Supporting Data-Aware Processes with MERODE

Monique Snoeck[1][0000-0002-3824-3214],Johannes De Smedt[1][0000-0003-0389-0275], and Jochen De Weerdt[1][0000-0001-6151-0504]

[1] KU Leuven, Belgium
{monique.snoeck, johannes.desmedt, jochen.deweerdt}@kuleuven.be

Abstract. Most data-aware process modelling approaches have been developed from a process perspective and lack a full-fledged data modelling approach. In addition, the evaluation of data-centric process approaches reveals that, even though their value is acknowledged, their usability is a point of concern. This paper presents a data-aware process modelling approach combining full-fledged domain modelling based on UML class diagrams and state charts with BPMN. The proof-of-concept has been implemented using the MERODE code generator, linking the resulting prototype application to a Camunda BPM engine, making use of RESTful web-services. The proof of concept is evaluated against 20 requirements for data-aware processes and demonstrates that the majority of these are already satisfied by this out-of-the-box prototyping approach.

Keywords: Conceptual Modelling, Process modelling, Data-Aware Processes, Model-Driven Engineering.

1 Introduction & motivation

For many years, data modelling, process modelling, and decision modelling have evolved as largely separate worlds, focusing on the respective modelling languages and methods, having different communities, conferences, and publication outlets [1]. While this "separation of concerns" allows focusing on the particularities of each domain, such silo-based approach comes with drawbacks as well. Data and processes are two concerns that underly different architectural viewpoints of a same system and integration is thus required to ensure consistency and correctness [2]. Architectural descriptions should come with defined correspondences and ensuing correspondence rules to express, record, enforce and analyse consistency between models and views. From an enterprise engineering perspective, defining the essential business concepts and their relationships through domain modelling and defining how the business operates through process modelling, should go hand in hand. Both perspectives should be aware of and integrated with the other perspective.

In recent years the importance of data aspects has been acknowledged by the process modelling community, and several approaches have been proposed, see [3]–[5] for overviews. Most of this research was initiated by experts from the process modelling domain, focusing on how to make processes data-aware, e.g. through case-based approaches [6], artefact-centric approaches [7], [8], object-centric approaches [9], developing connections to a database [10], or focusing on developing support for verifying process properties such as safety, liveness, etc., see for example [11], [12].

While research on data-ware processes provides progress towards an integrated approach, how data is addressed largely varies between approaches [3]. A full-fledged domain modelling approach focuses on defining business objects and their associations so as to provide an enterprise-wide definition of business concepts, as a common language shared by all business domains, and hence all business processes. A global perspective on the relationship between process modelling and domain modelling is still missing (e.g. in terms of an integrated meta-model), as well as a practical approach for modelers on how to tackle the balance between process modelling and domain modelling: what should come first, how are the models related to each other, and how do modelling decisions in one of the views affect the other view.

The goal of this paper is to investigate a data-aware process modelling approach that assumes the existence or joint development of a full-fledged domain model. In particular, the MERODE modelling method provides an approach to domain modelling [13] based on the Unified Modelling Language (UML), and formally grounded in process algebra [14]. While the MERODE-approach captures behavioural aspects through object lifecycle modelling and object interactions, it nevertheless also suggests the use of a business process layer to handle user and work-related aspects. Combining MERODE with process modelling results in data-aware process modelling, but -as opposed to most current approaches- the domain modelling is considered in its own right, rather than in function of process modelling. This paper contributes to the current state of the art by 1) providing a data-ware process modelling approach that relies on full-fledged domain modelling, 2) providing a concrete proof of concept for this suggested combination and 3) evaluating the resulting approach against the criteria presented in [4].

The remainder of this paper is structured as follows. Section 2 presents the state of the art on research that combines the process and data perspective. Section 3 describes the proposed approach based on the running example of [4]. Section 4 presents a detailed evaluation of the approach along the criteria defined in [4]. Section 5 presents a discussion and Section 6 concludes the paper.

2 Related work

In 2019, a systematic literature review on data-aware process modelling covering the period up till 2016 was published [3]. This review identified 17 different approaches to data-centric process modelling, described in 38 primary studies. While 13 papers relate to the Artefact-Centric approach proposed in [7], many other approaches have been developed as well. The results of this literature review also show that nearly each of the identified approaches have defined their own particular data representation construct. While some could be unified under the denominator of "Object" or "Entity", there still remains quite a large variation, and chosen constructs may not map to standard conceptual data modelling practices such as entity-relationship modelling or conceptual UML class diagrams. For example, certain approaches work on unstructured data like documents [15], others use Petri Nets to represent data [16]. As the authors state "*a general* *understanding of the inherent relationships that exist between processes and data is still missing*" [3].

Running the same query again in Web of Science and Scopus for the period 2017-2020 yielded 9 unique papers, 5 of them addressing an aspect of the artefact-centric approach (e.g. [17], [18]) or a specific subtopic of data and process integration like consistency, instance migration, the use of ontologies or process adaptation (e.g. [19]). No fundamentally new approach has been proposed.

