Data in Business Processes

Andreas Meyer, Sergey Smirnov, Mathias Weske

Technische Berichte Nr. 50

des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 50

Andreas Meyer | Sergey Smirnov | Mathias Weske

Data in Business Processes

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de/ abrufbar.

Universitätsverlag Potsdam 2011
http://info.ub.uni-potsdam.de/verlag.htm

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 4623 / Fax: 3474
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URL http://pub.ub.uni-potsdam.de/volltexte/2011/5304/
URN urn:nbn:de:kobv:517-opus-53046
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-53046

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-144-8

Data in Business Processes

Andreas Meyer, Sergey Smirnov, and Mathias Weske

Business Process Technology Group
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2–3, D-14482 Potsdam, Germany
{andreas.meyer,sergey.smirnov,mathias.weske}@hpi.uni-potsdam.de

Abstract. Process and data are equally important for business process management. Process data is especially relevant in the context of automated business processes, process controlling, and representation of organizations' core assets. One can discover many process modeling languages, each having a specific set of data modeling capabilities and the level of data awareness. The level of data awareness and data modeling capabilities vary significantly from one language to another.
This paper evaluates several process modeling languages with respect to the role of data. To find a common ground for comparison, we develop a framework, which systematically organizes process- and data-related aspects of the modeling languages elaborating on the data aspects. Once the framework is in place, we compare twelve process modeling languages against it. We generalize the results of the comparison and identify clusters of similar languages with respect to data awareness.

1 Introduction

In the mid-nineties, the primary focus of business process management (BPM) was on the design and documentation of processes. Business process models captured activities and their ordering necessary to achieve a business goal. In this way control flow was the dominant aspect in business process models. During the last decade, BPM received much attention as "a systematic and structured approach to analyze, improve, control, and manage business processes" [1]. Being widely adopted by industry, business process management faced new challenges and opportunities. Process models focusing only on control flow become insufficient. It turned out that additional aspects have to be addressed, including data.

We motivate the relevance of data within three areas: Service-oriented Architectures, representation of organizations' core assets, and process controlling. The emergence of Service-oriented Architecture (SOA) opened new horizons for *automated business process execution*, yet revealed new challenges. SOA transformed enterprise landscapes slicing the functionality of large software systems into services. As services are capable of accomplishing atomic business tasks, they can be effectively used for task automation. Thereafter, SOA catered for automation of business processes. While control flow oriented business processes made the process routing logic explicit, data was still "hidden" inside IT systems.

However, this data highly impacts process execution. For instance, many decisions in processes are data driven. As a result, the role of data in process models grew significantly. It became essential to model the data and data flow within the process. Thereby, process modeling languages that emerged in the last decade, e.g., Business Process Model and Notation (BPMN) [2], demonstrated higher data awareness.

Another driver for explicit data representation in process models is the *representation of core assets* which capture essential properties of an organization without the value creation cannot take place. Organizations' value creation mainly base on information about their own value chain, customers, production, and research and development cycles. This information is captured in terms of data in different IT systems which combine the enterprise-wide data utilized in the everyday work. All customer information, for instance, is stored in the customer relationship management (CRM) system. The organizations' actual value creation is performed by executing the organizations' business processes which depend on the information mentioned above. Following, these business processes need access to the IT systems and its contained data to keep an organization operational.

Finally, we refer to *business process controlling* as another motivator of data support in process modeling. For ensuring process quality by process controlling, key performance indicators (KPIs) are measured and interpreted using business process intelligence techniques. KPIs reflect business goals of an organization. These goals are defined referring to data. Following, KPIs rely on data. For instance, one business goal might be to achieve the highest customer satisfaction in the market. This goal is reflected by several KPIs; one of them deals with delayed credit applications. This number should be minimized for increasing the customer satisfaction. For process controlling, the activities contributing to this goal need to be identified for evaluating them. This is done by selecting the activities performing work on the appropriate data objects – in this case the credit application. Following, an explicit statement supports process controlling. Additionally, the data objects are considered for the evaluation itself as well. State and content changes provide insights with regard to the progress and long lasting steps can be identified amongst others. Against this background, process models focused on control flow and describing *how* to achieve business goals rather than *what* has to be achieved are insufficient. One approach to goal externalization is shifting the focus from control flow to data [3,4].

The three considered contexts motivate the need for extensive data modeling capabilities in business process modeling languages. The contribution of this paper is an evaluation of the data awareness level of current business process modeling languages. To organize the evaluation, we develop a framework. This framework assesses data awareness of process modeling languages against a set of criteria. We use the framework to compare the properties of twelve modeling languages emerged from industrial and academic initiatives. We also reflect on the results of the evaluation and organize the studied approaches according to their capabilities.

The remainder of this paper is structured as follows. Section 2 presents the evaluation framework enabling process modeling language comparison in Section 3. Afterwards, we discuss the summarized evaluation results and cluster the approaches amongst different criteria in Section 4. Section 5 discusses the related work. Section 6 concludes the paper.

2 Evaluation Framework

In this section, we present the framework for the evaluation of process modeling languages with a main focus on their data modeling capabilities. The framework assesses modeling languages against 23 criteria organized into four groups. The first group comprises criteria reflecting process modeling capabilities. The second group includes criteria that assess data modeling capabilities. The third group reveals how strongly process and data modeling capabilities are related. Finally, the criteria of the fourth group deal with general execution semantics and specific execution capabilities comprising the influence of data for execution.

2.1 Process Modeling Capabilities

Process modeling capabilities describe, whether the modeling language enables process modeling from the process perspective. A process perspective describes which activities have to be performed in which order to achieve the process' business goal. We distinguish the following process modeling capabilities:

1. *Activity Modeling.* An activity represents an unit of work. An activity example is *Create order*. Activity modeling capability indicates, whether the modeling language provides a construct for activity modeling.
2. *Event Modeling* An event represents something that happens within the course of the business process. Event examples are sending of a message, timing-incidents, and exception alerts. Event modeling capabilities show, whether the modeling language has a standalone construct for event modeling.
3. *Gateway Modeling.* A gateway is a model element realizing routing logic in the process. Gateways are used to model decisions or concurrency. A gateway example is the AND split, where two paths are executed concurrently. For instance, customer address details and customer bank details can be verified independently. Gateway modeling capability reveals whether the modeling language has specific constructs for gateways and specific gateway types.
4. *Control Flow Modeling.* The control flow represents the partial order between (if existing) activities, events, and gateways for one process. Therefore, it visualizes the set of all valid execution traces of a process. An example is the *Handle order* process, where the order will be received via a message event first, followed by creation and check of the order. Based on the outcome of the check, i.e. the order is approved or rejected, the order will be packed and sent to the customer or the order will be canceled. Control flow modeling capability evaluates whether the modeling language enables the modeler to order activities, events, and gateways explicitly.

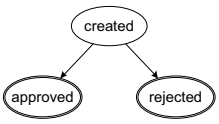

Fig. 1. Example of a life cycle for data object *Order*

2.2 Data Modeling Capabilities

Data structure modeling constructs introduce the capabilities representing data and of considering data possibly based on concrete data values or states. Additionally, relations between data objects play a key role for data modeling. The identified modeling constructs are as follows:

5. *Data Modeling.* Data modeling indicates whether a modeling language is capable of considering information, i.e. data, in the process specification by any means. This data awareness may be achieved via variables, (primitive) data types, or objects in the sense of object orientation (see below). These data representations may exist with any complexity. Examples are an *Order*, a *Product description*, or the *Value of an item*. Data modeling capability indicates, if a modeling language provides constructs to include the utilization of information and data within the process model.
6. *Data Object Modeling.* A data object is a model element that captures a unit of data manipulated during the business process. A data object is utilized as object in the sense of object orientation, i.e. a data object has an identity and encapsulates the behavior of the object: The life cycle, and either complex or a set of primitive variables and data types. An example of a data object is *Order* in the *Handle order* process. Data object modeling capability indicates, if a process modeling language provides constructs for capturing data objects explicitly.
7. *Data Object State Modeling.* A data object state is the set of property values characterizing the unique configuration of information of this data object. For instance, let data object *Order* has the only property *status* with the possible values *created*, *approved*, and *rejected*. Then *Order* can be in states *created*, *approved*, or *rejected*. Data object state modeling capability shows, if a modeling language enables the user to express data object states.
8. *Data Object Life Cycle Modeling.* Data object life cycle describes data object states and the allowed state transitions. The life cycle of data object *Order*, see Fig. 1, may allow the transitions from state *created* to state *approved* and from state *created* to state *rejected*. If the modeling language has data object life cycle modeling capability, the modeler can specify both data object states and transitions between them.
9. *Modeling of Data Object Collections.* Often business processes operate with several data objects of one type at a time. For instance, activity *Pack order* may consider several *Items* to be delivered within one *Order*. In this case modeling data objects collections is handful.
10. *Persistence Mechanism Modeling.* Certain process modeling scenarios require to show how data is persisted. The persistence mechanism can be

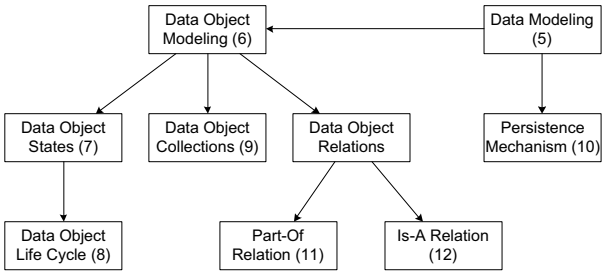

Fig. 2. Relations of data modeling capability criteria

realized, e.g., by means of a database management system. Returning to the example of order handling, the persistence mechanism may provide means to store information about the orders in the corporate IT infrastructure. Modeling of data persistence mechanisms enables the designer to specify explicitly in which storage data objects are persisted.

11. *Data Object part-of Relation Modeling.* While some data objects are primitives, others can be decomposed into more fine-grained objects. Such a decomposition is formalized with a *part-of* relation. For instance, a *Package* data object includes *Item*s, each of which is a part of this package. The *part-of* relation models this fact. Thereby, this criterion shows if the process modeling language enables modeling of *part-of* relation for data objects.
12. *Data Object is-a Relation Modeling.* The *is-a* relation is another fundamental modeling relation. Data object *is-a* relation shows that one data object type is the specialization of another data object type. An example is *Order* and *Purchase order*, where *Purchase order is-a Order*. The data object *is-a* relation modeling capability indicates, if the process modeling language supports modeling of the *is-a* relation for data objects.
13. *Are data objects mandatory elements in the process design phase?* Answering this question determines whether data must be added to the business process model. Thereby, we assume that the potential necessity is valid for design- as well as run-time representations, if both aspects are supported by a modeling language.

Figure 2 interrelates the different criteria of this category and visualizes existing dependencies. Each shown connection conforms to a *leads-to relation*. Exemplarily, this means that data object life cycle modeling might only be supported by a modeling language if data object states are supported as well.

2.3 Connection of Process and Data Modeling Capabilities

As we argued in Section 1, process and data aspects should not be examined separately. Hence, we assess the modeling languages with regards to their capabilities of *data flow* modeling. Data flow represents the evolution of data objects

within a business process. In particular, the data flow is captured by relations between the process elements and the data objects. Hence, data flow shows the order of manipulations for each data object along with activities performing these manipulations. This linkage is supported by the following modeling constructs:

14. *Modeling of Typed Relations between Process Elements and Data.* The relations between a data object and the process element indicate the fact of data access. These relations can be typed. The principle of relation typification can vary depending on the notation. One example is the distinction of *read* and *write* relations.
15. *Are data object life cycle state transitions associated with activities?* Associations between state transitions and activities connect the control flow with data of a business process and determine which actions manipulate which data object.

2.4 Execution Semantics

For automated processes, the execution semantics of the utilized modeling approach describes how the execution eventually takes place. Therefore, we evaluate the execution capabilities with a strong focus on data and highlight the capabilities of including data as important part into the execution of a process.

