Mathias Weske, Judith Michael (Hrsg.): Modellierung 2024,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2024 83

Transforming Object-Centric Process Models into BPMN 2.0 Models in the PHILharmonicFlows Framework

Marius Breitmayer [1], Lisa Arnold [1], Marko Pejic [1], Manfred Reichert [1]

Abstract: Business processes can be modeled using a plethora of different paradigms including activity-centric (e.g., imperative, declarative), and data-centric processes. The former focus on the process activities to be executed as well as their execution order and constraints, whereas the latter deal with the data required to progress during process execution. Both representations, however, allow describing the same process, but from different viewpoints. Consequently, a transformation between process representations based on the different paradigms yields promising perspectives for enabling a holistic view on both the behavior and the data perspective of a process and fosters a common understanding of different paradigms. This paper presents an approach for transforming object-centric processes (i.e., object lifecycle processes and their interactions) into corresponding activity-centric representations modeled in terms of BPMN 2.0. We present seven transformation rules for mapping an object- to an activity-centric process, illustrated along a running example. We evaluate the approach based on a proof-of-concept implementation that can automatically perform the necessary transformations and has been applied in multiple scenarios. Overall, our approach for transforming object-centric processes into BPMN 2.0 models provides new insights into the relationship between the two paradigms and enables a more flexible and effective way of modeling business processes in general.

Keywords: Object-centric Process; Data-centric Process; Activity-centric Process; Process Model Transformation

1 Introduction

Process-aware Information Systems (PAISs) are based on executable business process models expressed in terms of the *activity-centric paradigm* [DvTH05]. In the latter, a process model comprises a set of connected black-box activities that represent units of work or sub-processes, specified with an imperative language (e.g., BPMN 2.0). At runtime, the completion of these activities drives process execution. However, many processes (e.g., *unstructured*, and *knowledge-intensive* processes [Si09]) are data-centric, requiring the treatment of data as a first-class citizen at both design- and run-time. Due to the insufficient integration of processes and data, traditional PAISs do not adequately support such processes. To remedy this drawback, the *data-centric paradigm* has emerged [St19]. As opposed to activity-centric processes, the available data (values) drives process execution. Usually, approaches implementing this paradigm follow an object-centric approach, i.e., a business

[1] Ulm University, Institute of Databases and Information Systems, Helmholtzstraße 16, 89081 Ulm, Germany
firstname.lastname@uni-ulm.de

process corresponds to a multitude of concurrently processed business objects (of same or different types) that interact with each other to reach the overall process goal. Examples of object-centric process management approaches include case handling [vdAWG05], artifact-centric processes [CH09], object-centric process mining [vdA19], and object-centric/-aware process management [KR11]. Both activity-centric and object-centric paradigms have their pros and cons. An object-centric process model may be useful for understanding how data drives process execution, and how the various business objects involved in a business process interact with each other. In contrast, an activity-centric process model fosters our understanding on how work is executed. In general, the same process may be described from both viewpoints (i.e., object- and activity-centric), thus allowing for a more comprehensive approach of expressing process semantics and fostering process literacy. Despite the relevance of the different viewpoints, there exists a gap regarding the common use of activity- and object-centric process management approaches.

The contribution of this paper is twofold. On the one hand, we provide a conceptual approach for transforming object-centric processes (i.e., a PHILharmonicFlows process model) into an activity-centric representation (i.e., BPMN 2.0). As a benefit, the strengths of both paradigms can be exploited, increasing overall process information provided by the two representations. Furthermore, the approach allows for a more flexible and effective way of modeling processes, improving their compatibility and comparability. On the other hand, the paper provides valuable insights into how such transformation can be automatically performed using the FLOWS2BPMN approach.

This paper is structured as follows: Section 2 introduces fundamentals. Section 3 describes the proposed approach and presents the transformation rules of FLOWS2BPMN. Section 4 evaluates the approach and Section 5 relates it to existing works. Section 6 concludes the paper with a summary and outlook.

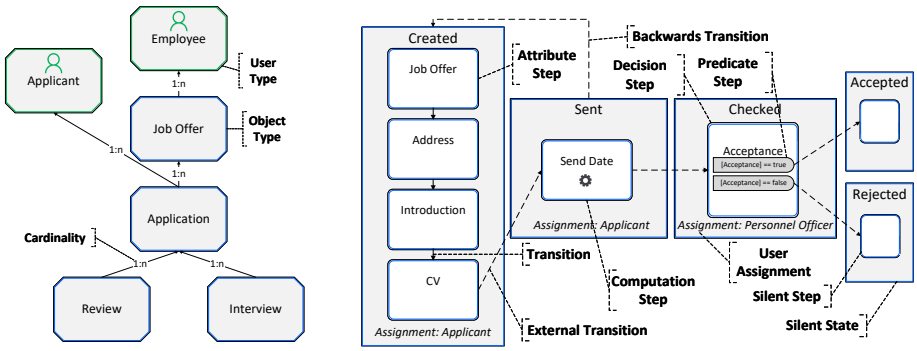
2 Fundamentals

As pillar of this work, we use the object-centric process management approach PHILharmonicFlows which implements the fundamental concepts of the object-centric paradigm, covering all lifecycle stages (i.e., modeling, execution, monitoring, and analysis/evolution). PHILharmonicFlows provides the most comprehensive approach as shown in a literature study [St19] as well as an integrated and usable implementation.

