OASIS, "WS-BPEL Extension for People (BPEL4People) Specification Version 1.1," *1.1 Working Draft*, 2008b, oasis-open.org.

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," 2005, Springer, pp. 130-147.

B. F. Van Dongen et al., "The ProM framework: A new era in process mining tool support," 2005, Springer, pp. 444-454.

WfMC, "Audit Data Specification. Version 2," Document Number WFMC-TC-1015, 1999, Workflow Management Coalition, Winchester, UK.

WfMC, "Business Process Analytics Format - Draft Specification. Document Number TC-1015," *1.0*, 2009, Workflow Management Coalition, Cohasset, MA.

M. zur Muehlen, and F. Klein, "AFRICA: Workflow Interoperability based on XML-messages," Stockholm, Sweden, 2000.