A major drawback of some data-aware process modelling approaches is that data is often considered on a per-process basis (e.g. by only modelling the data relevant for the process at hand, see language requirement 3 in [20], or [19]). In some approaches a global domain model is considered as a given, and data-awareness mainly resides in bridging the process model to an existing data model, e.g. by developing a data querying and manipulation language to allow for data-aware process execution such as DAPHNE [10]. In [21] the notion of Artefact acts as a collection of process variables to be associated to a process instance, and serves as interface between the process model and the classes in a pre-defined data model. While providing a practical solution to process execution, this does not constitute a fully data-aware process modelling approach, where process models are inherently aware of the enterprise-wide conceptual data model of the domain in which they operate [22].

Process-aware domain modelling on the other hand, seems a largely unexplored topic. In object-oriented (OO) conceptual modelling (as e.g. in OO-Method [23] and MERODE [13]) business objects can have a state chart imposing sequences on the invocation of an object's low-level methods that manipulate its data. Business process modelling is absent or not fully elaborated. Artefact-centric modelling (e.g. [7], [24]) equips business artefacts with a lifecycle, and considers that the business processes result from the composition of services, which are associated to the business artefacts and their lifecycles through associations. Both in the OO approach and in artefact-centric approaches, object lifecycles capture behavioural aspects on a per-object/artefact basis, but are not meant to address the user perspective and defining work organisation as business processes, which was one of the motivations behind the PHILharmonicFlows approach [9].

In terms of integrating the process and data perspective, a significant amount of research has been performed in consistency verification, e.g. [25], [26], [12]. While formal verification may provide useful support for modellers to verify their work, most of the approaches are formal, not intuitive nor practical from a business point of view [12]. Even the most practical approach does not come with a priori guidelines providing modellers intuitive insights in the relationship between constraints embodied by the conceptual data model and those included in the process model. The survey published in [5] reveals that even though the value of data-centric approaches is acknowledged, their usability remains a point of concern. Moreover, as previous research has demonstrated UML class diagrams and BPMN to be practitioners' favourite languages [27], it makes sense to look for a solution based on UML and BPMN.

3 Integrating Process and Domain modelling

3.1 Architectural Layers

Combining data and process modelling boils down to a multi-modelling approach, where each model captures a specific viewpoint of the architecture [2]. Typical viewpoints are:

- VP1 - the data or business objects viewpoint, addressing the information that a business creates and maintains;
- VP2 - the business object behaviour viewpoint, addressing the relevant states in the life of a business object, from its creation to its final disposition and archiving;
- VP3 - the shared services viewpoint, describing how a service may provide access to information or perform changes to one or more business objects;
- VP4 - business process behaviour viewpoint addressing units of work and how these are combined to coarser-grained processes and governed by constraints such as task precedence;
- VP5 - business actor viewpoint, addressing the distribution of work across actors;

A good practice from a software architecture perspective, is to organize software into layers. Typically, layers address specific viewpoints, and layers implementing stable aspects of a system are positioned in the kernel of the software architecture, whereas elements with higher needs for flexible adaptation should be implemented in outer layers [28]. Business processes are typical examples of elements with a higher need for flexible adaptability, whereas the data layer tends to be more stable. Above-mentioned viewpoints would typically be arranged as shown in Fig. 1. Current data-aware process approaches do not address all these viewpoints explicitly. And while it may be useful to allow bypassing layers (e.g. for performance), it is a good practice to avoid direct access to a database and instead install intermediate services layers (VP3) to isolate the business process layer from the persistence layer [28]. Many current data-aware process approaches however, let business process activities directly access the data layer, thus skipping the shared service layer (VP3). Artefact-centric approaches do not have a separate process layer. In BALSA [7], the Business Artefacts and the Lifecycles address VP1 and VP2. VP4 is addressed by the Services that define units of work, and the Associations that may define constraints governing the services' access to artefacts thus defining (among others) precedence relationships between services. VP3 and VP5 are not addressed. In BAUML [21], the class diagram and state charts address VP1 and VP2. Activity Diagrams address the associations and OCL is used to define contracts for services. This allows addressing aspects of VP4. VP3 and VP5 are not addressed. In approaches that combine process modelling with access to data (e.g. [10]), the process model addresses VP4 and VP5 and the data model captures the data viewpoint (VP1). Artefact behaviour (VP2) is not captured.

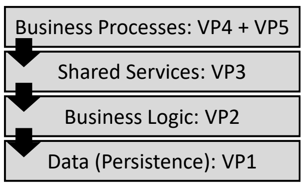


Fig. 1. Software Layers

PHILharmonicFlows [9] combines a data model (VP1) with Object Life Cycles (VP2) that define micro-processes, and defines macro processes too (VP4). Authorisations address VP5. VP3 is not addressed.

3.2 Layers in the MERODE approach

The MERODE method follows the principles of layers and identifies three major layers: the Enterprise layer (EL) is the bottom layer, the Business Process layer (BPL) is the top layer and in between sits an Information System Services layer (ISL). The Enterprise layer (EL) itself contains two sublayers. Business Objects are stored in the domain layer (DL). Additional logic is defined in the Object Life Cycles (OLCs). Transitions in OLCs are triggered by events, in MERODE called "Business Events". An Event-Handling Layer (EHL) offers an interface to invoke events and routes these to the relevant Business Objects that will handle the event by means of a corresponding operation effecting the required state changes. In between the EL and BPL sits the Information System Services layer (ISL) offering shared input and output services to access the EL. Output services allow querying the attributes and states of business objects. Input services capture input data but do not directly invoke operations on business objects. Rather, they achieve the requested operations by triggering one or several business events via the EHL. The business events and their handling through an EHL allows combining the advantages of an event-driven architecture with the advantages of the layered architecture, while also managing the transitioning to consistent states [29].

