Artifact-Centric Modeling of Business Processes Using UML Diagrams

David Grünert, Thomas Keller, and Elke Brucker-Kley

ZHAW School of Management and Law
Institute of Business Information Technology
8401 Winterthur, Switzerland

{grud, kell, brck}@zhaw.ch

Abstract

The modeling of business processes to date has focused on an activity-based perspective while business artifacts associated with the process have been modeled on an abstract and informal level. Ad hoc, dynamic business processes have recently emerged as a requirement. Subsequently, BPMN was extended with ad hoc sub-processes and a new standard, Case Management Modeling and Notation (CMMN), has been created by the Object Management Group (OMG). CMMN has an information-centric approach, whereas the extension of BPMN adheres to an activity-based perspective. The focus on BPMN and on processes in general has caused UML to fade into the background. UML combines an activity-based perspective (i.e., activity diagrams) with an information-centric perspective (i.e., state machines). This paper promotes an information-centric approach based on UML use case, state machine, and class diagrams that allows for an opportunistic execution of activities based solely on UML models.

Keywords: Content-oriented workflow models, bottom-up model creation, artifact-centric workflow models, document-oriented workflow models.

1 INTRODUCTION

The evolvement of business process management has at least partly been based on the need to enhance business IT alignment. At first, process models mainly served as documentation of requirements for the subsequent support by information technology. Later on, these process models were automatically converted into workflow definitions or were interpreted by a process engine [1]. Both approaches led to a better alignment of business with IT. However, not all business processes are suited to being implemented in such a workflow-oriented way. As soon as knowledge workers are involved, business processes need to be more flexible and require more detail. A purely activity-based perspective has to be complemented by an artifact-centric perspective. An expert knowledge worker in a collaborative setting differs from a transactional knowledge worker whose sequence of activities simply needs to be controlled and connected in an integrated setting. The knowledge worker's work is based on information (i.e., business objects), based on which she/he decides what needs to be done next. Given the unpredictable and varied range of situations, parameters, and expected outcomes of work, a much more situational and declarative, i.e., opportunistic, approach to modeling is required.

In response to this extended need, the information content of business processes gained in significance. The modeling of data is by no means a new modeling paradigm, but it has been either conducted separately or information flows have been subordinate and hidden in the process model. Accordingly, the positioning of information at the center of modeling was termed data- or information-centric process modeling [2], [3], [4], [5]. Information entities are modeled by state charts. Transitions are triggered by activities. Associated roles are defined by means of use case diagrams. In [5], the term opportunistic BPM (oBPM) was introduced for this kind of approach. The duality between activity- and information-centric models was shown in [6].

Many artifact-centric approaches defined new or extended model syntax [6]. However, a new or extended modeling syntax increases complexity for all parties involved in designing, reading, and implementing the modeled process and requires adapted modeling tools. Furthermore, not all models presented in the context of artifact-centric approaches are adequate for being executed by a process engine because there are no standardized workflows involved [7]. We wanted to find out if it was possible to define an artifact-centric model that was:

- Not domain-specific,
- **Executable** by an engine, and
- **Built on standard UML** diagrams without the need of new syntax elements.

We chose UML because it is tried and tested, receives broad tool support, and knowledge of UML diagrams is widespread. In this paper, we propose a solution based on UML use case, state machine, and class diagrams. We discuss advantages and disadvantages of the approach and we show how our approach extends previous work regarding the expressiveness of the model and the usage of standard UML.

2 RELATED WORK

A concept closely related to oBPM is the so-called content- or data-oriented workflow model. The term "content-oriented" first appeared in [8] and is used as an umbrella term for several scientific workflow approaches, namely "data-driven", "resource-driven", "artifact-centric", "object-aware" and "document-oriented". Common to all of these models is the definition of workflows based on documents, data records, or other objects containing process data. Content-oriented workflow models are a topic of ongoing research and numerous publications are available ([2], [3], [4], [5]).

Similar to oBPM, most content-oriented workflow models allow multiple execution paths of a business case, similar to the opportunistic task order of oBPM. However, only few approaches, e.g. [6], make the linking of the document state machines as explicit as oBPM. Because of the content-centric approach, content-oriented workflow models are typically well-suited for modeling ad-hoc events. What is new in oBPM is the combination of these aspects with the definition of a formalized process model in UML.

