Artifact-centric modeling using BPMN

Niels Lohmann and Martin Nyolt

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{niels.lohmann, martin.nyolt}@uni-rostock.de

Abstract. BPMN offers a rich pool of language constructs to model different aspects of choreographies, interorganizational business processes and service compositions. With collaborations and choreographies, BPMN enables the modeler concentrate on the control flow and the message flow, respectively. At the same time, data flow is only treated as a subordinate extension. In contrast, recent *artifact-centric* approaches model processes from the point of view of the data objects that are manipulated during the process. This paper investigates to what extend BPMN is suitable to model artifact-centric processes and which extensions are required to comfortably support this modeling approach.

1 Introduction

Business process modeling includes a specification of the order in which tasks are executed (control flow), the way data are processed (data flow), and how different branches in distributed and interorganizational business processes and services are invoked and coordinated (message flow). The current BPMN standard offers two different views on business processes: (1) collaboration diagrams (sometimes called interconnected models) that emphasize the local control flow of each participant of the process and (2) choreography diagrams (interaction models) that describe the process from the point of view of the messages that are exchanged among the participants. Conceptually, collaboration diagrams can be seen as control-flow centric models whereas choreography diagrams follow a message-flow centric view. In either diagram, data flow — if at all — only plays a subordinate role.

As third school of thought, artifact-centric models do not specify processes as a sequence of tasks to be executed or messages to be exchanged (i. e., imperatively), but from the point of view of the data objects (called *artifacts*) that are manipulated throughout the course of the process (i. e., declaratively). In recent work [10], we showed that artifact-centric models can be automatically transformed into choreographies and collaborations while guaranteeing certain correctness criteria such as soundness or compliance to business rules [9].

There currently does not exist a common conceptual modeling language for artifact-centric processes. In this paper, we investigate whether BPMN is suitable to express data aspects and to which extend BPMN needs to be extended to be used in an artifact-centric setting. The choice to study BPMN is motivated by its flexibility, comprehensibility, and its popularity among domain experts.

Organization. The rest of this paper is organized as follows. The next section introduces a shipping scenario and discusses several issues in BPMN's capability of modeling data flow and manipulation. In Sect. 3, we briefly introduce artifact-centric modeling. The

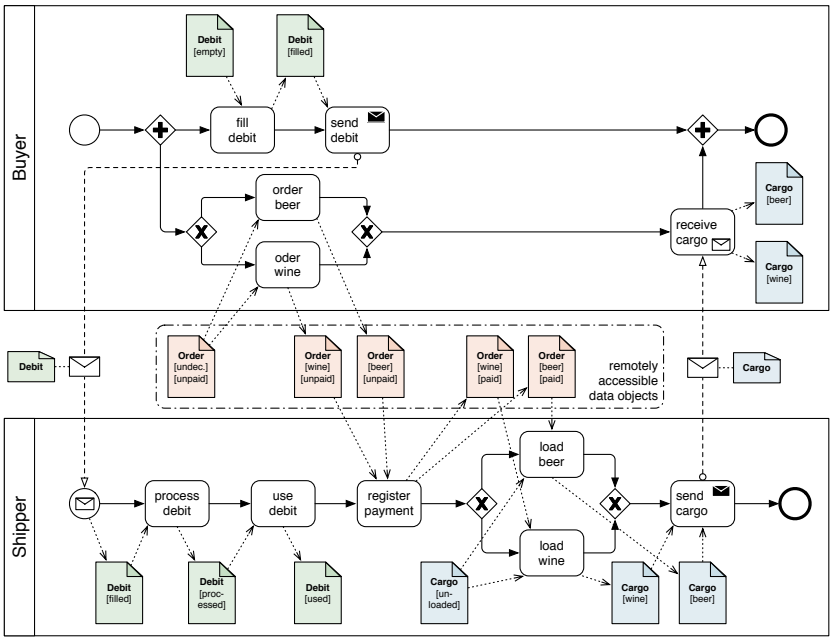


Fig. 1. Shipping scenario as BPMN collaborative process

main contribution of the paper is presented in Sect. 4. We introduce several BPMN extensions to model the different aspects of artifact-centric processes. We continue by discussing the resulting model and related work in Sect. 5 and 6, before Sect. 7 concludes the paper.

