SUPPORTING DATA-AWARE PROCESSES WITH MERODE

Monique Snoeck[1[0000-0002-3824-3214]], Charlotte Verbruggen[1[0000-0003-0418-2633]], Johannes De Smedt[1[0000-0003-0389-0275]], and Jochen De Weerdt[1[0000-0001-6151-0504]]

{monique.snoeck, charlotte.verbruggen, johannes.desmedt, jochen.deweerdt}@kuleuven.be

[1]Research Center for Information Systems Engineering (LIRIS); KU Leuven, Belgium

PREPRINT: This paper has been accepted for the SoSym Journal

Please cite as follows:
Snoeck, M., Verbruggen, C., De Smedt, J. *et al.* Supporting data-aware processes with MERODE. *Softw Syst Model* (2023). https://doi.org/10.1007/s10270-023-01095-4

Supporting Data-Aware Processes with MERODE

Monique Snoeck[1][0000-0002-3824-3214], Charlotte Verbruggen[1][0000-0003-0418-2633], Johannes De Smedt[1][0000-0003-0389-0275], and Jochen De Weerdt[1][0000-0001-6151-0504]

[1] KU Leuven, Belgium
{monique.snoeck, charlotte.verbruggen, 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 and DMN. The approach is illustrated by means of an elaborated example with multiple business processes on top of a joint domain model. A 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 proposed approach is evaluated against 20 requirements for data-aware processes and demonstrates that the majority of these are already satisfied by the out-of-the-box combination of the Camunda BPM engine with the prototype generated from a MERODE domain model.

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

1 Introduction & motivation

Addressing multiple concerns, multiple viewpoints and creating multiple models is an essential characteristic of software development [1]. However, for many years, and to name just three viewpoints, 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 [2]. While this "separation of concerns" allows focusing on the particularities of each domain, such silo-based approach comes with drawbacks as well. Data, decisions and processes are concerns that underlie different architectural viewpoints of a same system and integration is thus required to ensure consistency and correctness [1]. Architectural descriptions should come with defined correspondences and ensuing correspondence rules to express, record, enforce and analyse consistency between models and views [1], [2]. 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. In a true multi-modelling approach, all perspectives should be aware of and integrated with the other perspectives. While the need for multi-modelling is widely acknowledged, there remain significant gaps to address. To set a step towards a multi-modelling approach that is sufficiently concrete to use the models in a model-driven engineering approach while keeping the scope manageable, this paper focuses primarily on data and process modelling.

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-aware 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. In addition, object-oriented domain modelling allows defining additional business logic by defining operations (next to attributes) and object lifecycles. 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 propose a data-aware process modelling approach that assumes the existence or joint development of a full-fledged object-oriented domain model and to evaluate it along the criteria for data-aware process modelling formulated in [4]. In particular, the MERODE modelling method provides an approach to domain modelling [13] based on the Unified Modelling Language (UML), formally grounded in process algebra [14] when it comes to defining object behaviour. 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. The current paper presents the further elaboration of the approach of which a first draft was presented and evaluated in BPMDS 2021 [15]. In particular, the approach is further refined and more thoroughly evaluated, and it is extended with use of decision tables, modelled with DMN, and the corresponding use of Business Rules tasks. This paper contributes to the current state of the art by 1) providing a data-aware 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 presents a high-level architectural overview of combined domain modelling and process modelling. Section 4 then presents the process and domain co-modelling approach based on BPMN and MERODE and focuses in particular on how to express the relation between the process elements and domain model elements. Section 5 presents a detailed evaluation of the approach along the criteria defined in [4]. Section 6 presents a discussion and Section 7 concludes the paper.

2 Related work

Multi-modelling has been addressed in several domains of research. In the following paragraphs, we review how the combination of data and process modelling has been addressed in enterprise modelling, in model-driven engineering, and in single-viewpoint approaches augmented with an additional viewpoint: data-aware process modelling and process-aware domain modelling.

Enterprise Modelling

Enterprise Modelling approaches typically address several viewpoints of a domain of interest by proposing different types of models, called model kinds according to [1], and where each model kind captures a specific viewpoint of the architecture [1]. Enterprise Architecture Frameworks typically distinguish three architectural layers (the Business Architecture, the

Information System Architecture and the Technology Architecture) and may propose a set of model kinds for each of these layers [16]. While the Zachman [17] and TOGAF [18] frameworks remain vague in terms of the modelling languages to use, ArchiMate [19] provides a very concrete multi-view modelling language. In ArchiMate [19] both Business Processes and Business Objects are part of the Business Architecture Layer. However, the structure of the domain model and of the business processes are not drawn as part of the Business Architecture Layer. UML class diagrams are positioned in the Information Architecture Layer, and the Business Architecture Layer only lists the business processes without defining the business process models. For the detailing of a process in a process model, ArchiMate refers to the use of a business process modelling language, e.g. BPMN. Business objects may represent an information asset relevant from a business point of view and should then be realized by data objects. For representing relationships between these passive structure elements, the notation for aggregation, composition, association, and specialisation is borrowed from the UML [20], today's industry and research standard for software modelling. The meta-model of ArchiMate defines access relationships that document which business objects are accessed by which business processes, but operational-level details are not within ArchiMate's scope. Object lifecycles or state transition diagrams are not part of ArchiMate's set of proposed model kinds. Thus, for a detailed modelling of data-aware processes, the modeler is referred to using BPMN and UML class diagrams, without further support for the practical integration of process models and data models. Amongst the different model kinds in 4EM [21], we find the concepts model and the business process model. 4EM does not define the use of object lifecycles as a model kind, although permissible state transitions could be modelled as part of the business rule model. In 4EM the business process model has a 'use and create' relationship to the business concepts model. For each activity in a process model, a diagram allows detailing how the activity consumes input and produces output in terms of information and/or material. The contents of these information or material flows are detailed by referencing their definitions in the concepts model. Given that the process model and concepts model use proprietary notations, the integration remains at a high level and the gap to operational data-aware processes is significant. In MERODE, the domain modelling part of the method is very clearly established, and BPMN is suggested as process modelling language. While [13] already outlines some basic ideas for connecting the process layer to the domain layer, a clear explanation on how the modeler expresses the relation between the process in BPMN and the MERODE domain is missing, as well as how such relationship could be made operational. MEMO [22] also proposes a multi-modelling approach. According to its meta-model, business processes are located in the Organisation part of the meta-model, and classes in the Information System part. Processes are connected to classes via the invocation of class operations. Here too, object lifecycles seem not to be one of the suggested model kinds. All-in-all, while enterprise modelling approaches do propose the combination of full-fledged domain modelling with process modelling, they lack details on how exactly to operationalize the connection between a process model and a domain model. In addition, the combination of lifecycles of business objects with business process modelling seems lacking in these methods.

Model-driven Engineering

Besides the more abstract enterprise modelling approaches, more concrete model-driven engineering approaches can be considered as well. A notable example is OO-Method [23], which combines domain modelling with state-charts, a functional model and a presentation model, thus covering all is needed for application development. However, business process models are not covered in [23]. Investigating model-based and model-driven approaches to User Interface

Design [24], shows that most of these approaches start from task models, which can be considered very close to business process models. However, the study also shows that integration with the data aspects is lacking in these approaches. Generally speaking, the field of model-driven engineering has recognized the lack of integrated modelling of different perspectives as a significant gap to address in future research. The MMQEF [25] provides a framework to assess the quality of a set of modelling languages used in combination for model-driven engineering, yet for now, there is no suggested best combination of modelling languages. Finally, the approach also has a lot in common with Domain-Driven Design [26]. This approach favours model-driven engineering and couples the domain model with well-chosen patterns at the heart of systems design. While the domain model (defined as a class diagram) has a prominent place in this approach, business process modelling is not part of the approach. The focus rather lies on connecting the user interface layer in a proper way to the domain layer.

Data-aware process modelling and process-aware domain modelling

Finally, specific and detailed data-aware process modelling approaches have been suggested as well. 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 [27], others use Petri Nets to represent data [28]. 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 2017-2022 yielded 13 unique papers, 5 of them addressing an aspect of the artefact-centric approach (e.g. [29], [30]) or a specific subtopic of data and process integration like consistency, instance migration, the impact of data changes on a process [31], or the use of ontologies or process adaptation (e.g. [32]). 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 [33], or [32]). 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 [34] 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 [35].

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], [9], [36]) 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. [37], [38], [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 [39], it makes sense to look for a solution based on UML and BPMN.

In summary, we find that enterprise modelling approaches lack detail, model-driven engineering approaches are incomplete, and that when considering both data-aware process modelling and artefact-centric modelling for a more detailed approach, none of the proposed approaches combines all possible views and model kinds in a single approach. Either the focus lays on the business process layer that is made data-aware by means of a connection to a database (e.g. [10]), yet without considering an enterprise-wide domain model or object lifecycles, or the business process aspects are only captured by means of state charts per business artefact (e.g. BALSA [7] and BAUML [34]), and the business process modelling is the missing viewpoint. The only approach that combines the viewpoints of domain model, object lifecycles and business processes is MERODE, thus having the potential to cover a larger number of the 20 requirements formulated by Künzle et al. [4]. The remaining gap is nevertheless that the connection between the business process model and the domain model needs to be detailed further, which is addressed in the next sections.

3 Integrating Process and Domain modelling

Before presenting the details of a process and data co-modelling approach in section 4, this section first provides an overview of the research method that was followed to develop the approach. Section 3.2 presents a high-level architectural overview of combined domain modelling and process modelling. In particular, we review the notions of architectural layers and how these are typically stacked. Section 3.3 then reviews how these principles are instantiated in MERODE.

3.1 Research Methodology

The research presented in this paper follows a design science approach constituting of successive iterations of a design-evaluate cycle, and making use of a diverse set of evaluation methods as advocated by [40]. Four major cycles can be identified. The first main cycle developed MERODE as an "object-oriented analysis" method and was evaluated against criteria relating to syntax, semantics, and coverage of concepts of object oriented analysis [41] (Analytical/Static Analysis + Optimization), usability by practitioners [42] (Observational/Field Study).

A second iteration considered the addition of Business Process Modelling, used an Analytical (Static Analysis + optimization) evaluation to demonstrate the approach's ability to support formal verification [43]. The third iteration focused on the understanding of domain modelling and lifecycle modelling. The 'build' step resulted in the development of model simulation capabilities through the application of model-driven engineering concepts [44–47]. This cycle included a large series of experimental validations [48–50].

The current and fourth iteration expands this model understanding to the combination of domain modelling and business process modelling. To ensure the relevance of the approach, requirements by practitioners [51] as well as difficulties learners experience with multi-modelling [52] have been investigated. To ensure the rigor of the approach, existing approaches have been considered (see related research section) as well as criteria against which the approach should be evaluated. A first conceptual evaluation of the feasibility of the model-driven engineering of a prototype for a combined domain and process model was presented in [15]. The current paper presents the further elaboration of the approach based on the results of the evaluation. In particular, it further details how to express the relation between the process in BPMN and the MERODE models, extends the approach with use of decision tables, modelled with DMN, and the corresponding use of Business Rules tasks.

The evaluation in this fourth cycle constitutes of an analytical evaluation and makes use of the criteria established in [4], while the running example and elaborated case study provide a descriptive evaluation of the "Scenario" type as well as an Analytical evaluation of type "Dynamic Analysis" through the implementation of the example and the ability to test the resulting implementation.

3.2 Architectural Layers

Combining data and process modelling boils down to a multi-modelling approach. Typical viewpoints that can be distilled from the different works reviewed in section 2 follows :

- VP1 - the data or business objects viewpoint, addressing the information that a business creates and maintains. This viewpoint corresponds to the use of data models.
- 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. This viewpoint corresponds to the use of state charts to describe object lifecycles.
- VP3 - the shared services viewpoint, describing how a service may provide access to information or perform changes to one or more business objects. This viewpoint corresponds to defining micro-services to build modular (service-oriented) applications.
- 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. This viewpoint is typically captured by means of a process model or –in the case of model-driven UI design– by task models.
- VP5 - business actor viewpoint, addressing the distribution of work across actors. This viewpoint can be part of a business process model, or – e.g. in the case of model-driven UI-design– may be captured by separate user models.

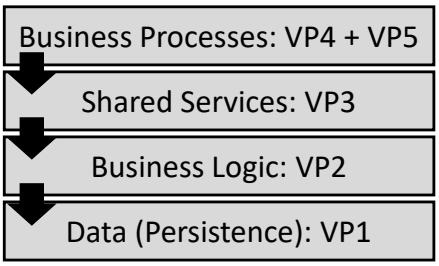

Fig. 1. Software Layers

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 upper layers [53]. 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 [53]. 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 address VP1 through their definition of data-related aspects and to a certain extent also VP2 in so far their operations also embody business logic. The Lifecycles address 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 BALSA. In BAUML [34], 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. 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.

