2012 45th Hawaii International Conference on System Sciences

Extending BPMN for Business Activity Monitoring

Jan-Philipp Friedenstab*, Christian Janiesch†, Martin Matzner* and Oliver Müller‡
*University of Muenster, European Research Center for Information Systems (ERCIS), Münster, Germany
jp.friedenstab@uni-muenster.de, martin.matzner@ercis.uni-muenster.de
† Karlsruhe Institute of Technology (KIT), AIFB, Karlsruhe, Germany
Email: christian.janiesch@kit.edu
‡University of Liechtenstein, Hilti Chair of Business Process Management, Vaduz, Liechtenstein
Email: oliver.mueller@uni.li

***Abstract*—Real-time access to key performance indicators is necessary to ensure timeliness and effectiveness of operational business processes. The concept of Business Activity Monitoring (BAM) refers to the observation, analysis, and presentation of real-time information about business activities across systems' and companies' borders. Designing and maintaining BAM applications is challenging, as the involved concepts (e.g., business processes, audit logs, performance measures) — though being strongly interrelated— are developed by different communities of practice. Also, they reside on different levels of abstraction, and are handled by different IT systems. Hence, we developed a conceptual modeling language which extends the widely accepted Business Process Modeling Notation (BPMN) by BAM-relevant concepts. The main results presented in this paper are: (1) a meta-model which formally describes the conceptual aspects of the developed BPMN extension (abstract syntax); (2) graphical symbols as an exemplary representation of this abstract syntax (concrete syntax); (3) a demo scenario that illustrates the application of the language in a fictitious scenario.**

I. Business Activity Monitoring

The term Business Activity Monitoring (BAM) was coined by the Gartner Group who referred to BAM as a tool for providing real-time access to business performance indicators to enhance the speed and effectiveness of operations. One of the main goals of BAM is to minimize the latency [3] of decision making by providing support for immediate operational decisions and for taking adequate actions [2] based on real-time information [20]. While Gartner's approach to BAM is rather broad, we have a more focused understanding of BAM, which concentrates on business processes (instead of general activities).[1] Accordingly, we consider real-time monitoring and control of business processes at runtime to be the essence of BAM.

A key concept of BAM is the aggregation of basic events about the current state and the results of business processes into quantitative measures, so-called Key Performance Indicators (KPIs) [1]. KPIs are typically derived from higher level business goals [19]. They can comprise generic metrics that are applicable to any process (such as process duration), or process-specific measures, that are typically based on the properties of process-relevant business objects [8]. The metrics can be calculated for one single process instance (case) or may be aggregated over several process instances to compute, e.g., average or minimum / maximum values with regards to a specific process definition [19]. Since information provided by BAM systems is targeted at managerial users, both an appropriate level of abstraction and a clear presentation of KPIs are important. Therefore, BAM applications typically resort to dashboards — i.e., visual displays of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen so that the information can be monitored at a glance [15].

According to this scope, the development of BAM applications can be generally divided into three tasks: the modeling of business processes, the specification of event processing logic which implements the actual monitoring and control functionality, and the design of dashboards which visualize KPIs. As these tasks are typically executed by different people, reside on different levels of abstraction, and are performed within different software systems, a number of challenges arise, especially with respect to usability and practicability.

The greatest challenge, according to our experience, is the lack of a coherent view on the overall BAM application and a clear awareness for the interrelations between the different design artifacts. In particular, there is a gap between the process models which are defined in BPM systems and the event processing models which are defined in the BAM system. To counter for these deficits, we developed an extension of the widely accepted Business Process Modeling Notation (BPMN) 2.0 specification that allows for integrating BAM-relevant concepts within business process models. The main results presented in this paper are: A meta-model that together with a textual description of the constructs semantic defines the conceptual aspects of the BPMN extension (Sec. 2.1), graphical symbols as an exemplary representation of this abstract syntax (Sec. 2.2), and a demo scenario that demonstrates the application of the language (Sec. 3). We

[1] DeFee and Harmon also state that BAM is a misleading term and the emphasis should have been on Business Process Monitoring. They assume that "Activity" was used instead to create a new acronym, since BPM referred to Business Process Management [4].

978-0-7695-4525-7/12 $26.00 © 2012 IEEE
DOI 10.1109/HICSS.2012.276

exhibit the novelty of our approach by a discussion against the background of related work (Sec. 4). We close with a conclusion and an outlook for further research (Sec. 5).

II. BPMN-EXTENSION FOR BAM

A. Abstract syntax and semantics

The abstract syntax of a modeling language defines the set of available language constructs and specifies the allowed relationships between them. It can be formally described by making use of a language-based meta-model [9]. The semantics of a modeling language, in contrast, refer to the meaning of the provided constructs in the context of the application domain and are usually provided in form of a natural language textual description [18]. Together, both are referred to as the conceptual aspects of a modeling language. Accordingly, in the following the conceptual aspects of the proposed BPMN extension are outlined using UML class diagrams and textual descriptions. Some of the models include concepts from the BPMN specification which are gray colored. They have primarily been derived from the current BPMN 2.0 specification. Additionally, we sometimes refer to the BPMN state change model which describes the general states a process activity and/or business process traverses through [20].

Duration. The measurement of process-related time spans on different levels of detail is a common task in BAM settings. To address this requirement, the language provides the *Duration* construct (Figure 1).