16. *Formal Token Flow Semantics.* Formal token flow semantics rely on a formally specified execution semantic.
17. *Informal Token Flow Semantics.* Informal token flow semantics rely on execution semantics which can be represented by using the token flow approach (including multiple token flow per instance), but a formal token flow specification is missing.
18. *Data-based Decisions.* Data-based decisions allow decisions based on data which directly influence the process control. An example is for instance the choice how to deal with an *Order* based on the order's current state. Path one is taken if the *Order* is in the state *approved* and path two is taken if the *Order* is in the state *rejected*. Alternatively, the current value of a data object may be used for decision making, e.g. the price of the *Order* is above or below *500 Euro*.
19. *Execution Controlled by Control Flow only.* Control flow controlled process execution is indicated by an exclusive process guidance by the elements introduced in group one (see Subsection 2.1), especially item number four: Control flow. An example is a sequence of activities to *Pack*, *Label*, and *Ship* an *Order* after the event *order is accepted* occurred.
20. *Execution always Controlled by Data and its Dependencies.* Data controlled process execution is indicated by process guidance always basing on data objects, their states and possibly their values or their pure existence. Thereby, the current state and value of a data object specifies upcoming actions, which map to activities, towards the business process goal.

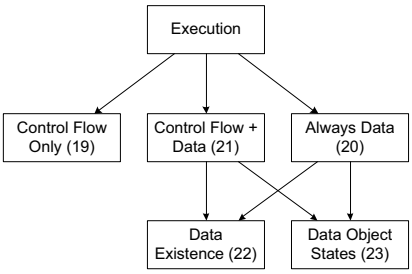

Fig. 3. Relations of execution semantics criteria

21. Execution Controlled by Data and Control flow. Process control influenced by both control flow and data is indicated if execution semantics and the execution order base on control flow dependencies as well as data dependencies, data states, data values or data existence, whereby we do not differentiate the driving force. An example for such collocated process control is that the control flow specifies the general execution order and that the data dependencies influence the enabling of activities due to state and value requirements. For instance, the activity *Check warehouse status* is planned to be enabled after *Verify order*, but it can only be enabled if the data objects *Internal order* and *Warehouse stock* are in the states *approved* and *updated* respectively to ensure proper execution of the activity.

22. Process Control via Existence of Data. This criterion indicates that process control is influenced by the existence or non-existence of a data object independently from the value and the state of the data object. Existence in our case means that the data object is defined and contains a value of business use. An example for this criterion is the activity *Verify customer* which needs a customer existing to be enabled and executed.

23. Process Control via Data Object States. This criterion builds up on the one before but contains stricter requirements for activity enabling. The data object does not only need to exist but it must also exist in the specified state to be executed. An *Order* in state *created* cannot enable the activity *Pack shipment*, but an *Order* in state *accepted* can do so with respect to the business process definition.

Figure 3 outlines the relations between the criteria of this category. Each edge of the graph conforms to a *leads-to relation*. Exemplarily, this means that criteria 22 can only be supported if criteria 20 or 21 are supported by a specific modeling language.

3 Evaluation

We evaluate twelve process modeling languages against the criteria introduced in Section 2. These are *Workflow nets*, *Web Services Business Process Execution*

Fig. 4. Value chain of *Handle order* business process

Language (BPEL), *Yet Another Workflow Language* (YAWL), *Event-driven Process Chains* (EPC), *UML activity diagrams*, *Business Process Model and Notation* (BPMN), *Corepro*, *Business Artifacts*, *Document-driven workflows*, *ADEPT*, *Case handling*, and *State charts* in that order.

The modeling capabilities of the evaluated approaches are illustrated by a running example of a *Handle order* business process. This process is presented in Fig. 4 and comprises seven main steps. To illustrate the modeling capabilities, we use an appropriate subset of these steps for each of the evaluated modeling languages. Following, we introduce the complete scenario.

Handle order business process. First, the order is received from a customer and created internally to deal with it. Based on the order information, customer details are extracted and checked. If the customer is already registered, nothing has to be done in this respect. Otherwise, the new customer needs to be registered and verified afterwards. The verification includes checks for the provided address details, the bank account information, and the bank account cash balance. If the customer could not be verified, the order is canceled and the process closed. Otherwise, the process continuous with the *Order preprocessing* step which includes a possibly iterative verification of the order. Each iteration includes an order refinement which is performed in close cooperation with the customer. A rejected order skips the following process steps and continuous with the *Order completion* step. Approved orders are passed to the *Warehouse management* step. There, the order packaging good supply are performed. After packaging, the *Order delivery* step is initiated. In case the ordered products are not on stock completely, the order can be split into two parts: The available and the pending one. The available part is packaged and the pending part is backordered from the customer point of view. Internally, the necessary products are reordered and put to the warehouse after arrival. Following, the pending order is executed again analogously to the original order. For instance, if the arrived products are not sufficient to handle all orders, still open orders are put into another backorder iteration. In case a product is not deliverable at all, the order is rejected and canceled. Then, *Order completion* is the next process step. For deliverable orders, the *Order delivery* comprises labeling, sending, and tracking of the order. Following, the *Invoice order* step comprises the creation and sending of the invoice as well as the receiving of the payment. In case the customer is not paying, the invoice can be re-send. Alternatively, a dunning letter might be created and sent to the customer. After receiving the payment, the process is continued with the *Order completion*. If the order has been split earlier, it will be consolidated again. The final task performed is to archive the order including all related information.

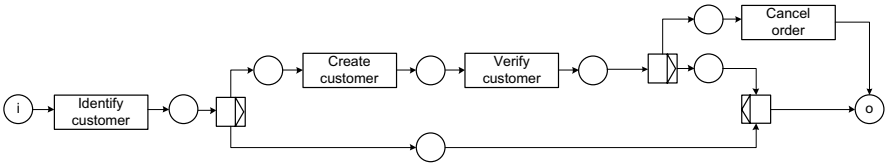

Fig. 5. Extract of business process *Handle order* modeled using Workflow nets

3.1 Workflow Nets

Workflow nets are based on Petri nets introduced by Petri in [5]. After several Petri net extensions covering for instance timing, hierarchies, and colors in Petri nets [6,7,8,9], they are often used to model or formalize business processes. Transitions model process activities, while places model process states and help to realize routing decisions. The edges capture the process control flow. Workflow nets again utilize petri nets for workflow modeling [10]. Compared to Petri nets, Workflow nets have been extended with explicit split and join nodes to highlight exclusiveness and parallelism relationships between activities. But generally, Workflow nets are Petri nets with additional modeling constructs.

Data Capabilities and Limitations. Data in general or data objects specifically are neither considered in the Petri net nor the Workflow net approach. Therefore, all criteria from the categories data modeling and connection of process and data and process modeling are not supported by Workflow nets. Execution semantics related, Workflow nets implement the formal token flow semantics of Petri nets and therefore, execution is controlled by control flow only. Data-based decisions as well as data-driven execution of any kind is not supported due to the lack of data awareness.

Example. Figure 5 introduces a Workflow net example comprising the second step of the overall scenario. As Workflow nets only cover control flow aspects, this net is limited to the four activities and the two decision points shown. After identifying the customer, the step can either be completed or the customer needs to be created and verified afterwards. Based on the verification, the process needs either be canceled and completed that way or it will be completed via the happy path, which leads to the upcoming order verification.

Conclusion. Workflow nets focus on modeling of process activities and control flow. No data aspect is part of this approach.

3.2 BPEL

WS-BPEL, the Web Services Business Process Execution Language, has been introduced by IBM, BEA Systems, Microsoft, Siebel Systems, and SAP. The

current version 2 has been introduced in [11] and focuses on activities, services linked to these activities, and their order visualized by an XML structure. The support for events and gateways is presented by the *pick*, *switch* and *flow* statements. The latter one introduces parallelism into BPEL whereas the first one allows decision taking based on external events. *switch* provides the capability to include decisions based on process aspects. Comprising these aspects, BPEL supports control flow specification which is the main driver for process execution utilizing BPEL. Currently, BPEL is the de-facto standard for web-service-based and IT systems supported enactment and execution of business processes.

Data Capabilities and Limitations. Data is modeled through variables contained in globally or locally visible data containers, which might be shared among the participants of the process. The variables represent in- and output messages of the activities. The exchange of specific data and the manipulation of variables is handled through BPEL's *assign* statement. This either copies specified data from one container the appropriate service cannot access to another container the service can access or it assigns a new value to the current variable. Execution is mainly driven by the control flow specification. However, data plays a role with respect to existence assumptions. Services linked together by the control flow can only be executed if the data assigned to this web service exists. Data-based decisions are supported by references to variables in the *switch* statement. Execution semantics neither follow a formal nor an informal token flow semantics.

To extend these capabilities, Habich et al. introduced a data aware extension to BPEL in [12]. They utilize so-called Data-Grey-Box Web Services [13], which are web services enhanced with an explicit data aspect specifying how and from where input information and is gathered and output information is stored. This information is added as data pointers to the SOAP message. Additionally, they introduce a new link type to connect services from the data perspective. Following, data dependencies and storage locations can be utilized. Altogether, the authors propose an orthogonal extension to the control flow concept: A separate data flow layer, which is eventually added on top of the specified process by a data modeling expert. However, specific aspects known from data modeling, e.g. data relations and states are not covered by this approach.

Altogether, data dependencies can be specified in BPEL, but data plays a supporting role only.

Example. The BPEL example presented in Listing 1.1 covers the passing of the order information from the *Seller_Administration* dealing with the first steps until and including the *Order verification* to the *Seller_Warehouse* where packaging and warehouse activities are performed.

```
1 <assign>
2  <copy>
3   <from container="Seller_Administration" part="InternalOrder"/>
4   <to container="Seller_Warehouse" part="InternalOrder"/>
5  </copy>
6 </assign>
```

Listing 1.1. Copying order information within *Handle order* business process in BPEL

As the aforementioned data containers usually only allow limited access, step three in the overall order process requires a transmission of the created and verified order from the administration to the warehouse department where packaging will continue the process. The data passing is solved in BPEL via the *copy* tag with indication of the information to be transferred and its source and its target.

The example in Listing 1.2 presents BPEL's capabilities with respect to data-based decision taking. Alongside the *Order* variable, a second variable stating the current state of the order is introduced. Based on the value of this variable, the order is changed in cooperation with the customer, the customer is informed about the rejection, or the warehouse staff deals with order packaging.

```
1  <switch>
2   <case condition="getVariableData(stateOfOrder)==undecided">
3    <invoke partnerLink="Customer" operation="Change order"/>
4   </case>
5   <case condition="getVariableData(stateOfOrder)==rejected">
6    <invoke partnerLink="Customer" operation="Reject order"/>
7   </case>
8   <case condition="getVariableData(stateOfOrder)==undecided">
9    <sequence>
10    <invoke partnerLink="Seller_Warehouse" operation="Check wh status"/>
11     ...
12   </sequence
13  </case>
14 </assign>
```

Listing 1.2. Modeling a Decision within *Handle order* business process in BPEL

Conclusion. BPEL focuses on the control flow with data playing a minor role by mainly shifting the responsibility to the executed services.

3.3 YAWL

YAWL, Yet Another Workflow Language, has been introduced at the Queensland University of Technology in [14]. The main driver for developing this language was to comprise all control flow patterns [15] initially discovered by the Workflow Patterns initiative as novelty in business process modeling. Therefore, the process modeling criteria are completely fulfilled. Visually, YAWL bases on Petri nets, but the semantics are completely new defined, independently from the Petri net semantics.