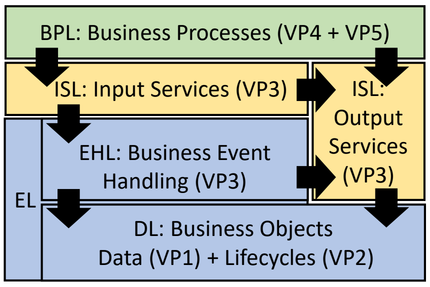

BPL: Business Processes (VP4 + VP5)
ISL: Input Services (VP3)
ISL: Output Services (VP3)
EHL: Business Event Handling (VP3)
EL
DL: Business Objects Data (VP1) + Lifecycles (VP2)

Fig. 2. MERODE layers: BPL (green), ISL (yellow), EL (blue) with two sublayers: EHL and DL.

The use of Business Events and an intermediate Event Handling layer is an important distinctive characteristic of the MERODE approach. Whereas usually a business process task's operational logic is defined in terms of SQL operations [10] or micro-processes defining read and write accesses to objects' attributes [9] (Fig. 3 left), in MERODE, the connection between the business process layer and the domain layer (or database layer) happens through the intermediary of input and out services and business events (Fig. 3, right). Input services can be kept simple or can incorporate logic that is reusable across different variants of similar tasks. Where to put what logic in view of balancing flexibility against business logic enforcement is discussed in [13], chapter 10

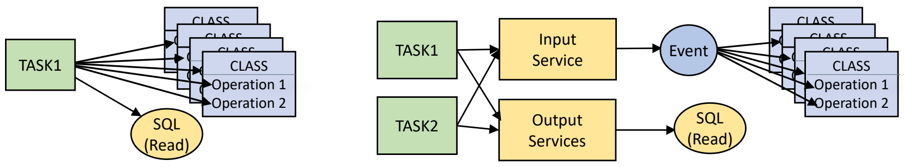


Fig. 3. Connecting the BPL to Data Objects: current approaches (left) vs. MERODE (right)

3.3 Example

We illustrate the proposed approach by means of the recruitment process from [4]. The example describes a process of people applying for a job, requiring reviews of their application forms before deciding to hire the candidate or not[1]. The following paragraphs describe the MERODE domain model used to generate the EL and ISL and how it can be connected to a BPL.

The MERODE Domain Model (EDG, OET and FSMs).

In the EL, the domain model defines the business objects and their associations by means of a UML class diagram in which all associations express existence dependency, therefore also called "Existence Dependency Graph" (EDG). It is obtained by means of systematic association reification for all associations that do not express existence dependency, thus identifying important "relators"[30] as explicit business concepts. For the given case, the class diagram is shown in Fig. 1. Each class in the class diagram is also equipped with a State Chart (Finite State Machine, FSM).

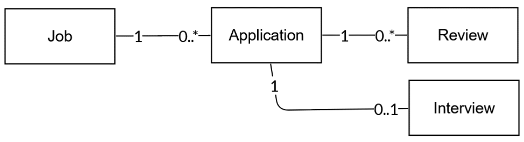
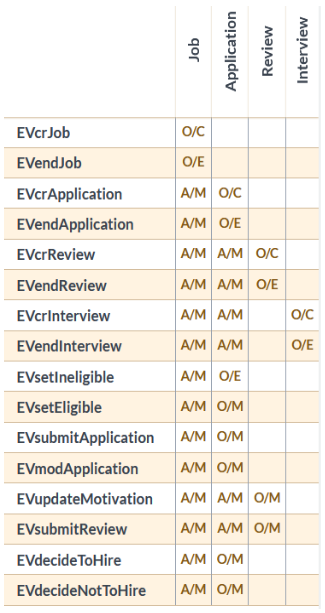


Fig. 4. UML class diagram (EDG)

MERODE defines business events as phenomena shared between the real-world and the information system, and operationalises these as call events (a subcategory of message events) that may trigger state changes in several business objects. The mapping of Business Events to Business Object types is captured through the Object-Event table (OET), where each cell indicates the type of state change that may be caused by the business event: C (creation), M (modification), or E(Ending). A marked cell thus means that the class of the corresponding column needs an operation to handle the event of the corresponding row.

The propagation rule defines a correspondence between the EDG and the OET: a master object will always be affected (at least indirectly) by the events affecting its dependents. This indirect participation is labelled 'A' (from Acquired), whereas the most dependent object affected by a business event is labelled as 'Owner' (O) of a business event. For example, 'decideToHire' is owned by Application, but will indirectly also affect the related Job as indicated in Fig. 5.



	Job	Application	Review	Interview
EVcrJob	O/C			
EVendJob	O/E			
EVcrApplication	A/M	O/C		
EVendApplication	A/M	O/E		
EVcrReview	A/M	A/M	O/C	
EVendReview	A/M	A/M	O/E	
EVcrInterview	A/M	A/M		O/C
EVendInterview	A/M	A/M		O/E
EVsetIneligible	A/M	O/E		
EVsetEligible	A/M	O/M		
EVsubmitApplication	A/M	O/M		
EVmodApplication	A/M	O/M		
EVupdateMotivation	A/M	A/M	O/M	
EVsubmitReview	A/M	A/M	O/M	
EVdecideToHire	A/M	O/M		
EVdecideNotToHire	A/M	O/M		

Fig. 5. OET

[1] For the ease of reading, the description can also be downloaded here.