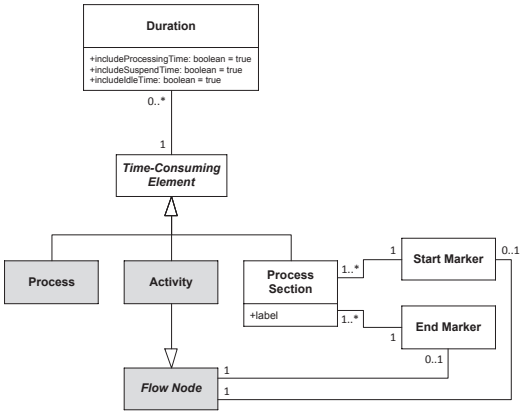

Figure 1. Meta model: Duration

A *Duration* object always references exactly one *Time-Consuming Element* which constitutes the basis for measuring the elapsed time. The latter is modeled as an abstract superclass of *Process*, *Activity*, and *Process Section*. The semantics of these sub-classes can be described in further detail:

The *Process* class stems from the BPMN specification [14] and hence represents a complete business process. If a *Duration* object is associated with a *Process* object, this implies —by default— that the period between process start (its state first changes to 'Running' according to the BPMN state model) and that moment where it is finished (the state changes to a sub-state of 'Closed') is to be measured.

The *Activity* class (from the BPMN specification) represents a unit of work within a business process. For the measurement of the duration, analogous state-based semantics as for a *Process* are valid. However, in case of an activity performed by a human, the measured duration will by default also include the time in which the *Activity* is in state 'Assigned' (i.e., idle time).

The *Process Section* element facilitates flexible definition of time periods to be measured, bridging the gap between measuring single *Activities* and whole *Processes*. A Process Section can be described by label attributes and is defined by one *Start Marker* and one *End Marker* which are in turn associated with *Flow Nodes* from BPMN. Thus, the duration of arbitrary sub-sections of a business process can be measured.

With these three different types of reference objects, it is possible to conveniently specify duration measures on different levels of granularity. Furthermore, the attributes of the *Duration* class enable a more differentiated measurement of time spans, since they determine whether the time spent in specific execution states should be included in the measure or not. As indicated above, the total turnaround time —i.e., the sum of the actual processing time as well as possible idle and suspend times— will be measured by default. This is expressed by the default value 'true' for the three attributes includeProcessingTime, includeSuspendTime and includeIdleTime. However, if not the total duration of a *Time-Consuming Element* is of interest, but only a sub measures (or a combination of them), the attribute values can bet set accordingly.

Frequency. Apart from measuring the duration of a business process (or a part of it), another typical task is to count the number of times something has happened within the course of a process. E.g., one might be interested in how often an activity has been executed or how often a specific process instance has been suspended or resumed. Here, the *Frequency* element is used (Figure 2). It counts different types of occurrences within the scope of one single process instance.

A *Frequency* object is always associated with exactly one *Countable Element*. Equivalent to the specification of the *Time-Consuming Element* in the context of the Duration measure, the Countable Element is an abstract class that expresses the commonalities of its subclasses and aims at simplifying the meta-model. In particular, the following two elements are considered to be countable and can thus be referenced by a *Frequency* object:

An *Activity* from BPMN can be used to define a basic counting mechanism: The associated Frequency measure will comprise the number of times the specific *Activity* has

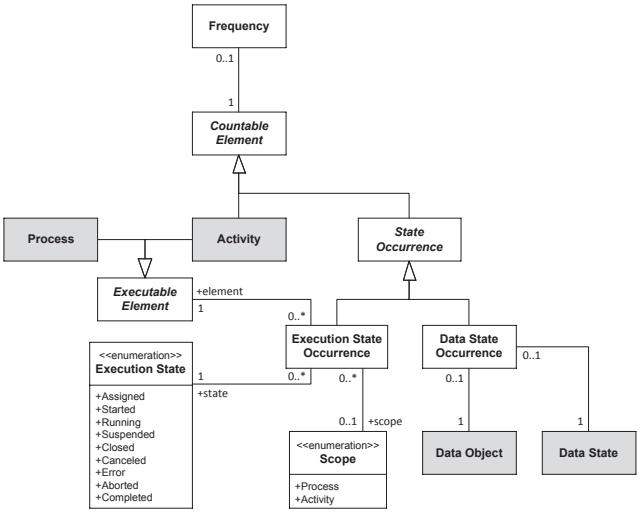

Figure 2. Meta model: Frequency

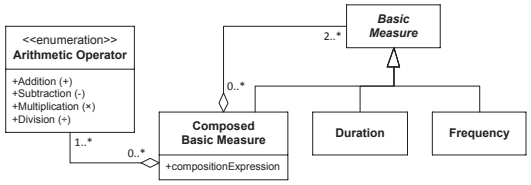

Figure 3. Meta model: Composed Basic Measure

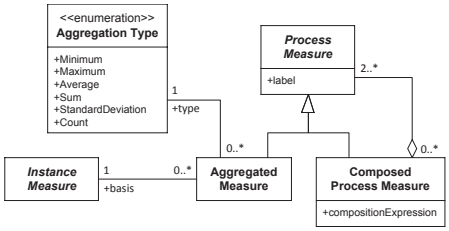

Figure 4. Meta model: Aggregated Measure

been successfully executed within a single process instance (in case of loops in the control flow of the process).

The *State Occurrence* class (respectively its two subclasses) provides a more generic concept to define a *Frequency* measure, based on the number of times a specific state occurs during process execution. The first subclass named *Data State Occurrence* can be used to count how often a BPMN *Data Object* passes through a specific *Data State*. In contrast, the second subclass, *Execution State Occurrence*, refers to a *Process* or an *Activity* as an *Executable Element* and a particular *Execution State* (the listed states in the enumeration class are derived from the BPMN execution state model). The semantics of an *Execution State Occurrence* object (and its associated Frequency measure) differ depending on the type of the element it references: If an *Activity* is referenced, the measure comprises the number of times that particular *Activity* resides in the specified state (one might, e.g., ask how often a particular activity has been reassigned). If a *Process* is referenced the *Scope* of the measure has to be additionally specified to avoid ambiguities: Here, '*Process*' means that the associated *Execution State* represents the process instance state; *Frequency* refers to the question of how often the process has been in the particular state (e.g., "How often has the process been suspended?"). '*Activity*' implies that the state is applied to all *Activities* of the *Process* (to answer questions such as: "How many activities of the process have been aborted?").