Several sample implementations of content-oriented workflow models were carried out for research purposes and were able to demonstrate their capabilities ([9], [10]). However, most of these implementations address specific application domains such as the health sector or the automotive industry. While no general purpose business process modeling tool has so far been developed to implement a content-oriented workflow model, there are software providers conducting research on artifact-centric workflow models (see [11]).

3 ARTIFACT-CENTRIC BUSINESS PROCESS MODELING USING UML

The term "information-centric BPM" stands for designing processes with a minimum of control flow by modeling the states of artifacts involved in business processes. The rationale behind this approach is outlined in Section 1 and can also be found in [5]. The oBPM model [5] is both user- and artifact-centric[1] and has two different perspectives. The first perspective shows the top-down view based on standard UML diagrams. This perspective is useful for process owners or system administrators. The second perspective is used for bottom-up model creation and allows knowledge workers to read, change, and define their own processes as suggested in [16]. We presented a first version of the oBPM model in [5] including both perspectives. This section describes the UML top-down perspective in more detail and suggests several extensions of the previously presented model.

3.1 The oBPM Model in UML

The oBPM model defines roles, tasks modeled as use cases as well as artifacts and their dependencies. The model also defines hierarchies for tasks and artifacts. It allows us to define the workflow of any business case with one UML use case diagram, one UML class diagram, and as many UML state machine diagrams as there are documents or artifacts used in the workflow. This section introduces all elements of the model and shows how they can be represented in the above-mentioned UML diagrams.

Use Case Diagram for Role, Task, and Document Associations[2]. The first diagram used for oBPM is a use case diagram. It illustrates a system's overall capabilities by connecting roles, tasks, and artifacts. An example of such a model is shown in **Fig. 1.** The use case diagram contains the following elements:

- **User roles** define all roles available to users interacting with the tool. User roles can be defined specifically for an oBPM model, or they can be taken from existing role-based access control systems (RBAC) [12]. A UML model uses the stickman symbol to represent the different roles as actors. The example shown in **Fig. 1** defines the roles <Sales>, <Accounting>, and <Customer Service>.
- **Tasks** represent one or more activities which define a meaningful operation in the business context. Tasks are represented in the UML model as use cases. The example shown in **Fig. 1** defines the tasks <Sell product>, <File receipt of payment>, <Send reminder>, <File complaint>, and <Handle complaint>.

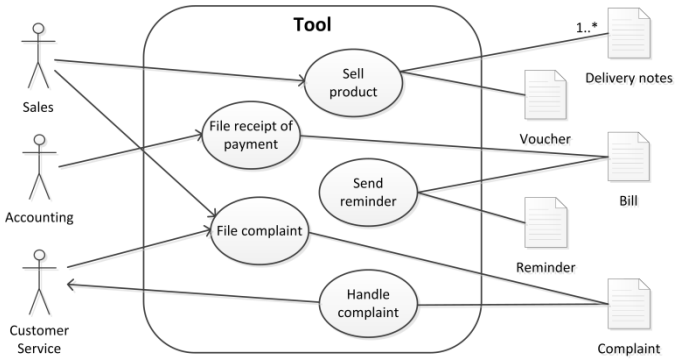

Fig. 1. Use case diagram defining associations between roles, tasks, and documents