2.1 PHILharmonicFlows

In the PHILharmonicFlows approach, we developed a framework for data-centric and -driven process management and enhanced it with the concept of *objects*. Generally, an object-centric business process comprises multiple interacting objects (e.g., *Job Offer*, *Application*, *Review*, and *Interview*) with each object representing a real-world business

object. A (semantic) *data model* (see Fig. 1a) is used to organize all process-relevant objects (including their attributes) as well as their semantic relations [KR11]. The latter also consider cardinality constraints. Finally, object behavior (i.e., the data-driven processing of the respective object lifecycle) is expressed in terms of a state-based *object lifecycle process* model. Thereby, each *state* of a specific lifecycle process (e.g., the lifecycle process of object Application in Fig. 1b comprises states *Created*, *Sent*, *Checked*, *Accepted*, and *Rejected*) may comprise multiple *steps*. Each step refers to a specific object attribute to be written before completing the respective state. After all required attributes (i.e., steps) of the present state have assigned values, the object may transition to the next state, i.e., the execution of a lifecycle process is data-driven.



(a) PHILharmonicFlows Data Model

(b) Lifecycle of Object Application

Fig. 1: Simplified Data Model and Lifecycle Process

At runtime, each object may be instantiated multiple times with the corresponding lifecycle process instances being executed concurrently [ASR21]. Furthermore, relations (e.g., the one between an *Application* instance and a *Review* instance) may be instantiated multiple times, enabling 1-to-many or many-to-many associations between individual object instances. Overall, this results in a large relational process structure at runtime [SAR18b]. The interactions between the various object instances of such a process structure, are managed by a *coordination process* [SAR18a]. In Fig. 2, for example, an application may only be rejected if either the review or the interview proposes rejection. Conversely, an application is accepted if the job offer is closed, the application is checked, and the interview proposes hiring the applicant. In a nutshell, the coordination process enables or prohibits objects to change to another state depending on pre-defined constraints of that object in relation to the states of other objects (see [SAR18a] for details).

2.2 Business Process Model and Notation

Business Process Model and Notation (BPMN) 2.0 constitutes an established standard for representing business processes [Ko15]. BPMN is based on the activity-centric process

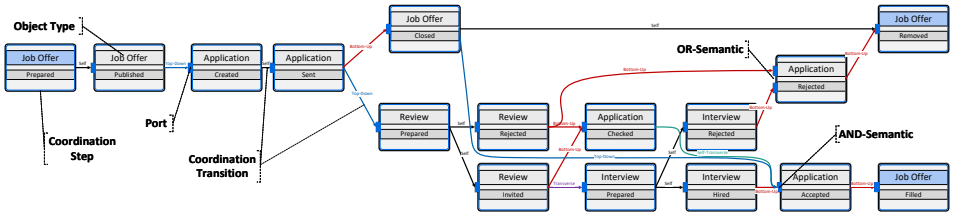


Fig. 2: Coordination Process Example Recruitment

management paradigm, providing a standardized multi-domain modeling notation [We19]. Generally, the completion of activities drives process execution, with the order in which these activities are executed being managed by the control flow. The latter comprises *sequence flows*, *message flows*, and *gateways*. In addition, *data objects* allow modeling data and may be read or written by activities, enriching the latter with specific information. Modeling the data perspective, however, is often neglected as activities are treated as first-class citizens [Re12]. Fig. 3 depicts a simplified process model of a recruitment process in which a *Job Offer* is created and published by a *Personnel Officer*. Then, *Applicants* may create and sent their *Application*. Afterwards, a *Department Expert* evaluates the application and either accepts or rejects it. Finally, the *Applicant* is notified. Note that the execution of this process is driven by the execution of activities rather than the data becoming available (cf. Section 2.1).

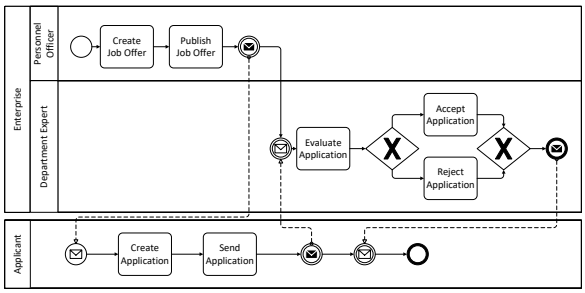


Fig. 3: Example Process Recruitment in BPMN 2.0 (simplified)