While none of the reviewed approaches combines object lifecycles and business processes to describe behavioural aspects, these two model kinds do however constitute different views on behaviour, and both are needed to define the behavioural aspects of a sociotechnical system well. As explained in [54] "t*he interplay between data and processes can be very subtle, it is not sufficient to only study information and process models in isolation.*". In [29] this is illustrated by means of a tax payment process in the Hungarian administration: focusing on a process approach hampers the development of a one-stop-shop administration, whereas starting from the artefacts lifecycle and using the final state as the goal to reach, fosters a broader analysis and results in a more efficient administration. In line with Enterprise Architecture Frameworks such as Zachman [17] and TOGAF [18], we consider both the business process model and the domain model (including business object lifecycles) to belong to the "Business Architecture". The business process view captures behaviour from the perspective of work organisation but will typically only address a part of an artefact's lifecycle. A business object may be affected by several business processes, and the lifecycle will provide a global and unifying perspective on all processes that affect its state. For example, an order will be created by the ordering process, but its further lifecycle will be affected by shipping, invoicing, payment, return, and refund processes. The business process and object lifecycle (OLC) perspectives thus provide orthogonal and complementary views on a same reality: the business process view groups elements of behaviour according to work organisation, whereas an OLC groups behaviour on a per-object

basis. Both perspectives interact by the fact that business process activities may affect object states. The different perspectives must be considered and designed jointly to capture all relevant aspects and ensure the proper functioning of an organization and its supporting information system. Having only the business process perspective or the OLC perspective poses a limitation to the correct and complete modelling of a business's operations. Moreover, a multi-modelling approach also needs to clarify how the different models are related and should be organized in a global architecture that strives for a minimisation of ripple effects when some aspect of the business changes.

3.3 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 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 [55].

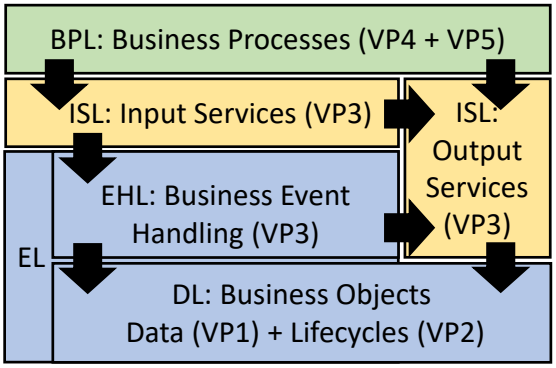

Fig. 2. MERODE layers: BPL (green), ISL (yellow), EL (blue) with two sublayers: EHL and DL.

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 direct access to a class' operations [56] and 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 output 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 superficially in [13], chapter 10.

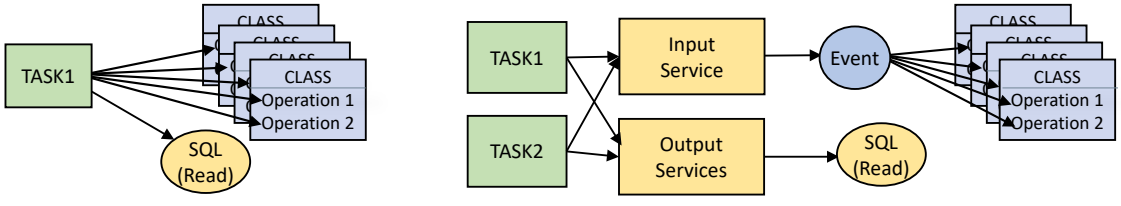

Fig. 3. Connecting the BPL to Data Objects: current approaches (left) vs. MERODE (right)

It should be noted that the layers in Fig. 2 are conceptual layers. In line with model-driven engineering, platform choices will dictate the transformation to an implementation architecture, and will entail choices related to, e.g., the centralised or distributed character of the implemented system.

4 Co-modelling Domain and Business Process Layer

Given the need of jointly addressing the business process modelling and domain modelling, this section explains how to combine a domain modelling focus with a process focus. We illustrate the proposed approach by means of an expanded version of the recruitment process from [4]. The example describes a process of people applying for a PhD position, requiring reviews of their application forms before deciding to hire the candidate or not. The example is expanded with additional processes to better illustrate the co-modelling of the domain and business processes. The following paragraphs first describe the general steps of a domain and BP co-modelling process (Section 4.1). Subsequently, section 4.2 describe the MERODE domain model that can be used to generate the EL and ISL. Section 4.3 focuses on the modelling of the business processes, and how the concepts in the business process models are connected to the concepts of the EL.

4.1 General Steps

The layers discussed above are a software organisation instrument, and the order of the layers is independent from how the requirements gathering and engineering is organized. Current software development practices mostly organize the requirements gathering step along the concept of "User Stories" (Agile Software Development) or "Use Cases" (UML). As software organisation is none of their concern, when users formulate system requirements during requirements elicitation, they usually tend to mix domain knowledge specifications with information system service requirements and business process requirements. The requirements cycle thus requires additional steps besides requirements gathering. A first step deals with classifying the different requirements according to the different layers, meaning that we will separate requirements that describe objects in the problem domain, information system service requirements and business process requirements, and classify all these requirements in the appropriate layer. Next, the models can be developed, starting with the stable layers first, and proceeding to the upper layers. Finally, the requirements validation step will verify the mutual consistency of all models, and validate the models with the user, e.g. by means of prototyping.

Fig. 4 illustrates the four steps of the requirements cycle: starting with requirements gathering and classifying the requirements according to the layers, followed by the bottom-up creation of the domain model, Information System Services and (re)designing the business processes, and finally validating with the users, e.g. through prototyping. The whole process is highly iterative, including micro-iterations within and between steps, and ends when a stable solution is obtained.

For the example on hiring PhDs, each of the steps can be illustrated as follows

a) During the requirements gathering, a user may utter a statement like: "*When I'm requested to perform a review of an application, I'd like to be able to refuse. In case I accepted, to facilitate the review process, I'd like to be able to see the text of the vacancy next to the applicant's file. It's also important that I can download the whole application as one pdf file so that I can easily read it on my tablet. When I have*

finalized and submitted my review, I'd like to receive a copy of my review in my mailbox.".

b) In the classification step, the statements of users need to be decorticated and separated to atomic requirements. The elementary requirements related to the process (e.g. the possibility to refuse a review request), to information system services (required service for downloading an application as a single pdf), and elements of the domain model (required information on business objects APPLICATION, VACANCY and REVIEW) are isolated into the right layer.

c) In the model building step, the business object REVIEW can be added to the domain layer, as well as the status 'submitted' in the corresponding OLC. A message event *refuse review request* may lead to a backward loop in the process model to find another reviewer.

d) The (updated) models can then be validated, making use of prototyping. The MERODE approach supports the fast prototyping of the domain model with default information system services and manual execution of the business processes. It is also possible to generate REST service interfaces to this application to connect Business Processes for simulation of processes in the Camunda BP Engine.

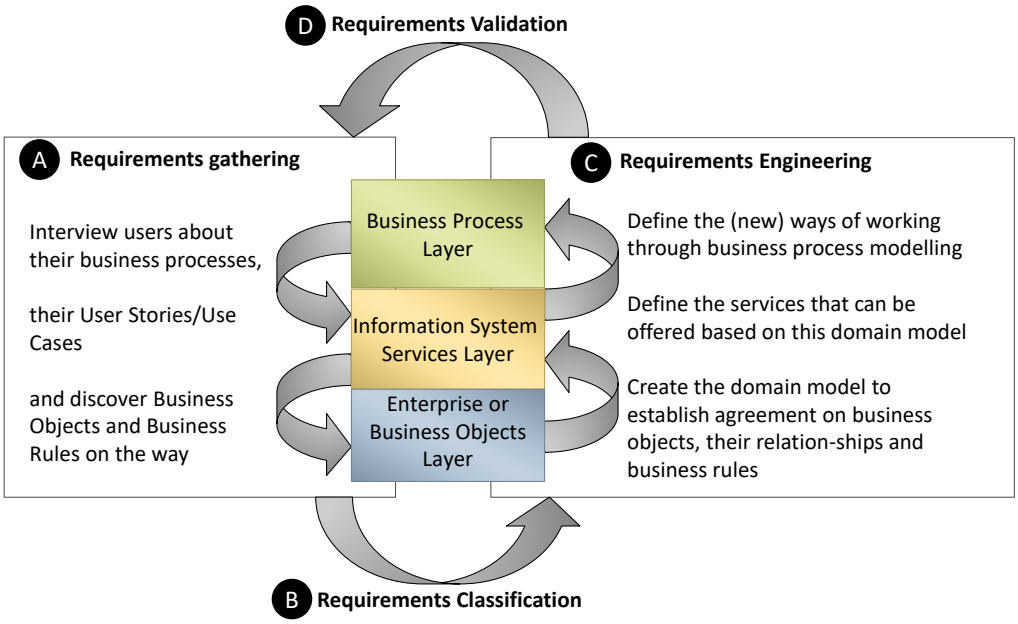

Fig. 4. Requirements gathering versus requirements engineering with MERODE

The following paragraphs further detail the creation of different models and a set of tables specifying the links between the models. Fig. 5. provides an overview of the created models and their links. At the bottom, the three MERODE-models are shown: on the right a class diagram (1) defining the conceptual data model for the domain; on the left one state chart per object type in this class diagram (3), and in the middle the object event table (2), linking the object types in the class diagram to the state charts. This is further explained in section 4.2. At the top-right, a collection of business process models (6) defines how work is performed. Business Rules tasks will link to a decision model defined using DMN (7). Activities may also require information from the domain model, or to register some changes to information (5). The former is realized by means of an output service, while the latter is realized by means of input services (4). This is further explained in section 4.3.

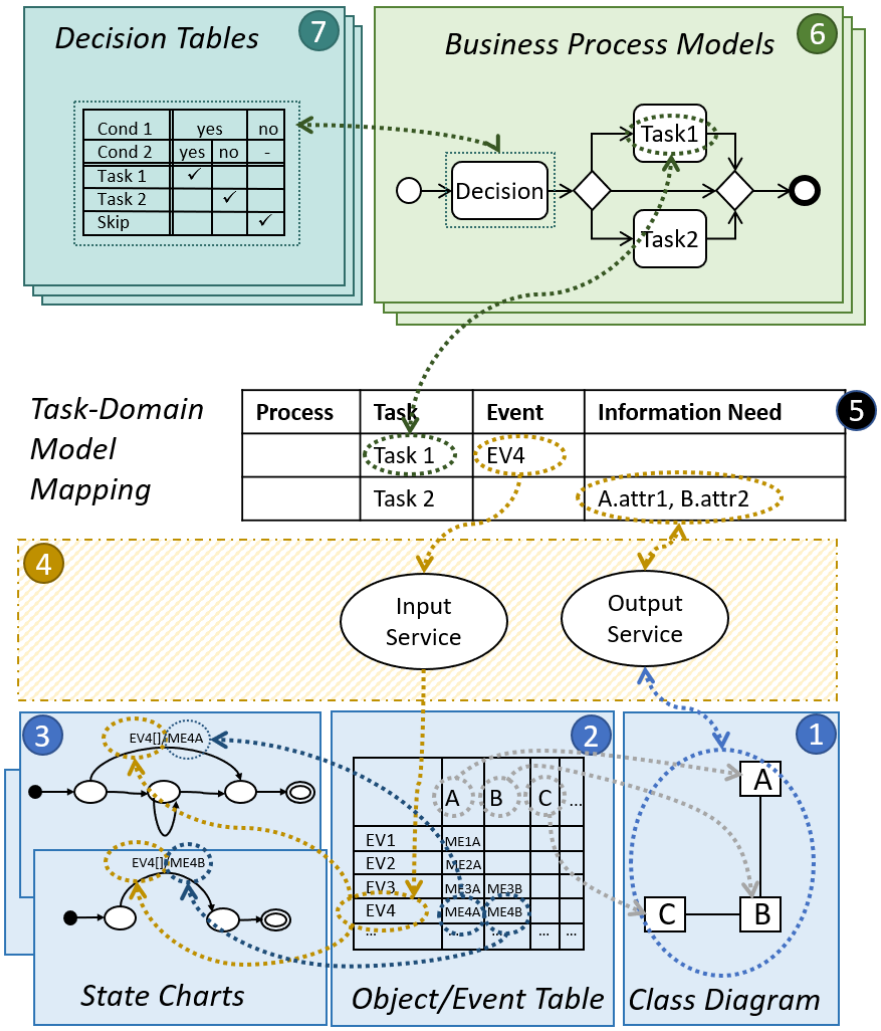

Fig. 5. Overview of the combined BPMN + DMN + MERODE approach

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

To illustrate the construction of a MERODE domain model, we use the small example from [9] and start from the list of requirements for hiring PhD Students[1]. The classification into layers is performed implicitly during the elaboration of the example. In the EL, the domain model defines the business object types 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' [57] as explicit business concepts. For the given case, information about the vacancy, the applications, the reviews of the application and notes from the interview are identified as business objects (VACANCY, APPLICATION, REVIEW, INTERVIEW). The class diagram is shown in **Fig. 6**. Note that while attributes are not visible in the class diagram, they can be defined and explored for

[1] The document with the full description of the case and the resulting models is provided in the Appendix. In addition, a more elaborated version of the same example is provided as well.

each class using the inspector tool of MERLIN. The same tool can be used to document a class's definition, preconditions and postconditions, to visualize the class's operations and associations, and to define constraints.