- **Artifacts** are business-relevant objects that are created, manipulated, and archived as they pass through a business process [13]. All artifacts used in oBPM are typed. The type defines the data format[3] and a lifecycle model. Artifacts are represented in the UML model as actors with a document symbol. The example shown in **Fig. 1** defines the artifacts <Delivery notes>, <Voucher>, <Bill>, <Reminder>, and <Complaint>. While the modeling of roles as actors is an obvious choice, modeling artifacts as actors is not. We nevertheless suggest doing this for two reasons: First, just like roles artifacts in oBPM can trigger the execution of tasks. Artifacts can therefore be seen as a third-party system interacting with the process tool. Second, while the oBPM tool will manage or at least monitor the artifact's lifecycle the content of the artifact will typically be edited outside the tool. Therefore, the document is part of the system's context rather than part of the system.
- **Role-task associations** identify all tasks where a given role is involved. The association is represented in the UML model as a directed or undirected association between the actor symbol of the role and the use case symbol of the task. Tasks with an association pointing towards the task can be triggered by the user whenever all associated artifacts are in the required lifecycle state (see section <State Machine Diagram> for details on the lifecycle). Associations pointing towards the role indicate tasks triggered by the system. Undirected associations allow triggering from both. Use cases without a relation to a task are triggered and executed without involving a user. The example shown in **Fig. 1** defines that users can trigger the tasks <Sell product>, <File receipt of payment>, and <File complaint>, while the tool can trigger the execution of the tasks <Send reminder> and <Handle complaint>.
- **Artifact-task associations** relate artifacts with tasks and identify all tasks a document is used in. The association is represented in the UML model as an undirected association between the actor symbol of the artifact and the use case symbol of the task. Multiple associations pointing towards the same task define an AND operation. Therefore, all associated artifacts must have the required lifecycle state to execute the task (see section <State Machine Diagram> for details on the lifecycle). The artifact-task association can be decorated with multiplicity. Default multiplicity of one is assumed if no multiplicity is defined. Multiplicity is used when more than one instance of an artifact is used for a single task execution (1..*) or when an artifact is optional (0..1). The example shown in **Fig. 1** defines that the task <Sell product> uses an artifact of type <Voucher> and one or more artifacts of type <Delivery notes>.
- **Task-task associations** are used to define hierarchies for tasks. The association is represented in the UML model with an association of type <<include>>. The example

[1] We use the term <artifact> as a synonym for any kind of structured and unstructured business data.

[2] All the following examples can be downloaded from https://drive.google.com/file/d/0B4Pg8YZ0eoLUNjM2dTl6ZWYwX1U/view?usp=sharing

[3] An artifact may also contain unstructured data of any format.

shown in **Fig. 2** defines that the task <Sell product> and <Write offer> triggers the execution of <Check credit ratings>.

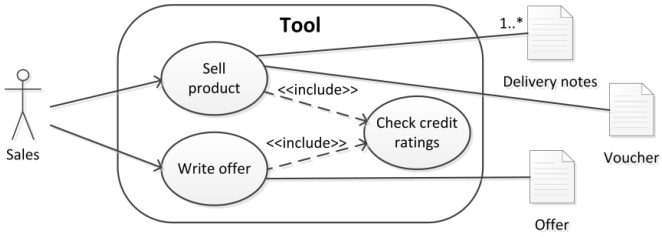

Fig. 2. Use case diagram defining hierarchies of tasks

Class Diagram for Document Associations. The second diagram used for oBPM is a UML class diagram. It defines the details of the artifacts being used. An example of such a model is shown in **Fig. 3**. Each use case diagram is associated with exactly one class diagram. It shows how many instances of an artifact can be created per case (see explanation below), how these artifacts are related, and it can also define alternative documents. The class diagram contains the following elements:

- **The class <Case>:** The class diagram used for the oBPM model must contain exactly one class named <Case>. This class is not an artifact; it represents the case. An instance of the class <Case> is created automatically by the system whenever a new case is opened.
- **Normal classes** define all artifacts used in the case. The example shown in **Fig. 3** contains five artifacts: <Delivery notes>, <Bill>, <Reminder>, <Complaint> and <Transaction receipt>.
- **Normal associations** define relations between the artifacts and their cardinality per case. The example shown in **Fig. 3** defines that each case can have any number of delivery notes and complaints and zero or one vouchers. In addition, each bill can have one reminder.
- **Classes of type interface and associations of type <implements>** define alternative artifacts. The example shown in **Fig. 3** defines that <Transaction receipt> and <Bill> are alternative artifacts. A task that requires an artifact of type <Voucher> can either use the artifact <Transaction receipt> or <Bill>. While AND relations between multiple artifacts and a task are modeled in the use case diagram, OR relations must be modeled in the class diagram.