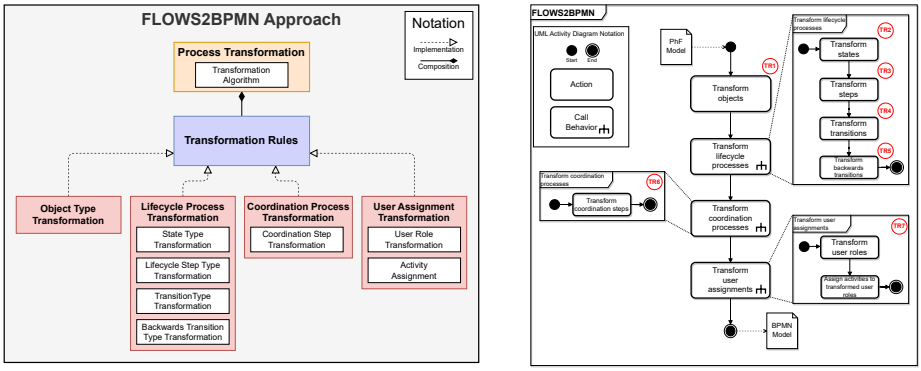
3 Transformation Approach

The goal of FLOWS2BPMN, the approach we propose for transforming object-centric process models to BPMN 2.0 models, has been three-fold: First, we developed a concept for transforming object-centric processes into an activity-centric representation, i.e., a BPMN 2.0 process model. Second, we implemented a proof-of-concept prototype capable of automatically realizing this model transformation. Third, the transformation between object-centric and activity-centric approaches enables a holistic view on business processes and facilitates our understanding of processes expressed with these different paradigms. To enable the transformation between object- and activity-centric processes, we derived a set of transformation rules that allow transforming object types, lifecycle processes, coordination

processes, and user assignments into a suitable BPMN 2.0 representation, enabling the application of existing process management tools to object-centric processes.

Fig. 4a depicts the main components of the FLOWS2BPMN approach, which enables the generic transformation of object-centric process models into BPMN 2.0 models.

The core of the approach comprises 7 *transformation rules (TR)*, each belonging to one of the following categories: *Object Type Transformation*, *Lifecycle Process Transformation*, *Coordination Process Transformation*, and *User Assignment Transformation*. The *Transformation procedure* is illustrated by Fig. 4b. It describes the order in which the *transformation rules* are applied to generate a BPMN 2.0 process model from an object-centric process representation. The remainder of this section introduces the 7 transformation rules along the running example of a recruitment process (cf. Sec 2) and the transformation procedure.



(a) Main Components

(b) Transformation Procedure

Fig. 4: The FLOWS2BPMN Approach

3.1 Object Type Transformation

TR1 (Object Type Transformation):
An object type of an object-centric process is mapped to a pool of a BPMN collaboration diagram.

Transformation rule TR1 maps object types to BPMN elements. Each object type is transformed into a separate pool. Note that the generation of a pool implies adding a start and end event to it. As a consequence, we map different object types to different pools, each having corresponding start and end events. Fig. 5 illustrates the application of TR1 to object type *Application*. Generally, multiple instances of an object type may be created at runtime, each being executed by a separate lifecycle process instance. In the transformation of object types to BPMN pools, this is reflected by the use of *multi-instance* pools (MI pool), i.e., each pool generated by TR1 is tagged as a MI pool.

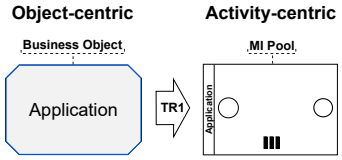


Fig. 5: Example *Object Type Transformation*

3.2 Lifecycle Process Transformation

Based on TR1, each object type can be transformed to a multi-instance pool. The second step of the transformation procedure (see Fig. 4b) then transforms each object lifecycle process to a semantically corresponding BPMN 2.0 representation. This includes the transformation of lifecycle states, steps (i.e., attributes), and transitions between them to corresponding BPMN elements.

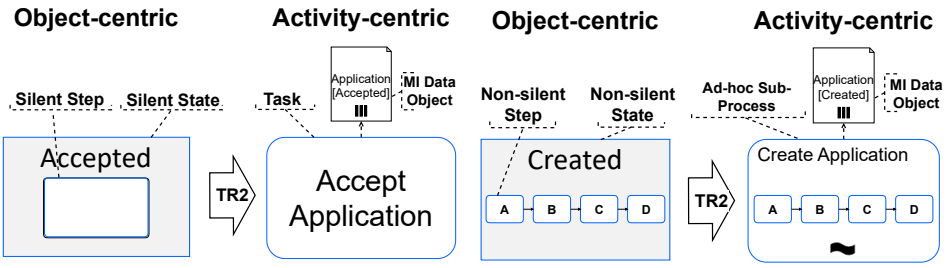
TR2 (State Type Transformation):
Each state type of the object lifecycle process is transformed into a BPMN activity. Depending on the respective state type, this activity corresponds to an atomic *task* or a *sub-process* reflecting the internal logic of the steps within an object state. Additionally, activities write data objects that indicate the state of an object type.