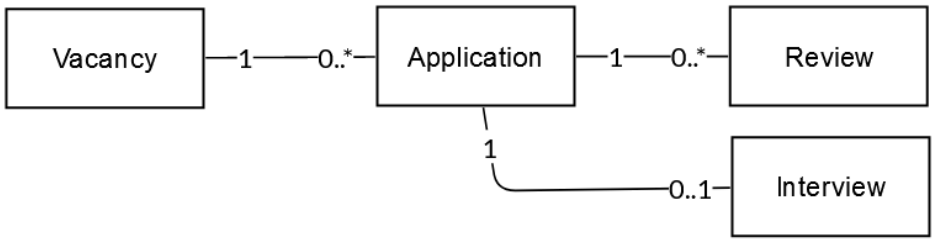

Fig. 6. UML class diagram (EDG)

Each class in the class diagram is also equipped with a State Chart (Finite State Machine, FSM) to capture the relevant state of a business object. For example, a VACANCY may be in draft mode, published, or closed. An APPLICATION can be in draft, submitted, reviewed, evaluated, etc. MERODE defines *business events* as phenomena shared between the real-world and the information system [58], and operationalises these as 'call events' (in UML these are 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.

By default, MERODE assumes at least two business events per business object: one for the start of its life, and one for the end of its life, thus leading to default events such as *EVcrVacancy, EVcrApplication, EVendVacancy*, etc. The elicitation of additional business events happens through the analysis of the business processes. The process of opening the vacancy makes clear that additional events are needed for triggering a state change due to submitting it for approval (*EVsubmitVacancy*), the outcome of the verification process, *EVapprove, EVdisapprove*), and publishing it (*EVpublish*). The next big step is receiving and evaluating applications, leading to the discovery of business events triggering state changes in the APPLICATION business object type.

Each of these events constitutes a row in the OET, and the involvement of a business object in an event gives rise to the definition of an operation in the corresponding class. Hence, the VACANCY business object will be equipped with a *MEsubmit* operation to handle the occurrence of the *EVsubmit* event, a *MEapprove* operation triggered by the *EVapprove* event, etc. The propagation rule defines a correspondence (consistency rule) 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, *EVdecideToHire* is owned by APPLICATION, but will indirectly also affect the related VACANCY as indicated in Fig. 5. Both APPLICATION and VACANCY are thus each equipped with an *MEdecideToHire* operation that will affect their state when this event happens: APPLICATION.*MEdecideToHire* will move the APPLICATION to a state 'CandidateToHire' and the VACANCY.*MEdecideToHire* will move the VACANCY to a state 'CandidateHired'.

	Vacancy	Application	Review	Interview
EVcrVacancy	O/C			
EVmodVacancy	O/M			
EVsubmitVacancy	O/M			
EVapprove	O/M			
EVdisapprove	O/M			
EVpublish	O/M			
EVclose	O/E			
EVendVacancy	O/E			
EVcrApplication	A/M	O/C		
EVsubmitApplication	A/M	O/M		
EVdecideNotToHire	A/M	O/M		
EVdecideToHire	A/M	O/M		
EVmodApplication	A/M	O/M		
EVsetEligible	A/M	O/M		
EVsetIneligible	A/M	O/E		
EVendApplication	A/M	O/E		
EVcrReview	A/M	A/M	O/C	
EVmodReview	A/M	A/M	O/M	
EVsubmitReview	A/M	A/M	O/M	
EVendReview	A/M	A/M	O/E	
EVcrInterview	A/M	A/M		O/C
EVendInterview	A/M	A/M		O/E

Fig. 7. OET

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. 8** shows the FSMs for REVIEW, APPLICATION and VACANCY. 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 (according to the semantics defined in the CSP process algebra [59]). 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 has been taken. In the lifecycle of VACANCY, the decision to hire a candidate will cause a state change for the vacancy to the state CandidateHired ensuring that other candidates can no longer be hired.

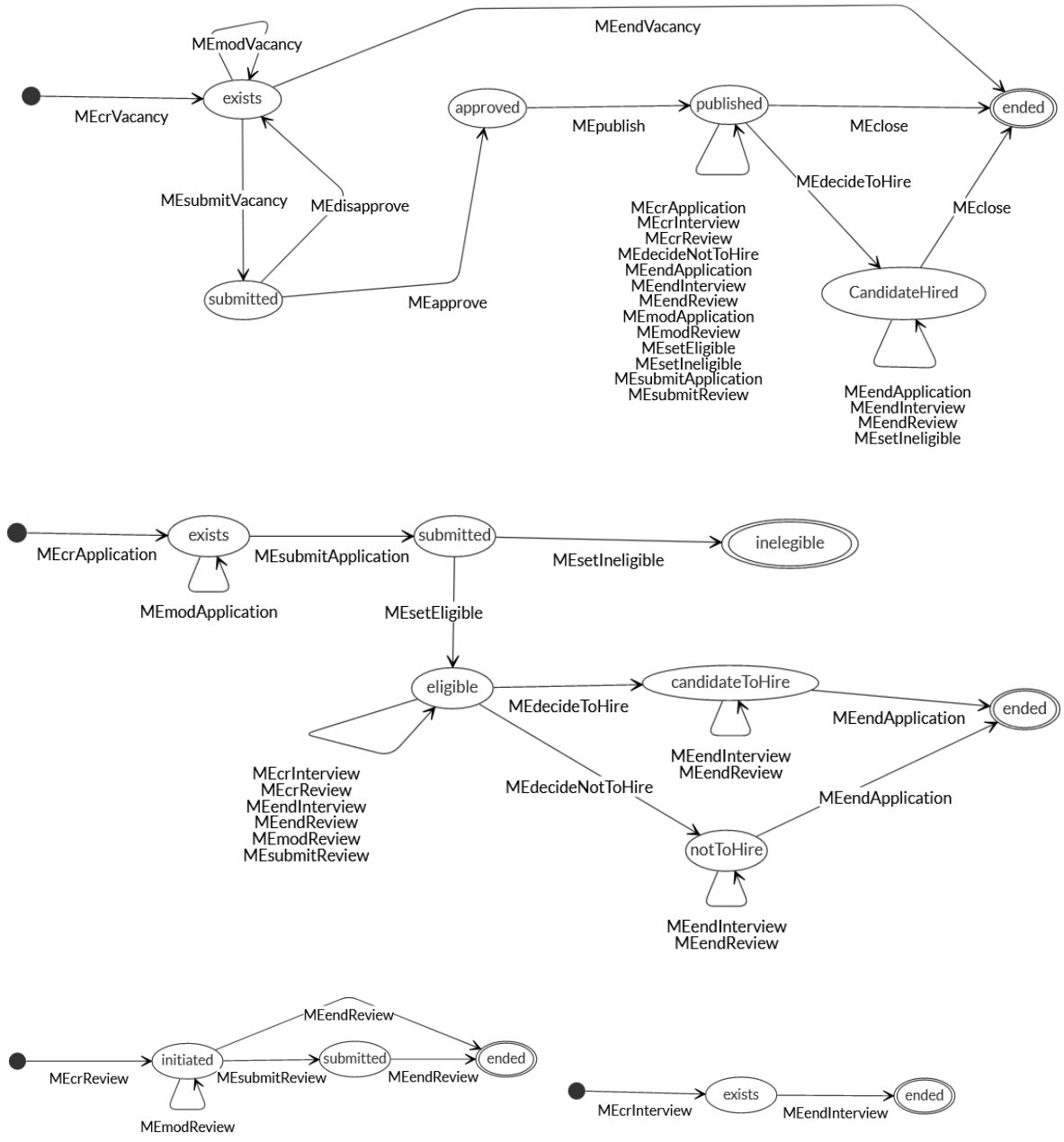

Fig. 8. Lifecycles of Application, Vacancy Review, and Interview

At this stage, it is already possible to simulate the domain model. Thereto, default input and output services and a default user interface can be generated. **Fig. 9** shows the interface of the prototype application for this domain model.

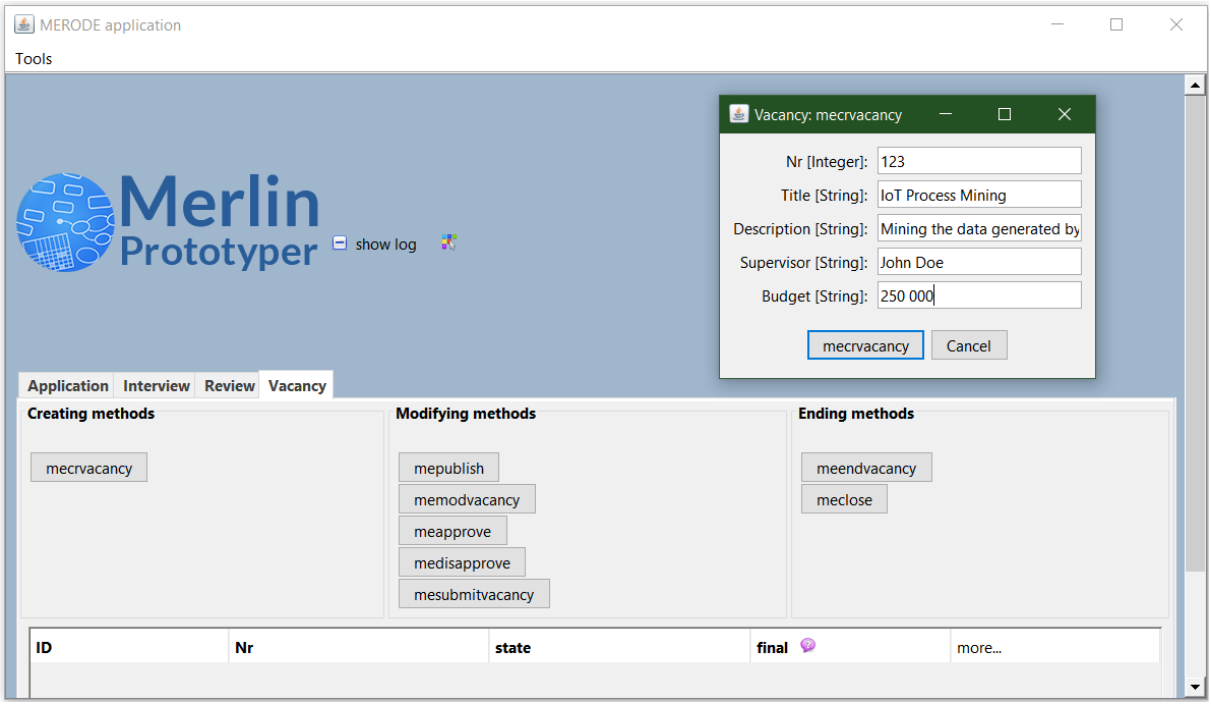

Fig. 9. Interface of the prototype application, with the pop-up resulting from triggering the Event *EVcrVacancy*.

4.3 The Business Process Layer: Business Process Models

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. **Fig. 10** shows the top-level process for hiring a PhD student, and **Fig. 11** shows the expanded subprocess for opening a vacancy. To obtain an executable process model, 'Open Vacancy' and 'Evaluate Application' are modelled as Call Activities so that the subprocess can be invoked from the global process (as required by Camunda).

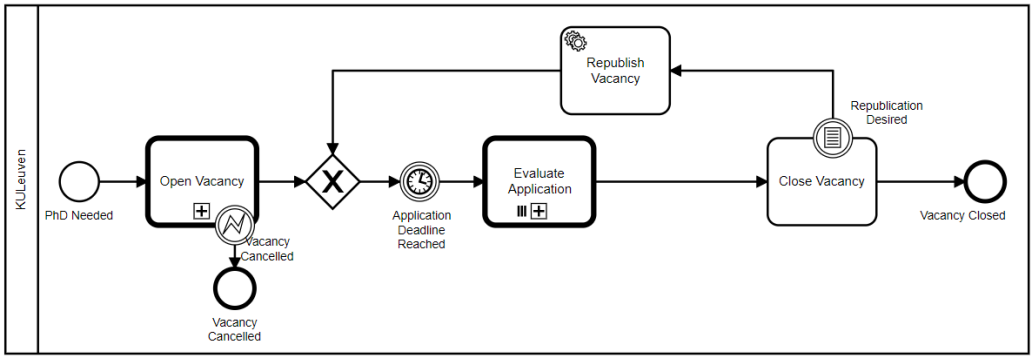

Fig. 10. Top-level process for PhD Hiring.