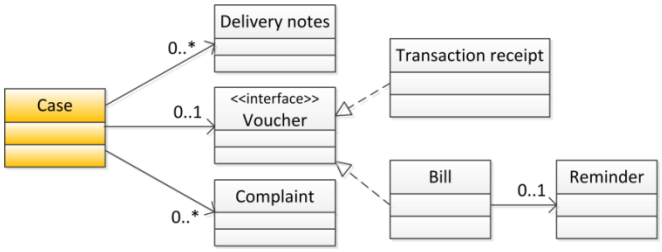

Fig. 3. Class diagram defining dependencies between documents and case

State Machine Diagram for Artifact States. The last diagram type used to complete the oBPM model is the state machine diagram. This diagram is the key element of oBPM. It exists once for each normal class defined in the class diagram. An instance of the diagram is created for each instance of the respective class. The purpose of the state machine diagram is to define all possible states of artifacts and the availability of the tasks defined in the use case diagram. Examples of state machine diagrams are shown in **Fig. 4** to **Fig. 6**. The state machine diagram contains the following elements:

- The **frame** of the state machine identifies the artifact it belongs to. The example shown in **Fig. 4** contains the state machine for the artifact <Constraint>.
- **Start and final states** are used to indicate the start and the end of the artifact's lifecycle. Multiple start and final states can be used in hierarchical state machines (see **Fig. 5**).
- **States** define the possible states of the artifact. The example shown in **Fig. 5** defines five states for the artifact <Bill>: created, reminded, paid, hold, and canceled.
- **State transitions** are used to restrict the execution of tasks by defining the possible state transitions of artifacts. Restriction is achieved by linking all state transitions with one or multiple tasks defined in the use case diagram. The state machine diagram indicates this link with the event name defined for the transition. In the example shown in **Fig. 5**, the transition from state <created> to state <paid> is linked with the task <File receipt of payment>. On the one hand, this transition defines that the state <paid> can only be reached with the task <File receipt of payment>. On the other hand, it defines that the task <File receipt of payment> is only available if the artifact of type <Bill> is either in the state <created> or <reminded>. The state transition can be decorated with a guard, using the UML square bracket notation. The guard may contain a time restriction, a role restriction, or a result restriction. An example of a time restriction is shown in **Fig. 5**, where the task <Send reminder> is executed when the artifact has been in the state <created> for 30 days. An example of a result restriction is shown in **Fig. 6** for the task <Handle complaint>.
- **Combined states** can be used to model exceptions or other transitions that can happen from multiple states. The example shown in **Fig. 5** defines <No pending complaint> as a combined state. The states <created>, <reminded>, and <paid> are left when the task <File complaint> is executed. Depending on the outcome of the task <Handle complaint>, the original state is re-entered via the history state or the document ends in the state <canceled>.

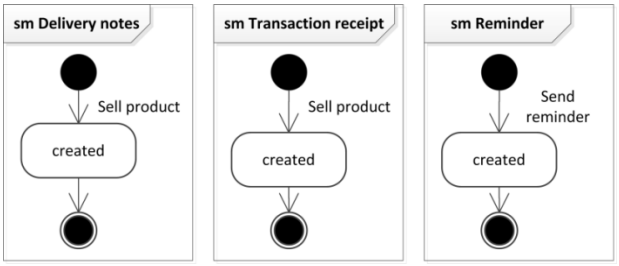

Fig. 4. Example of three state machines with transitions triggered by tasks

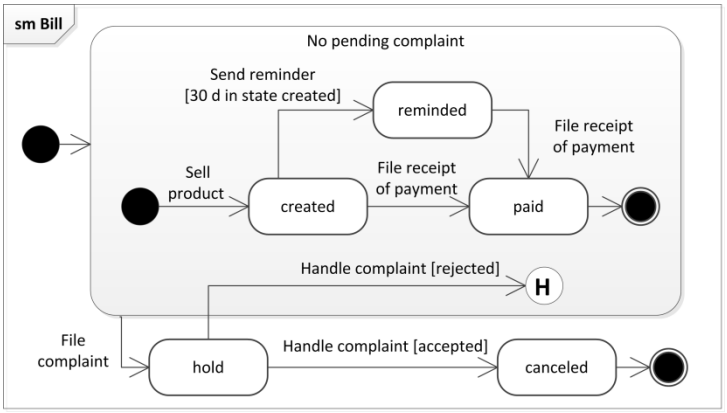

Fig. 5. Example of a state machine with combined states

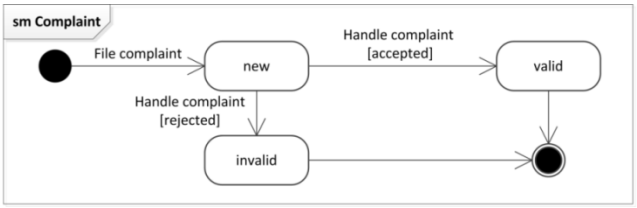

Fig. 6. Example of a state machine with a result restriction

3.2 Model Execution

The previous sections introduced the modeling aspect of oBPM. In the following, we will address the question how a system can process this information and how the workflow is finally presented to the user.