Remember that a state of an object lifecycle process comprises a set of ordered steps, each representing atomic actions (i.e., writing an attribute) of the object. In BPMN, this behavior can be encapsulated by a *sub-process*. When transforming a state to BPMN elements, two cases need to be distinguished:

1. The state comprises a number of connected steps that reflect the attributes to be written (e.g., in a form) before leaving the state. In this case, the state is mapped to a (collapsed) BPMN sub-process. Particularly, this sub-process requires the transformation of the steps within the respective state as well (see Fig. 6b and TR3).
2. The state is silent, i.e., it does not comprise any step and action respectively, (e.g., the silent state *Accepted* in Fig. 6a). Consequently, representing the internal logic of the state is not required. In this case, the state is transformed into a BPMN *task*.

The resulting BPMN process model reflects the state-based view of the lifecycle process (see Fig. 1b). Expanding sub-processes, in turn, displays the internal logic of a state and its respective steps (see TR3). Consequently, both granularity levels of an object lifecycle process (i.e., state and step level) can be represented. Moreover, each created BPMN activity is connected to a data object that refers to the object in the corresponding state. Note that this data object corresponds to a *multi-instance* data object.

Lifecycle steps represent atomic actions to write or update object attributes, e.g., by filling in form fields or sensing a data value from the physical environment. Mapping a step to BPMN results in a task (i.e., atomic activity), embedded in the respective sub-process generated



(a) Transforming Silent State Type

(b) Transforming Non-silent State Type

Fig. 6: Example *Lifecycle State Type Transformation*

by TR2. The created sub-process and its tasks reflect the attributes to be updated when processing an object state as well as the order (including if-then-else constraints) in which the corresponding attribute values may be written during runtime. Note that, for example, this information may be exploited by process implementers to create corresponding forms (which are auto-generated in PHILharmonicFlows).

TR3 (Lifecycle Step Type Transformation):
When mapping a lifecycle step to a BPMN element, different cases need to be distinguished, depending on the properties of the step (cf. Fig 1b):

1. *Attribute steps* update object attributes and correspond to the default step type. They are transformed to a BPMN task.
2. *Computation steps* enable the automatic computation of attribute values (e.g., the current date or a price including VAT). They are mapped to a *Service Task* in BPMN.
3. *Decision steps* comprise predicate steps and are mapped to tasks.
4. *Predicate steps* enable choices during lifecycle process execution, i.e., the respective object may transition to different successors depending on attribute values. Predicate steps are not mapped to BPMN tasks, but to the labels of the sequence flows outgoing from a predicate step.
5. *Silent steps* (i.e., steps without associated action) are represented by the state generated in TR2 – no transformation is required.

In a nutshell, only attribute, computation, decision, and predicate steps need to be mapped to the BPMN model. In particular, attribute and decision steps are mapped to *tasks*, whereas computation steps are mapped to a *service task*. They further require the integration of data in the BPMN model. For this purpose, each task is associated with a corresponding data object in the BPMN model. As steps are allocated to states, these data objects exist within the created sub-process. Note that these data objects are labelled as multi-instance data objects. The mapping of steps to BPMN elements is depicted in Fig. 7.

Note that attribute values need not necessarily be provided in the pre-specified order of the lifecycle process steps. Transitions between the steps of a state, therefore, only define a default execution order of steps [SAR19], e.g., an applicant may upload her CV before providing

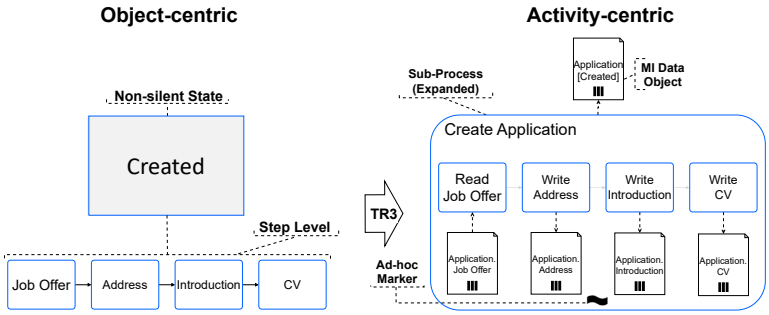


Fig. 7: Example *Lifecycle Step Type Transformation* Application State Created

her address when creating an application. Consequently, the sub-process representing a state may be declared as a BPMN ad-hoc activity, i.e., the order in which attributes are written is arbitrary. In particular, the sub-process may only contain activities, data objects, sequence flows, and gateways. If sequence flows are omitted, the execution order of the internal tasks will be arbitrary. Consequently, the flexible execution (i.e., arbitrary order of how attributes are written) of the lifecycle process is adopted in the resulting BPMN model. Note that we greyed out the (yet to-be transformed) sequence flows in Fig. 7 as they are not mandatory, but optional. Again, note that the created sub-process provides a specification to process engineers that is useful for implementing the process (e.g., user form design). In general, the transitions of a lifecycle process correspond to sequence flows of a BPMN model as both possess the same semantics in their respective process modeling paradigm. Mapping the exact semantics of a transition to a sequence flow, however, depends on its context.

TR4 (Transition Type Transformation):
A transition type is transformed into a sequence flow. Expressions specified in lifecycle predicate steps are mapped to labels of the corresponding sequence flows. To minimize the routing paths per element, exclusive gateways are generated to group these sequence flows and to ensure that activities only have one incoming and outgoing sequence flow.