2 Data flow modeling in BPMN

We shall employ a shipping scenario taken from [10] as running example for this paper. It consists of a buyer that (1) fills a debit order and (2) may choose between ordering wine or beer and a shipper who (3) processes the debit order, (4) loads a cargo with the ordered goods, (5) and sends it to the buyer. Figure 1 depicts a collaborative BPMN process model of this scenario.

Several data objects, or *artifacts*, are involved in the process: the debit, the order, and the cargo. Apparently, the cargo is not a classical data object, but a physical artifact. Nonetheless, a conceptual model should be able to cope with such objects. *This paper does not focus on actual implementations which would be done in standard languages such as WS-BPEL.* The state of these artifacts can be changed by tasks. For instance, the buyer's task "fill debit" changes the debit's state from "[empty]" to "[filled]". Usually, the access to artifacts is restricted and an artifact is implicitly owned by a participant of the process. At the same time, it may be necessary to change the location of an artifact; that is, to send it to another participant. In the example, the debit is eventually sent to the shipper. As BPMN does not support this directly, we used an association to specify that the debit artifact *is* actually the message that is sent to the shipper. Beside artifacts that may change their location, it is also common to assume artifacts that can be remotely accessed. Such artifacts do not have a canonic owner or pool to be associated

to. In the shipping scenario, the order artifact is an electronic document that can be remotely accessed by both participants: It does not need to be explicitly sent to the shipper to be evaluated. Again, BPMN does not define a canonic way to specify this. Finally, artifacts may contain several data fields that can be manipulated independently. The order artifact in the example process contains information on the desired goods ("unspecified", "wine", or "beer") and the payment status ("unpaid" and "paid"). A partial state change (e. g., setting the payment status to "paid" without considering the ordered items) is not supported by BPMN, which makes the shipper's "register payment" task appear clumsy and underspecified.

The shipping scenario shows that BPMN's capability to specify data objects and data flow is rather limited and a shift to an artifact-centric approach is impossible without making several extensions and adjustments.

3 Artifact-centric modeling in a nutshell

Artifact-centric modeling promotes the data objects of a process and their life cycles to first class citizens. An artifact-centric model of the shipping scenario is consequently specified from the point of view of the *artifacts*; that is, the order, the debit, and the cargo. As we see in Fig. 1, the states of the artifacts may evolve over time, and each state change is performed by an *agent*, namely the buyer or the shipper. Specifying which agent may change an artifact's state also requires information on the location and the access control of an artifact: Physical artifacts (such as the cargo) need to be sent to an agent to be processed, whereas logical artifacts (such as a data base) can be remotely accessed.

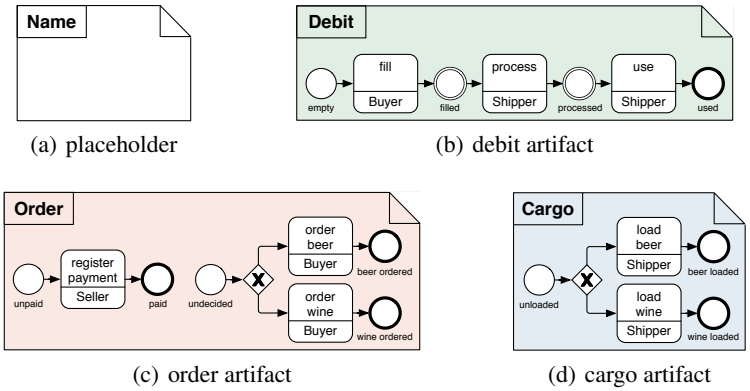
The life cycle of each artifact can be seen as a small business process with an initial state and at least one final state. The latter models a successful completion such as "order paid" or "cargo loaded". As the artifacts can evolve independently, the control flow of the whole business process is specified *declaratively*. Each execution that brings each artifact to a final state can be seen as sound. Apparently, this includes unreasonable executions, for instance those where an implicit order of tasks is violated (sending the cargo before placing an order). Also the final states of artifacts may not be arbitrarily mixed, because otherwise executions in which beer is ordered and wine is delivered would be possible. In recent work [10], we proposed *policies* and *goal states* to exclude such undesired behavior: Policies constrain the execution order of tasks in different artifacts and goal states exclude certain combinations of artifacts' final states.