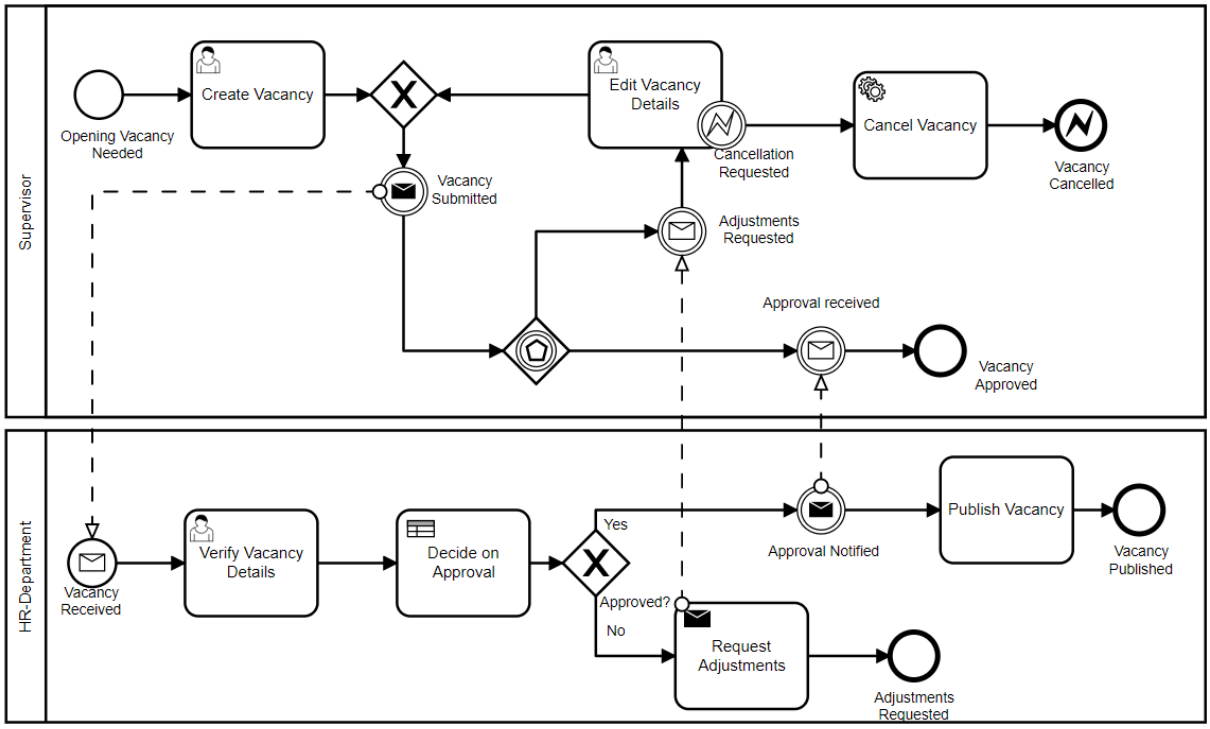

Fig. 11. Open Vacancy subprocess.

Activities in the business processes may invoke input and output services to obtain information from the data layer and update information. The distinction between input and output services follows the command-query separation principle advocated in domain-driven design [26]. In particular, input services are commands that trigger one or several events. Output services query the domain objects, and must be free of side effects. In the requirements engineering and modelling phase, we temporarily skip the detailed definition of information system services, and establish the correspondence between the process model and the domain layer directly. We do this by setting the correspondences with the business object types and their attributes and with the business event types. On the one hand, information needs per task will identify which business objects and attributes are needed as input, and on the other hand a task/business event mapping can be performed. We choose not to represent the data artefacts in the business process model, but rather to document the mapping using tables in a similar way as in [60]. Table 1 shows this mapping for the 'Open Vacancy' subprocess and the top-level process. For each process, all simple tasks are listed, and per task their information needs in terms of required domain objects and their attributes are identified as well as the business events needed to register the creation of new information or the update or deletion of existing information. As such, these tasks allow identifying the required attributes for a vacancy. In addition to the business events already explained above, the 'Edit Vacancy Details' task requires defining an *EVmodVacancy* business event. In the top-level business process, the tasks 'Republish Vacancy' and 'Close Vacancy' trigger a state change in the VACANCY object using the corresponding business events *EVpublish* and *EVclose*. Finally, some tasks do not need specific elements in the domain layer, as their execution is performed by a service not interacting with data (e.g. it is assumed that the content of the 'Request Adjustments message' is not stored in the domain model). It should be noted that in its current form, Table 1 is used as a requirements engineering tool, and cannot yet be used for automated code generation. As explained in section 4.4, further detailing of information system services is required.

Table 1 Task-Domain Model mapping for the Open Vacancy subprocess

Process / DT	Task	Invoked Business Event	Information Needs
Open Vacancy (Supervisor)	Create Vacancy	EVcrVacancy	Vacancy.*
	Edit Vacancy Details	EVmodVacancy	Vacancy.*
	Cancel Vacancy	EVendVacancy	Vacancy.*
Open Vacancy (HR)	Verify Vacancy Details	EVmodVacancy	Vacancy: details, description, available budget, DescriptionQuality
	DecideOnApproval	EVapprove, EVdisapprove	Vacancy.*
	Publish Vacancy	EVpublish	Vacancy.*
	Request Adjustments	--	--
Top-level Process	Close Vacancy	EVclose	Vacancy
	Republish Vacancy	EVpublish	Vacancy.*
Approve Vacancy	Approve	EVapprove	Vacancy.*
	Disapprove	EVdisapprove	Vacancy.*

The decision on approval could be modelled as a business rule (BR) task. However, Camunda assumes that BR tasks are executed automatically. Therefore, we precede the (automated) decision task with a human check task 'Verify Vacancy Details', where a human actor sets the values of all decision parameters: are details complete, is the description quality OK, and is there sufficient budget? The subsequent decision task can be executed based on the decision table shown in Table 2.

Table 2 Decision Table for Approving a Vacancy

Approve Vacancy | **Hit Policy:** Unique

	When Input string	And Input string	And Input string	Then Output string	Annotations
1	Available Budget < RequiredBudget	-	-	Disapprove	Invoke the EV_disapprove event
2	Available Budget >= RequiredBudget	Details = Incomplete	-	Disapprove	Invoke the EV_disapprove event
3	Available Budget >= RequiredBudget	Details = Complete	DescriptionQuality = Not OK	Disapprove	Invoke the EV_disapprove event
4	Available Budget >= RequiredBudget	Details = Complete	DescriptionQuality = OK	Approve	Invoke the EV_approve event if all vacancy details have been provided both in Dutch and in English and proof of sufficient budget is available.
+	-	-	-		

As shown in the last two rows of Table 1, the DT's output corresponds to the invocation of the corresponding business events (*EVapprove*, *EVdisapprove*) via an information system service. All required input data can be mapped to attributes of the VACANCY business object.

The process for evaluating an application is depicted in **Fig. 12**. The Faculty's HR consultant will start the process when the deadline for application has been reached. The task 'Check Eligibility' is a BR task that will use output services to inspect the application file and then use an input service to trigger either the *EVsetEligible* or the *EVsetIneligible* business event for this application. 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. In order for the data-based gateway to decide which path to follow, the outcome of the decision needs to be passed on from the 'Decide on Review Request' task to the gateway. In the simplified version we choose not to treat the request as a business object, but to keep the information about issuing the request, its acceptance or refusal as events in the business process layer. These

events can be logged for process monitoring purposes, but the consequence is that for a refused request no object is created in the domain layer but it is only kept as data object in the process. In the more elaborated solution in the appendix, an alternative is given where a domain object is created as result of the request for a review[2].

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', will trigger the business event *EVcrReview* when started, possibly trigger a number of *EVmodReview* events during its execution, and finally end by triggering an *EVsubmit* business event.

Working with a business process layer allows adding or changing business processes, and such addition or change may not always require modification of the domain layer. It could e.g. be decided that a review by the international office is needed in case of international candidates. The process model in **Fig. 12** can easily be adjusted to reflect this new requirement, without needing to change the domain model. This aspect relates to work organisation, and the criteria to request a review by international office may change over time. the criteria for routing an application through the international office can be modelled at the level of a gateway in this process. By managing these criteria in the BPL, maximal flexibility for adjusting the criteria for performing this task is ensured.

The invocation of the 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 by means of a table suffices for simple one-to-one mappings. A more complex mapping would require integration with MERODE's extension for UI Design [61].

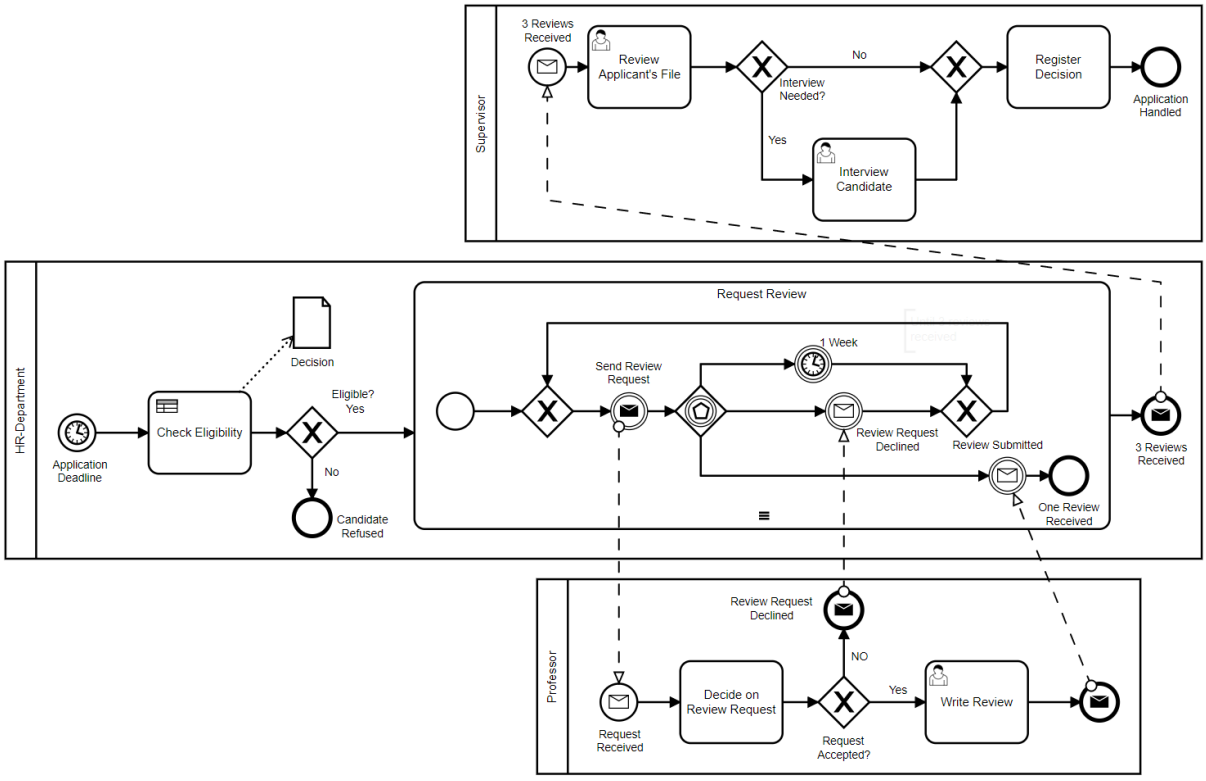

Fig. 12. Business Process for evaluating an application.

[2] The data objects are not included in Figs. 10-12 to keep them more legible. In the appendix, the data objects have been added to Fig. 12.

4.4 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 decision. **Fig. 13** 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[3] was chosen for being open source Java-based and providing a free demo account. In the Camunda BPM platform, Task lists 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. In addition, Camunda supports the integration of DMN: decision tables such as Table 2 can be handled in a fully automated way.

As Table 1 is at this point in time still an informal requirements engineering instrument, some additional work is required to connect the MERODE application to the Camunda BPM platform. Most importantly, the process models need to be made executable by defining the proper process variables, and linking tasks to application services. To obtain the services needed by the process engine, input and output services need to be defined. The Proof of Concept makes use of the default input and output services obtained from the domain model: the EL and the EHL are wrapped and exposed as REST web-services by using the corresponding code-generator's option [62]. 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 generated default RESTful web services [63]. For now, the forms are created manually, but they could be generated from UI models [61]. A demo of the implemented example can be found online[4], as well as the description of a more elaborated example in the appendix. Fig. 12, right shows the corresponding layered structure. The EHL ensures that sequence constraints as specified in the lifecycles are respected. The MERODE checking algorithms ensure that these lifecycles together define deadlock-free system behaviour [14].