Data Capabilities and Limitations. Because of focusing on control flow patterns during language development, data was out of scope for YAWL. Therefore, the role of data is not formally specified in YAWL. This includes the absence of data objects and all related concepts. Consequently, data is not modeled in YAWL. However, YAWL is a business process language meant to be executed by the YAWL engine which is part of the YAWL editor and following, data needs to be considered at this stage. In fact, YAWL's data support is completely handed over to the tool. There, some of the also determined data patterns [16]

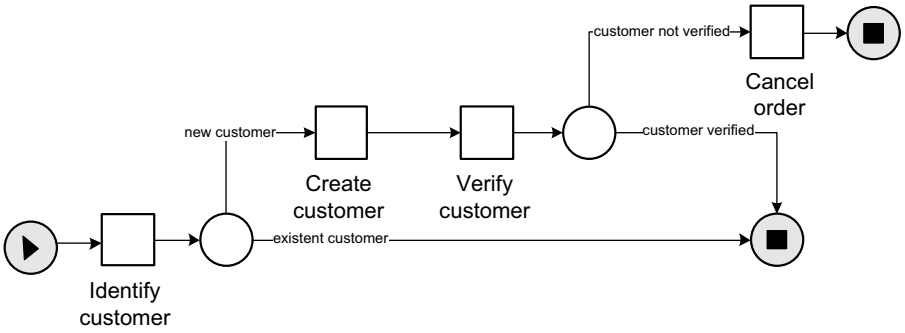

Fig. 6. Extract of business process *Handle order* modeled using YAWL

are considered. Besides the process modeling constructs activities, events, and gateways their connection in terms of control flow, a YAWL process generally contains a global, probably complex process variable as XML structure. A data variable in YAWL is not to be seen as data object in the sense of object orientation as discussed in the framework introduction. Additionally, tasks may work with locally visible variables. Further, input and output behavior of variables can be specified, whereas all data dependencies are hidden and not visualized explicitly. The access to the variables is handled via XPath and XQuery. Decisions may be taken based on variables.

Additional data aspects like states and relations between data are not available in the YAWL editor. The execution semantics follow a formal token flow semantics and is based on both: control flow and data, whereas existence of data is satisfactory for process control.

Example. The YAWL model in Figure 6 covers the *customer processing* step from the scenario. The utilized data is hidden in variables and becomes visible only at decision points for path evaluation. After the customer is identified, this step is completed successfully in case the customer is already registered what can be examined by evaluating the variable customer and check this one against the production customer database. Otherwise, the customer needs to be created and afterwards verified. The failed verification is followed by an order cancellation, whereas an approved verification leads to the next step in the process of the overall scenario.

Conclusion. Specification-wise, YAWL does not support any data-related aspect. However, the YAWL editor provides the concept of global and local variables to allow process execution based on data input and output requirements.

3.4 EPC

Event-driven Process Chains, EPCs, have been introduced by Keller et al. in [17] and are widely accepted by industry. EPC is a graph-based notation, where the

nodes are distinguished into activities, events, and gateways (called connectors), while the edges represent the control flow (called connections). Connectors are distinguished into ORs, XORs, and ANDs. The industry demand motivated the development of extended EPCs, eEPCs, introduced by ARIS. eEPC provide capabilities to model organizational entities, data objects, interfaces to processes and input/output records for functions.

Data Capabilities and Limitations. EPCs capture no data related information, whereas eEPCs, as we argued earlier, are far more expressive. eEPCs model data objects as a graph node that does not belong to the control flow and is optional for process specification. Data object states are not explicitly modeled, but can be deduced from events. Hence, no support of data object life cycles is observed except derivation from events and utilizing them as states. eEPC do not address data object collections, relations between data objects and persistence mechanisms. However, eEPCs enable modeling of undirected and directed relations between data objects and activities. Data flow is visualized through directed relations, where the association direction points to the data object being the activities' input or output. Decisions are taken event-based whereat such event may refer to a data object.

Neither EPCs, nor eEPCs do have formally defined execution semantics.

Example. The example in Figure 7 illustrates the *Customer processing* step. The process assumes the existence of the order modeled as *Order created* event. The subsequent activity *Identify customer* accesses the *Order* data object and outputs the customer details. Next, the XOR gateway represents the decision. If the details match an existing customer, no verification is needed and the process is completed. Otherwise, a new customer entry is created. The new entry is used for verification. The verification updates the *Customer* data object. If the verification succeeds, the customer is verified. Whereas if the verification fails, the order is canceled and the appropriate data object is updated.

Conclusion. Although eEPCs provide facilities for data modeling, data plays the secondary role. The process design is driven by the control flow aspects. At the same time, the semantics of data objects and their use is not formalized and is vague. Original EPCs do not cover data at all.

3.5 UML Activity Diagrams

Activity diagrams became part of the UML specification in version 2.0 [18]. Activity diagrams formalize processes as graphs. Graph nodes are typed into activities, data nodes, and flow nodes. The latter node type is further refined to *XOR* and *AND* gateways. Activities may be triggered by events utilizing the *AcceptEventAction*. Basically, events are supported as activity trigger but not as concrete modeling construct. The edges represent the control flow. Data

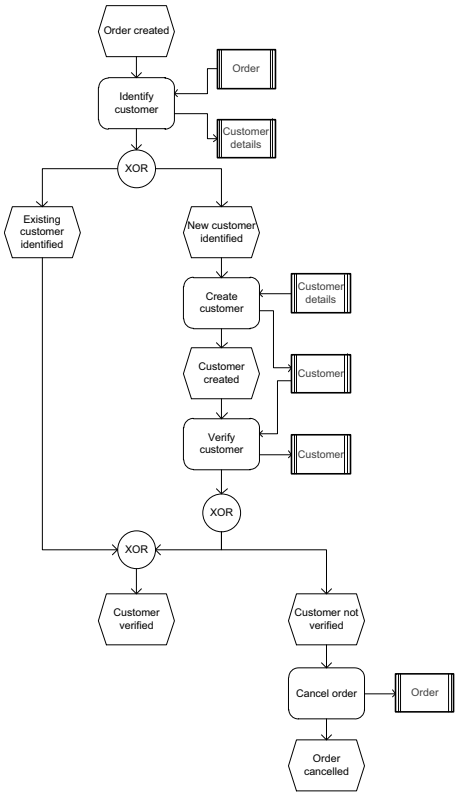

Fig. 7. Extract of business process *Handle order* modeled using eEPC

nodes are distinguished into *pins* and *datastores*, which both are optional in activity diagram modeling. *Datastores* introduce persistence locations in terms of databases. *Pins* allow data object flow modeling and therefore, they specify input and output data conditions of activities.

Data Capabilities and Limitations. These pins represent the association of data objects to activities. Generally, states of data objects are not part of the specification. But due to UML's extensibility, the modeler may annotate each pin with a data object state. Following, state changes of data objects throughout process execution are captured by activity diagrams. However, the object life cycle is not given explicitly, but can be derived based on these conditions. Additionally, data objects' structures of any type cannot be modeled: Part-of and is-a relations as well as data object collections are out of scope for this approach.

The execution semantics follow an informal token flow approach which is driven by control flow aspects and, if specified, data dependencies, which consider the actual state of the appropriate data object. Moreover, data-based decisions are supported.

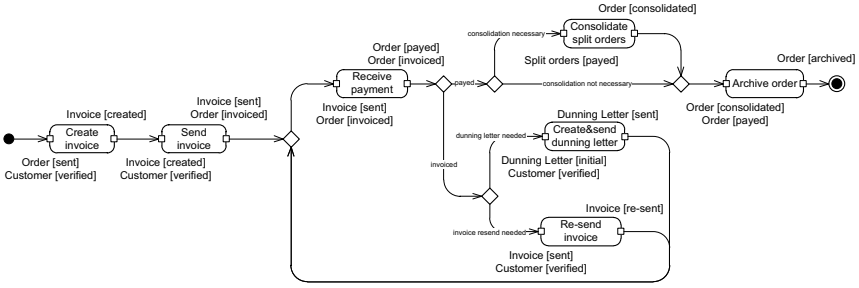

Fig. 8. Extract of business process *Handle order* modeled using UML activity diagrams

Example. The example in Fig. 8 comprises the last two steps of the order process. The first activity utilizes the data objects *Order* and *Customer* in the states *sent* and *verified* respectively. Note that we assume *and* relations between all specified data objects, if they are different, and *or* relations if they are the same. The result of the first activity is the created *Invoice* which is sent to the customer afterwards. This send operation also changes the state of the *Order* data object to *invoiced*. Then, the organization is awaiting the payment from the customer. If the payment is not received within a defined timespan, the activity is completed without any changes to the data objects. If the payment has not been received, the next activity is either to re-send the invoice to remember the customer or to create and send the dunning letter. Which path needs to be chosen here is based on an external decision. If the payment arrives at the organization, the step of *order invoicing* is completed and following, the last step of the order process is initiated. In this step, the first decision is basically based on the historical information of the *Order*. If it had to be split earlier, a consolidation is performed and afterwards the archiving is initiated. Otherwise, the *Order* is archived directly. After archiving, the process step and therefore, the overall process, is completed with the data object *Order* in the business state *archived*.

Conclusion. Data modeling capabilities of activity diagrams have supporting character. Although, data flow is modeled explicitly and therefore, object life cycles can be derived, data modeling is optional in UML activity diagrams. Additionally, the models focus on the specification of control flow aspects.

3.6 BPMN

The initial version of the Business Process Modeling Notation, BPMN 1.0, was introduced in 2004 in [19]. Within years, BPMN 1.X has experienced a large uptake by the industry and matured to the current version 2.0 [2]. BPMN 2.0 evolves with new model elements, diagram types, and model execution semantics.

BPMN has rich expressiveness and provides numerous modeling constructs. In essence, BPMN is a graph-based modeling notation. The graph nodes correspond to modeling constructs as events, activities, data objects, and gateways. Graph edges represent object relations, e.g., control flow, message flow, and associations. Those graph nodes that are related by the control flow relation are referenced as flow nodes, e.g. events, activities, and gateways. The other nodes like data objects are non-flow nodes.

Data Capabilities and Limitations. BPMN enables explicit specification of data objects by associating them with flow nodes. An association indicates that the flow object accesses the data object. While undirected associations capture only the fact of data access, directed associations indicate, whether the data object is read or written. The modeler can also specify data objects as process input and output to show that the specific data object is read from or written to an process external source. A data object can be also associated with a sequence flow to visualize data passing. In this context, it is relevant to mention the message exchange mechanism of BPMN: Different organizations communicate with each other via message exchange. BPMN enables capturing of messages as graph nodes, simplifying modeling of message content.

Additionally, each data object may get assigned a data object state. While the data object life cycle can be derived from the data object states and associations connecting flow elements with data objects, there is no explicit life cycle modeling support. BPMN enables modeling of neither *is-a* nor *part-of* relations of data objects. BPMN 2.0 introduces the concept of data object collections, organizing similar data objects. Additionally, *data stores* are introduced as a data persistence instrument. Notice that the process flow is mainly driven by the control flow elements and data is not a mandatory aspect within the process design phase.

BPMN specification prescribes the execution semantics based on tokens informally. The basic execution order is determined by the control flow. At the same time, data impacts the process execution in terms of data-based decisions on gateways or as prerequisite of activity to allow activity execution. Following, the process is driven by both the control flow and data.