The system executing oBPM needs to manage the states of all artifacts. By applying the use case, class, and state machine diagrams, the executing system is able to derive a role-dependent task list from these states. How this can be done is shown in the following example: Assuming the system needs to evaluate if users with the role <Accounting> are currently allowed to execute the task <File receipt of payment>, the system needs to parse the use case diagram (**Fig. 1**) to find that the only required artifact is of the type <Bill>. By analyzing the state machine diagram (**Fig. 5**), the system finds that the artifact <Bill> must either be in the state <created> or <reminded>. The system can now apply this restriction to all open cases by navigating to the artifact <Bill> and then following the associations defined in the class diagram (**Fig. 3**). The result of this operation is a list of cases that currently allow the execution of the task.

Finally, the result of this operation must be visualized for users. A possible solution is a role-dependent task list in combination with a filter. The filter defines which tasks are shown in the list and allows choosing between all tasks and the tasks for a selected case.

4 PROPERTY DAMAGE CLAIM USE CASE

This section applies the oBPM modeling approach introduced above to the property damage claim example[4] from [6]. It is meant to show the applicability of oBPM in a more comprehensive context. **Fig. 7** shows the use case diagram consisting of three actors, three artifacts, and a couple of tasks that have varying associations with actors and artifacts. Fig. 9 to Fig. 10 show the respective state machines for the three artifacts.

A new instance of the property damage claim case is initiated by the customer executing the task "Notify Claim". There is no other way to initiate a new instance since the task "Record Claim" is only available once the artifact "Loss Event" is in the state "notified". As soon as this has happened, the task "Record Claim" is made available to the clerk. By executing this task, a new claim instance is generated and the state of "Loss Event" changes to "recorded". The clerk then needs to validate the claim to move it to the state "validated". At this point, the role "Clerk" has no more tasks to complete and the role "Investigator" takes over. There are two possible tasks at this stage: "Decide on Claim" and "Analyze". The task "Analyze" will not change the state of the artifact "Claim" and can be executed as many times as needed by the "Investigator". Only after executing the task "Decide on Claim", the state will either change to "accepted" or "rejected", depending on the result of the task execution. If the artifact is in the state "accepted", only the task "Offer Benefit" is possible to be executed by the "Investigator". By executing it the first time, a payment artifact is generated and put into the state "created". From this point on, the "Investigator" is offered the task "Discharge Claim" in addition to "Offer Benefit". The rest of this path is then straightforward and is not commented any further.

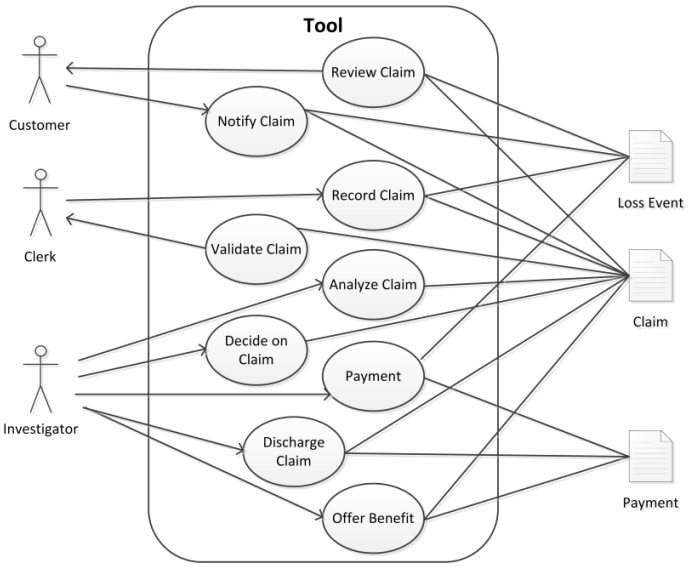

Fig. 7. Use case diagram for property damage claim

If the claim is rejected by the "Investigator", the state moves to "rejected". By reaching this state, the role "Customer" is given an opportunity to comment on the rejection by executing the task "Review Claim". If he or she accepts the rejection, the state machines of "LossEvent" and "Claim" move on to their respective end states. An instance of "Payment" has not been generated for this case.