More precisely, a sequence flow in the BPMN model must be solely labeled if the source element of the transition corresponds to a *predicate step* (see TR3). In this case, the label of the created sequence flow should correspond to the expression attached to the predicate step. Fig. 8 shows the mapping of a predicate step to labeled sequence flows according to TR4.

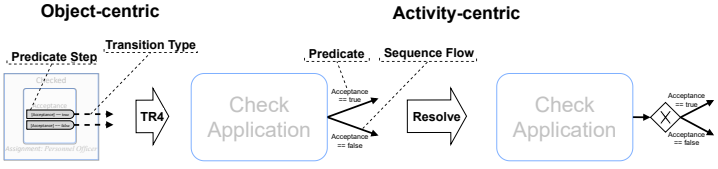


Fig. 8: Example *Transition Type Transformation*

TR5 (Backwards Transition Type Transformation):
A backwards transition type of a lifecycle process is transformed to a loop in the corresponding BPMN model.

Backwards transitions allow the users involved in the execution of a lifecycle process to return to previous states. When mapping a backwards transition to a BPMN model, certain activities of the BPMN model need to be repeated. For this purpose, each backwards transition is mapped to a *loop* in the BPMN model. Fig. 9 illustrates this transformation.

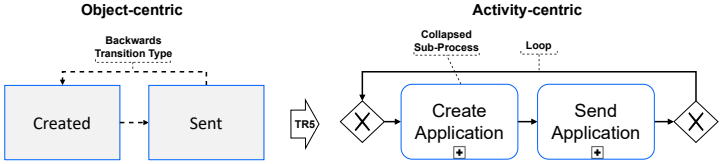


Fig. 9: Example *Backwards Transition Type Transformation*

3.3 Coordination Process Transformation

The previous sections have introduced the transformation rules for object types and their lifecycle processes. This section shows how interactions between different object lifecycle processes are considered in the model transformation procedure. In PHILharmonicFlows, an interaction between lifecycle processes refers to the states of different objects and is managed by a coordination process (see Fig. 2) [SAR18a]. The latter comprises a number of coordination steps of which each reflects a semantic relationship between object states.

TR6 (Coordination Process Transformation):
The information contained in a coordination process is transformed to BPMN elements as well. In particular, this transformation considers the constraints attached to the coordination steps of the coordination process, i.e., the semantics of each individual coordination step needs to be mapped to its BPMN counterpart. For this purpose, intermediate message events (catching) are applied to *catch* messages. Moreover, intermediate parallel multiple events are used to catch multiple messages. In turn, intermediate message events (throwing) enable sending messages. Finally, event-based gateways allow coordinating alternative paths.

The semantics of a coordination step is defined by the ports associated with it as well as the transitions connected to these ports (see Fig. 2). Different cases need to be distinguished:

1. Multiple ports attached to a coordination step reflect an (X)OR-semantics (cf. coordination step *Application Rejected* in Fig. 2). State *Rejected* of object *Application* may be only executed after executing one of the preceding states. In BPMN, we realize this behavior using an *Event-based Gateway*. This transformation is depicted in Fig. 10 (1).

2. Multiple transitions targeting at a single port of a coordination step express AND-semantics. Consider coordination step *Application Accepted* in Fig. 2. Regarding this step, multiple constraints (e.g., *Job Offer Closed, Application Checked*, and *Interview Hired* in Fig. 2) need to be met. When mapping this behavior to a BPMN process model, *intermediate parallel multiple events* are leveraged, i.e., process execution is delayed upon arrival of corresponding messages. These messages are modeled via *intermediate message events (throwing)* following the corresponding activities in the respective (object) pool. This transformation is depicted in Fig. 10 (2).
3. If a coordination step has exactly one port with one incoming transition, the activation of the state necessitates the previous completion of another state corresponding to a different object (see coordination step *Application Created* in Fig. 2). Such behavior can be expressed with BPMN using a *message exchange*, i.e., a particular coordinating activity sends a message received by another coordination activity. For this purpose, an *intermediate message event (throwing)* as well as an *intermediate message event (catching)* need to be added to the process model. Note that the control flow of the BPMN model is delayed until completing this message exchange. This transformation is illustrated by Fig. 10 (3).

For object-centric processes these different semantics are not mutually exclusive. The actual complexity might increase when using the various coordination constraints in combination.

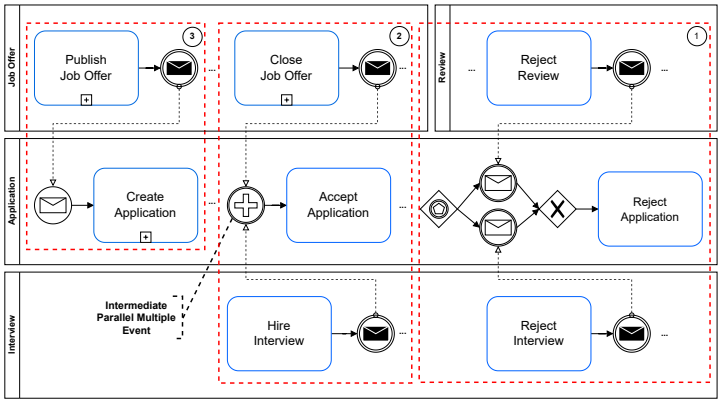


Fig. 10: Example *Coordination Process Transformation*

3.4 User Assignment Transformation

To complete the transformation rules of the FLOWS2BPMN approach, the activities created by the previous TRs need to be associated with their respective user roles. The lifecycle process model associates each state with a user assignment, which determines the user roles responsible for processing the respective state.