Composed Basic Measure. It is possible to recursively define complex measures based on two or more basic measures, resulting in a *Composed Basic Measure* (Figure 3). A *Composed Basic Measure* comprises of a compositionExpression that defines how the related measures are to be combined by arithmetic operations, i.e., addition, subtraction, multiplication, or division (one example for such a composed measure might be the percentage of the duration of a single activity with respect to the duration of the whole process).

In many BAM scenarios, expressing measures on instance-level is not enough. Rather, most KPIs constitute process-level measures that aggregate data from many instances [5]. To cater for the definition of such measures, the BAM modeling language provides the construct *Aggregated Measure* (Figure 4), which represents a single meaningful value related to one process.

Aggregated Measure. Each *Aggregated Measure* references exactly one *Instance Measure* which is the basis for the aggregation and provides the connection to the process (or process element) to which the *Aggregated Measure* refers to. An *Aggregated Measure* is linked to an *Aggregation Type* which determines how the instance-level values are to be condensed. The following types are possible: Extreme values can be selected ('Minimum' and 'Maximum'), aggregated metrics can be calculated ('Average', 'Sum', and 'StandardDeviation'), or the values are counted ('Count').

An *Aggregated Measure* is a specialization of a *Process Measure* which, like *Instance Measures*, can also be combined by arithmetic operations. For this purpose, the *Composed Process Measure* is provided (e.g., the number of successfully completed processes with respect to the overall number of closed processes). The class is specified like the *Composed Basic Measure* class introduced above and thus contains a compositionExpression which defines how two or more measures are to be composed. It is itself a subclass of the *Process Measure* so that a *Composed Process Measure* can be used in another composition again.

Filter. Measures may only be defined over a subset of the available process instances. E.g., one might be interested in the average duration of the last ten instances only or might want to exclude instances that do not match a particular

condition (e.g., instances that did not complete successfully but were canceled should not be regarded). Therefore, the BAM modeling language provides the *Filter* (Figure 5) which can be applied to restrict the measurement basis for a particular measure.

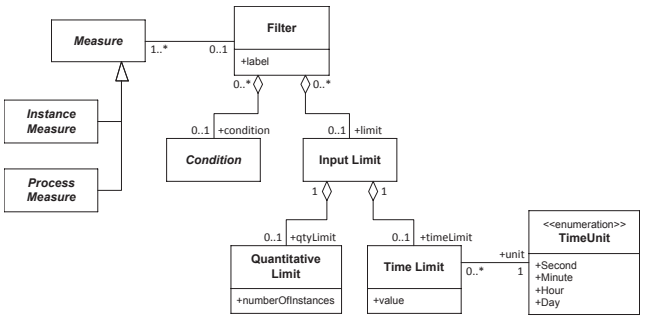

Figure 5. Meta model: Filter

Both *Instance Measures* and *Process Measures* —which are subsumed to the generic class *Measure*— can optionally be associated with a *Filter*. Each *Filter* definition can contain two components: a *Condition* or an *Input Limit*. At least one of both has to be specified. A *Condition* expresses that only instances for which the condition holds are to be considered. Concerning the type of condition no limitations exists, so that all of the above-described Conditions may be used.

An *Input Limit* comprises two different constructs for restricting the amount of process instances to be included. First, the *Quantitative Limit* states that only a limited quantity of the recent instances (set by the attribute numberOfInstances) is supposed to be used (e.g., only the last ten instances of a particular process). Second, the *Time Limit* defines a time window —reaching backwards from the moment of analysis— in which the process instances have to be completed to be considered. Time windows are configured by the attributes value and unit. Both limits can also be combined. In this case, the limit that is reached first is decisive.

Target definition. A common requirement in BAM is to specify target or threshold values for particular KPIs. Therefore, the BAM modeling language provides the concept of a *Target Definition* (Figure 6). In general, each Measure can have an associated *Target Definition*. A *Target Definition* is composed of two components: The specification of the actual target and an optional specification of deviations:

The target attribute is of the type *Range*. A *Range* represents the domain the value of the respective *Measure* is allowed or supposed to be within. Its limits are defined by the lowerBound and upperBound attributes. Both attributes are marked as optional to allow for the specification of (one-sided) unbounded ranges. At least one of them has to be set. If both are set, the upperBound has to be greater than the lower-Bound.

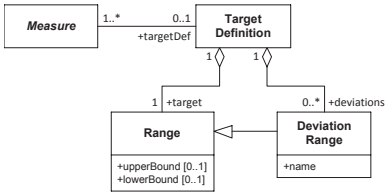

Figure 6. Meta model: Target Definition

The semantics of a *Range* definition are as follows: If only one of the attributes is set, the value of that particular attribute represents a threshold value for the respective *Measure*. Depending on whether the lowerBound or the upperBound is specified, the threshold either constitutes a minimum or a maximum threshold. E.g., if the average duration of a process should not exceed 24 hours, the upperBound has to be set accordingly. In contrast, the lowerBound attribute would be set if one wanted to express that the percentage of successfully finished processes should be at least 90 percent (minimum threshold). Finally, if both attributes are specified, an explicit target range is defined instead of discrete threshold values.

The second component of a *Target Definition* is the optional specification of *Deviation Ranges*. A *Deviation Range* specifies a critical sector of values that represents a deviation from the intended target range. With respect to the degree of deviation, a *Target Definition* may contain several Deviation Ranges (which might also be the basis for different kinds of reactions).

Actions. *Target Definitions* are only of limited use if used solely. Rather, it is often required to automatically initiate reactions if a particular measure is out of a value range. Here, the BAM modeling language provides the concept of an *Action* which is syntactically specified by the meta-model presented in Figure 7.