Object behaviour is defined by means of FSMs showing how the events will cause state transitions. Each object type has a default lifecycle consisting of creating an object (triggered by any of the */C business events), having an arbitrary number of modifications in a random order (triggered by its */M events). Transitions triggered by a */E business event bring the object to the final state. A more specific FSM can be defined when needed. Fig. 6 show the FSMs for Review, Applications and Job. Interview has a default lifecycle. Because of the fact that a same business event may be reacted upon by several business objects, objects will synchronise and interact by means of joint participation to business events. The propagation rule allows a master to adjust its state upon activities and/or to restrict activities of its dependent object types. For example, the lifecycle of Application shows how events relating to reviews can only happen after an application has been considered eligible, and new reviews cannot be initiated once a final decision to hire or not to hire has been taken. In the lifecycle of Job, the decision to hire a candidate will cause a state change for the job ensuring that other candidates can no longer be hired.

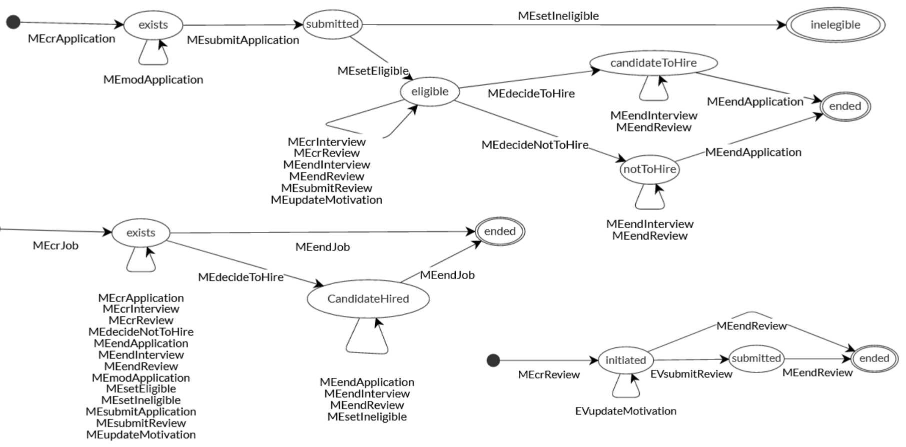


Fig. 6. Lifecycles of Application, Job and Review

The business process layer: Business process models

Activities in the business processes may invoke the output and input services to obtain information from the data layer and update information. While the EL captures behaviour on a per business object type basis, the BPL will capture other aspects of behaviour relating to users, task attribution and permissions. Assume the process for collecting reviews for a PhD candidate depicted in Fig. 7. The Faculty's HR consultant will start the process when an application arrives. The task "Check Eligibility" will use an output service to inspect the application file and then use an input service to trigger either the EVsetEligible or the EVsetIneligible business event for this application (see explanations in section 3.4). A review by the International Office is only needed in case of international candidates. This aspect relates to work organisation, and the criteria to request a review by International Office may change over time. By managing these criteria in the BPL, maximal flexibility for adjusting the criteria for performing this task is ensured. The next task of the HR Consultant is to ask for reviews from three professors, to be looped until three professors have accepted. Each professor may accept or refuse the request. As opposed to the solution proposed in [4], we choose not to model acceptance and refusal of the tasks as part of the lifecycle of the review object type. A Review object will only be created when the Reviewer actually writes a review. The reviews *requests* are thus distinguished from the actual reviews, the former being managed in the BPL and the latter being persisted and managed in the EL. The "Write Review" task may include updates if a professor decides to take time to think it over, and will be concluded by submitting the review. Thus, the task "Write Review", be it executed by International Office or by a professor, will trigger the business event "create review" when started, possibly trigger a number of "update motivation" events during its execution, and finally end by triggering a "submit" business event. The invocation of these events through the event-handling layer will trigger the necessary changes in the data layer while being subject to the constraints defined by the associations, multiplicities and FSMs in the EL. Connecting the BPL to the EL happens by means of a table mapping tasks to input and output services, as explained in [13], chapter 10. This technique suffices for simple one-to-one mappings. A more complex mapping would require integration with MERODE's extension for UI Design [31].