Example. The BPMN example presented in Fig. 9 comprises the *Order invoicing* and *Order completion* steps of the *Handle order* process. First, the invoice needs to be created. This requires data objects *Customer* and *Order* to be retrieved from an external source: *Order* from the *ERP System* and *Customer* from *Customer Database*. Next, the invoice is sent to the customer, the state of the *Invoice* data object is set to *sent*, and the *Order* state is set to *invoiced*. After sending the invoice, the organization awaits the payment from the customer completing the step of *Order invoicing*. If the payment is not received within a defined timespan, the *Receive payment* activity is terminated. Based on environmental aspects, either the invoice is re-sent or the dunning letter is created and sent to the customer, each utilizing the specified data objects. After reminding the customer, the *Receive Payment* activity is enabled again. If the payment is received, prior

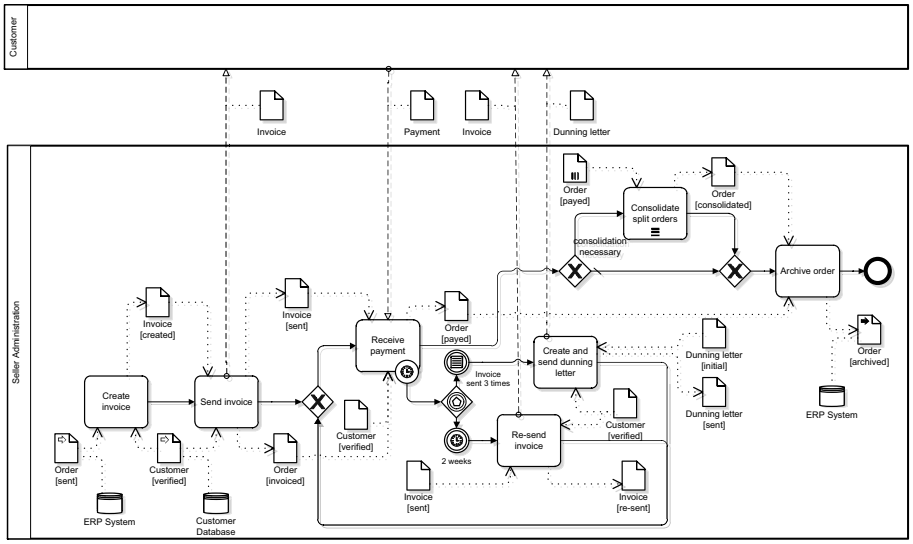

Fig. 9. Extract of business process *Handle order* modeled using BPMN

split orders need to be consolidated. If no split occurred earlier, the default path is taken. The existence of a split order leads to consolidation realized as a sequentialized multiple instance activity where each instance adds one list element to the consolidated *Order*. The final step is archiving the *Order* within an *ERP System*.

Conclusion. BPMN generally provides a high data awareness in terms of data object specifications, implicit data dependencies, and derivable object life cycles. However, data modeling is optional in BPMN. Following, BPMN remains an approach focusing on control flow with modeling constructs of this type being the only mandatory ones. Therefore, data objects remain second class modeling constructs supporting the control flow.

3.7 Corepro

Corepro is a framework providing an approach for enacting and changing data driven process structures belonging to the same complex process structure. Basically, there this complex process structure has an initial state, several intermediate states, and one final state with respect to the business goal. Between these states, processes perform business related steps to achieve the subgoals (intermediate states), whereas branching is allowed. The Corepro approach is presented in [20] and [21].

The core idea is to automatically create data driven process structures. Corepro is a four-step-approach. First, the data model is defined. The data model comprises the involved data objects and their part-of relations determining the

dependencies independently from specific representations of the data object, i.e. instances. Afterwards, the life cycle coordination model is determined which comprises the data object life cycles of each data object specified in the data model and its dependencies for state transitions. This also includes dependencies between states of life cycles of different data objects. Therefore, the inter-relation between all involved data objects is defined. These both steps influence the model level and act as schema for the instance level tackled by the remaining steps. Step three deals with the definition of actual data structures, i.e. deriving dependencies for actually utilized data objects from the model level. Based on these data structures and the prior defined life cycle coordination model, the data driven process structures are created automatically in step four. Following, due to the given complex process structure and the defined data aspects, process execution is highly influenced from both: Control flow and data. Based thereon, support for all process modeling criteria can be derived.

Data Capabilities and Limitations. The execution of the processes being part of the complex structure leads to state changes within the life cycles towards the ultimate business goal. Within these processes, events and gateways may be utilized. Additionally, gateways are supported on a higher level by the created data driven process structures. For instance, parallel execution of different lifecycle fragments and their according processes may occur.

Due to the high data involvement and the object orientation during creation of the data driven process structures, data modeling in terms of data object modeling is covered by Corepro. As mentioned above, each data object has assigned a life cycle comprising all states of the object, whereas these data objects are not categorized in collections nor the concrete storage locations are specified. Step one and three describe the part-of relations of data objects. In contrast, is-a relations are not part of the approach. Finally, the data object definitions are mandatory and therefore, they need to be completely specified at all times.

In Corepro, relations between the data objects and the appropriate processes are defined by stating the process or process fragment to be executed to achieve a state transition. Therefore, a direct coupling of activities and state transitions does exist. As aforementioned, the execution is driven by control flow and data, whereby the states of the data objects and their transitions play the main role. Execution semantics neither follow a formal nor an informal token flow semantics.

Corepro also allows process adaptation during run-time. Therefore, the modifications made to data structures are automatically translated into the data driven process structures. These modifications include but are not limited to changes like adding or deleting data objects, changing relations of data objects, or adding external state transitions. For not yet activated elements of the process, the authors provide simple rules for the adaptation. For modifications of already started instances, specific correctness criteria are formulated and need to be fulfilled for allowing the intended adaptation.

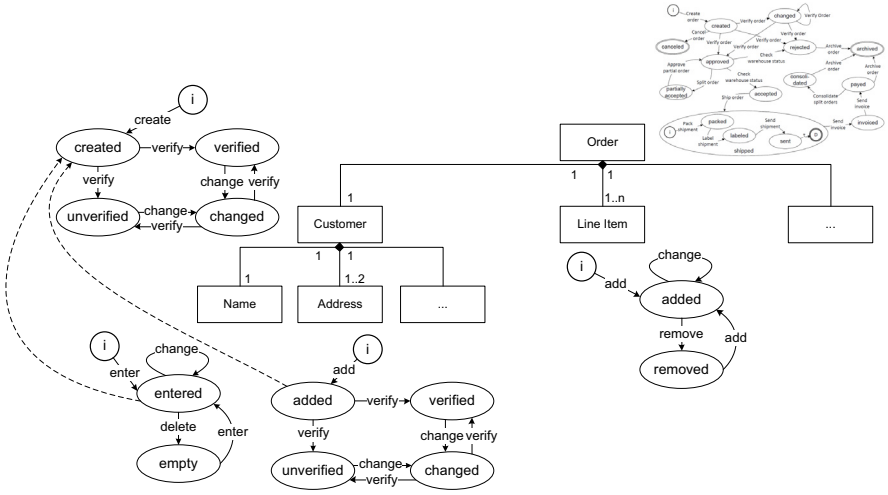

Fig. 10. Extract of business process *Handle order* modeled using Corepro

Example. The example presented in Fig. 10 consists of two parts: The part-of relations between the super data object *Order* and its sub data objects and the data object life cycles for each element of the part-of relation. For complexity reasons, we only show a part of the whole model. The notation for part-of relations is similar to UML class diagrams. Each *Order* consists of exactly one *Customer*, of one to n *Line Items*, and of further not presented sub data objects. Analogously, the *Customer* also exists of sub data objects. The data object life cycles are given as State charts [22]. Each is placed next to the data object it describes.

After definition of these single State charts and the composed data objects, linkings between the states of each data object need to be determined. In this example, the *Customer* can reach the state *created* only, if the *Name* and *Address* are *entered* or *added* respectively as visualized exemplarily by the dotted arrows. Further sub data objects may provide more dependencies than just the two mentioned. These dependencies highly drive the process execution and therefore, influence the process control equally as the process structures.

Conclusion. Corepro closely relates control flow and data structures within a complex process structure. Following, process execution is equally influenced by data structures and dependencies as well as control flow aspects.

3.8 Business Artifacts

Compared to traditional process modeling languages, in Business Artifacts, the focus changed from actions taken (control flow) to data objects on which actions are performed. Initially introduced by Nigam and Caswell in [23], the approach has been discussed in a series of papers, for instance [24,25], and thoroughly

formalized by Bhattacharya et al. in [3]. The formalization identifies Business Artifacts, schemata, services, and business rules as main concepts of the approach.

The main idea of Business Artifacts is to achieve a closer coupling between data and processes. Generally, Business Artifacts are information entities capturing business process goals including the path and information to achieve them and to enable judgment of the goal accomplishment. Examples for Business Artifacts are the waiter's guest check in a restaurant presented in [23] or the tracking form of a postal organization. The tracking form, for instance, contains the goal to deliver the shipment to the recipient, the intermediate steps to be handled, the information needed to manage the delivery, and the final signature to indicate a successful delivery.

Business Artifacts modeling relies on three constructs: Stages, guards, and milestones. A rectangle with rounded corners represents the stage which maps to a collection of activities to be performed to achieve a milestone. The stages can be hierarchically structured. A rotated square represents the guard which maps to an event or a condition. A guard enables a stage. A circle represents the milestone which maps to an event or a condition as well. A milestone is considered to be a business relevant goal and completes a stage once it is achieved. Decisions are not supported with modeling construct. But these are captured by the guards and milestones which guide process execution.

The relations of the formalized aspects are as follows: Services correspond to activities. A service acts on Business Artifacts manipulating the content of the informational model and changing the artifacts' life cycle states. Business rules determine the use cases and conditions which need to appear for allowing a service to access a Business Artifact. Following, control flow is implicitly represented by these modeling constructs.

Notice that in the new revision process of the Business Artifacts approach, Business Artifacts are renamed to business entities with life cycles. But we will still use the term Business Artifacts in the upcoming discussion.

Data Capabilities and Limitations. Within a process, one Business Artifact is the key object that steers process execution. Therefore, it holds the aforementioned executional information for the whole process. This also includes references to further Business Artifacts, the key object involves in process execution. These referenced Business Artifacts only hold the information necessary for their purpose instead of all process information.

According to the formalization in [3], a Business Artifact is a connection of two models: The informational model and the life cycle model. The informational model describes the artifact properties relevant to the process, e.g. a database schema. The life cycle model defines the states and allowed state transitions, e.g. via State charts or Petri nets. The life cycle states correspond to high-level states on the path towards the business process goal, i.e. the intermediate steps mentioned above. Further data capabilities are not supported. Linkage between process and data aspects is achieved via the services.

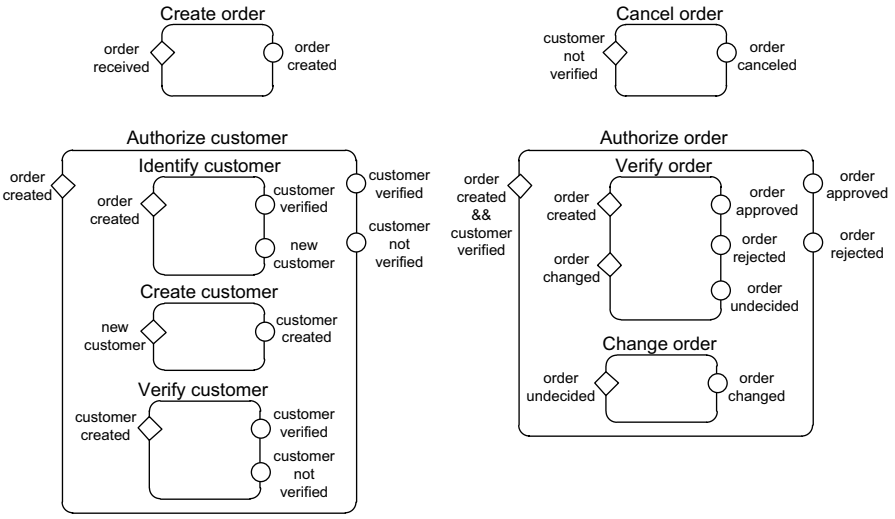

Fig. 11. Extract of business process *Handle order* modeled using Business Artifacts

Process execution does not follow a predefined order, but bases only on the availability of Business Artifacts in the specified state containing defined content. Token flow semantics of any kind cannot be associated with Business Artifacts. Data-based decisions are supported as basically all decisions are taken based on data within this approach.