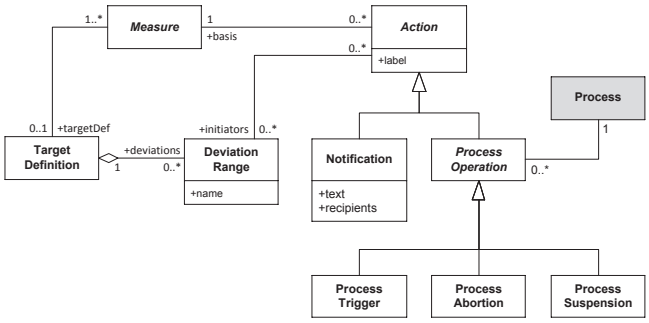

Figure 7. Meta model: Actions

An *Action* element —that can be described by its label attribute— is always associated with one *Measure* which constitutes the basis for the action to be triggered. A *Measure*, in turn, can cause several *Actions*. Generally, an *Action* is initiated every time the value of the associated

Measure is updated, regardless of the value. However, to be able to specify that an *Action* is only initiated in case of deviations from a target value, it can be associated with one or more *Deviation Ranges*. It has to be considered that only those *Deviation Ranges* can be used that are defined for the *Target Definition* of that *Measure* the *Action* refers to.

The *Action* class itself is modeled as an abstract class and is a generalization of the *Notification* and the *Process Operation*:

A *Notification* can be used to inform responsible business users or decision makers by sending a message. Therefore, each *Notification* object has a message text and a list of one or more recipients. A recipient can be a concrete person, an organizational unit or a role which is assigned to several persons. The concrete communication channel is abstracted from by the modeling language.

A *Process Operation* is an abstract super-class for different types of actions related to business processes. A *Process Operation* references exactly one *Process*. The following operations are possible: A *Process Trigger* is used to start a new instance of the associated *Process*, the *Process Abortion* represents the termination of a process instance, and the *Process Suspension* can be used to pause a process instance. While the *Process Trigger* is applicable to all *Measures*, the use of the other two *Process Operations* is restricted: They can only be associated to an *Instance Measure*, since in case of a *Process Measure*, it could not be deduced which of the instances that contributed to the aggregation should be aborted or suspended.

Apart from the described *Process Operations*, the BAM modeling language needs to support the manipulation of the control flow of a process instance. Here, the existing possibilities of BPMN —that includes condition expressions for *Sequence Flows* which determine if that particular process path is executed or not— are enhanced. A BPMN Expression is enabled to include one or more *Measures* defined in the BAM modeling language (Figure 8). Thereby, it is possible to activate a specific *Sequence Flow* based on externally calculated metrics (e.g., for redirecting a process if the duration of a preceding part of the process has exceeded a certain threshold).

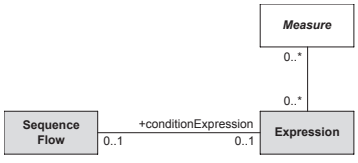

Figure 8. Meta model: Measure-based Expressions

Finally, the BAM modeling language comprises a *Dashboard* element (Figure 9) which is used to specify which of the defined measures are to be visualized. As a process dashboard typically comprises an overview of several related KPIs, a *Dashboard* element can reference one or more *Measures*. In turn, a *Measure* can be part of several *Dashboards* or it is not associated with a *Dashboard* at all. Finally, a *Dashboard* can be further described by its label attribute.

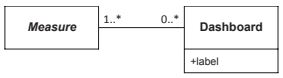

Figure 9. Meta model: Dashboard

B. Concrete syntax

In order to create graphical models with a particular modeling language, the concrete syntax has to be specified by providing visual representations (i.e., graphical symbols and textual labels that concrete their behavior). The notation of the proposed modeling language is presented in Figure 10.

III. Demo Scenario

We introduce a fictitious 'Purchase Order Process' process to demonstrate our BAM modeling approach. We take the perspective of a manufacturing company that produces products on stock and operates in a business-to-business supply chain environment. Under these conditions —esp. if the company's customers produce just-in-time and their orders are thus time-critical— the monitoring and control of the order handling process is important.

A. Preparatory phase

Model business process. First, the BPMN model of the purchase order process was captured (Figure 11, bottom): Having received an order from a customer, the order is analyzed (i.e., check availability of items check delivery date, and check customer's credit). If the order is accomplishable as specified by the customer, a confirmation message is sent. If the order cannot be directly fulfilled the responsible person has to take further action (not exhibited). If an order becomes confirmed, the order has to be processed (i.e., picking goods, packing, preparation for shipment). Then goods are shipped, either with a standard carrier or an express carrier (faster, more expensive). Finally, the reception of the goods at the customer is acknowledged.

Define requirements for BAM. From the various conceivable requirements to an order processing BAM system, the fictive company decides to first focus on time-related measures in order to assess the performance of its purchase order processing: Here, the average cycle time of the complete order process (from order reception to successful delivery) is an important KPI, since the company assures its customers of a delivery time of 48 hours. In case of deviations from this target, a notification should be sent to the process owner. The measure should also be part of a process dashboard and is supposed to direct the flow of a process instance: If the average cycle time