There is no need to show the class model for "LossEvent", "Claim", and "Payment" since it is trivial. As in [6], "Claim" is not detailed any further since it does not lead to any more insights regarding the interaction between actors and artifacts.

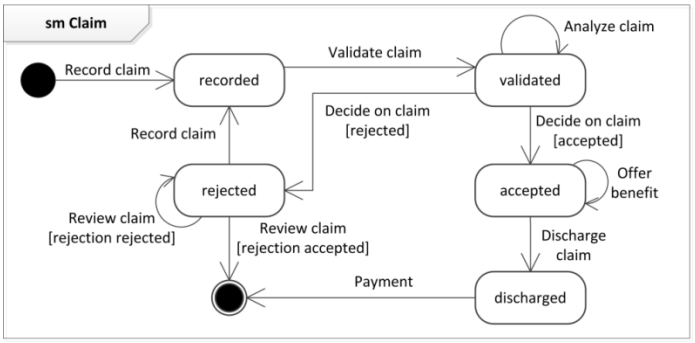

Fig. 8. State machine for artifact claim

[4] All the drawings are available at https://drive.google.com/file/d/0B4Pg8YZ0eoLUNjM2dTl6ZWYwX1U/view?usp=sharing

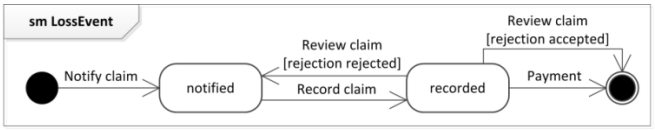

Fig. 9. State machine for artifact loss event

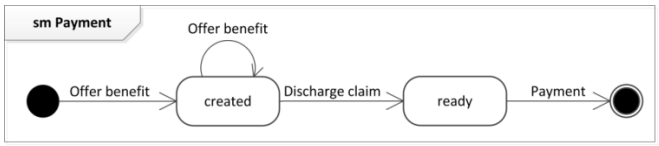

Fig. 10. State machine for artifact payment

The property damage claim example shows that the state machines of the three artifacts are interleaved and that this interleaving is defined by the use case diagram. The use case further specifies which roles have responsibility for which task. And the tasks act as triggers for transitions of the state machines. Although no business process is explicitly modelled, the workflow and all its variants can be derived from the state machines and the use case diagram.[5]

5 BENEFITS OF OBPM

Apart from the general advantages of any information-centric approach, we claim that oBPM features the following advantages:

- **Standard UML**: The presented oBPM approach makes only use of standard UML diagrams. This allows the creation of models using any UML 2.x-compliant modeling tool. The Object Management Group (OMG), which has created UML, has also defined an XML-based exchange format for UML models called XMI [14]. This data format can be used to import the model into a process engine, which often has native support for XML documents. Unfortunately, there are several incompatibilities between different implementations of XMI for UML [15]. Therefore, each combination of modeling tool and process engine needs to be validated.
- **Roles**: Unlike other proposed notations such as CMMN, oBPM includes the definition of user roles and their relations to tasks and subtasks. Defining user roles as part of the model allows automatic translation of artifact states into tasks for users and roles. The definition of roles also allows us to make use of existing user and role management IT infrastructure.
- **Task and artifact hierarchies**: oBPM allows the definition of hierarchies for both artifacts and tasks. The hierarchy of artifacts makes it possible to define the multiplicity for each artifact individually relative to the case or other artifacts. The multiplicity can also be used to define an artifact as optional. Furthermore, it is possible to define AND as well as OR relations between artifacts and tasks. AND relations can be defined in the use case diagram by connecting multiple artifacts with a task. OR relations can be defined in the class diagram by using an interface class. Having AND and OR relations not defined in the same diagram can be seen as a disadvantage of the oBPM approach but it also helps to reduce the complexity of the individual diagrams. Hierarchies for tasks allow a more efficient modeling by sharing common subtasks. This also helps in increasing maintainability by reducing duplicated parts of the model.
- **Artifact restrictions**: When creating an artifact-centric model, the final structure or exact content of an artifact is often not known in detail. Unlike other models [6], oBPM allows the definition of an entire model without knowledge of any details of the artifacts used. The only requirement is that their behavior can be modeled with a state machine and that the tasks can be linked using the corresponding transitions. Taking the artifact as a black box makes it possible to continuously improve the document structure without the necessity of changing other parts of the model.