TR7 (User Assignment Transformation):
A user role is transformed to a lane within a pool. Based on this transformation, a human task (i.e., state) is assigned to that lane whose role corresponds to the user assignment set out by the respective state of the lifecycle process model. A *system* user role (lane) comprises non-assigned states.

We accomplish the transformation of user roles in a two-step procedure. First, we map the user role to a new lane of the pool created by TR1. Further, we label the newly added lane with the name of the respective user role. The activities of the corresponding pool need to be assigned to their proper lanes. To be more precise, each task resulting from the application of TR2 is assigned to the lane that represents the user role in the context of the respective the state. In turn, lifecycle states without user assignments (see states *Accepted* and *Rejected* in Fig. 1b) are executed by a software system. Modeling such behavior in terms of BPMN requires an additional lane representing a *System-user role*. Similar to user roles, this lane indicates that a *(computer) system* auto-executes the activities of this lane. Fig. 11 illustrates the transformation of object *Application* (cf. Fig. 1b) including user assignments and collapsed sub-processes.

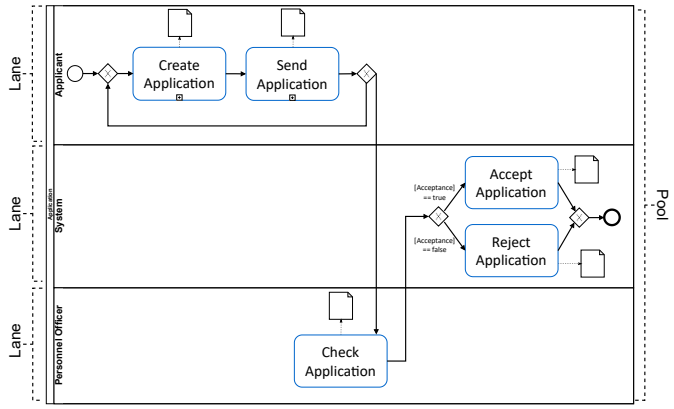


Fig. 11: Example *User Assignment Transformation*

4 Evaluation

4.1 Proof-of-concept Prototype

We implemented a proof-of-concept prototype that realizes the presented transformations. It leverages standard data interchange formats for representing object-centric processes (i.e., JSON) and BPMN processes (i.e., XML) respectively. Generated BPMN process models can be imported to any tool capable of visualizing BPMN models (e.g., Signavio, bpmn.io or Camunda). The source code is available via GitHub[2].

[2] https://github.com/markopejic-git/Transforming-Object-Centric-Processes-into-BPMN

4.2 Case Studies

We conducted case studies in which we applied the transformation procedure to process models of three different real-world scenarios initially modeled using the PHILharmonicFlows framework. Each of these object-centric process models focuses on the complexity in different parts of the process model (e.g., objects, lifecycles, or coordination). Supplementary material on PHILharmonicFlows, the different object-centric process models, and the resulting BPMN process models are provided in a cloudstore[3].

Recruitment The recruitment process served as a running example in this paper and stems from a long-term collaboration we have had with an ERP software provider. In total, the recruitment model comprises 4 object types, 2 user types, and 5 relations (see Fig. 1a). Corresponding lifecycle processes consist of 20 states and 43 steps (see Fig. 1b for the lifecycle of object *Application*), while the coordination includes 16 steps and 19 transitions (see Fig. 2). This coordination process has the highest complexity as it includes a plethora of coordination steps, various combinations of ports and transitions (cf. TR 6).

PHoodle The e-learning system *PHoodle*, an object- and process-centric information system, we implemented with PHILharmonicFlows, includes 7 object types (e.g., Lecture, Exercise, or Submission), 2 user types, 11 relations, and corresponding lifecycle processes. The latter comprise 20 states and 52 steps. Moreover, a coordination process exists that consists of 6 steps and 9 transitions. This process has also been applied in a real-world deployment at Ulm University to organize the lecture, exercises, and exams of a course over one semester. In this real-world scenario, *Phoodle* managed 2 teaching employees, 5 Exercises, 6 Tutors, 14 Downloads, 51 Tutorials, 128 Students, and 487 Submissions.

Diagnosis & Treatment The *Diagnosis & Treatment* process deals with the admission, diagnosis, tests, treatment, and discharge of patients in a hospital scenario. The object-centric process model comprises 3 object types, 2 user types, and 4 relations. Their corresponding lifecycle processes contain 14 states and 22 steps. The coordination process consists of 10 steps and 10 transitions, and mainly focuses on the sequential coordination of objects.