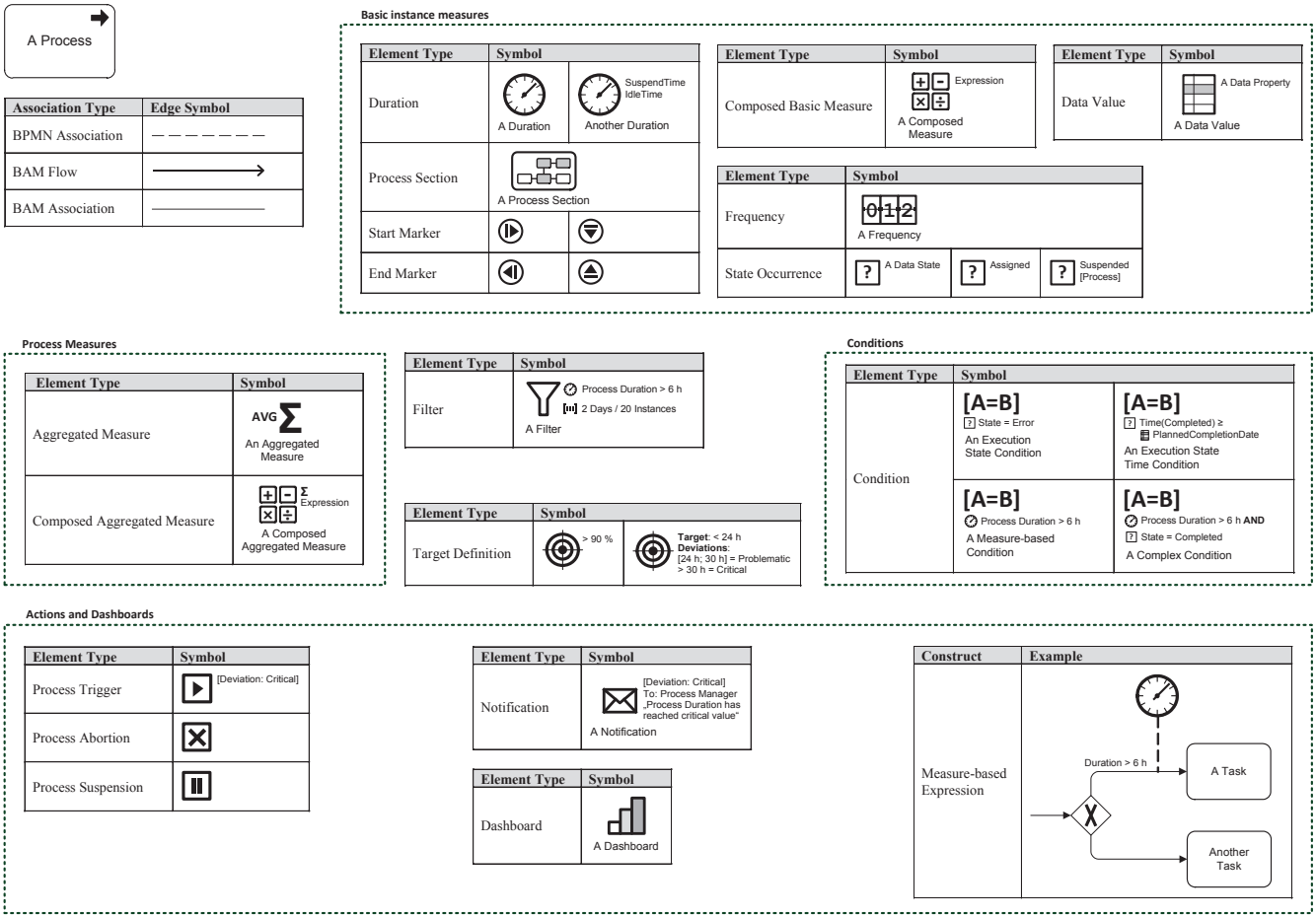

Figure 10. Concrete syntax of the BAM modeling language

is above the threshold of 48 hours, an express shipment should be initiated automatically. Further, only processes which have completed successfully should be considered for the calculation of the average cycle time so that the values are not distorted by canceled processes. Besides the cycle time of the whole process, the average percentage of the company-internal part of the process (which does not include the shipment) with respect to the overall cycle time is required. The process managers are also interested in a visualization of the cycle times of the last 50 orders to identify current trends. These values are hence not supposed to be aggregated.

B. BAM modeling phase

Each of the desired KPIs or actions has to be transferred into the respective constructs of the BAM modeling language (Figure 11). I.e., for each of the KPIs, one has to first determine the Basic Measure(s) the KPI is based on as well as the BPMN elements they refer to. Subsequently, Aggregated Measures are modeled, Filters and Target Definitions are specified and, finally, Actions and Dashboards are defined.

(1) Average cycle time. The average cycle time of the order process describes a time span; therefore a Duration measure (as a concrete type of a Basic Measure) is selected. It is attached to the Process element for the 'Purchase Order Process', as the complete duration of the processes instances has to be considered. The Duration is connected with a Filter containing an Execution State Condition to express that only successfully completed process instances are relevant. Since not the cycle times of single process instances are of interest but the average value, the Duration element has to be followed by the respective Aggregated Measure. The latter is in turn supplemented by a Target Definition, because a threshold value of 48 hours has been defined for the average cycle time. To account for the information needs of the process owner, the Aggregated Measure is succeeded by a Notification element. Moreover, it is connected with a Sequence Flow of the BPMN process model, to initiate the express shipment. The Aggregated Measure also has an outgoing connection to a Dashboard element.

(2) Average percentage of the company-internal part of the process with respect to the overall cycle time. A