From the artifact life cycles, the policies, and the goal states, a process model (such as the one in Fig. 1) can be *automatically synthesized* which is sound by design; that is, correctness of the model follows from the synthesis algorithm. The interested reader is referred to [10] for a detailed discussion.

4 BPMN extensions for artifact-centric modeling

In this section, we give more details on the ingredients of artifact-centric processes and propose BPMN extensions for each aspect. In particular, we consider

- artifacts: the process model's basic building blocks;
- object life cycles: a specification of the artifacts' states;
- location information: a means to specify how artifacts change their location;
- access control: a specification of remote accessibility of artifacts;



(a) placeholder (b) debit artifact

(c) order artifact (d) cargo artifact

Fig. 2. BPMN representation of artifacts and object life cycles

- goal states: a specification of desired final states; and
- policies: a means to remove undesired behavior.

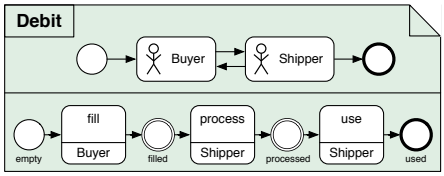
The structure of this section can be seen as a recipe specifying the natural order in which artifact-centric models are created. We thereby exploit that the artifact-centric approach is modular in the sense that artifacts can be specified independently from one another. Hence, each artifact can be refined locally, and later synchronization (e. g., removing undesired behavior by applying policies) usually only affects a small subset of all artifacts.

4.1 Artifacts and object life cycles

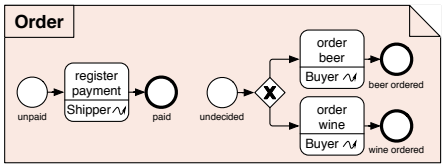
The basic building blocks of the artifact-centric approach are *artifacts* which are represented just like data objects in BPMN, cf. Fig. 2(a). The artifact's name is noted in the upper left corner. Without any further information, it can be treated as a placeholder symbol just like an empty pool in a collaboration. This placeholder symbol can also be used to hide details, for instance when policies are modeled and the focus lies on the dependencies among artifacts.

The object life cycle of an artifact (cf. Fig. 2(b)–2(d)) can be modeled using the symbols for tasks, events, and gateways. However, artifacts have no behavior per se, but are passive objects whose state changes are triggered by agents, for instance "Buyer" or "Shipper". Consequently, each task is annotated by an agent who may execute it. Furthermore, events are annotated by adjectives that describe the current state of the artifact (e. g., "empty" or "filled"). Initial and final states are denoted using the standard BPMN symbols for start and end events, respectively. Note that the life cycle of an artifact may have more than one symbol denoting an initial state. As an example for this, consider for instance the order artifact in Fig. 2(c): This artifact keeps track of the payment status and the purchase order. Each of these information has an individual initial state. The order is initially "unpaid" and the purchase order is initially "undecided".

As we can see from Fig. 2, the local life cycles of the artifacts are rather brief and clear compared to the data handling in the process of Fig. 1. We achieved this by "abusing" control flow constructs to model the evolution of object life cycles. As the



(a) debit artifact with location information (message exchange)



(b) order artifact with location information (remote access)

Fig. 3. BPMN representation of location information

semantics of the gateways remains the same, the object life cycle models should be intuitively understandable by BPMN modelers.