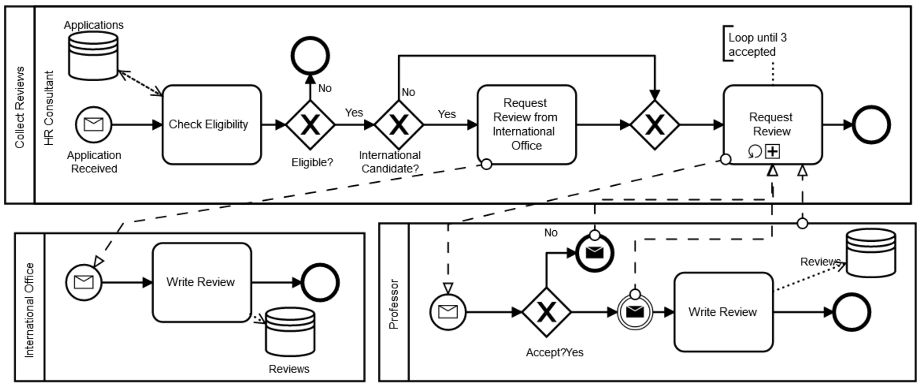


Fig. 7. Business Process for collecting reviews

3.4 Proof of Concept of Model integration

MERODE allows generating Java applications as prototypes of the EL with default IS services in an ISL. Assume the steps of reviewing an application and taking a decistion. Fig. 8 on the left shows the layered structure of such application. The generated Java Swing interface allows to "View" the details of an Application (❶) and from there to navigate to the details of its Reviews (❷). The User Interface (UI) accesses the objects making use of SQL. When a decision is taken to hire a candidate, a corresponding button will trigger the "EVDecideToHire" event (❸). The event-handler will check the permissibility of hiring the candidate against the status of that application (❹). If allowed, the state changes are performed by invoking the corresponding class's operation (❺). The result (error or success) is notified to the UI (❻).

To add a BPL layer, we used the Camunda BPM platform and the Camunda Modeler. Camunda[2] was chosen for being open source Java-based and providing a free demo account. In the Camunda BPM platform, Tasklists manage users' interactions with their tasks; The Camunda Cockpit web application presents the users facilities to monitor the implemented process and its operations; Camunda Admin is used to manage the users and their access to the system. For example, groups can be created and different authorizations can be managed for distinct participants.

To connect the MERODE application to the Camunda BPM platform, the EL and the EHL are wrapped and exposed as REST web-services by using the corresponding code-generator's option [32]. The Java user interface is then replaced by Camunda Task Forms and Service Tasks. The forms take the structure of an HTML document and manipulate business objects through the RESTful web services [33]. For now, these are created manually, but they could be generated from UI models [31]. Fig. 8, right shows the corresponding layered structure. The EHL ensures that sequences constraints as specified in the lifecycles are respected. The MERODE checking algorithms ensure that these lifecycles together define deadlock-free system behaviour [14].

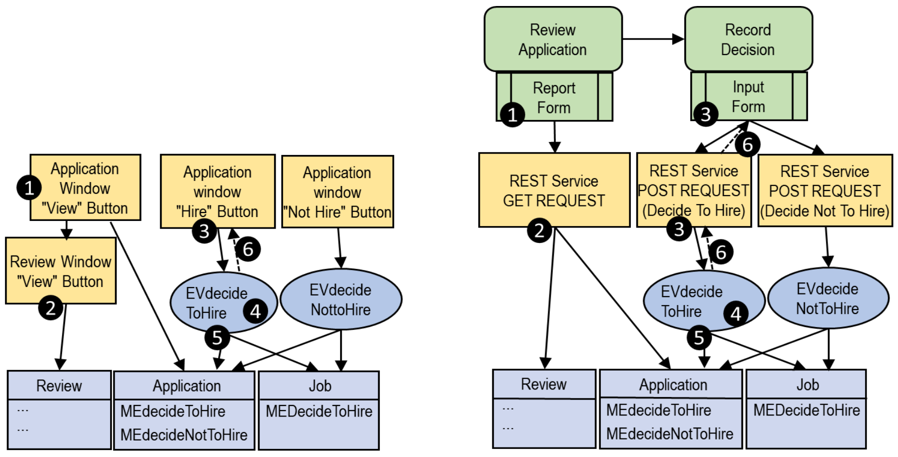

Review Application
Record Decision
Report Form
Input Form
Application Window "View" Button
Application window "Hire" Button
Application window "Not Hire" Button
Review Window "View" Button
REST Service GET REQUEST
REST Service POST REQUEST (Decide To Hire)
REST Service POST REQUEST (Decide Not To Hire)
EVdecide ToHire
EVdecide NottoHire
EVdecide NotToHire
Review
Application
MEdecideToHire
MEdecideNotToHire
Job
MEDecideToHire
1
2
3
4
5
6

Fig. 8. Layered architecture of a generated Java prototype (left) and after integration with a BP Engine (Right).

4 Evaluation

To evaluate to what extent the combination of MERODE and BPMN may support data-aware process modelling, we evaluate the prototype resulting from combining a generated MERODE-application interacting with a Camunda BPM engine through REST interfaces against the 20 requirements formulated by Künzle et al. [4]. These requirements are developed around four sets of important properties for object-aware processes. First, we elaborate on these four properties. Then, the different requirements are evaluated one by one.

[2] https://camunda.com/

4.1 General properties

Properties relating to data. Data should be managed based on object types (including attributes) which are related to each other. The EDG-part of the MERODE model addresses these requirements. Furthermore, [4] identifies a hierarchy between objects, whereby an object that references another object is considered a "lower level" object and the referred to object the "higher-level" object, e.g. a job application being the higher-level object instance of a set of associated reviews. This corresponds exactly with the notions of master and dependent as specified in the MERODE method, where the Job object type would be the master of the Application object type, which in turn is the master of Review and Interview object types.

Properties relating to activities. The different types of activities that are identified in [4] can be addressed. Per default the triggering of a single business event, and therefore input tasks relating to a single instance, are supported, as well as viewing the details of individual objects or lists of objects and navigating to related objects. While not provided per default in the prototype, more complex queries and transactions triggering multiple events can be programmed (cfr. chapter 9, [13]).