4.3 Limitations

The presented approach faces several limitations:

1. The transformation requires the extended set of BPMN elements. On the one hand, the extended set of BPMN elements allows reducing the number of process model elements (i.e., model size and complexity). On the other, resulting models might be harder to comprehend, especially for inexperienced modelers, and the syntax of the extended BPMN elements might not fully match the one of the object-centric model.
2. Pools generated for each object according to TR1 do not fully conform with the traditional representation of pools (i.e., participants) in BPMN. However, in the context of object-centric processes, a variety of (interacting) objects participate in the process rather than traditional participants (i.e., organizations or roles). In future work, we will extend the approach to further address this issue.

[3] https://cloudstore.uni-ulm.de/s/d9Mq3kBHbiyKNac

3. The multi-instance symbol in BPMN assumes that the number of instances is known beforehand, which is not always the case in object-centric processes. In the running example, the number of application object associated with a job offer might be arbitrary. The latter corresponds to unbounded interleaving behavior, for which BPMN does not have a special symbol.
4. The execution syntax of a lifecycle process state and the ad-hoc sub-process generated by TR3 are not identical. The ad-hoc sub-process in BPMN specifies that the performer determines the sequence and number of an activity. The sequence, in which the steps of a lifecycle process are organized, specifies the guidance provided at runtime. However, a lifecycle process state may be completed upon availability of all required values. We incorporated the execution guidance of lifecycles through sequence flows within the ad-hoc sub-process rather than losing this information during the transformation.

4.4 Benefits

The presented approach enables the exploration of object-centric process for modelers unfamiliar with object-centric processes. The representation uses an established modeling language (i.e., BPMN 2.0), facilitating the understanding of the fundamental differences between the two process management paradigms. Especially, this is beneficial in education or training, during which understanding the differences of the modeling paradigms is of utmost importance. In a nutshell, the presented transformation might increase understandability of object-centric processes in general. Furthermore, the representation of object-centric processes in terms of BPMN enables applying a plethora of existing approaches for process management to object-centric processes. Consequently, this further strengthens the use of the object-centric process paradigm. This includes approaches towards modeling, analyzing, and evolving & optimizing business processes [Du18].

5 Related Work

The presented work is related to process model transformations, especially between activity-centric and data-centric process representations. Despite their fundamental differences, activity- and data-centric processes are not mutually exclusive [Re12] and approaches often combine existing principles. In Case Handling [vdAWG05], for example, activities are completed upon provision of data and do not constitute atomic work units, i.e., process execution is data-driven. The work presented in [Me13] enriches activities with SQL-enabling data support.

[KLW08] formally defines activity-centric process models and presents an approach for transforming activity- to information-centric models. UML state charts are used for representing lifecycle processes. These state charts, however, have limited capabilities regarding the communication between tasks and objects, i.e., coordination aspects are neglected. The work presented in [EVG16] overcomes these issues by enabling parallelism as well as event communication. However, processes are represented in terms of UML state charts, and no explicit support for BPMN is provided. As opposed to [KLW08, EVG16], the presented approach also considers the transformation of object coordination constraints using message events in the resulting BPMN model.

A similar approach is presented in [MW14]. It allows transforming artifact- to activity-centric models and vice versa, but requires intermediate steps (e.g., a synchronized object lifecycle), potentially increasing complexity. Besides, no implementation is provided. Similar to [KLW08], the approach lacks the integration of coordination constraints.

[SMW07] presents an approach for enabling users to define semantic correspondences between different syntax elements using mapping operators, and to execute the transformation between EPC and UML process models. Both EPC and UML process models represent activity-centric process models. In contrast, our approach is able to automatically derive the BPMN model without mapping operators specified by users.

[Es13] presents a framework for representing artifact-centric process models in UML. It considers elements of artifact-centric process models (i.e., business artifacts, lifecycles, services, and associations), similar to our approach's consideration of different granularity levels (i.e., data models, object lifecycles, and coordination processes). However, [Es13] focuses representing artifact-centric elements in UML, rather than performing a complete transformation of process models. Consequently, specific UML methods (e.g., state machines and activity diagrams) are required for representing artifact-centric elements. In contrast, our approach provides a complete transformation of all levels of granularity, integrating them into one BPMN 2.0 process model.

[Ou06] presents an approach for transforming BPMN and UML processes to BPEL-based workflows. However, no data-centric approaches were considered. In contrast, the presented approach tackles the transformation of data-centric processes to BPMN. Finally, [KRG07] presents an approach for generating compliant business process models from a set of reference object lifecycles. Synchronization points between lifecycles need to be manually identified first. In contrast, our approach derives coordination constructs based on the information available from object-centric coordination processes (i.e., coordination steps).