4.2 Location information

We already sketched in Sect. 3 that artifacts may be logical or physical objects that need to be transported between agents to perform state changes (e. g., the cargo artifact) or data objects that can be remotely accessed (e. g., the order artifact). This differentiation has an impact on the execution of the process, because artifacts may need to be sent to an agent before he can execute a task.

To this end, we extend the artifact models with *location information* and information about accessibility. We assumed that the debit artifact can be sent among the buyer and the shipper. This is modeled by an additional life cycle in the upper part of the debit artifact, cf. Fig. 3(a). Thereby, each agent is treated as a location (depicted by a stick-figure) and the arrows specify message channels between these locations. In addition, a start event and an end event specify that the debit artifact resides initially at the buyer and a successful processing of the artifact is only possible at the shipper agent. The cargo artifact is extended similarly. When synthesizing a process such as the model in Fig. 1, respective message events are inserted automatically and make sure that, for instance, the debit artifact is sent to the shipper before the task "process debit" is executed. Again, we refer to [10] for more information on the synthesis.

In contrast, the order artifact does not have a location, but can be remotely accessed, for instance using an URL or a different form of addressing mechanism. This is depicted by a small lightning symbol next to the agent's names, cf. Fig. 3(b). This means that the agent can always execute the task and does not need to receive the artifact before.

4.3 Access control

Up to now, we assumed that an artifact is either an object — physical or logical — that needs to change its location to be accessed or a remotely accessible data object whose location (e. g., its URL) is known to the agents. The latter abstraction fails short in describing situations in which data objects are created or the access is only granted after

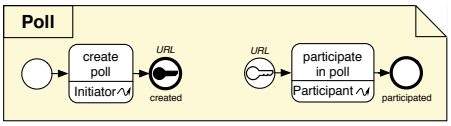


Fig. 4. BPMN representation of the creation and distribution of access control tokens

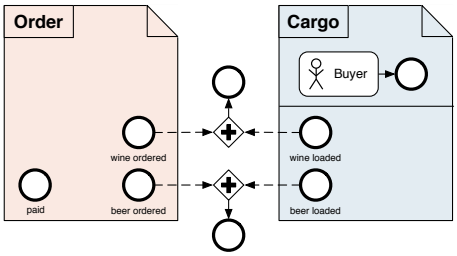


Fig. 5. BPMN representation of goal states

other tasks have been executed. In such situations, the *access control* must be explicitly modeled.

As an example, consider the Doodle Web service (`http://doodle.com`) to schedule meetings or events. After creating a poll, the service returns a URL to that poll that can be sent to colleagues that should take part in the meeting. Without this URL, a participation in the poll is impossible, and the URL can hence be seen as an access token. We visualize this access control granting by a novel event type with a key symbol, cf. Fig. 4. In the example, the initiator of the poll creates a key with name "URL", and the participant can only execute the task "participate in poll" after receiving this URL.

The extensions we presented so far only affect single artifacts. Hence, each artifact can be modeled from a mere placeholder to a fully specified model including object life cycles, location information, and access control. The remaining extensions concentrate on the interdependencies between artifacts.

4.4 Goal states

Up to now, any execution of tasks that lead each artifact to a final state would be seen as a successful outcome of the shipping process. Such executions would, however, include runs that reach the final state "beer ordered" of the order artifact (cf. Fig. 2(c)) and "wine loaded" of the cargo artifact (cf. Fig. 2(d)). Apparently, such executions are undesired as the final states of the artifacts do not match. To exclude such undesired combinations, we specify *goal states*. A goal state is a combination of artifact's final states, and each combination that is not mentioned as goal state is implicitly excluded during the synthesis to a collaborative process model.

To give goal states a BPMN representation, we connect the desired end events of the artifacts with a parallel gateway, cf. Fig. 5. In this example, connecting "beer ordered" and "beer loaded" respectively "wine ordered" and "wine loaded" has the effect that only these combinations are valid. Considering all artifacts, this yields two valid final states of the overall process:

1. The order is paid and beer was ordered. The cargo is at the buyer and loaded with beer. The debit is used and at the shipper.