Example. The key object within the *Handle order* process is *Order*. Fig. 11 represents the first three steps and includes the linkage to a second artifact, the *Customer*. If the external *order received* event is observed, the stage *Create order* is performed leading to the event *order created*. After observing that event, the *Authorize customer* stage is activated. This stage consists of three substages which might, but do not necessarily need to be performed. First, *Identify customer* is performed and based on the achieved milestone, the stage is either completed or the stages *Create customer* and *Verify customer* are executed in this order. Afterwards, the stage *Authorize customer* is definitely completed with either milestone. In case the customer could not be verified, the stage *Cancel order* performs the completion of the process. Otherwise, the stage *Authorize order* performs the order verification and if needed an order refinement (stage *Change order*). After each refinement, the verification will be performed again. Theoretically, this iteration can be performed unlimitedly. Based on the verification result, the appropriate next step of the *Handle order* process is triggered.

Conclusion. Process execution does not follow a predefined order but depends on the availability of Business Artifacts in a specific state or Business Artifacts

holding certain information. Hence, this approach utilizes data as first class modeling construct which alone drives the business process execution.

3.9 Document-driven Workflows

Document-driven workflows have been introduced by Wang and Kumar in [26]. Within their approach, no control flow is specified explicitly. The execution of a process is driven by documents, i.e. data objects, which are the input and output to single tasks. Following, the specification of them is mandatory. Based on the existence and availability of a specific document (or part of a document), execution of a task can be started. Thereby, they distinguish between so-called hard and soft constraints which relate to data dependencies (strict dependencies guiding process execution without exception) on the one hand and to business policy constraints (execution order not fixed and depends on organizations' decisions) on the other hand. Therefore, we consider process control of this approach to be always data-driven.

The underlying framework relies on four layers: schema, runtime, scheduling, and application layer. Within the schema layer, workflow processes and the appropriate tasks and resources (in the sense of humans and data) are defined conceptually. Events are not part of this approach and gateways only support data-based decisions. The runtime layer deals with process execution and determines how tasks are started and ended. For each case, one process instance is instantiated and the tasks are executed driven by data considering the hard and soft constraints. Following, the control flow is implicitly given by by the documents and the gateways. The scheduling layer ensures the assignment of resources to the tasks. Finally, the application layer provides an abstraction layer between applications and workflows and links the application data to documents.

The authors position their framework to be very useful especially in the field of ad-hoc processes because flexibility, i.e. execution structure changes, is introduced easily by changing the constraints, i.e. input and output documents, instead of completely rerouting and remodeling the process.

Data Capabilities and Limitations. The Document-driven workflow approach fundamentally deals with data. The existence of data objects fulfilling specific requirements is the criteria which influences process execution. Thereby, an informal token flow semantics can be applied. Following, each task needs to get assigned at least one document to work with. This assignment is a directed relation. For modeling, this means that this approach focuses on mandatory input-output-representations of the utilized data objects. The access to documents may occur concurrently. The framework allows this via three locking mechanisms (shared, exclusive, append) and via the opportunity to split and duplicate documents. However, a large amount of concurrent data changes may lead to conflict resolution issues, because the probability to access the same part of a document increases. If two tasks access identical parts of a document, synchronization between these accesses needs to be achieved, whereas the framework

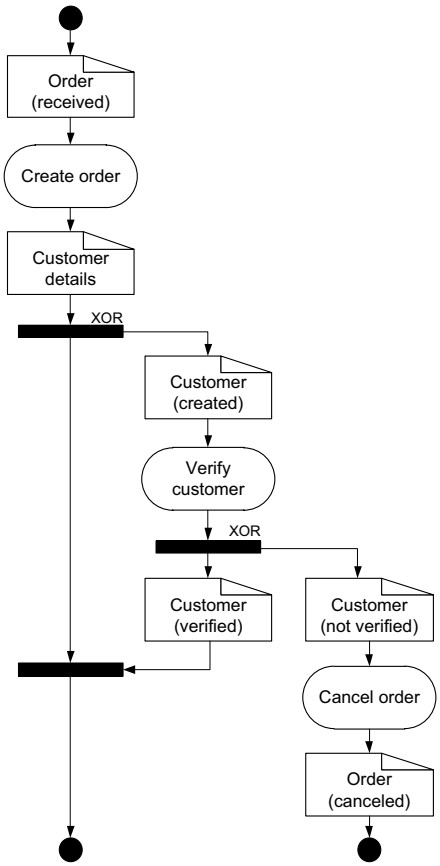

Fig. 12. Extract of business process *Handle order* modeled using Document-driven workflows

does not provide such means. Besides splitting, merging of prior split documents is possible. Therefore, the framework allows data object aggregation and part-of relations are supported, whereas is-a relations are out of scope as well as the support of document collections and persistence mechanism modeling. In contrast, the association of states to a document is available by annotating the document with the state. However, states of documents and following data object life cycle modeling have not been in focus of this approach.

The visualization of this approach is difficult. As the data-based constraints are modeled using connectors, large models run into complexity and understandability problems and the model itself ends up as spaghetti model.

Example. The example presented in Fig. 12 deals with the very beginning of the order process until the customer who placed the order is verified or the order

canceled due to an unverified customer. First, the order is received. Afterwards the order must be created and the *Customer details* be derived as these are the information needed for the next step of the process. Based on the *Customer details*, the decision is taken and either this process step be completed or a new customer created. This new customer must be verified to either end again the process step leading to the next one or to refuse the order which results in a cancellation, which requires the unverified customer as input or rather precondition and which outputs the order in a final state, i.e. no more work will be performed on this document.

Conclusion. Document-driven workflows are purely data-driven. Following, data is the first class modeling construct. Process execution relies on data dependencies and business policies only.

3.10 ADEPT

Reichert and Dadam introduced ADEPT to support dynamic changes in business process models [27]. The approach formalizes process models as graphs: nodes correspond to activities and gateways, while edges represent the control flow. Activities are shown with rectangles connected by the directed edges - the control flow. Events are not part of this approach. ADPET allows only block-structured process models. As the introduction of changes to process models in general and to running process instances in particular highly depends on the process data, Reichert and Dadam elaborate on the modeling of data, which is represented by variables being input or output of an activity, what is modeled by means of data links. In addition, the authors introduce correctness criteria for data flow schemata for helping to judge, if the required process change is valid or not.

Data Capabilities and Limitations. Data is exchanged between activities by means of global variables. A variable contains a unique identifier and the domain. In general, the user can define data objects, specify, whether a data object is an activity input or output, and define the data object value. Additionally, data objects must be modeled in ADEPT. But besides, no further data aspects like relations between data objects are covered. The concept of a data flow schema allows to model the current state of a process. A data flow schema comprises the set of all data links, i.e. associations between activities and data elements. Based on the current execution of data links, the current status of the process can be derived. For correct execution, ADEPT assumes that all specified inputs are supplied prior execution and that all specified outputs exist after execution. Additionally, unsynchronized activities are not allowed to write the same data element. These developed correctness criteria enhance traditional data modeling capabilities.

Execution of an ADEPT model follows an informal token semantic and is driven by control flow as well as data aspects. Thereby, only data existence is considered. The gateways mentioned earlier are operated in three ways and one is the capability to base decisions on values of a data object.

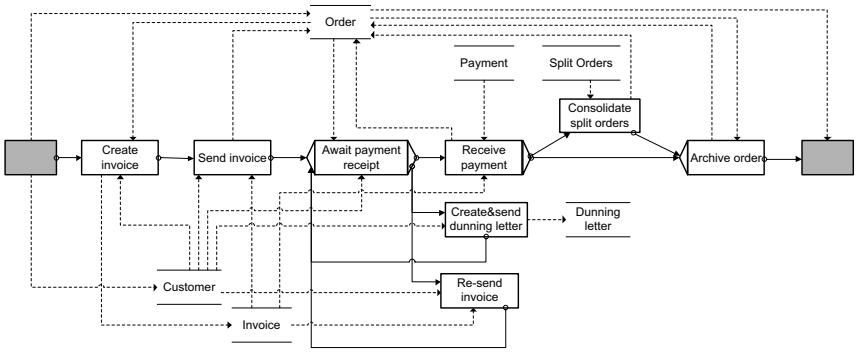

Fig. 13. Extract of business process *Handle order* modeled using ADEPT

Example. Fig. 13 provides an example representing the last two steps of the overall order process scenario. The data objects *Order* and *Customer* are process inputs as indicated through the association to the first gray rectangle. They are utilized to create the *Invoice* data object which is part of the input of the two upcoming activities *Send invoice* and *Receive payment*. The latter succeeds *Await payment receipt* which acts as buffer to receive the payment by the customer. If the payment is received, the according *Payment* data object exists process execution is continued along the main path. Otherwise, if the payment does not reach the organization, one out of two activities follows the waiting period. Based on environment information, either the invoice should be resent or the dunning letter created and sent. The latter shall happen, if two reminders have been sent. After receiving the payment, one of two follow-up tasks are executed. If there exist split orders, they will be consolidated and afterwards the consolidated order archived. Otherwise, the single *Order* is archived directly. The last aspect to be mentioned is the association of the final *Order* data object to the last gray rectangle to symbolize that this data object is the process output.

Conclusion. ADEPT is an approach focusing on changes to process models as well as process model instances. Therefore, it focuses on data dependencies and checks these dependencies to verify requested process changes. The general data support is limited to the linkage of data elements to activities as input or output. Additionally, process control is steered by the existence of data.

3.11 Case Handling

Case handling has been introduced by van der Aalst, Weske, and Grünbauer in [4]. The authors designed a process modeling approach intended to capture knowledge-intensive business processes. Within such processes, process control needs to be flexible to improve the process' execution by reducing the overall execution time though parallelism. Therefore, Case handling relies on process control by data

and its dependencies. In this approach, data objects are represented as forms with several data fields. The dependencies are specified between the data fields of different forms. Activities visualize the actions to be performed utilizing the forms for achieving the case goal. However, process execution does not follow a predefined order of these activities. Instead, an activity is triggered when the needed data is available, i.e. the specified data fields are filled by prior activities. These fields may be part of different forms on which work may be performed by different activities. Events and gateways are not supported in this approach. However, gateways can be regarded as considered as for instance parallelism exists if the data dependencies are fulfilled for several activities simultaneously or overlapping in execution. Decisions cannot be regarded.

Data Capabilities and Limitations. [4] introduced the meta model for Case handling as well as its formal framework. The framework defines the main concepts of Case handling and specifies case execution semantics. The formalization of a case distinguishes two object types: activities and data objects. Activities can be in the precedence relation, which corresponds to the control flow. However, activity execution is driven by data only and does not follow any token flow semantic. More specific, the existence of data, i.e. information within the data fields of forms, drive the execution. Therefore, each activity is linked to at least one data object, i.e. form. Following, data objects are mandatory modeling elements. For these links between activities and data objects, two types of relations exist: *Restricted* relation and *mandatory* relation. A mandatory relation assures that a data object is available, i.e. the specific data fields of the form are filled, after the according activity completes. However, it does not mean that the according activity is responsible for adding the information. It might be done in an earlier step of the process. In contrast, a *restricted* relation indicates that the according activity is responsible of filling the specific form fields. Further, for each activity the mapping *condition* specifies data objects and their values that enable execution of the activity. The activities require user interaction to enter the information into the form fields.

Additionally, the forms are not linked to or consist specific states and therefore, object life cycles and state transitions cannot be captured. Comprising data objects in a collection, defining persistence aspects as well as relations between forms are not captured by Case handling.