Properties relating to processes. The modelling and execution of processes is based on two levels of granularity: *object behaviour* and *object interactions,* a requirement that is satisfied by the MERODE method. In addition, the ISL and BPL allow for defining coarser-grained levels of behaviour (complex transactions and processes).

Properties relating to users. The notion of a user is not part of a default prototype MERODE-application: per default any user has access to any operation. But the Camunda Admin can be used to manage the users and their access to the system.

Monitoring. The overall state of the process is made transparent by means of default output services allowing to view the state of individual objects. If needed, specific queries can be run on the database to provide for more specific reports. The Camunda Cockpit provides additional information.

4.2 Individual Requirements

In what follows, we go over the different categories in more depth and clarify the twenty different requirements for the evaluation of the prototype. A requirement is labelled ✓✓ when already fully satisfied by the proposed approach; ✓ when minor extensions or adjustments would be needed, ○ when complex adjustments or extensions would be required the basic ideas of which have already been described, and with a '✗' if not supported.

Data.

R1 (data integration, ✓✓) describes the need for data objects that should comprise attributes and have connections to other objects [4]. This requirement is met by the MERODE EDG which presents the connected structure of the business object types.

R2 (Access to data, ○) pertains to authorisations. While the Camunda Admin allows managing this partly, data-based authorisation management would require setting an authorisation system in place. Full satisfaction of this requirement is possible, but it is not yet satisfied by the out-of-the-box approach.

R3 (cardinalities, ✓) requires the possibility to set cardinalities on relationships. The MERODE-approach allows setting a minimum constraint of 1, but for maximal constraints higher than 1, it uses the UML default of many (denoted as "*"). Setting a specific maximum number larger than one is possible but would require (straightforward) application specific coding. This requirement is thus largely satisfied.

R4 (mandatory information, ✓✓) requires the ability to distinguish between optional and mandatory attributes and to forbid proceeding further when mandatory attributes are missing. Per default, the generated code considers all attributes mandatory and will refuse the entering of incomplete data. Allowing for optional attributes is straightforward when hand coding or with minor adaptations of the code generator.

Activities.

R5 (Form-based activities, ✓✓) defines form-based activities as “comprising a set of atomic actions. Each of them corresponds to either an input field for writing or a data field for reading the value of an object attribute”. Making use of the REST interfaces and custom UIs, any type of form can be developed, or even generated automatically at runtime. Thus, R5 is satisfied through custom development.

R6 (black-box activities, ✓✓) activities enable complex computations or integration of advanced functionalities (e.g., sending e-mails or invoking web services). This requirement can be satisfied through custom coding and using the REST interfaces.

R7 (Variable granularity, ✗) requires the ability to distinguish between instance-specific, context-sensitive and batch activities so that users can to choose the most suitable action. The EL and ISL layers allow for providing these services, but to allow users choosing at run-time, CMMN should be used for the BPL rather than BPMN.

R8 (Mandatory and optional activities, ✓✓). Both at the level of FSMs, and at the level of the Business Processes, mandatory and optional events/activities can be defined. E.g. asking a review by International Office, may or may not be requested.

R9 (Control-flow within user forms, ✓✓) refers to adjusting the mandatory or optional character of an attribute on-the-fly while a user fills a form. Task Forms in Camunda allow for making certain attributes mandatory for the execution of an activity. The on-the-fly aspect of the requirement requires some custom-made logic.

Processes

R10 (Object behaviour, ✓✓) requires object type behaviour to be defined in terms of states and transitions. This requirement is obviously satisfied. Driving process execution based on states needs to be implemented at the business process layer, e.g. by means of rule-based events that react to conditions becoming true.

R11 (Object interactions, ✓✓) requires the possibility to process object instances concurrently while synchronising them when needed. In MERODE, creation dependencies are naturally enforced through the rules on existence dependency. A master object also has access to all information of its (direct and indirect) dependents, thus satisfying the need for aggregative information. Execution dependencies, e.g. when switching an object instance to a certain state depends on the state of another object instance, can be enforced by a master object managing execution sequences across all its dependents. Some execution dependencies may need to be managed by defining transactions that group events, or by defining a process that implements the required logic. For example, initiating a re-order when a product is out of stock would be implemented in the BPL, while the hiring of a candidate resulting in the automatic rejection of other candidates can be implemented as a transaction in the ISL.

R13 (Flexible process execution, ✗). When using BPMN to define the processes, flexibility of processes as described by Künzle et al. will not be possible. A possible solution could be using a case-based approach instead of BPMN.

R14 (Re-execution of activities, ✓✓) states that the re-execution of activities should be allowed, even if mandatory attributes are already set. The example that a person may change his/her application arbitrarily often until s/he explicitly agrees to submit it, is modelled by the self-loop 'EVmodApplication' in the Application FSM.

R15 (Explicit user decisions, ✗) requires allowing users to choose between execution paths. In the proposed approach, this would boil down to having gateways relying on user decisions rather than data to choose the next activity. Such user-based decisions could be captured by combining BPMN with DMN [34]. This is thus only partly satisfied by the proposed approach, unless DMN-support would be added.

User Integration

R16-R19 (○) deal with different forms of authorisations. Camunda offers a number of functionalities relating to the authorisations. A full-fledged authorisation system, combining the notions of user roles, their tasks and access to the required data is beyond the scope of the current proof-of-concept. The general design of such authorisation system has been described in [13], yet a practical implementation has not been made yet. These requirements are therefore considered as not yet satisfied.