[5] See https://www.lucidchart.com/invitations/accept/132be91d-ed0f-40b2-94d6-3045e8c73932 for an additional example comprising all features introduced in the previous paragraphs.

6 CONCLUSION

With the approach presented in this paper, we hope to show that it is possible to define an artifact-centric model for business processes using standard UML diagrams. The proposed approach allows the modeling of all aspects of a business case including artifacts, tasks, roles, artifact hierarchies, task hierarchies, and artifact states. The approach does not define any restrictions on the artifacts being used. It can deal with any type of artifact that can be associated with a lifecycle and it does not require any information on the content or the structure of the artifacts.

The approach distributes the information on a business case over three types of diagrams. This reduces the complexity of the individual diagram by separating different aspects of the business case definition. The challenge of interlinking multiple diagrams has been dealt with in a role-centric, bottom-up perspective as proposed in [5].

We believe that oBPM represents an approach to adapting IT support for the use by collaborative knowledge workers that have been missed out by traditional activity-centric BPM and workflow systems. We are currently investigating how existing document management solutions can be adapted to oBPM in order to build a prototype. Especially emerging NoSQL data bases include support for many key requirements of oBPM. In addition, we are evaluating user acceptance of the oBPM approach together with knowledge workers and process owners and we are compare different procedures to develop the process definition.

7 REFERENCES

[1] Workflow Management Coalition, «WfMC, the Workflow Reference Model,» Winchester, UK, 1995.

[2] S. Kumaran, P. Nandi, T. Heath, K. Bhaskaran und R. Das, «ADoc-oriented programming,» in *Symposium on Applications and the Internet (SAINT)*, 2003.

[3] P. Nandi und S. Kumaran, «Adaptive business objects - a new component model for business integration,» in *Proceedings of International Conference on Enterprise Information Systems*, 2005.

[4] A. Nigam und N. Caswell, «Business artifacts: An approach to operational specification,» in *IBM Systems Journal 42(3)*, 2003.

[5] D. Grünert, E. Brucker-Kley und T. Keller, «oBPM – An Opportunistic Approach to Business Process Modeling and Execution,» in *Business Process Management Workshops*, 2014.

[6] S. Kumaran, R. Liu und F. Y. Wu, «On the Duality of Information-Centric and Activity-Centric Models of Business Processes,» in *Advanced Information Systems Engineering*, Springer Berlin Heidelberg, 2008.

[7] V. K. a. M. Reichert, «Towards Object-aware Process Management Systems: Issues, Challenges, Benefits,» in *Proc. 10th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'09)*, Amsterdam, 2009.

[8] C. Neumann und R. Lenz, «The Alpha-Flow Use-Case of Breast Can-

cer Treatment - Modeling Inter-institutional Healthcare Workflows by Active Documents,» in *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, Larissa, 2010.

[9] COREPRO, Configuration-based Release Management Processes in the Automotive Sector, University of Twente, 2005-2007.

[10] P. Flows, *Process, Humans and Information Linkage for harmonic Business Flows*, University of Ulm, Institute of Databases and Information Systems.

[11] IBM Research, «Business Artifacts Research,» [Online]. Available: http://researcher.watson.ibm.com/researcher/view_group.php?id=2501. [Zugriff am 22 05 2015].

[12] D. F. Ferraiolo und R. D. Kuhn, «Role-Based Access Controls,» in *15th National Computer Security Conference*, Baltimore, 1992.

[13] D. Cohn und R. Hull, «A Data-centric Approach to Modeling Business Operations and Processes,» *IEEE Data Eng. Bull. 32(3)*, 3 2009.

[14] Object Management Group (OMG), XML Metadata Interchange (XMI), Version 2.4.2, April 2014.

[15] e. a. K. Lausdahl, «Connecting UML and VDM++ with Open Tool Support.,» in *Volume 5850 of Lecture Notes in Computer Science*, Proceedings of the 2nd World Congress on Formal Methods, 2009.