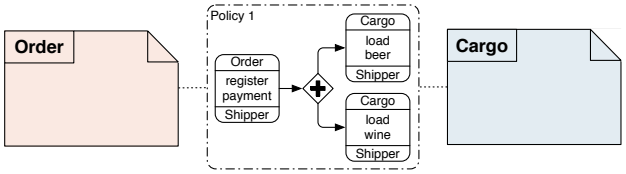


Fig. 6. BPMN representation of policies

2. The order is paid and wine was ordered. The cargo is at the buyer and loaded with wine. The debit is used and at the shipper.

In Fig. 5, we used placeholder symbols for the order and the cargo artifacts and only depicted the end events. Such a view on the end events should be provided by a modeling tool to make the specification of goal states as simple as possible.

4.5 Policies

Artifact-centric approaches follow a declarative modeling style: Instead of explicitly modeling global state changes, we only modeled the local object life cycle of each artifact. Consequently, the order of tasks in the generated process is only constrained with respect to goal states. As a downside of this approach, a lot of unreasonable behavior is exposed. For instance, sending out an unloaded cargo or even sending without prior payment is possible. To rule out this undesired behavior, we employ four *policies*:

1. Only load the cargo after payment has been registered.
2. Only register the payment when the filled debit form is at the shipper.
3. Do not send an unloaded cargo to the buyer.
4. Only send the debit form if it is filled and at the buyer.

A policy specifies dependencies between the tasks of one or more artifacts. For instance, the first policy expresses a causality between the order and the cargo artifact. To model policies in BPMN, we use tasks and gateways to express the additional dependencies and add an association to all involved artifacts. As an example, consider Fig. 6: In the upper part of each task, we also specify the artifact it belongs to. Policy 1 specifies that the "register payment" task of the order artifact must be executed before the "load beer" or the "load wine" task of the cargo artifact are executed. The parallel gateway is used, because a policy should not exhibit choices (i. e., we do not allow exclusive splits). The fact that the last two tasks are mutually exclusive follows from the life cycle of the order artifact, cf. Fig. 2(c). Again, we used placeholder symbols for the involved artifacts and a modeling tool should support the hiding of the object life cycles. The other policies can be modeled similarly.

5 Discussion

Figure 7 depicts the overall artifact-centric model. It consists of the three artifacts (with their object life cycles and the location information), four policies, and two goal states. This model can be automatically transformed into the process depicted in Fig. 1 using the synthesis approach described in [10]. Conceptually, this synthesis is defined in terms of Petri nets [12], but the BPMN constructs can be straightforwardly mapped to Petri nets using the formal semantics of Dijkman et al. [6]. More details can be found in [11].