[3] https://camunda.com/

[4] http://merode.econ.kuleuven.ac.be/MERODExBPMN.html

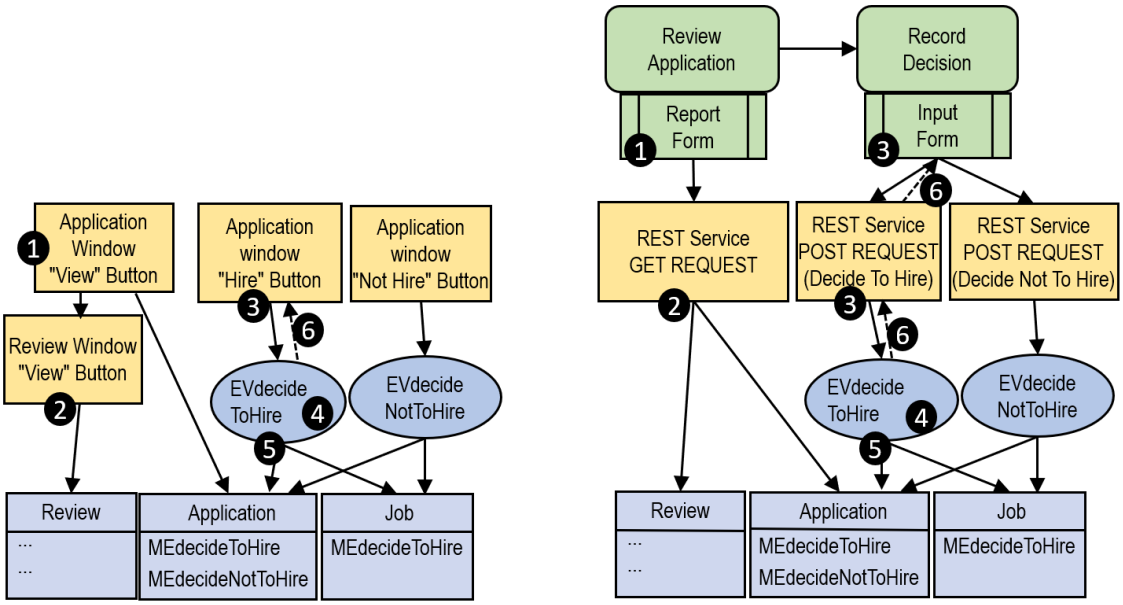

Fig. 13. Layered architecture of a generated Java prototype (left) and after integration with a BP Engine (Right).

5 Evaluation

To evaluate to what extent the combination of MERODE and BPMN may support data-aware process modelling, different evaluation methods, in line with [40] can be considered. To perform an analytical evaluation, different sets of evaluation criteria can be considered. The most elaborated set of functional evaluation criteria are the 20 requirements formulated by Künzle et al. [4]. The DALEC framework [3] also provides a large set of criteria. However, the functional and modelling perspectives are covered in less detail compared to the requirements of [4], while on the other hand more criteria can be found addressing process deployment, execution and evolution. As the present paper focuses on the modelling aspects, an analytical evaluation is performed according to the 20 requirements in [4], based on the experiences with implementing the prototype (both for the smaller and the larger case study) resulting from combining a generated MERODE-application interacting with a Camunda BPM engine through REST interfaces. Künzle's requirements are developed around four sets of important properties for object-aware processes. First, section 5.1 elaborates on these four properties. Then, the different requirements are evaluated one by one in section 5.2. An extensive comparative evaluation of other approaches can be found in [4] for approaches until 2011, and in [3] for approaches until 2016. Both comparative evaluations show that existing approaches are still failing to meet a substantial set of criteria. The PHILharmonicFlows approach aims to fill these gaps. Section 5.3 therefore compares the MERODE+BPMN approach to PHILharmonicFlows as this is likely the most complete approach today. Finally, the running example and elaborated case study provide a descriptive evaluation of the "Scenario" type as well as an Analytical evaluation of type "Dynamic Analysis" through the implementation of the example and the ability to test the resulting implementation.

5.1 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 Vacancy 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.

5.2 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.

R12 (Process-oriented view, ✓). Refers to the possibility of distinguishing between optional and mandatory activities in a process. In CMMN, this can be captured by distinguishing between mandatory and discretionary items. Until CMMN is integrated in the approach, in BPMN a gateway can be used to allow skipping an optional activity.

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 (e;g. CMMN) 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. This boils down to having gateways relying on user decisions rather than domain data to choose the next activity. As shown in the above examples, this requires the outcome of the user-based decisions to be captured as a data element. In the case of the Vacancy Approval process, the user reviews the vacancy and adds judgements to the vacancy object (e.g. set Description-Quality to OK), upon which a Business Rule task can be used to execute and register the decision. The new object's state can be read by the gateway to route the process accordingly. In the case of the Review process, the user decision is registered as process variable that is then used by the gateway. This illustrates the different ways of combining BPMN with DMN [64] or having simple user-based routing.

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]. A practical implementation has been elaborated in the context of B-MERODE [65]: in the EDG relevant business objects are included and labelled as participants. A participant-event table determines what participant is allowed to trigger what event. Furthermore, constraints can be added to manage the "vertical" authorisation: e.g. ensuring that reviewers can only update their own reviews. Considering the separate requirements, R16 (Data permissions) can likely be addressed following the B-MERODE approach. R17 (Process authorization) should be handled in the Camunda Cockpit. R18 (Differentiating authorization and user assignment) cannot be addressed easily, whereas R19 (Vertical authorization and user assignment) can be addressed using the B-MERODE approach. How exactly to combine the B-MERODE approach with the authorisations provided by Camunda requires however further research. 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. In the prototype application, the database can be directly queried by using the database interface (HSQL database manager). The Camunda Cockpit provides information on tasks and users. Event logging is another source of information that may provide useful insights. Per default, the prototype application logs all actions in a log file, with the purpose of providing the business analyst information on test coverage [66].

In summary, most requirements are 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) would require further elaboration of the approach.

5.3 MERODE versus PHILharmonicFlows

To compare MERODE with the PHILharmonicFlows framework (PF for short), we considered the papers listed on the PHILharmonicFlows project page [67] and selected the publications

that, according to their title or abstract, describe the PF framework or tools. The following sources were consulted: [9], [68–70].We will not provide a detailed explanation of the PF framework in this paper, but direct the reader to the mentioned sources for the substantiation of the comparison below.

Both approaches show important similarities:

1. Both approaches combine a data model with lifecycles per object type.
2. Both approaches have split levels of process definition: micro and macro processes in PF versus FSMs and BP models in MERODE.
3. Both approaches are supported by a modelling tool that provides model verification .

There are nevertheless some important differences as well. The discussion of these differences can be structured according to the properties listed in section 5.1

Differences related to data

1. PF has a semi-hierarchical relational data model (data levels and unidirectional relations) that still allows self-references and cyclic relations, while in MERODE the EDG has stronger semantics for ED relationships that have been formalized by means of process algebra, and, based on this formalization, forbids self-references and cyclic relationship [14, 71].
2. PF has a "creation context" that ensures the cardinality constraints from the data model are maintained at runtime. MERODE automatically ensures the observation of maximum one multiplicities, issues warnings for the non-observation of minimum one cardinalities, enforces referential integrity constraints resulting from the existence dependency associations, and allows the modeller to define path constraints in the modelling tool. Other types of constraints are formally defined in the specification of the approach, but are not (yet) included in the implementation of the modelling tool (e.g. uniqueness constraints).

Differences related to activities

3. In PF, activity forms are generated based on the states of micro processes. In MERODE, activities are part of the business process layer.

Differences related to processes

4. A major difference is that PF is data-driven while MERODE is event-driven, resulting in a different definition of the lifecycles. In PF, micro process types can be described as lifecycles of object types in which each state corresponds to an activity of one or more users and specifies the attributes for which a value can be set at that point in the lifecycle of an object type. Because FP is data-driven, the enabled state of an object can be determined based on its available attribute values. The interactions between states of different object types have to be modelled in separate macro process types. In case more than two objects need to interact, this can be achieved by the use of parallel paths in the macro process. In MERODE, lifecycles are defined as simple FSMs where the label of a transition represents all events that can trigger the transition. The states represent the progression of a single object instance through its lifecycle, rather than being defined in terms of the values of its attributes. Consequently, object types will typically have less attributes in MERODE than in PF, given that some attributes become obsolete (e.g.: the Boolean attribute "ended" can be discarded as the final "ended" state of the FSM captures all relevant information). Given that the OET defines which object types participate in shared events, the FSMs inherently defines object interaction in an event-driven way. Importantly, this interaction is a multi-party interaction: more than two objects can react to the same event, thus providing very compact modelling compared to how this is

addressed in macro process types in PF. The order of entering/modifying attribute values is not defined and user roles are not allocated to given process tasks.

Differences related to users

5. PF defines data object types and user types separately, while MERODE makes no distinction: user types are to be included in the EDG as data object types. In PF, User types have attributes describing the user role, and authorization is implemented with constraints on these attributes (e.g. course coordinator and professors must have the same value for the attribute "faculty"). Relation between users and user roles is defined at runtime. Although MERODE does not explicitly define user roles, in many cases, the existence dependency allows modelers to easily embed the required constraints as path constraints. The generated application allows to create new instances of the data object types representing the user and the user roles, and managing them as defined in the models.
6. PF includes information flow modelling regulating the authorization of the users and the generation of forms that allow the authorized users to complete their tasks. This is not included in MERODE.

Differences related to monitoring

7. Both approaches provide suitable monitoring at runtime.

It is important to observe that the PHILharmonicFlows framework was developed with the main goal of meeting the requirements listed in section 5.2 [72], while MERODE was developed independently. Therefore, the symbiosis between PF and the requirements is much larger than between MERODE and the requirements. In order to counteract this bias, we list some additional differences between MERODE and PF below:

8. PF is a proprietary modelling language, given that the business process has to be modelled as micro and macro process types as defined by the framework. A major issue for practitioners is the high effort required in learning how to apply new model-based software engineering approaches and how to use the corresponding tools [51]. MERODE mitigates this problem by using a set of industry standards as modelling languages for the enterprise layer. The business process layer allows the modeller to use their process modelling language of choice for specifying the processes platform independent. Because of its combination of several independent modelling language, MERODE truly is a multi-modelling approach.
9. In PF, the micro and macro process types only capture the to-be-implemented business process. In MERODE, the FSMs are a complete representation of all possible paths through the lifecycle of an object type irrespective of a specifically considered to-be-implemented business process, whereas the to-be-implemented business process is modelled in the business process layer. The sequence in which events are invoked in the business process must always comply with all FSMs. This allows the modeller to separate the allowed object behaviour (FSMs) from the actual workflow (BP layer) which is more likely to change over time. The models in PF are more intertwined with each other given that the interaction of micro processes is explicitly modelled in a macro process (see also bullet point 4). Therefore, the layered architecture of MERODE provides a separation of concerns, allows the modeller a certain level of flexibility when it comes modelling the business process. The cost of this flexibility is the lack of formal integration between the BP layer and the other layers. This paper has addressed that challenge by integrating the

MERODE approach and BPMN, but it would equally be possible to define a formal integration with other process modelling languages, e.g. CMMN.

10. The PF modelling tool and runtime environment do not seem to be publicly available (at least not through the PF website), whereas MERODE provides a dedicated website[5] with a publicly available modelling tool and code generator. Additionally, the website contains supplementary materials such as an FAQ for the tool installation, course slides and a database of cases.

6 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.

While the modelling part has been already extensively applied, the main limitation of this research is that the operational validation of the transition from model to code is limited 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 and DMN to realize a running process and data-aware application 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]. 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 along the proposal of B-MERODE requires further investigation and would be key to satisfy the authorisation-related requirements.

A 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 [39]. Combining these "modelling favourites" with BPMN could meet the usability concerns and stimulate the uptake of data-centric process management. While the above evaluation shows that the combination of MERODE, BPMN and DMN does not yet address all the requirements set forward in [4], the fact that the approach proposed in this paper is based on industry standards and publicly available tools can be considered as important advantages that may contribute to the uptake of data-ware process modelling approaches. An in-depth evaluation of the approach for its usability and effectiveness could be based on the Method Evaluation Model [73], possibly enriched with dimensions of other models on technology acceptance to e.g. evaluate the impact of facilitating conditions, habit, price, etc [74]. Preliminary insights were already gained from the method's use so far. Teaching the MERODE + BPMN approach to students for more than 10 years has provided insights into what makes the approach easy to use or not, as a result of which teaching material and the approach itself have been systematically clarified. The approach has also been taught to Enterprise Architects for more than 5 years, thus allowing for an evaluation by approximately 100 professionals. The course evaluation questions gauging for the relevance, utility and

[5] http://merode.econ.kuleuven.be/Tools.html

interestingness of the approach systematically obtain scores above 6 out of 7. In their (informal) comments, the 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 [63], but could ideally be done with less hassle. The ultimate 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. To proceed beyond prototyping and avoid the costly manual coding of forms, a full-fledged low-code or even no-code approach should be aimed for. This would be require expanding the current set of models with presentation models and functional models as in [23, 75].