6 Summary and Outlook

This paper presents an approach for transforming object-centric processes into activity-centric processes with the latter being modeled in terms of BPMN. In particular, we want to bridge the gap between the two paradigms. In detail, we introduced 7 transformation rules that cover different aspects of object-centric processes (i.e., data model, lifecycle processes, and coordination process) and enable their mapping to activity-centric process models (i.e., BPMN 2.0). The technical feasibility of the automated transformation procedure was demonstrated by a proof-of-concept prototype. Furthermore, we applied the transformation to three processes of varying complexity and from different domains. In future work, we will further enhance the transformation by examining the comprehensibility of the generated BPMN 2.0 models, investigate different labels of object-centric processes, and simplify the resulting model. Furthermore, we will enable the reverse transformation (i.e., mapping BPMN 2.0 models to object-centric models) by inverting the transformation rules.

Bibliography

[ASR21] Andrews, Kevin; Steinau, Sebastian; Reichert, Manfred: Enabling runtime flexibility in data-centric and data-driven process execution engines. Inf Sys, 101:101447, 2021.

[CH09] Cohn, David; Hull, Richard: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng Bull, 32(3):3–9, 2009.

[Du18] Dumas, Marlon; Rosa, Marcello La; Mendling, Jan; Reijers, Hajo A.: Fundamentals of Business Process Management. Springer, 2nd edition, 2018.

[DvTH05] Dumas, Marlon; van der Aalst, Wil M P; Ter Hofstede, Arthur H: Process-Aware Information Systems. John Wiley & Sons, 2005.

[Es13] Estañol, Montserrat; Queralt, Anna; Sancho, Maria-Ribera; Teniente, Ernest: Artifact-Centric Business Process Models in UML. volume 132, pp. 292–303, 01 2013.

[EVG16] Eshuis, Rik; Van Gorp, Pieter: Synthesizing data-centric models from business process models. Computing, 98(4):345–373, 2016.

[KLW08] Kumaran, Santhosh; Liu, Rong; Wu, Frederick Y: On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Advanced Information Systems Engineering. Springer, pp. 32–47, 2008.

[Ko15] Kocbek Bule, Mateja; Jost, Gregor; Hericko, Marjan; Polančič, Gregor: Business Process Model and Notation: The Current State of Affairs. Computer Science and Information Systems, 12(2):509–539, 2015.

[KR11] Künzle, Vera; Reichert, Manfred: PHILharmonicFlows: towards a framework for object-aware process management. J of Soft Maint & Evo, 23(4):205–244, 2011.

[KRG07] Küster, Jochen M.; Ryndina, Ksenia; Gall, Harald: Generation of Business Process Models for Object Life Cycle Compliance. In: Business Process Management. Springer, 2007.

[Me13] Meyer, Andreas; Pufahl, Luise; Fahland, Dirk; Weske, Mathias: Modeling and Enacting Complex Data Dependencies in Business Processes. In: BPM, pp. 171–186. Springer, 2013.

[MW14] Meyer, Andreas; Weske, Mathias: Activity-centric and artifact-centric process model roundtrip. In: Int'l Conf on BPM. Springer, pp. 167–181, 2014.

[Ou06] Ouyang, Chun; Dumas, Marlon; Breutel, Stephan; ter Hofstede, Arthur: Translating Standard Process Models to BPEL. In: CAiSE 2006. Springer, pp. 417–432, 2006.

[Re12] Reichert, Manfred: Process and Data: Two Sides of the Same Coin? In: 20th Int'l Conf on Cooperative Information Systems (CoopIS'12). Springer, pp. 2–19, 2012.

[SAR18a] Steinau, Sebastian; Andrews, Kevin; Reichert, Manfred: Modeling Process Interactions with Coordination Processes. In: CoopIS'18. LNCS. Springer, pp. 21–39, 2018.

[SAR18b] Steinau, Sebastian; Andrews, Kevin; Reichert, Manfred: The Relational Process Structure. In: CAiSE 2018. LNCS 10816. Springer, pp. 53–67, 2018.

[SAR19] Steinau, Sebastian; Andrews, Kevin; Reichert, Manfred: Executing Lifecycle Processes in Object-Aware Process Management. In: SIMPDA. Springer, pp. 25–44, 2019.

[Si09] Silver, B: Case management: Addressing unique BPM requirements. Taming the unpredictable: real-world adaptive case management, pp. 1–12, 2009.

[SMW07] Strommer, Michael; Murzek, Marion; Wimmer, Manuel: Applying Model Transformation By-Example on Business Process Modeling Languages. In: Advances in Conceptual Modeling – Foundations and Applications. Springer Berlin Heidelberg, pp. 116–125, 2007.

[St19] Steinau, Sebastian; Marrella, Andrea; Andrews, Kevin; Leotta, Francesco; Mecella, Massimo; Reichert, Manfred: DALEC: A Framework for the Systematic Evaluation of Data-centric Approaches to Process Management Software. Softw & Sys Modeling, 18(4):2679–2716, 2019.

[vdA19] van der Aalst, Wil M. P.: Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data. In: Software Engineering and Formal Methods. Springer, Cham, pp. 3–25, 2019.

[vdAWG05] van der Aalst, Wil M. P.; Weske, Mathias; Grünbauer, Dolf: Case handling: a new paradigm for business process support. DKE, 53(2):129–162, 2005.

[We19] Weske, Mathias: Business Process Management: Concepts, Languages, Architectures. Springer, 2019.