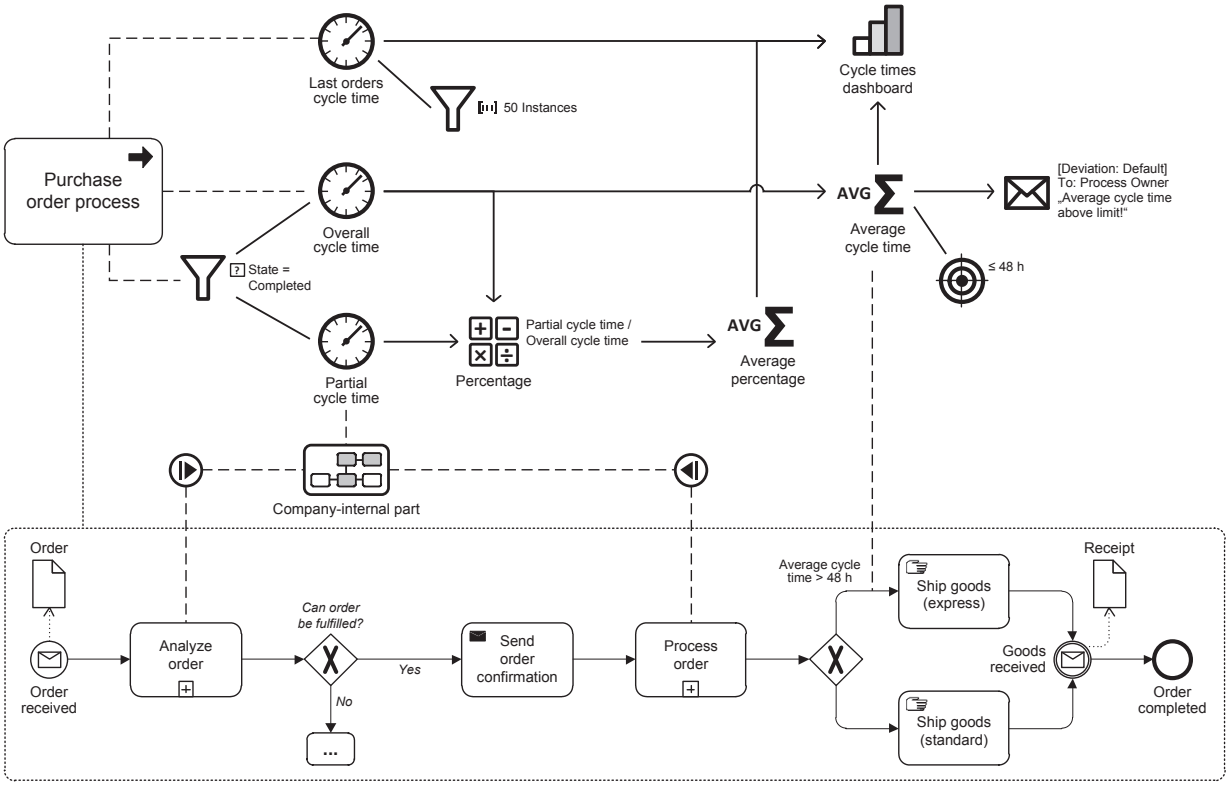

Figure 11. BAM model: cycle times

Process Section is needed to describe the company-internal part of the business process. In particular, the Start Marker is attached to the 'Analyze Order' Sub Process, while the End Marker is connected with the 'Process Order' Sub Process. To measure the cycle time of this Process Section, it is connected with a Duration measure which reuses the above-described Filter to include only successfully completed processes. Though, not the cycle time of the Process Section itself, but its percentage with respect to the cycle time of the overall process is required. Therefore, a Composed Basic Measure has to be applied that relates both measures to each other. Finally, this measure is aggregated and connected with the cycle times Dashboard.

(3) Cycle times of the last 50 orders. For the third time-related measure, another Duration element is necessary. This has to be supplemented by a Filter element that specifies a Quantitative Limit, since only the last 50 process instances should be regarded. As the cycle times are supposed to be visualized, the Duration element is also connected with the above mentioned Dashboard.

IV. Review of Related Approaches and Discussion

When proposing a new modeling method, there is a need to demonstrate its "worthiness" against the background of the "blooming production" of modeling methods [17]. We therefore subsequently review the state-of-art in BAM modeling and will then discuss our approach against the identified body of methods.

A. Review of related approaches

Del-Río-Ortega, Resinas and Ruiz-Cortés strive for a better integration of Process Performance Indicators (PPIs) into the business process lifecycle (definition, execution, analysis). PPIs should be modeled together with the business processes. Thus, they present an ontology to define PPIs which comprises a comprehensive classification of PPIs and explicitly defines how the PPIs are related to elements of a BPMN business process model (e.g., data objects or activities) [5]. Due to the clear orientation to BPMN, the authors provide a valuable contribution for closer integration of business process modeling and monitoring aspects. However, while a connection between BPMN constructs and PPIs is established, a graphical notation to include the measures into process models is missing. Further, the sole definition and modeling of process metrics is of limited value in a BAM context as threshold values and the corresponding exception handling mechanisms need to be expressed as well.

A related approach that connects process KPIs and process models is provided by Wetzstein, Ma and Leymann who propose a semantic framework for BAM which aims

at increasing the degree of automation in measuring KPIs [19]. The key concept of the framework is to enable business users to model KPIs by referencing semantic annotations of business processes. These annotations explicitly specify the semantics of process activities with respect to state changes of process-related business objects which in turn have to be described in domain ontologies. For KPI specification, the authors also present a KPI ontology which, though, is not as differentiated. However, it expresses that process metrics are calculated on the basis of events triggered during the execution of a business process and, thus, accounts for the event-driven character of BAM systems. The corresponding IT-level event-based models that specify how KPIs are computed are automatically generated based on the business-level KPI models. The authors do not only achieve a closer integration of process modeling and monitoring concerns, but also tackle the problem that event-driven BAM solutions require the manual development of complex technical models. However, their approach is rather specific as it is only applicable in the context of semantic BPM and requires the corresponding annotation of business processes as well as the design of domain ontologies. The KPIs are specified in form of text-based formulas [19, p. 231] without graphical modeling support. In contrast, the presented approach aims at providing a generally applicable modeling language that provides a graphical notation to define KPIs.

Such a graphical notation is included in the work of LIST and KORHERR who pose the problem that prevalent business process modeling languages do not explicitly support the modeling of process goals and performance measures [11]. The authors extend the meta models of the EPC as well as BPMN and provide notation elements for including performance aspects into process models. The aim is to make goals and measures —which are classified into quality, cost and process cycle time measures— conceptually visible [10]. The proposed approach fits into the BAM context, as the performance measurement of business processes is a crucial aspect of BAM systems. It further addresses the strict separation between process modeling and process monitoring and measurement. However, it also exhibits some limitations: The strict classification of measures into three categories and the restriction to cycle time measurement on a process instance level [9, p. 292] hampers a flexible definition of other types of performance metrics (e.g., average process runtime). The approach supports only limited possibilities to define actions that should be taken if measures exceed or deviate from their target values. Finally, it has a clear conceptual focus and does not support event-based calculation of measures.