Finally, process verification has not been addressed in this paper. The process algebra formalisation of MERODE provides extensive consistency checking [76], but checking the consistency of the combined state charts against a business process model needs further investigation. An initial study has been published in [43], but requires further extension to achieve support for process verification as a complement to the above-mentioned validation through integrated prototyping. The same holds for verifying the consistency of the data flows (including e.g. read activities against the domain model. An in-depth analysis of the level to which data-flow anomalies [60] are catered for by the consistency checks in the MERODE domain model remains to be done, as well as specifying modelling guidelines similar to those ensuring a correct combination of decision modelling and process modelling [2, 64].

7 Conclusion

Combining MERODE with BPMN and DMN allows addressing more perspectives than currently possible with Enterprise Modelling approaches, MDE approaches or data-aware approaches. The crux of combining model kinds resides in the operationalisation of the interrelationships between the different model kinds, both in terms of how to express this as a modeler as in terms of how to transform these models to a working application. MERODE already provides a no-code prototyping approach for artefact-centric modelling, while Camunda offers the integration of BPMN and DMN. Combining both approaches yields a process, data and decision-aware modelling approach with a demonstrated operationalisation to an executable process and decision-aware information system. 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 CMMN can provide additional pathways for future research. The addition of a presentation model and elaborating a meta-model for task/domain model mappings will allow generating forms instead of hand-coding them. 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. ISO: ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description, https://www.iso.org/standard/50508.html, last accessed 2021/12/10.

2. Hasić, F., De Smedt, J., Vanthienen, J.: Augmenting processes with decision intelligence: Principles for integrated modelling. Decis. Support Syst. 107, 1–12 (2018). https://doi.org/https://doi.org/10.1016/j.dss.2017.12.008.
3. Steinau, S., Marrella, A., Andrews, K., Leotta, F., Mecella, M., Reichert, M.: DALEC: a framework for the systematic evaluation of data-centric approaches to process management software. Softw. Syst. Model. 18, 2679–2716 (2019). https://doi.org/10.1007/s10270-018-0695-0.
4. Künzle, V., Weber, B., Reichert, M.: Object-aware business processes: Fundamental requirements and their support in existing approaches. In: Krogstie, J. (ed.) Frameworks for Developing Efficient Information Systems: Models, Theory, and Practice. pp. 1–29. IGI Global, Hershey, PA (2013). https://doi.org/http://doi:10.4018/978-1-4666-4161-7.ch001.
5. Reijers, H.A., Vanderfeesten, I., Plomp, M.G.A., Van Gorp, P., Fahland, D., van der Crommert, W.L.M., Garcia, H.D.D.: Evaluating data-centric process approaches: Does the human factor factor in? Softw. Syst. Model. 16, 649–662 (2017). https://doi.org/10.1007/s10270-015-0491-z.
6. Haarmann, S., Holfter, A., Pufahl, L., Weske, M.: Formal Framework for Checking Compliance of Data-Driven Case Management. J. Data Semant. (2021). https://doi.org/10.1007/s13740-021-00120-3.
7. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: Meersman, R. and Tari, Z. (eds.) On the Move to Meaningful Internet Systems: OTM 2008. pp. 1152–1163. Springer Berlin Heidelberg, Berlin, Heidelberg (2008).
8. Calvanese, D., Montali, M., Estañol, M., Teniente, E.: Verifiable UML Artifact-Centric Business Process Models. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 1289–1298. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2661829.2662050.
9. Künzle, V., Reichert, M.: PHILharmonicFlows: Towards a Framework for Objectaware Process Management. J. Softw. Maint. Evol. Res. Pract. 23, 205–244 (2011).
10. Calvanese, D., Montali, M., Patrizi, F., Rivkin, A.: Modeling and In-Database Management of Relational, Data-Aware Processes. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 11483 LNCS, 328–345 (2019). https://doi.org/10.1007/978-3-030-21290-2_21.
11. Artale, A., Calvanese, D., Montali, M., van der Aalst, W.M.P.: Enriching Data Models with Behavioral Constraints. Ontol. Makes Sense. 316, 257–277 (2019). https://doi.org/10.3233/978-1-61499-955-3-257.
12. Estañol, M., Sancho, M.R., Teniente, E.: Ensuring the semantic correctness of a BAUML artifact-centric BPM. Inf. Softw. Technol. 93, 147–162 (2018). https://doi.org/10.1016/j.infsof.2017.09.003.
13. Snoeck, M.: Enterprise Information Systems Engineering. (2014). https://doi.org/10.1007/978-3-319-10145-3.
14. Snoeck, M., Dedene, G.: Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types. IEEE Trans. Softw. Eng. 24, 233–251 (1998). https://doi.org/10.1109/32.677182.
15. Snoeck, M., De Smedt, J., De Weerdt, J.: Supporting Data-Aware Processes with MERODE. In: Augusto, A., Gill, A., Nurcan, S., Reinhartz-Berger, I., Schmidt, R., and Zdravkovic, J. (eds.) Enterprise, Business-Process and Information Systems Modeling.

pp. 131–146. Springer International Publishing, Cham (2021).
16. Bernaert, M., Poels, G., Snoeck, M., De Backer, M.: CHOOSE: Towards a metamodel for enterprise architecture in small and medium-sized enterprises. Inf. Syst. Front. 18, 781–818 (2016). https://doi.org/10.1007/s10796-015-9559-0.
17. Zachman, J.A.: The zachman framework for enterprise architecture. Prim. Enterp. Eng. Manuf. Zachman Int. (2003).
18. The Open Group: The TOGAF® Standard, Version 9.2, https://www.opengroup.org/togaf, last accessed 2022/02/26.
19. The Open Group: Archimate, https://www.opengroup.org/archimate-home, last accessed 2022/02/02.
20. OMG: Unified Modeling Language, https://www.omg.org/spec/UML/2.5.1/About-UML/, last accessed 2022/02/26.
21. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: Enteprise Modeling: Tackling Business Challenges with the 4EM Method. Springer-Verlag Berlin Heidelberg (2014). https://doi.org/10.1007/978-3-662-43725-4.
22. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. Softw. Syst. Model. 13, 941–962 (2014). https://doi.org/10.1007/s10270-012-0273-9.
23. Pastor, O., Molina, J.C.: Model-driven architecture in practice: A software production environment based on conceptual modeling. Springer Berlin Heidelberg (2007). https://doi.org/10.1007/978-3-540-71868-0.
24. Ruiz, J., Serral, E., Snoeck, M.: Evaluating user interface generation approaches: model-based versus model-driven development. Softw. Syst. Model. 18, 2753–2776 (2019). https://doi.org/10.1007/s10270-018-0698-x.
25. Giraldo, F.D., España, S., Giraldo, W.J., Pastor, Ó.: Evaluating the quality of a set of modelling languages used in combination: A method and a tool. Inf. Syst. 77, 48–70 (2018). https://doi.org/https://doi.org/10.1016/j.is.2018.06.002.
26. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley (2004).
27. Neumann, C.P., Lenz, R.: α− Flow: A Document-Based Approach to Inter-institutional Process Support in Healthcare BT - Business Process Management Workshops. Presented at the (2010).
28. van der Aalst, W.M.P., Stahl, C., Westergaard, M.: Strategies for Modeling Complex Processes Using Colored Petri Nets BT - Transactions on Petri Nets and Other Models of Concurrency VII. Presented at the (2013).
29. Kiss, P.J., Klimkó, G.: A Reverse Data-Centric Process Design Methodology for Public Administration Processes. In: K\Ho, A., Francesconi, E., Anderst-Kotsis, G., Tjoa, A.M., and Khalil, I. (eds.) Electronic Government and the Information Systems Perspective. pp. 85–99. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-27523-5_7.
30. Ouali, N.H., Tmar, M., Haddar, N., Tmar, M.: Models and Run-Time Systems for Data Intensive Workflow Applications. In: 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). pp. 429–436 (2017). https://doi.org/10.1109/PDCAT.2017.00075.
31. Tsoury, A., Soffer, P., Reinhartz-Berger, I.: Data Impact Analysis in Business Processes. Bus. Inf. Syst. Eng. 62, 41–60 (2020). https://doi.org/10.1007/s12599-019-00611-5.
32. Rietzke, E., Bergmann, R., Kuhn, N.: ODD-BP - an Ontology- and Data-Driven Business Process Model. CEUR Workshop Proc. 2454, 409–415 (2019).

https://doi.org/10.1007/978-3-319-77525-8_107.
33. Mertens, S., Gailly, F., Poels, G.: A Generic Framework for Flexible and Data-Aware Business Process Engines. In: Proper, H.A. and Stirna, J. (eds.) Advanced Information Systems Engineering Workshops. pp. 201–213. Springer International Publishing, Cham (2019).
34. De Giacomo, G., Oriol, X., Estañol, M., Teniente, E.: Linking Data and BPMN Processes to Achieve Executable Models BT - Advanced Information Systems Engineering. Presented at the (2017).
35. Hasić, F., Smedt, J. De, Broucke, S. Vanden, Asensio, E.S.: Decision as a Service (DaaS): A Service-Oriented Architecture Approach for Decisions in Processes. IEEE Trans. Serv. Comput. 1 (2020). https://doi.org/10.1109/TSC.2020.2965516.
36. Estañol, M., Munoz-Gama, J., Carmona, J., Teniente, E.: Conformance checking in UML artifact-centric business process models. Softw. Syst. Model. 18, 2531–2555 (2019). https://doi.org/10.1007/s10270-018-0681-6.
37. Deutsch, A., Hull, R.: Automatic Verification of Database-Centric Systems. 43, 1–13 (2014).
38. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: SMT-based verification of data-aware processes: a model-theoretic approach. Math. Struct. Comput. Sci. 30, 271–313 (2020). https://doi.org/10.1017/S0960129520000067.
39. van der Linden, D., Hadar, I., Zamansky, A.: What practitioners really want: requirements for visual notations in conceptual modeling. Softw. Syst. Model. 18, 1813–1831 (2019). https://doi.org/10.1007/s10270-018-0667-4.
40. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. 28, 75–105 (2004). https://doi.org/10.2307/25148625.
41. Dedene, G., Snoeck, M.: M.E.R.O.DE.: A Model-Driven Entity-Relationship Object-Oriented Development Method. SIGSOFT Softw. Eng. Notes. 19, 51–61 (1994). https://doi.org/10.1145/182824.182838.
42. Dedene, G., Snoeck, M.: Experience from M.E.R.O.D.E. Cases: on object-oriented model-driven approach. In: O'Callaghan, A. and Thornes, S. (eds.) Practical experiences of object technology. pp. 1–17 (1996).
43. De Backer, M., Snoeck, M., Monsieur, G., Lemahieu, W., Dedene, G.: A scenario-based verification technique to assess the compatibility of collaborative business processes. Data Knowl. Eng. 68, 531–551 (2009). https://doi.org/10.1016/j.datak.2008.12.002.
44. Sedrakyan, G., Snoeck, M., Poelmans, S.: Assessing the effectiveness of feedback enabled simulation in teaching conceptual modeling. Comput. Educ. 78, 367–382 (2014). https://doi.org/10.1016/j.compedu.2014.06.014.
45. Snoeck, M., Haesen, R., Buelens, H., De Backer, M., Monsieur, G.: Computer aided modelling exercises. Informatics Educ. (2007).
46. Sedrakyan, G., Snoeck, M., Weerdt, J. De: Computers in Human Behavior Process mining analysis of conceptual modeling behavior of novices – empirical study using JMermaid modeling and experimental logging environment. Comput. Human Behav. 41, 486–503 (2014). https://doi.org/10.1016/j.chb.2014.09.054.
47. Weerdt, D.: Process - mining enabled feedback : " tell me what I did wrong " vs . " tell me how to do it right ." (2016).
48. Sedrakyan, G., Snoeck, M.: Feedback-enabled MDA-prototyping effects on modeling knowledge. (2013). https://doi.org/10.1007/978-3-642-38484-4_29.
49. Sedrakyan, G., Poelmans, S., Snoeck, M.: Assessing the influence of feedback-inclusive rapid prototyping on understanding the semantics of parallel UML statecharts by novice