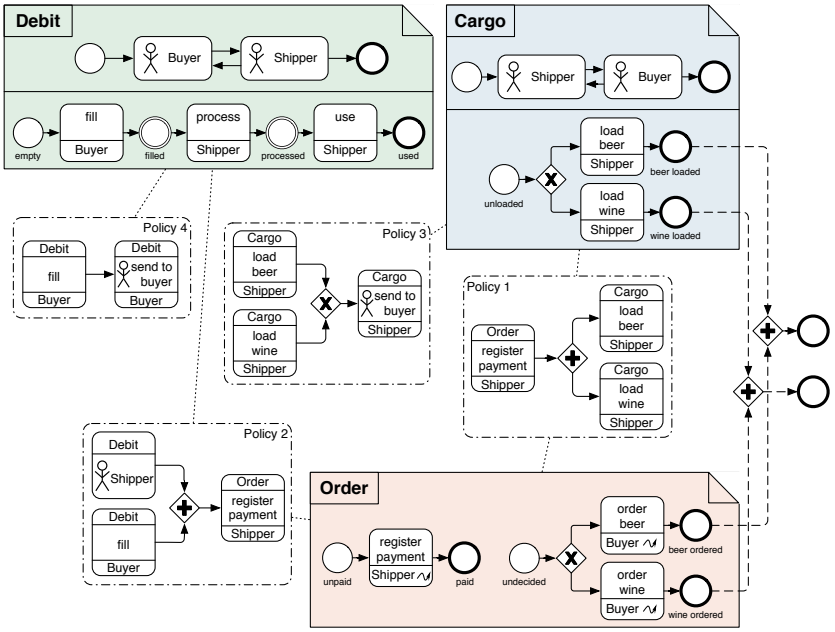


Fig. 7. Complete artifact-centric BPMN model of the shipping scenario

At this point, a comparison between the artifact-centric model (cf. Fig. 7) and the collaborative process model (cf. Fig. 1) is difficult and premature, because without empirical studies, it is impossible to decide which model is easier to understand or which model can be created in a shorter period of time. In particular, such questions heavily rely on the education of the modelers and proper tool support. However, we would like to mention some aspects that are unique to, or at least enforced by, artifact-centric process models.

Modularity. We already mentioned that artifacts can be modeled independently. That is, a modeler only needs to concentrate on a single artifact model at a time. This partition into smaller, "brain-sized", units may not only improve the understandability and the model quality, but should also facilitate the concurrent modeling of all required artifacts of a process. Furthermore, adding or refining an artifact does not influence the other artifacts which adds flexibility to the approach.

The moment dependencies between several artifacts need to be expressed, views may help the modeler focus on the relevant aspects. For goal states, only the artifact's end events need to be considered. If a tool supports such a view, the specification of a goal state should boil down to a few selection actions. Likewise, the creation of a policy only depends on the names of the artifact's tasks and could be facilitated by a modeling tool by corresponding menus and support for autocompletion.

To summarize, the modularity allows to explore and create a model gradually. Compared to the model in Fig. 1, less global dependencies need to be understood at once. As a side effect, the artifact-centric model (cf. Fig. 7) is already graphically partitioned into smaller units.

Declarative modeling. The declarative modeling style is a liberal specification of all possible behavior. Thereafter, undesired behavior needs to be ruled out by adding goal states and policies. Given the artifacts, the goal states, and the policies, a sound process model can be automatically synthesized. In case an aspect changes, this synthesis can be repeated. This may increase the flexibility and the modeling speed of this approach. Even if the change *affects* large parts of the process, only a few parts of the original artifact-centric model need to be changed due to the modularity. For instance, adding the possibility to perform a third-party debit check would have a large impact on the overall process, but could be realized by a small adjustment to the debit artifact.

The automatic synthesis further allows to abstract from certain error-prone aspects such as message protocols: The message flow directly follows from the location information. It ensures that artifacts are only sent to agents to perform tasks on them or to satisfy goal states.

6 Related Work

Since the first draft of BPMN, several extensions have been proposed that aim at giving modelers the constructs at hand to model additional aspects of their processes: Decker et al. [5,3] studied extensions toward choreography modeling. These extension have already been picked up by the OMG and choreography models are now a part of the current BPMN standard. Other extensions focus on nonfunctional properties such as performance measures [8], time requirements and constraints [7], service quality requirements [14], authorization [15] and security [13], transactions and compensation [2], or process compliance [1].

All extensions share the evaluation that BPMN is the de facto standard in process modeling and that it is promising to integrate novel aspects to BPMN rather than to propose an alternative language. This results in careful extensions that try to reuse BPMN constructs as much as possible and to make extensions appear as natural as possible. Best practices such as abstraction by sub processes or hiding of unnecessary details are also frequently adopted. Depending on whether the extension is just conceptual or already aimed at execution, also the meta model needs to be adjusted.

To the best of our knowledge, this paper provides the first extension of BPMN toward artifact-centric modeling. As our extensions are at a conceptual model, we did not specify an extension to the BPMN meta model at this point.