Example. The example visualized in Figure 14 contains the activities of the *Customer processing* step. This includes the activity *Create customer* and the ad hoc subprocess *Verify customer* from which all tasks need to be executed. With respect to the Case handling approach, execution of each of the three tasks can be started even if the customer creation is not completed but the data fields in the form are filled instead. For instance, *Verify cash balance* can be activated as soon as the data fields *Acc No* and *BIC* are filled. Following, parallelism and therefore throughput is highly increased. The *restricted* relation indicates that the customer form is exclusively filled by the *Create customer* activity.

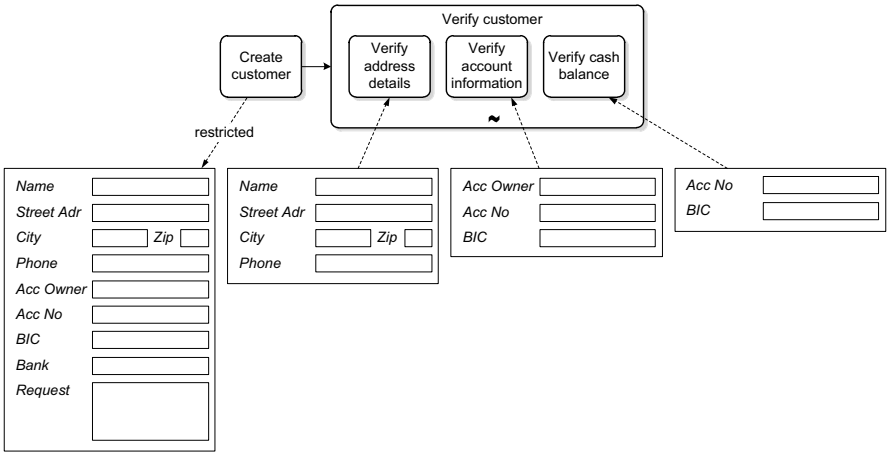

Fig. 14. Extract of business process *Handle order* modeled using Case handling

Conclusion. To address flexibility in process control, Case handling considers data as a first class modeling construct and studies the relations between activities and data in a business process. The activity execution order depends on the availability of required data objects, i.e. on completed data fields of according forms.

3.12 State Charts

State chart diagrams represent the behavior of usually one object, e.g. the life cycle of a data object. They have been introduced by Harel in [22] to extend state transition systems like automatons [28,29] with means to model hierarchies, concurrency, and communication. State charts are a graph-based notation, where the nodes correspond to states of an object and the edges to state transitions. A state transition identifies the action which may be performed next on the object described by the State chart. Therefore, the actions map to activities with respect to our evaluation framework. The actions comprise the full set of operations which may be performed on the appropriate object. Each state node can be a container for several other state nodes to specify hierarchies. Entry and exit points are possible from each contained state. Concurrency is modeled through subgraphs divided by a continuous line within one container. Except the aforementioned activities, no criteria from the group of process modeling capabilities is supported by State charts.

Data Capabilities and Limitations. A State chart represents the life cycle of one data object. Following, data object specification is mandatory for this approach. Therefore, each diagram is linked to exactly one data object and visualizes each state transition and the appropriate action which are associated to the data object. However, the life cycles of two different data objects can be aligned by utilizing so-called extension points. These synchronization points get

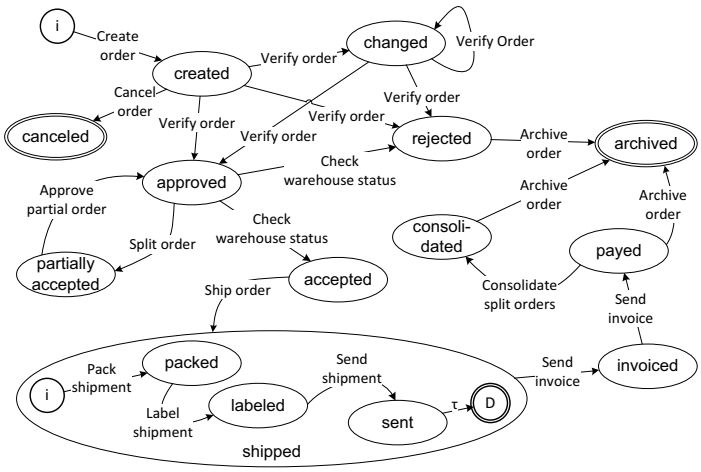

Fig. 15. Extract of business process *Handle order* modeled using State charts

attached specific dependencies which include notification of another State chart that a certain state is reached or waiting for such a notification from another State chart before the next action can be performed. Following, dependencies between data objects are covered by State charts. The states itself can be organized into a hierarchy, but the data objects represented by State charts cannot. Besides the already mentioned data capabilities, State charts do not support others. Following, part-of and is-a relations as well as collections and persistence mechanisms are not supported. Execution semantics are also not defined for State charts.

Considering newer specifications of State charts like the state machines part of the UML 2 documentation [18], some additional features are supported. For instance, UML state machines allow annotations to edges specifying conditions for actions. Using the terminology *event[guard]/action* of these state machines, an action can only be performed after the specified event occurred and the guard condition is true. Following, UML 2 state machines support implicit event modeling as well as data-based routing decisions.

Example. The example in Fig. 15 presents the valid state transitions of the *Order* data object of the complete order process introduced in the beginning of this section. After initialization, the activity *Create order* leads to the state *created* which subsequently allows two different activities to be executed from the *Order* data object's point of view. These activities do not necessarily follow directly in the process model, but these are the next ones which utilize the *Order*. Therefore, either *Cancel order* ends the process by changing the state into the final one *canceled* or *Verify order* leads to one out of three states based on the result of the verification. The outcome *changed* requires rework on the *Order* and finally reactivates the activity *Verify order* allowing the same upcoming states as before. A rejection is followed by the archival activity and a proper

termination in the state *archived*. The expected path leads to *approved*. The following activity *Check warehouse status* may lead to a rejection with equal follow ups as mentioned before and to *accepted* as expected path. Additionally, the *Order*'s state may transition to *partially accepted*. At this point in time, a copy of the current situation need to be done covering the part of the *Order* which got accepted and the part which need further work. The notation cannot handle this as compositions are not supported. However, assuming this partition, the activity *Split order* leads to *accepted* for the first part or it leads to *approved* for the other currently not handleable part. Afterwards, for this now as full *Order* handled part, the warehouse checking is performed again with the same consequences as above. An *accepted Order* gets shipped next which is indicated by the composed state at the bottom of the example. Shipping is composed of packaging, labeling, and sending the *Order*. Following, the last two steps of the overall order process are following. *Send invoice* leads to *invoiced* followed by *payed*. The last decision goes back to the partial accept earlier and consolidates the different split *Orders*. Basically, the different copies of State charts for one process instance need to be integrated into one which again is not supported by the notation. Independently whether the *Order* needed to be consolidated or not, the last activity of the process is *Archive order* and it leads to the final state *archived* which is the same as for the rejected *Order*s. However, the distinction is done due to process context information assigned to the *Order*s. Within this example, we used the notation introduced by Harel. But UML 2 state machines can be created similarly.

Conclusion. State charts usually describe the life cycle of one data object only. But it is possible to connect State charts of different data objects in specific synchronization states to visualize dependencies between these data objects. Altogether, State charts are highly focused on data and represent the actual evolution of data objects – usually from the business point of view towards the final business goal.

4 Discussion of Evaluated Approaches

This section compares the discussed process modeling languages. We not only summarize the modeling languages' features, but cluster also the languages identifying their common properties.

4.1 Modeling Language Clustering

The evaluated process modeling languages can be clustered according to different criteria. We introduce three clustering schemata. First, we distinguish between industrial and academic approaches according to their place of origin. Second, we cluster the approaches with respect to their execution semantics. And lastly, we consider the degree of data support as clustering criteria.

Clustering According to Origin Industrial approaches are widely used in commercial contexts, whereas the academic approaches are designed at universities or research facilities of companies but lack a high degree of practical usage. Among the evaluated approaches, we consider *BPEL*, *BPMN*, *EPC*s, *UML activity diagrams*, and *Workflow net*s to be industrial approaches. The others are academic ones.

Clustering According to Execution Semantics We distinguish three approach clusters according to the execution semantics: *control flow driven*, *data driven*, and a mixture of both named *control and data flow driven* cluster (compare with criteria 19 to 21 in Section 2). As State charts and EPCs have no execution semantics specified, they belong to none of the three clusters. Table 1 visualizes the results.

Control flow	Control flow and data	Data
Workflow nets	BPEL	Business Artifacts
	YAWL	Document-driven WF
	UML AD	Case handling
	BPMN	
	Corepro	
	ADEPT	

Table 1. Clustering of business process modeling languages based on execution semantics

Clustering According to Data Support This classification organizes process modeling languages according to their general data support capabilities. We cluster all approaches based on their capabilities at design- as well as run-time. Modeling languages without specified execution semantics are only considered for clustering at design-time. Fig. 16 sketches the dependencies between these clusters. The *control flow driven* and the *data driven* clusters indicate both extremes and they meet in the middle for the equally *control flow and data driven* cluster.

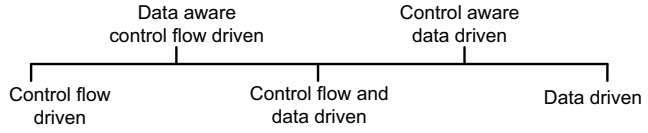

Fig. 16. From control flow to data driven processes

Table 2 outlines the applied clustering criteria. Thereby, run-time clustering considers all four sections of the table, whereas design-time clustering only considers criteria from table sections one to three. All criteria marked with a "+" need to be supported for the specific cluster, whereas all criteria marked with a "–" must not be supported for belonging to that specific cluster. The marking "*o*/–" indicates that a criterion may be implicitly but is not explicitly supported. Markings "*x*" indicate an either or support between the criteria marked that way. Lastly, all empty entries denote that the support can be of any strength as it does not influence clustering.

Criteria	Cluster	Control flow driven	Data aware control flow driven	Control and data driven	Control aware data driven	Data driven
1 Activity Modeling		+	+	+	+	
2 Event Modeling		+	+	+	+	o/–
3 Gateway Modeling		+	+	+		
4 Control Flow Modeling		+	+	+	o/–	o/–
5 Data Modeling		–	+	+	+	+
6 Data Object Modeling				+	+	+
7 Data Object State Modeling				+	+	+
13 Mandatory Data Objects in Design Phase?		–	–	+	+	+
14 Modeling of Typed Relations		–	+	+		
15 Associations b/t. State Transitions and Activities		–		+	+	
18 Data-based Decisions		–	+	+	+	+
19 Execution via Control Flow Only		+				
20 Execution Always via Data					x	+
21 Execution via Control Flow and Data			+	+	x	
23 Process Control via Data Object States						+

Table 2. Overview of clustering criteria

The *control flow driven* languages are characterized by a complete support of *process modeling capabilities*; process control is driven by control flow only. Due to the data unawareness, data, data-based decisions, and relations between data and traditional process modeling constructs are not supported by these languages.

For the *data aware control flow driven* languages, we relax the requirements with respect to capturing of activities and routing decisions, i.e. gateways, combined with a explicit control flow representation. Additionally, data linked to activities is a model part. Decisions may rely on data and the process control is affected by control flow and data.

The cluster comprising approaches driven by *control flow and data* is characterized by a support of all traditional process modeling constructs and basic data constructs comprising 1) data objects as a mandatory process model part, 2) data object states, and 3) associations between activities and state transitions of data objects. The process control is driven by control flow and data. Decisions based on data is obligatory.

Control aware data driven approaches support the mandatory existence of data objects, associations between activities and state transitions, states being assigned to data objects, and data-based decisions. Process control is driven either by data, or by a mixture of data and control flow. Control flow related aspects needed to be comprised in an approach being part of this cluster are activity and event modeling capabilities. However, control flow modeling is not explicit in contrast to the aforementioned clusters.