modellers. Inf. Softw. Technol. 82, (2017). https://doi.org/10.1016/j.infsof.2016.11.001.
50. Sedrakyan, G., Snoeck, M.: Effects of simulation on Novices' understanding of the concept of inheritance in conceptual modeling. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 9382, 327–336 (2015). https://doi.org/10.1007/978-3-319-25747-1_32.
51. Verbruggen, C., Snoeck, M.: Practitioners' experiences with model-driven engineering: a meta-review. Softw. Syst. Model. (2022). https://doi.org/10.1007/s10270-022-01020-1.
52. Verbruggen, C., Snoeck, M.: Exploratory Study on Students' Understanding of Multi-perspective Modelling. In: Augusto, A., Gill, A., Bork, D., Nurcan, S., Reinhartz-Berger, I., and Schmidt, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. pp. 321–335. Springer International Publishing, Cham (2022).
53. Richards, M.: Software Architecture Patterns, https://www.oreilly.com/content/software-architecture-patterns/, last accessed 2021/03/13.
54. van der Werf, J.M.E.M., Polyvyanyy, A.: The Information Systems Modeling Suite: Modeling the Interplay Between Information and Processes. In: Janicki, R., Sidorova, N., and Chatain, T. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 414–425. Springer International Publishing, Cham (2020).
55. Snoeck, M., Lemahieu, W., Goethals, F., Dedene, G., Vandenbulcke, J.: Events as atomic contracts for component integration. Data Knowl. Eng. 51, 81–107 (2004). https://doi.org/http://dx.doi.org/10.1016/j.datak.2004.03.007.
56. Frank, U.: Multi-perspective enterprise modeling: Foundational concepts, prospects and future research challenges. Softw. Syst. Model. 13, 941–962 (2014). https://doi.org/10.1007/s10270-012-0273-9.
57. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, http://www.loa.istc.cnr.it/Guizzardi/SELMAS-CR.pdf, (2005).
58. Jackson, M.: The World and the Machine. In: Proceedings of the 17th International Conference on Software Engineering. pp. 283–292. Association for Computing Machinery, New York, NY, USA (1995). https://doi.org/10.1145/225014.225041.
59. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall International (2004).
60. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Sheng, O.R.L.: Formulating the Data-Flow Perspective for Business Process Management. Inf. Syst. Res. 17, 374–391 (2006). https://doi.org/10.1287/isre.1060.0105.
61. Ruiz, J., Sedrakyan, G., Snoeck, M.: Generating user interface from conceptual, presentation and user models with JMermaid in a learning approach. ACM Int. Conf. Proceeding Ser. 07-09-Sept, 25–32 (2015). https://doi.org/10.1145/2829875.2829893.
62. Scheynen, N.: Construction of web services using the MERODE approach, (2016).
63. Mohout, I., Leyse, T.: Enriching Business Process Simulation by integration with MERODE prototype applications, (2020).
64. Hasić, F., Serral, E., Snoeck, M.: Comparing BPMN to BPMN + DMN for IoT Process Modelling: A Case-Based Inquiry. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. pp. 53–60. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3341105.3373881.
65. Amaral de Sousa, V., Burnay, C., Snoeck, M.: B-MERODE: A Model-Driven Engineering and Artifact-Centric Approach to Generate Blockchain-Based Information Systems. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-49435-3_8.

66. Marín, B., Alarcón, S., Giachetti, G., Snoeck, M.: TesCaV: An Approach for Learning Model-Based Testing and Coverage in Practice BT - Research Challenges in Information Science. Presented at the (2020).
67. PHILharmonic Flows - Process, Humans and Information Linkage for harmonic Business Flows, https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/philharmonic-flows/, last accessed 2022/10/15.
68. Chiao, C.M., Künzle, V., Reichert, M.: Integrated modeling of process-and data-centric software systems with PHILharmonicFlows. In: 2013 IEEE 1st International Workshop on Communicating Business Process and Software Models Quality, Understandability, and Maintainability (CPSM). pp. 1–10. IEEE (2013).
69. Chiao, C.M., Künzle, V., Andrews, K., Reichert, M.: A tool for supporting object-aware processes. In: 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations. pp. 410–413. IEEE (2014).
70. Steinau, S., Andrews, K., Reichert, M.: A modeling tool for PHILharmonicFlows objects and lifecycle processes. (2017).
71. Dedene, G., Snoeck, M.: Formal deadlock elimination in an object oriented conceptual schema. Data Knowl. Eng. 15, 1–30 (1995). https://doi.org/10.1016/0169-023X(94)00031-9.
72. Künzle, V., Reichert, M.: Philharmonicflows: Research and design methodology. (2012).
73. Moody, D.L.: The Method Evaluation Model : A Theoretical Model for Validating Information Systems Design Methods The Method Evaluation Model : A Theoretical Model for Validating Information Systems Design Methods. 9–12 (2003).
74. Tamilmani, K., Rana, N.P., Wamba, S.F., Dwivedi, R.: The extended Unified Theory of Acceptance and Use of Technology (UTAUT2): A systematic literature review and theory evaluation. Int. J. Inf. Manage. 57, 102269 (2021). https://doi.org/https://doi.org/10.1016/j.ijinfomgt.2020.102269.
75. Sánchez-González, L., García, F., Ruiz, F., Piattini, M.: A case study about the improvement of business process models driven by indicators. Softw. Syst. Model. (2015). https://doi.org/10.1007/s10270-015-0482-0.
76. Snoeck, M., Michiels, C., Dedene, G.: Consistency by construction: The case of MERODE. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 2814, 105–117 (2003). https://doi.org/10.1007/978-3-540-39597-3_11.

Appendix: Hiring PhD students at KU Leuven

Case Description

To hire a PhD, first a vacancy must be opened. A deadline will be specified, and until that deadline, applications can be submitted. After the deadline, each application is evaluated. After the evaluation, a decision is taken whether or not to hire a candidate and the vacancy is closed. In case no good candidate was found, in collaboration with the HR department, the supervisor can decide to republish the vacancy. Otherwise, the process ends.

Opening a vacancy is subject to verification by the HR department. The supervisor will create the vacancy in the ERP system, and provide a number of details. The supervisor will then submit the vacancy proposal to HR, where the vacancy details will be verified for the completeness and quality of the information provided as well as the availability of a sufficient budget to pay the future PhD's salary for at least 4 years. The vacancy can either be approved, upon which it will be published online, or it can be rejected, with a clarification of the requested adjustments to obtain approval. The supervisor can then edit the vacancy and resubmit it, or decide to abandon the vacancy, and cancel it.

Once published, applicants can submit their application. They can do this in several steps, creating first a draft, editing it, and finally submitting it. After submission, the application cannot be modified any more. Once the application deadline has passed, the HR department initiates the evaluation process. It first performs a sanity check on the submitted applications, by verifying a number of eligibility criteria: the applicant must have the right academic degree, the issuing university must be of sufficient quality, and the applicant must provide proofs of decent mastery of the English language, as well as supply a GMAT or GRE score of sufficient level. If the applicant is deemed eligible, three colleagues of the supervisor are asked for a review of the applicant's file. The supervisor is notified when the three reviews are in, and will then proceed to review the applicant's file. The supervisor may decide to have an interview with the candidate. This interview can be done online or in person, which will require different organization processes. Interview notes are also registered in the system. Finally, for each candidate, the hiring decision for this candidate is registered. As soon as one candidate is marked as to be hired, the vacancy moves to a 'Candidate Hired' state to avoid hiring a second person on the same vacancy (something that is not allowed by government's regulations).

People are assigned to departments. Such assignment has a certain percentage, and together these percentages for a same person amount to maximum 100%.
The supervisor is considered the “owner” of a vacancy, and cannot be modified. Next to the supervisor, a number of people can be assigned as HR administrator to the vacancy.
When assigning reviewers, independent opinions are sought for. Thus, the supervisor cannot act as reviewer.

Modelling process

Below we present the models created during the requirements analysis and engineering process. It should be noted that the models were created iteratively, requiring approximately four iterations to stabilize. For example, initial process models were not executable, and had to be refined until conforming Camunda's requirements for executable models. We illustrate the difference between

the initial process model and the final process model for the global process only. Al the other processes underwent a same type of refinement.
As we evolved from a smaller version of the model to this larger version, some models had to be expanded. In the following, the difference between the expanded example and the smaller running case used in the paper are described in “Model evolution notes”.
While analyzing a process, ingredients for the domain model were registered in a table, and the domain model was created in parallel using the Merlin modelling tool. The model was checked using the Merlin Prototyper. Once a sufficiently stable version of the domain model was reached, the refinement of the process models in Camunda and the refinement of the domain model was further performed in parallel.
Below we first provide the process models, decision models and their mapping to domain model elements, and we conclude with the domain model.
The zipped archive containing the model as Merlin file and the application generated from this model can be downloaded here.

Global Process

The initial draft model of the global process was as follows:

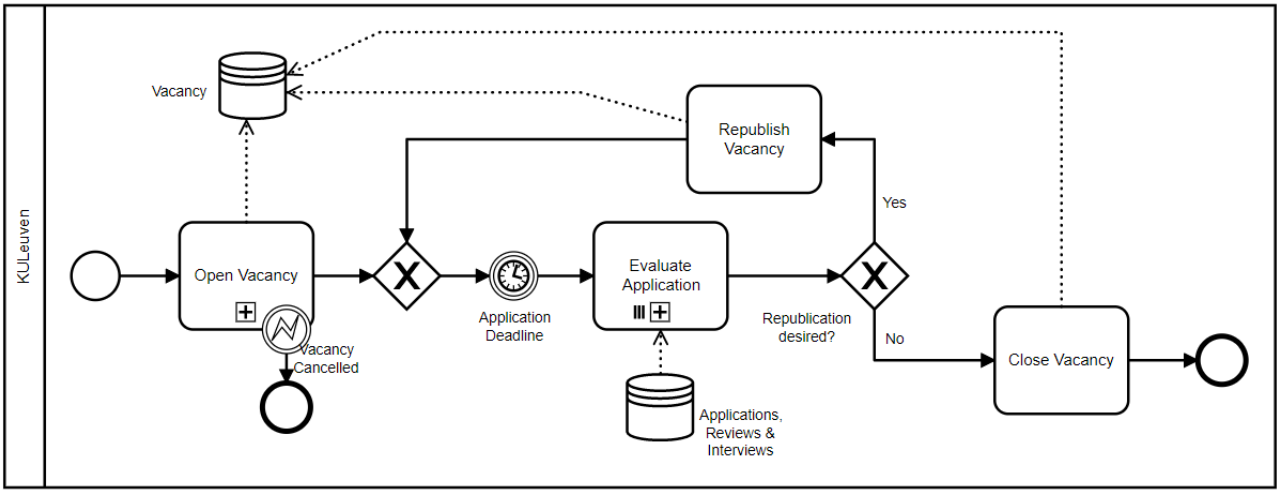

Fig. 1. Draft model for the Global Process

While this model still has several flaws (element names need to be added or improved, and the process needs to be made executable), it suffices for a first analysis of required ingredients for the domain model. A first set of domain model elements can be identified as follows.

Table 1. Process/Tasks mapped to invoked Business Event Types and required Business Object Types for the Global Process

(Sub)Process	Task	Invoked Business Event Type	Required Business Object Type
Open Vacancy			Business Object: Vacancy
Evaluate Application			Business Object: Application
Top-level Process	Close Vacancy	EVclose	Vacancy
	Republish Vacancy	EVpublish	Vacancy

The tasks 'Republish Vacancy' and 'Close Vacancy' will use an IS Service to trigger a state change in the VACANCY object using the corresponding business events *EVpublish* and *EVclose*. For now,

we assume that the generated default REST services suffice for such implementation. More complex services can be developed manually if needed.

In the above diagram, the VACANCY data store represents the 'master' data object from the data model part that contains the data relevant for this process. It was eventually decided not to include the data stores in the process models as this would lead to cluttered diagrams. In addition, data stores are not required to obtain executable processes in Camunda. On the other hand, to obtain an executable process model, 'Open Vacancy' and 'Evaluate Application' are modelled as Call Activities such that the subprocess can be invoked from the global process (required by Camunda). Furthermore, the decision to republish the vacancy was modelled as an exception to the task of closing the vacancy: in Camunda, an Input Form has a "submit" button that triggers the normal outgoing flow. Another "exit" can be modelled as an exception to this task, invoking other services. Thus, in the form for 'Close Vacancy' the "submit" button will close the vacancy, and the alternative exit will lead to the republication of the vacancy. The final model for the global process is shown in Fig. 2.