Monitoring

R20 (Aggregated view, ✓✓) states that process monitoring should provide an aggregated view of all object instances involved in a process as well as their interdependencies. The database in the MERODE-application provides information about the objects, their dependencies and their states. The Camunda Cockpit provides information on tasks and users. Event logging is another source of information that may provide useful insights.

In summary, most requirements are at satisfied immediately or easily by the out-of-the box approach, though custom coding may be needed in addition to the default code generation. The addition of an authorisation layer (R2, R16-19) and support for different forms of flexibility (R7, R13, R15) need elaborating the approach further.

5 Discussion

Ideally, process modelling should be "data-aware" in the sense that an existing domain model is presumed to exist or to be developed jointly. Possibly process modelling may require revisiting the domain model. Similarly, domain modelling should be conscious of the business processes that need to be supported. As constraints set by a domain model will impact processes, the conceptual domain modeller should be aware of which processes are hindered or made possible in order to make the right decisions during his/her data modelling.

The main limitation of this research is that it limited itself to an out-of-the-box implementation using the default application generated by the MERODE-code generator and linking it to simple Camunda service and form tasks by means of the default REST web-services generated by the code generator. Nevertheless, this basic proof-of-concept combining MERODE with BPMN is able to satisfy a majority of the 20 requirements defined in [4]. This comes as no surprise given that the MERODE approach contains the main ingredients defined in the BALSA framework [7]. Augmenting the proposed architecture with DMN, and providing integration of MERODE with a case-based approach next to BPMN, could help to achieve the for now unsatisfied requirements on process flexibility. Investigating Camunda Admin's possibilities more deeply and implementing a data-based authorisation system requires further investigation and would be key to satisfy the authorisation-related requirements.

The review of data-centric approaches in [5] reveals that their usability is a source of concern. On the other hand, research also shows that UML-class diagrams and UML state charts are amongst the most-used modelling languages [27]. Combining these "modelling favourites" with BPMN could meet the usability concerns and stimulate the uptake of data-centric process management. Teaching of the MERODE + BPMN approach to students and to Enterprise Architects has already proven its ease of use. Enterprise Architects in particular value the innate data-centric process aspects embedded in the MERODE approach.

The whole process of generating and starting the web services, setting up the connection with Camunda, etc. requires several steps [33], but could ideally be done with less hazzle. The utopian goal would be to achieve this through code generation as well, to allow for process validation through the integrated prototyping of a collection of processes and the supporting information system with just a few clicks.

Finally, process verification has not been addressed in this paper. The process algebra formalisation of MERODE provides extensive consistency checking [35], but checking the consistency of the combined state charts against a business process model needs further investigation. An initial study has been published in [36], but requires further extension to achieve support for process verification as a complement to the above-mentioned validation through integrated prototyping.

6 Conclusion

While being preliminary, the proof-of-concept of MERODE and Camunda presented in this paper provides interesting opportunities to elaborate its functionalities. Considering other process implementation platforms and augmenting the proposed approach with DMN can provide additional pathways for future research. On the other hand, addressing the authorisation issues could prove a challenge. Besides addressing the unfulfilled requirements, development of a proof of concept with more complex models including completing the generated code by hand and using more elaborate BPMN models would allow to gain deeper insights into the merits of this combination. A formal evaluation of the approach could shed light on remaining issues, and how to make data-centric process management easier to use.

References

[1] F. Hasić, J. De Smedt, and J. Vanthienen, “Augmenting processes with decision intelligence: Principles for integrated modelling,” *Decis. Support Syst.*, vol. 107, pp. 1–12, 2018.

[2] “ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description.” [Online]. Available: https://www.iso.org/standard/50508.html. [Accessed: 11-Jan-2021].

[3] S. Steinau, A. Marrella, K. Andrews, F. Leotta, M. Mecella, and M. Reichert, “DALEC: a framework for the systematic evaluation of data-centric approaches to process management software,” *Softw. Syst. Model.*, vol. 18, no. 4, pp. 2679–2716, 2019.

[4] V. Künzle, B. Weber, and M. Reichert, “Object-aware business processes: Fundamental requirements and their support in existing approaches,” in *Frameworks for Developing Efficient Information Systems: Models, Theory, and Practice*, J. Krogstie, Ed. Hershey, PA: IGI Global, 2013, pp. 1–29.

[5] H. A. Reijers *et al.*, “Evaluating data-centric process approaches: Does the human factor factor in?,” *Softw. Syst. Model.*, vol. 16, no. 3, pp. 649–662, 2017.

[6] S. Haarmann, A. Holfter, L. Pufahl, and M. Weske, “Formal Framework for Checking Compliance of Data-Driven Case Management,” *J. Data Semant.*, 2021.

[7] R. Hull, “Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges,” in *On the Move to Meaningful Internet Systems: OTM 2008*, 2008, pp. 1152–1163.

[8] D. Calvanese, M. Montali, M. Estañol, and E. Teniente, “Verifiable UML Artifact-Centric Business Process Models,” in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 1289–1298.

[9] V. Künzle and M. Reichert, “PHILharmonicFlows: Towards a Framework for Objectaware Process Management,” *J. Softw. Maint. Evol. Res. Pract.*, vol. 23, no. 4, pp. 205–244, 2011.