Finally, the *data driven* cluster concentrates on mandatory data object modeling. These data objects get states representing the evolution and decisions taken with respect to data. The process control is driven by data object states, rather than data object existence. Regarding the traditional process modeling constructs, event and control flow modeling is not explicitly supported.

Based thereon, Table 3 presents the clustering of all evaluated modeling languages. A "+" refers to an assignment of the approach to the specific cluster, whereas a "–" indicates that the approach is not part of that cluster. Two symbols parted by a slash ("/") correspond to different cluster assignments at design- and run-time due to missing execution semantics for *EPCs* and *State charts*. An "*o*" indicates that we assign an approach to a cluster, yet it does not fulfill all the cluster criteria. For instance, ADEPT is assigned to *Data aware control flow driven* cluster despite the missing support of event modeling and the link between data state transitions and activities. In general, one can realize the reasons for "*o*" comparing Table 2 and Table 3.

Cluster	Approach	Workflow nets	BPEL	YAWL	eEPC	UML AD	BPMN	Corepro	Business Artifacts	Document-driven WF	ADEPT	Case handling	State charts
Control flow driven		+	–	–	–	–	–	–	–	–	–	–	–
Data aware control flow driven		–	+	+	+/–	+	+	–	–	–	o	–	–
Control flow and data driven		–	–	–	–	–	–	+	–	–	–	–	–
Control aware data driven		–	–	–	–	–	–	–	+	–	–	–	–
Data driven		–	–	–	–	–	–	–	–	+	–	o	o/–

Table 3. Clustering of business process modeling languages based on data support at design-time/ run-time

The majority of the approaches is aware of data and incorporates it for process execution and process control as visualized in Table 3. The industrial approaches are driven by control flow, while data is considered as a conditional factor for enabling activities. Data driven approaches, aware as well as unaware of control flow, currently play rather an academic role.

Evaluated Aspect	Modeling Language	Workflow nets [10]	BPEL [11]	YAWL [14]	eEPC [17]	UML AD [18]	BPMN [2]	Corepro [20,21]	Business Artifacts [23,3]	Document-driven WF [26]	ADEPT [27]	Case handling [4]	State charts [22,18]
1 Activity Modeling		+	+	+	+	+	+	+	+	+	+	+	+
2 Event Modeling		+	+	+	+	+	+	+	+	–	–	–	o
3 Gateway Modeling		+	+	+	+	+	+	+	o	+	+	–	o
4 Control Flow Modeling		+	+	+	+	+	+	+	o	o	+	+	–
5 Data Modeling		–	+	+	+	+	+	+	+	+	+	+	+
6 Data Object Modeling		–	–	–	+	+	+	+	+	+	–	+	+
7 Data Object State Modeling		–	–	–	o	+	+	+	+	+	–	–	+
8 Data Object Life Cycle Modeling		–	–	–	o	o	o	+	+	–	–	–	+
9 Modeling of Data Object Collections		–	–	–	–	–	+	–	–	–	–	–	–
10 Persistence Mechanism Modeling		–	–	–	–	+	+	–	–	–	–	–	–
11 Data Object “part-of” Relation Modeling		–	–	–	–	–	–	+	–	+	–	–	–
12 Data Object “is-a” Relation Modeling		–	–	–	–	–	–	–	–	–	–	–	–
13 Mandatory Data Objects in Design Phase?		–	–	–	–	–	–	+	+	+	+	+	+
14 Modeling of Typed Relations		–	+	+	+	+	+	+	–	+	+	+	+
15 Associations b/t. State Transitions and Activities		–	–	–	–	+	+	+	+	–	–	–	+
16 Formal Token Flow Semantics		+	–	+	–	–	–	–	–	–	–	–	–
17 Informal Token Flow Semantics		–	–	–	–	+	+	–	–	+	+	–	–
18 Data-based Decisions		–	+	+	o	+	+	+	+	+	+	–	+
19 Execution via Control Flow Only		+	–	–	–	–	–	–	–	–	–	–	–
20 Execution Always via Data		–	–	–	–	–	–	–	+	+	–	+	–
21 Execution via Control Flow and Data		–	+	+	–	+	+	+	–	–	+	–	–
22 Process Control via Data Existence		–	+	+	–	+	+	–	–	+	+	+	–
23 Process Control via Data Object States		–	–	–	–	–	–	+	+	–	–	–	–

Table 4. Comparison of the evaluated business process modeling languages

4.2 Feature Discussion

Table 4 summarizes the results of our evaluation. It witnesses that the evaluated approaches are very diverse. However, we observe that the data awareness of modeling languages increases. Among the evaluated approaches, Workflow nets are the only completely data unaware approach. With their focus on utilizing Petri nets for process modeling, data aspects have never been in scope.

In Table 4, a "+" symbol indicates a feature support, a "–" represents a lack of feature, and an "*o*" represents an implicitly supported feature. Implicitly refers to situations where, for instance, the necessary information can be derived from the process model, but the feature is not part of the specification.

Control flow aware as well as control flow unaware data driven approaches lack support for traditional modeling constructs as their support of the first group of criteria is below average for each representative. The support of data aspects is two fold. Basically, non of these approaches achieves the highest support, but except Case handling, they are above average and close to the top score. However, data support is rather primitive. Data support in terms of objects is obligatory, but data object relations and data object collections are barely supported.

The data aware control flow driven approaches are complete in terms of traditional process modeling, but vary significantly in terms of data support. All have in common that data is utilized within the business process – either as variable or as object. The approaches, which capture objects, also support states as mapping to business milestones and goals. Generally, these approaches have the same issues as the data driven pendants: Complex data aspects, especially relations, are rarely supported. Following, shallow data support is widespread in process modeling approaches. But data modeling including amongst others hierarchies, specifications, and aggregations is mostly not supported. A complete support is reached by none of the evaluated modeling languages. Therefore, we identify a need for data-related features. Altogether, Corepro is the only approach coupling the worlds of traditional process and data modeling closely as both aspects are explicitly considered for process specification and process control.

BPMN, Corepro, and UML activity diagrams are the most feature complete approaches according to the introduced framework, see Section 2. These three approaches support simple data modeling completely. But with respect to the complex criteria also these approaches only achieve a limited support, i.e. one to two complex out of four complex data modeling criteria. Following, even for these most complete approaches, the aforementioned need for features with respect to data modeling capabilities is observed.

However, not all features identified in the evaluation framework from Section 2 need to be supported by one modeling language for two main reasons: First, complexity increases with a higher degree of criteria coverage. And second, most modeling languages are created to cope with a specific problem domain and therefore, the scope is limited. With respect to complexity, in [30,31], the authors discuss optimal and maximum numbers of modeling constructs. They identified 20 as an optimal number of modeling constructs. But full data modeling support within a process modeling language increases the complexity from the point

of view of visualized elements for a business scenario. This holds especially for connectors and associations representing all data dependencies. A solution for the complexity issues might be the introduction of data-oriented views on process models or the specification of standardized interfaces between process and data models. With respect to the scope of modeling languages, data aspects only need to be supported to a degree the scope comprises.

For instance, we recognized a stronger focus on data in approaches like Case handling and ADEPT which are made for business processes requiring a large degree of flexibility. A use case application is amongst others the field of dynamic activity reordering or parallelizing for higher throughput during run-time. Because of their strong focus, these approaches do not necessarily comprise a large set of covered criteria from our evaluation framework.

Approaches utilized to represent business processes meant to be implemented provide a strong data criteria coverage due to the need of considering data dependencies during process execution. Lastly, independent from specific modeling approaches, data consistency can be ensured easier by fully modeling and specifying the dependencies.

However, data capabilities are not the only factor in choosing an approach to use within a project or an organization. Based on the field of application and the scope of process modeling in a particular case, the decision should be taken. For instance, the activity driven approaches, data aware or unaware, may the pick of choice regarding documentation of business processes. Data driven approaches are more applicable in areas which require flexible process redesign, probably even during process execution, or fields of knowledge intensive processes.

5 Related Work

In this section, we will introduce different works dealing with data capabilities. We structure the section by distinguishing two basic aspects. On the one hand, we discuss further recent evaluations of process modeling approaches with respect to data. On the other hand, we discuss approaches dealing with or tangenting data in the fields of business process management or workflow modeling. These discussed approaches i) act as preparation step for developing process models with one of the evaluated approaches, ii) utilize evaluated approaches for process modeling, or iii) are out of scope for our evaluation.

Further Recent Evaluations. Another evaluation in the field of business process management with respect to data is part of [32]. Müller focuses on the continuous interplay of control flow and data within modeling, execution, adaptation, and exception handling by assessing vendor tools as well as theoretical approaches. Therefore, he arranges the approaches and tools in a four by four matrix indicating the degree of process and data support on each axis. In contrast, we mainly focus on the data aspects and assess them more detailed.

[33] evaluates different process modeling approaches with respect to their capability to support case management. Case management focuses on process

execution aspects. Therefore, it basically makes use of data relations and aspects, so-called case data, to verify and propose next execution steps especially based on data dependencies and predefined rules. Moreover, the authors classify the evaluated approaches based on the main modeling artifact into activity, data, and communication driven approaches. Compared to our evaluation, the authors considered a very limited scope and established a coarse-grained classification. Additional evaluations are unknown to us for the best of our knowledge.

Further Approaches. The approach of product-based workflows by Reijers et al. [34] addresses the problem of designing a workflow optimally with respect to time or cost requirements. Thereby, a product is the business goal of a specific workflow and corresponds to the appropriate data object as final output. To achieve this goal, certain sub-products need to be created, which in turn may have further sub-products. Additionally, only subsets of sub-products may be needed to create the parent-product. This leads to a tree structure representing the dependencies the authors call production rules. For each production rule, the probability of use as well as costs and time consumption are annotated. An example omitting these annotations for clarity reasons is presented in Figure 17. Product A can be created following three different production rules: i) A is created based on product B only, ii) A is created based on product D only, or iii) A is created based on a combination of products B, C, and D.

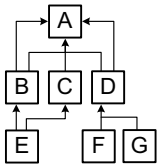

Fig. 17. Visualization of production rules in product-based workflows

Altogether, product-based workflows is an approach which can be considered as a pre-step to the definition of the control flow of a process. Based on data dependencies and a well defined goal, the optimal execution path is calculated considering side aspects like time and cost specifications. The optimal path then is represented as a Workflow net highly influenced by data aspects.

Proclets are leight-weight processes which communicate via structured messages [35]. Each of these processes focus on the behavior of one specific case instead of overloading single processes with information from several cases. Proclets are basically a communication framework being able to utilize different types of modeling languages, for instance the ones we discussed in Section 4. Adapting this idea, each process, i.e. proclet, may represent a data object and the activities needed to be performed on this data object. Synchronization, i.e. fulfilling existing data dependencies, is achieved by utilizing the structured messages for communication. Though, proclets still utilize business process modeling languages evaluated in this paper to represent the work performed on each data

object. Therefore, main aspects of the data support are driven by the utilized modeling language.

Data provenance [36] deals with scientific workflows instead of business processes as the approaches do we evaluate in this paper. The main difference is the data traffic. Scientific workflows deal with data updates in the millisecond range rather than on a daily or even monthly rate as business processes typically do. Additionally, the focus of scientific workflows lies on querying the stream of data to get real time answers instead of describing intended process flows. One main challenge there is to provide meaningful answers to the queries, i.e. understandable for the scientist performing the queries. In [37], an end-user-centric provenance model has been introduced which bases on three operations on data only: read, write, and state reset. The authors cluster the data operations into sessions and as long as no state reset appears, read and write operations belong to the same collection. Based on this information, the workflow traces can be queried and consequently, the origin and evolution of data can be identified. Altogether, data provenance approaches are highly data centric but not meant to represent business processes. Nevertheless, the querying capabilities are interesting in the field of business process monitoring. Enriching business process modeling languages with such capabilities may lead to more detailed real-time information during business process execution.