GONZÁLEZ, CASALLAS and DERIDDER introduced a language to supplement business process models with monitoring, measurement and control (MMC) aspects [6], [7]. Process metrics and rule-based actions that may trigger notifications or control the execution of a process can be defined. The authors' goal is to enable MMC specifications from a business perspective at the process definition level, considering the specific concepts of the modeled domain. Explicit reference to elements from a BPMN process model is supported. A closer integration of process modeling and event-based monitoring is facilitated. Though, the basic intention of providing a high-level, business-oriented language is a bit thwarted, as the developed language is a textual, declarative one: This development style is probably not very intuitive and user-friendly for business analysts who are used to graphical process modeling languages.

Further work on the integration of process modeling and monitoring aspects comprises the following: MOMM, MALEC and ABECK apply the principles of a Model Driven Architecture (MDA) to the development of process monitoring systems. They propose to combine the BPMN meta model with a —rather simple— meta model for the specification of PPIs that should allow for a stepwise generation of executable process models extended by monitoring functionality [13]. Graphical modeling of KPIs is not touched. LIU ET AL. aim at turning performance monitoring into an integrated part of process modeling by also following an model-driven approach [12]. A performance monitoring model is automatically derived from extended process models centered on business artifacts which define so-called monitoring contexts. Process events are automatically matched with monitoring contexts to calculate process metrics. Finally, COSTELLO and MOLLOY present an approach for incorporating event definitions, performance metrics and thresholds into (XML-based) process models to be used in a BAM setting [2], [3]. The authors take account of the need for a closer integration of process modeling and BAM. However, their XML schema definition is rather restrictive, since it does not allow for a flexible definition of process metrics, but only supports a limited set of them.

B. Discussion against the related work

Against the presented previous research contributions, we summarize that to our best knowledge a high-level graphical modeling language that is specifically tailored to cover the full spectrum of BAM applications and enables an integration with business process models has previously not been proposed. Our approach, though, aims at closing this research gap. In particular, the new language contributes to the research field by addressing the following objectives:

Comprehensive feature set: The presented language covers the entire functional range of BAM systems. This implies that, apart from modeling different types of process related KPIs and defining notification and alerting mechanisms, it is especially supposed to express process interaction functionalities. Hence, it is targeted towards the specification of both process monitoring and control aspects.

Explicit modeling of BAM functionality: The modeling language is purpose-specific —instead of being a multi-

purpose, generally applicable one— that offers explicit features for the BAM application context. It thus is able to directly express all BAM functionality by providing corresponding language constructs with easily interpretable semantics. Thereby, a good comprehensibility of the resulting models is ensured.

Integration with BPMN: To overcome the strict separation of process definitions and specification of BAM functionality, the presented language provides connection points to BPMN to allow for a better integration and combination of the models. The modeler can directly specify dependencies between business processes modeled in BPMN and the corresponding BAM operations. It is traceable, e.g., which processes or process activities are used for the calculation of KPIs, on which basis escalation processes are triggered or where a process control flow is influenced. The language hence facilitates a holistic view on BAM applications. Thus, it allows for easier identification of necessary adaptations in BAM models in case of changes in the business processes. The risk of inconsistencies between different design artifacts is reduced, which positively influences the maintainability and facilitates a more frictionless implementation of BAM scenarios.

User-friendly graphical notation: The presented language provides a graphical notation for the specification of BAM functionality. The language's claim is hence to be intuitively usable for non-technical users and process modelers that are typically familiar with graphical modeling languages. A graphical notation, compared to a textual programming language, allows for clearer identification of the interrelations between the different components of a BAM application, which increases understandability of the overall application.

Abstraction from technical details: The presented modeling language provides constructs that abstract from technical details and allows for the modeling of BAM scenarios from a rather conceptual, business level point of view. In particular, the manual specification of CEP logic is replaced by a modeling approach in which the developer is relieved from technical system details (e.g., the internal identifier of a process activity) and the handling of low-level process events. The language provides the means to logically describe the desired BAM aspects at a higher level of abstraction.

In summary, the modeling language provides a BAM-specific graphical notation that allows for the reference to BPMN process models to specify both process monitoring and control aspects from a business-level perspective. The scope of the language cannot be compared with that of existing CEP-based implementation techniques for BAM: It does not work on single process events and does not directly yield executable models. Advanced mechanisms like the automatic generation of IT-level event-based models from high-level BAM models would be necessary to change the prevailing development practices. The proposed language can thus only be a first step towards a more effective development of event-driven BAM applications. Here, the presented specification contributes a precise definition of the semantics to facilitate a later transformation into executable implementation models.

V. Conclusions and Outlook

Having faced challenges in current development practices for event-driven BAM systems, we developed a graphical modeling language. The language enables an explicit modeling of BAM aspects at a conceptual level and allows for the integration with BPMN process models. It supports the definition of different types of process-related KPIs which can either refer to single process instances or can be located on an aggregated process level. Further, reactions can be specified which should be taken if measures exhibit critical values.

However, the presented approach is subject to limitations. Design science research literature advocates that IT artifacts (such as our language) should be subjected to *demonstration* and *evaluation* to assess their "validity". The demonstration serves as a proof of concept to show that the general idea works. The artifact is applied "to solve one or more instances of the problem" [16, p. 55]. The evaluation strives to "observe and measure how well an artifact supports a solution to the problem" [16, p. 56], based on metrics and analysis techniques or based on comparing the artifact's functionality with the solution's objectives as defined in earlier stages of the research. This paper is limited to the demonstration of our approach in a fictitious case only. Further efforts will be needed to demonstrate and systematically evaluate the language's application in real-world business scenarios.