[10] D. Calvanese, M. Montali, F. Patrizi, and A. Rivkin, “Modeling and In-Database Management of Relational, Data-Aware Processes,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11483 LNCS, pp. 328–345, 2019.

[11] A. Artale, D. Calvanese, M. Montali, and W. M. P. van der Aalst, “Enriching Data Models with Behavioral Constraints,” *Ontol. Makes Sense*, vol. 316, no. Dynamics 365, pp. 257–277, 2019.

[12] M. Estañol, M. R. Sancho, and E. Teniente, “Ensuring the semantic correctness of a BAUML artifact-centric BPM,” *Inf. Softw. Technol.*, vol. 93, pp. 147–162, 2018.

[13] M. Snoeck, *Enterprise Information Systems Engineering*, vol. 141. 2014.

[14] M. Snoeck and G. Dedene, “Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types,” *IEEE Trans. Softw. Eng.*, vol. 24, no. 4, pp. 233–251, 1998.

[15] C. P. Neumann and R. Lenz, “α− Flow: A Document-Based Approach to Inter-institutional Process Support in Healthcare BT - Business Process Management Workshops,” 2010, pp. 569–580.

[16] W. M. P. van der Aalst, C. Stahl, and M. Westergaard, “Strategies for Modeling Complex Processes Using Colored Petri Nets BT - Transactions on Petri Nets and Other Models of Concurrency VII,” 2013, pp. 6–55.

[17] P. J. Kiss and G. Klimkó, “A Reverse Data-Centric Process Design Methodology for Public

Administration Processes," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11709 LNCS, pp. 85–99, 2019.

[18] N. H. Ouali, M. Tmar, N. Haddar, and M. Tmar, "Models and Run-Time Systems for Data Intensive Workflow Applications," in *2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2017, vol. 2017-Decem, pp. 429–436.

[19] E. Rietzke, R. Bergmann, and N. Kuhn, "ODD-BP - an Ontology- and Data-Driven Business Process Model," *CEUR Workshop Proc.*, vol. 2454, pp. 409–415, 2019.

[20] S. Mertens, F. Gailly, and G. Poels, "A Generic Framework for Flexible and Data-Aware Business Process Engines," in *Advanced Information Systems Engineering Workshops*, 2019, pp. 201–213.

[21] G. De Giacomo, X. Oriol, M. Estañol, and E. Teniente, "Linking Data and BPMN Processes to Achieve Executable Models BT - Advanced Information Systems Engineering," 2017, pp. 612–628.

[22] F. Hasić, J. De Smedt, S. Vanden Broucke, and E. S. Asensio, "Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes," *IEEE Trans. Serv. Comput.*, p. 1, 2020.

[23] O. Pastor and J. C. Molina, *Model-driven architecture in practice: A software production environment based on conceptual modeling*. Springer Berlin Heidelberg, 2007.

[24] M. Estañol, J. Munoz-Gama, J. Carmona, and E. Teniente, "Conformance checking in UML artifact-centric business process models," *Softw. Syst. Model.*, vol. 18, no. 4, pp. 2531–2555, 2019.

[25] A. Deutsch and R. Hull, "Automatic Verification of Database-Centric Systems," vol. 43, no. 3, pp. 1–13, 2014.

[26] D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, and A. Rivkin, "SMT-based verification of data-aware processes: a model-theoretic approach," *Math. Struct. Comput. Sci.*, vol. 30, no. 3, pp. 271–313, Mar. 2020.

[27] D. van der Linden, I. Hadar, and A. Zamansky, "What practitioners really want: requirements for visual notations in conceptual modeling," *Softw. Syst. Model.*, vol. 18, no. 3, pp. 1813–1831, 2019.

[28] M. Richards, "Software Architecture Patterns." [Online]. Available: https://www.oreilly.com/content/software-architecture-patterns/. [Accessed: 13-Mar-2021].

[29] M. Snoeck, W. Lemahieu, F. Goethals, G. Dedene, and J. Vandenbulcke, "Events as atomic contracts for component integration," *Data Knowl. Eng.*, vol. 51, no. 1, pp. 81–107, 2004.

[30] G. Guizzardi, "Ontological Foundations for Structural Conceptual Models," University of Twente, 2005.

[31] J. Ruiz, G. Sedrakyan, and M. Snoeck, "Generating User Interface from Conceptual , Presentation and User models with JMermaid in a learning approach," in *Proceedings of the XVI International Conference on Human Computer Interaction*, 2015, pp. 25–32.

[32] N. Scheynen, "Construction of web services using the MERODE approach," Leuven : KU Leuven. Faculteit Economie en Bedrijfswetenschappen, 2016.

[33] I. Mohout and T. Leyse, "Enriching Business Process Simulation by integration with MERODE prototype applications," KU Leuven, 2020.

[34] F. Hasić, E. Serral, and M. Snoeck, "Comparing BPMN to BPMN + DMN for IoT Process Modelling: A Case-Based Inquiry," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 53–60.

[35] M. Snoeck, C. Michiels, and G. Dedene, "Consistency by construction: The case of MERODE," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2814, pp. 105–117, 2003.

[36] M. De Backer, M. Snoeck, G. Monsieur, W. Lemahieu, and G. Dedene, "A scenario-based verification technique to assess the compatibility of collaborative business processes," *Data Knowl. Eng.*, vol. 68, no. 6, pp. 531–551, 2009.