Petri nets [5] as one representative of place-transition-nets are usually unaware of data. But using a mapping introduced in [38], a subset of BPMN is mapped to petri nets focusing on the data flow implicitly modeled within BPMN. Thereby, the evolution of a single data object within the business process is visualized. The places represent the states of the data object and the transitions represent the activities performed on that data object. As this utilization of place-transition-nets is only a further interpretation and not explicitly covered by the defined semantics of such approaches, we omit evaluating this idea alongside the other business process modeling approaches.

In 2004, the Workflow Patterns initiative evaluated a range of at that time current workflow systems [16]. Based on this evaluation, they derived a set of 39 patterns describing the common usage of data aspects within these systems. This comprises data visibility, internal and external interaction, transfer, and routing. Data visibility deals with questions with respect to who is allowed to read data, i.e. setting the scope of a data object. Data interaction and transfer deal with the access on data objects either with shared objects (interaction) or separate but passed objects (transfer). Data transfer may generally occur in external communications. Finally, routing deals with process control based on data values and dependencies. Later, different evaluations have been obtained to check the degree of coverage with respect to these patterns for business process modeling notations. These include evaluations of BPMN 1.x [39], UML activity diagrams [40], BPEL [41], and Oracle BPEL [42].

6 Conclusion

In this paper, we discussed the evaluation of a set of twelve process modeling languages ranging from industry standards like BPMN, BPEL, UML activity diagrams and industry driven approaches like EPCs to scientific approaches like Business Artifacts, ADEPT, and Corepro with respect to their capabilities to deal with data. Due to space limitations, we evaluated four representatives in detail. But for the general discussion, we comprised the findings with respect to all modeling languages. The underlying framework for our evaluation consists of 23 criteria in four groups: *Process modeling capabilities* (control flow), *data modeling capabilities*, *connection of control flow and data modeling*, and *execution semantics*.

The results of the evaluation show that no process modeling notation supports all criteria specified in the framework. Settled standards like Workflow nets and Petri nets do not support data at all, except the modeling constructs are redefined to represent, for instance, data object states. The highest coverage is achieved by Corepro, UML activity diagrams, and BPMN. However, data support remains on a shallow level in general: Only basic criteria like the representation of data objects are widely supported, complex ones like interrelations between data objects are mostly omitted. Also the three modeling languages with the highest coverage only support a subset of the complex criteria.

Following, we identified a general need for features with respect to full support modeling languages. However, the reason for shallow coverage is mostly intended with respect to the scope of the modeling language. A full support approach increases the complexity as, for instance, the number of modeling elements increases. But wide acceptance requires a high understandability and an easy use so that many modeling languages are restricted to specific use cases.

Based on our evaluation, we propose a clustering of the modeling languages based on the awareness of process and data capabilities. Therefore, we distinguish the groups of *control flow driven*, *data aware control flow driven*, *control flow and data driven*, *control aware data driven*, and *data driven* process modeling languages.

References

1. Elzinga, D., Horak, T., Lee, C.Y., Bruner, C.: Business process management: survey and methodology. IEEE Transactions on Engineering Management **42**(2) (1995) 119–128
2. OMG: Business Process Model and Notation (BPMN), Version 2.0 (January 2011) `http://www.omg.org/spec/BPMN/2.0/` accessed November 3, 2011.
3. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards Formal Analysis of Artifact-Centric Business Process Models. In: Business Process Management, Springer (2007) 288–304
4. van der Aalst, W., Weske, M., Grünbauer, D.: Case Handling: A New Paradigm for Business Process Support. Data and Knowledge Engineering **53**(2) (2005) 129–162
5. Petri, C.: Kommunikation mit Automaten. PhD thesis, Institut für Instrumentelle Mathematik, University of Bonn (1962)

6. Jensen, K.: Coloured Petri Nets. Petri nets: Central Models and Their Properties **254** (1987) 248–299
7. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis methods, and Practical Use. Volume 1–3. (1996)
8. van der Aalst, W.: Timed Coloured Petri Nets and their Application to Logistics. PhD thesis, Eindhoven University of Technology (1992)
9. van der Aalst, W.: Putting high-level Petri nets to work in industry. Computers in Industry **25**(1) (1994) 45–54
10. Van der Aalst, W.: The Application of Petri Nets to Workflow Management. Circuits Systems and Computers **8** (1998) 21–66
11. OASIS: Web Services Business Process Execution Language, Version 2.0 (April 2007) `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html` accessed November 3, 2011.
12. Habich, D., Richly, S., Preissler, S., Grasselt, M., Lehner, W., Maier, A.: BPEL-DT - Data-Aware Extension of BPEL to Support Data-Intensive Service Applications. In: Emerging Web Services Technology. (2007) 111–128
13. Habich, D., Richly, S., Grasselt, M.: Data-Grey-Boxweb Services in Data-Centric Environments. In: IEEE International Conference on Web Services. (2007) 976–983
14. van der Aalst, W., ter Hofstede, A.: YAWL: Yet Another Workflow Language. Information Systems **30**(4) (2005) 245–275
15. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
16. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Data Patterns. Technical report, Queensland University of Technology (2004)
17. Keller, G., Nüttgens, M., Scheer, A.: Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Technical Report Heft 89, Institut für Wirtschaftsinformatik, University of Saarland (1992)
18. OMG: Unified Modeling Language (UML), Version 2.2 (February 2009) `http://www.omg.org/spec/UML/2.2/` accessed November 3, 2011.
19. BPMI: Business Process Modeling Notation (BPMN), Version 1.0 (May 2004) `http://www.bpmi.org/downloads/BPMN-V1.0.pdf` accessed November 3, 2011.
20. Müller, D., Reichert, M., Herbst, J.: Flexibility of Data-driven Process Structures. In: Business Process Management Workshops, Springer (2006) 181–192
21. Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-driven Process Structures. In: Advanced Information Systems Engineering, Springer (2008) 48–63
22. Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming **8**(3) (1987) 231–274
23. Nigam, A., Caswell, N.: Business artifacts: An approach to operational specification. IBM Systems Journal **42**(3) (2003) 428–445
24. Bhattacharya, K., Hull, R., Su, J.: A Data-Centric Design Methodology for Business Processes. Handbook of Research on Business Process Management (2009)
25. Kumaran, S., Liu, R., Wu, F.: On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Advanced Information Systems Engineering, Springer (2008) 32–47
26. Wang, J., Kumar, A.: A Framework for Document-Driven Workflow Systems. In: Business Process Management, Springer (2005) 285–301
27. Reichert, M., Dadam, P.: Adept$_{\text{flex}}$-supporting dynamic changes of workflows without losing control. Intelligent Information Systems **10**(2) (1998) 93–129
28. Mealy, G.H.: A Method for Synthesizing Sequential Circuits. Bell Systems Technical Journal **34** (1955) 1045–1079

29. Moore, E.: Gedanken-Experiments on Sequential Machines. Automata Studies **34** (1956) 129–153
30. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. Volume 6 of LNBIP. Springer (2008)
31. Mendling, J., Reijers, H.A., van der Aalst, W.: Seven Process Modeling Guidelines (7PMG). Information & Software Technology **52**(2) (2010) 127–136
32. Müller, D.: Management datengetriebener Prozessstrukturen. PhD thesis, University of Ulm (2009)
33. de Man, H.: Case Management: A Review of Modeling Approaches. Technical report, BPTrends (2009)
34. Reijers, H., Limam, S., van der Aalst, W.: Product-Based Workflow Design. Management Information Systems **20**(1) (2003) 229–262
35. van der Aalst, W., Barthelmess, P., Ellis, C., Wainer, J.: Workflow Modeling using Proclets. In: Cooperative Information Systems, Springer (2000) 198–209
36. Goble, C.: Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics. In: Workshop on Data Derivation and Provenance. (2002)
37. Bowers, S., McPhillips, T., Ludäscher, B., Cohen, S., Davidson, S.: A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows. Provenance and Annotation of Data (2006) 133–147
38. Awad, A., Decker, G., Lohmann, N.: Diagnosing and Repairing Data Anomalies in Process Models. In: Business Process Management Workshops, Springer (2010) 5–16
39. Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A., Russell, N.: On the Suitability of BPMN for Business Process Modelling. In: Business Process Management, Springer (2006) 161–176
40. Russell, N., van der Aalst, W., ter Hofstede, A., Wohed, P.: On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. In: Conceptual Modelling, Australian Computer Society, Inc. (2006) 95–104
41. van der Aalst, W., Dumas, M., ter Hofstede, A., Russell, N., Verbeek, H., Wohed, P.: Life after BPEL? In: Web Services and Formal Methods, Springer (2005) 35–50
42. Mulyar, N.: Pattern-based Evaluation of Oracle-BPEL (v. 10.1. 2). Technical report, Eindhoven University of Technology (2005)

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
49	978-3-86956-143-1	**Adaptive Windows for Duplicate Detection**	Uwe Draisbach, Felix Naumann, Sascha Szott, Oliver Wonneberg
48	978-3-86956-134-9	**CSOM/PL: A Virtual Machine Product Line**	Michael Haupt, Stefan Marr, Robert Hirschfeld
47	978-3-86956-130-1	**State Propagation in Abstracted Business Processes**	Sergey Smirnov, Armin Zamani Farahani, Mathias Weske
46	978-3-86956-129-5	**Proceedings of the 5th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
45	978-3-86956-128-8	**Survey on Healthcare IT systems: Standards, Regulations and Security**	Christian Neuhaus, Andreas Polze, Mohammad M. R. Chowdhuryy
44	978-3-86956-113-4	**Virtualisierung und Cloud Computing: Konzepte, Technologiestudie, Marktübersicht**	Christoph Meinel, Christian Willems, Sebastian Roschke, Maxim Schnjakin
43	978-3-86956-110-3	**SOA-Security 2010 : Symposium für Sicherheit in Service-orientierten Architekturen ; 28. / 29. Oktober 2010 am Hasso-Plattner-Institut**	Christoph Meinel, Ivonne Thomas, Robert Warschofsky et al.
42	978-3-86956-114-1	**Proceedings of the Fall 2010 Future SOC Lab Day**	Hrsg. von Christoph Meinel, Andreas Polze, Alexander Zeier et al.
41	978-3-86956-108-0	**The effect of tangible media on individuals in business process modeling: A controlled experiment**	Alexander Lübbe
40	978-3-86956-106-6	**Selected Papers of the International Workshop on Smalltalk Technologies (IWST'10)**	Hrsg. von Michael Haupt, Robert Hirschfeld
39	978-3-86956-092-2	**Dritter Deutscher IPv6 Gipfel 2010**	Hrsg. von Christoph Meinel und Harald Sack
38	978-3-86956-081-6	**Extracting Structured Information from Wikipedia Articles to Populate Infoboxes**	Dustin Lange, Christoph Böhm, Felix Naumann
37	978-3-86956-078-6	**Toward Bridging the Gap Between Formal Semantics and Implementation of Triple Graph Grammars**	Holger Giese, Stephan Hildebrandt, Leen Lambers
36	978-3-86956-065-6	**Pattern Matching for an Object-oriented and Dynamically Typed Programming Language**	Felix Geller, Robert Hirschfeld, Gilad Bracha
35	978-3-86956-054-0	**Business Process Model Abstraction: Theory and Practice**	Sergey Smirnov, Hajo A. Reijers, Thijs Nugteren, Mathias Weske
34	978-3-86956-048-9	**Efficient and exact computation of inclusion dependencies for data integration**	Jana Bauckmann, Ulf Leser, Felix Naumann
33	978-3-86956-043-4	**Proceedings of the 9th Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS '10)**	Hrsg. von Bram Adams, Michael Haupt, Daniel Lohmann

ISBN 978-3-86956-144-8
ISSN 1613-5652