Further, due to the strict focus on business processes and BPMN, only process-related KPIs can be expressed. Other categories of KPIs which might also be of interest (e.g., financial) are not supported. External data which is not handled within a business process cannot be used for the definition of measures. Finally, as the BAM modeling language is rather conceptual it does e.g. not yield executable models. Hence, it is not a direct alternative to prevailing BAM development approaches, but would have to be supplemented by additional techniques and concepts. E.g., implementation-level models could be automatically generated from the high-level BAM models. However, the models would then need to be described more formalized and a precise mapping of the different language constructs to event-based operations would be necessary. In addition, an appropriate software infrastructure is required.

From this need we draw implications and ideas for system design. First of all, an appropriate modeling environment for developing BPMN process models and BAM models is needed. So far, there exists only the language definition and Visio Stencils of the concrete syntax. A specific modeling tool could support the definition of multiple perspectives on the overall model, so that only those elements are

visible which are relevant in a certain modeling situation. In particular, the whole process model might not always be necessary for the definition of the desired BAM operations. The modeling tool could also provide better possibilities to define the attributes of the model elements. On execution layer, a closer integration between the BPM engine that interprets the process models and a CEP module that handles process-related events is desirable to support automated transformation from conceptual BAM models to event-based implementation models. Further, as the BAM model defines which measures are to be visualized in a dashboard, a connection could make use of this information (e.g., by generating pre-configured dashboard files).

REFERENCES

[1] R. Bruns and J. Dunkel, *Event-Driven Architecture*. Berlin, Heidelberg: Springer, 2010.

[2] C. Costello and O. Molloy, "Building a Process Performance Model for Business Activity Monitoring," in *Information Systems Development*, W. Wojtkowski, G. Wojtkowski, M. Lang, K. Conboy, and C. Barry, Eds. Boston, MA: Springer US, 2009, pp. 237–248.

[3] ——, "Towards a Semantic Framework for Business Activity Monitoring and Management," in *Papers from the AAAI Spring Symposium AI Meets Business Rules and Process Management*, K. Hinkelmann, Ed. Stanford, CA, USA: AAAI Press, 2008.

[4] J. DeFee and P. Harmon, "Business Activity Monitoring and Simulation," in *Workflow Handbook*, L. Fischer, Ed. Lighthouse Point, FL, USA: Future Strategies, 2005, pp. 53–74.

[5] A. Del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés, "Defining Process Performance Indicators," *Lecture Notes in Computer Science*, vol. 6426, pp. 555–572, 2010.

[6] O. González, "Monitoring and Analysis of Workflow Applications," Dissertation, Universidad de los Andes / Vrije Universiteit Brussel, Bogotá/Brussels, 2010.

[7] O. González, R. Casallas, and D. Deridder, "MMC-BPM: A Domain-Specific Language for Business Processes Analysis," in *Proceedings of the 12th International Conference on Business Information Systems (BIS)*, Poznan, Poland, 2009, pp. 157–168.

[8] D. Grigori, F. Casati, M. Castellanos, and U. Dayal, "Business process intelligence," *Computers in Industry*, vol. 53, no. 3, pp. 321–343, 2004.

[9] G. Guizzardi, L. Ferreira Pires, and M. van Sinderen, "On the role of Domain Ontologies in the design of Domain-Specific Visual Modeling Languages," in *Proceedings of the 2nd OOPSLA Workshop on Domain-Specific Visual Modeling Language*, J.-P. Tolvanen, J. Gray, and M. Rossi, Eds., Seattle, WA, USA, 2002, pp. 25–38.

[10] B. Korherr and B. List, "Extending the EPC and the BPMN with Business Process Goals and Performance Measures," in *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS)*, Funchal, 2007, pp. 287–294.

[11] B. List and B. Korherr, "An evaluation of conceptual business process modelling languages," in *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06*, no. Section 3, Dijon, France, 2006, pp. 1532–1539.

[12] R. Liu, A. Nigam, J. Jeng, and C.-R. Shieh, "Integrated Modeling of Performance Monitoring with Business Artifacts," in *Proceedings of the 7th IEEE International Conference on e-Business Engineering (ICEBE)*, Shanghai, China, 2010, pp. 64–71.

[13] C. Momm, R. Malec, and S. Abeck, "Towards a model-driven development of monitored processes," in *Proceedings of the 8. Internationale Tagung Wirtschaftsinformatik*, Karlsruhe, Germany, 2007, pp. 319–336.

[14] Object Management Group, "Business process model and notification," OMG, Tech. Rep. 2011-01-03, 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF/

[15] T. Palpanas, P. Chowdhary, G. Mihaila, and F. Pinel, "Integrated model-driven dashboard development," *Information Systems Frontiers*, vol. 9, no. 2-3, pp. 195–208, 2007.

[16] K. Peffers, T. Tuunanen, M. A. Rothenberger, and A. S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2008.

[17] K. Siau and M. Rossi, "Evaluation techniques for systems analysis and design modelling methods - a review and comparative analysis," *Information Systems Journal*, vol. 21, no. 3, pp. 249–268, May 2011.

[18] Y. Wand, D. E. Monarchi, J. Parsons, and C. C. Woo, "Theoretical foundations for conceptual modelling in information systems development," *Decision Support Systems*, vol. 15, no. 4, pp. 285–304, Dec. 1995.

[19] B. Wetzstein, Z. Ma, and F. Leymann, "Towards Measuring Key Performance Indicators of Semantic Business Processes," *Lecture Notes in Business Information Processing*, vol. 7, no. 7, pp. 227–238, 2008.

[20] M. zur Muehlen and R. Shapiro, "Business Process Analytics," in *Handbook on Business Process Management*, J. vom Brocke and M. Rosemann, Eds. Berlin: Springer, 2010, vol. 2, pp. 137–157.