7 Conclusion

Summary. We investigated a small process model of a shipping scenario and showed that certain data-flow aspects are not faithfully supported by the current BPMN standard. To model artifact-centric processes, we proposed several BPMN extensions to represent the parts of the artifact-centric approach. Thereby, we tried to reuse existing graphical notations as much as possible. We also tried to follow modeling best practices and define different views on the process to help the modeler focus on relevant details.

Lessons learnt. The current BPMN standard has only limited modeling support for data aspects. In particular, modeling the life cycle of data objects results in cluttered models. By “abusing” control flow constructs such as tasks, events, and gateways, artifacts and their life cycle can be effectively modeled. As a matter of fact, the whole school of artifact-centric process design can be expressed with only a few adjustments to BPMN.

As artifact-centric models are naturally partitioned into smaller parts (i. e., artifacts, policies, and goal states), they facilitate a gradual modeling approach. Beside the modularity, the declarative nature further improves flexibility, because even if changes affect large parts of the model (e. g., if a policy or an artifact changes), a sound process can be automatically synthesized. This process can then be realized by several services; that is, the process is replaced by a service composition.

Future work. This paper is only the first step toward BPMN support for artifact-centric modeling. One obvious direction of future work is the integration of the proposed extensions into a BPMN modeling tool such as Oryx [4]. Beside the mere support of the concepts, also an implementation of different process views (i. e., a view on a single artifact, a view on goal states, and a view to model policies) is required.

Based on a prototyping implementation, case studies and empirical experiments need to be conducted. The question which kind of processes may benefit from an artifact-centric modeling approach is still open. This includes questions regarding modeling speed, model quality, adaptability and flexibility, as well as understandability of the models.

References

1. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: BPM. LNCS, vol. 5240, pp. 326–341. Springer (2008)
2. Bocchi, L., Guanciale, R., Strollo, D., Tuosto, E.: BPMN modelling of services with dynamically reconfigurable transactions. In: ICSOC 2010. LNCS, vol. 6470, pp. 396–410 (2010)
3. Decker, G., Barros, A.P.: Interaction modeling using BPMN. In: BPM Workshops 2007. LNCS, vol. 4928, pp. 208–219. Springer (2007)
4. Decker, G., Overdick, H., Weske, M.: Oryx - an open modeling platform for the BPM community. In: BPM 2008. pp. 382–385. LNCS 5240, Springer (2008)
5. Decker, G., Puhlmann, F.: Extending BPMN for modeling complex choreographies. In: OTM Conferences (1). LNCS, vol. 4803, pp. 24–40. Springer (2007)
6. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology 50(12), 1281–1294 (2008)
7. Gagné, D., Trudel, A.: Time-BPMN. In: CEC 2009. pp. 361–367. IEEE (2009)
8. Korherr, B., List, B.: Extending the EPC and the BPMN with business process goals and performance measures. In: ICEIS 2007. pp. 287–294 (2007)
9. Lohmann, N.: Compliance by design for artifact-centric business processes. In: BPM 2011. pp. 99–115. LNCS 6896, Springer (2011)
10. Lohmann, N., Wolf, K.: Artifact-centric choreographies. In: ICSOC 2010. pp. 32–46. LNCS 6470, Springer (2010)
11. Nyolt, M.: Modellierung artefaktzentrierter Geschäftsprozesse. Bachelorarbeit, Universität Rostock, Rostock, Germany (2011), (In German)
12. Reisig, W.: Petri Nets. Springer, EATCS Monographs on Theoretical Computer Science edn. (1985)
13. Rodríguez, A., Fernández-Medina, E., Piattini, M.: A BPMN extension for the modeling of security requirements in business processes. IEICE Transactions 90-D(4), 745–752 (2007)
14. Saeedi, K., Zhao, L., Sampaio, P.R.F.: Extending BPMN for supporting customer-facing service quality requirements. In: ICWS 2010. pp. 616–623. IEEE (2010)
15. Wolter, C., Schaad, A.: Modeling of task-based authorization constraints in BPMN. In: BPM 2007. LNCS, vol. 4714, pp. 64–79. Springer (2007)