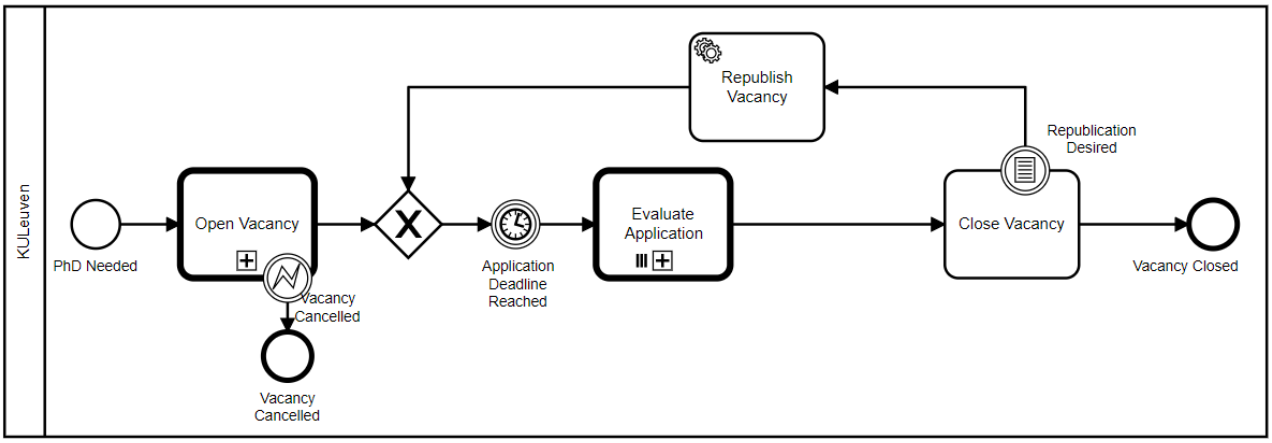

Fig. 2. Final model for the Global Process

Open Vacancy Process

Fig. 3 depicts the executable model for the Open Vacancy Process and **Table 2** defines its mapping to business object types and business event types.

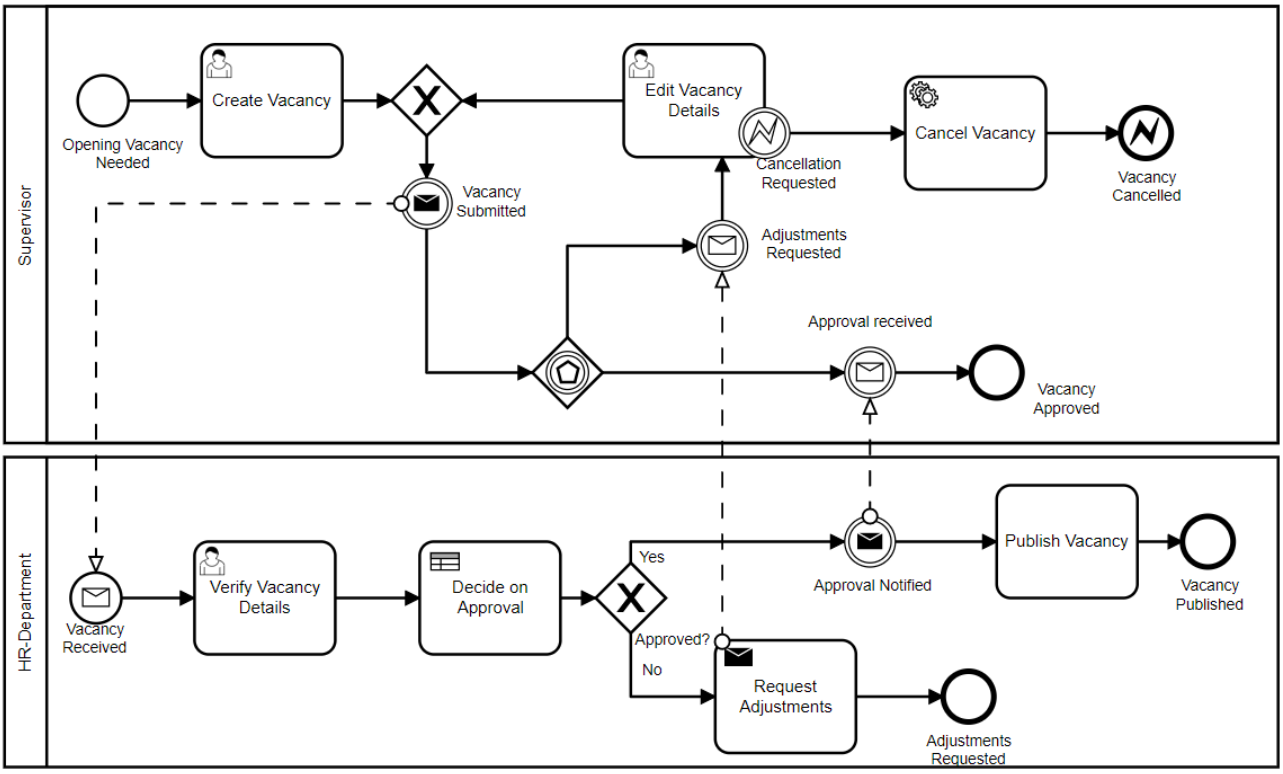

Fig. 3. Open Vacancy Process

Table 2. Process/Tasks mapped to invoked Business Event Types and required Business Object Types for the Open Vacancy Process

Process	Input Task	Invoked Business Event Type	Required Business Object Type
Open Vacancy (Supervisor)	Create Vacancy	EVcrVacancy	Vacancy
	Edit Vacancy Details	EVmodVacancy, EVcrHRAdministratorDuty, EVendHRAdministratorDuty	Vacancy
	Cancel Vacancy	EVendVacancy	Vacancy
Open Vacancy (HR Dept.)	Verify Vacancy Details	EVmodVacancy	Vacancy
	Decide on Approval	EVapprove EVdisapprove	Vacancy
	Request Adjustments	--	--
	Publish Vacancy	EVpublish	Vacancy

Model evolution note: The initial data model did not contain data about people and the departments they are member of. We added the object type HRADMINISTRATIONDUTY to indicate who has access to a vacancy in order to manage authorisations. This access can be set and modified during the 'Edit Vacancy Details' task.

The rules to decide on approval of the vacancy can be modelled by means of a Decision Table, shown in Fig. 4. Camunda assumes that BR tasks are executed automatically, but in this case, some of the rules need a human verification. We therefore decided -for the sake if the illustration- to precede the automated decision task with a human verification task where a human actor sets the values of all decision parameters, possibly using the decision table as a guide. By setting all the parameters (e.g. a Boolean Complete), the BR task can then automatically take the decision according to the following decision table:

Approve Vacancy | **Hit Policy:** Unique

	When Input string	And Input string	And Input string	Then Output string	Annotations
1	Available Budget < RequiredBudget	-	-	Disapprove	Invoke the EV_disapprove event
2	Available Budget >= RequiredBudget	Details = Incomplete	-	Disapprove	Invoke the EV_disapprove event
3	Available Budget >= RequiredBudget	Details = Complete	DescriptionQuality = Not OK	Disapprove	Invoke the EV_disapprove event
4	Available Budget >= RequiredBudget	Details = Complete	DescriptionQuality = OK	Approve	Invoke the EV_approve event if all vacancy details have been provided both in Dutch and in English and proof of sufficient budget is available.
+	-	-	-		

Fig. 4. Decision Table for approving a vacancy

The DT's output corresponds to the invocation of the corresponding business events (*EVapprove, EVdisapprove*) via an information system service as indicated in the table below. All required input data can be mapped to attributes of the VACANCY business object.

Evaluate Application

Fig. 5 depicts the executable model for the Open Vacancy Process and Table 3 defines its mapping to business object types and business event types. Here we included the data objects to illustrate how.

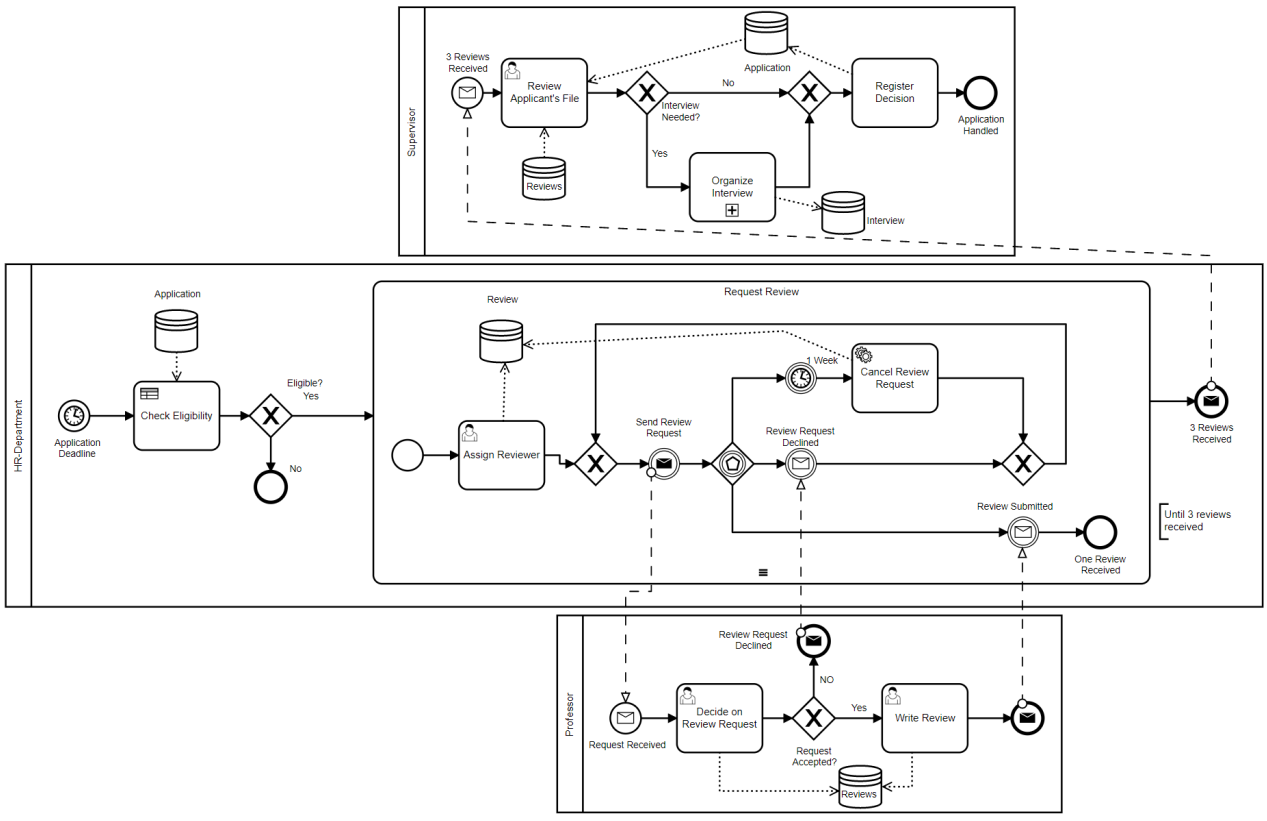

Fig. 5. Evaluate Application Process

Model evolution note: In the expanded example, review requests are also stored in the database. This mostly impacted this process: the 'Assign Reviewer' task will create the review request, the service task 'Cancel Review Request' was added to register the cancellation, and the 'Decide on Review Request' will trigger the acceptance or refusal of the review request.
Furthermore, the organization of the interview is expanded to a complex process, distinguishing between an online interview and an in-person interview.

Table 3. Process/Tasks mapped to invoked Business Event Types and required Business Object Types for the Evaluate Application Process

Process / DT	Task	Invoked Business Event	Information Needs
Evaluate Application (HR)	Check Eligibility	EVsetEligible EVsetIneligible	Vacancy.*
	Assign Reviewer	EVcrReview	Application, Person
	CancelReviewRequest	EVcancelRequest	Review
Evaluate Application (Professor)	Decide on Review Request	EVacceptToReview EVrefuseToReview	Review
	Write Review	EVcrReview EVmodReview EVsubmitReview	Vacancy: details, description Application detailsReview
Evaluate Application (Supervisor)	Review Applicant File		Application, Reviews
	Interview Candidate	EVcrInterview	Interview
	Register Decision	EVdecideToHire, EVdecideNotToHire	Application

The 'Write Review' and 'Review Applicant File' task will use output services to consult the data about the applicant. 'Interview Candidate' is a subprocess to set up and conduct the interview. It is not further specified, but will result in creating the INTERVIEW object with basic data such as date of the interview, location and notes.

'Check Eligibility' is a Business Rules task. It is associated to the DMN model consisting of a DRD depicted in Fig. 6 and four Decision Tables (Fig. 7, Fig. 8, Fig. 9, Fig. 10).
Decisions are taken using attributes of the APPLICATION object. The outcome of the top-level decision is recorded by triggering either the *EVsetEligible* or the *EVsetIneligible* business event according to the results.

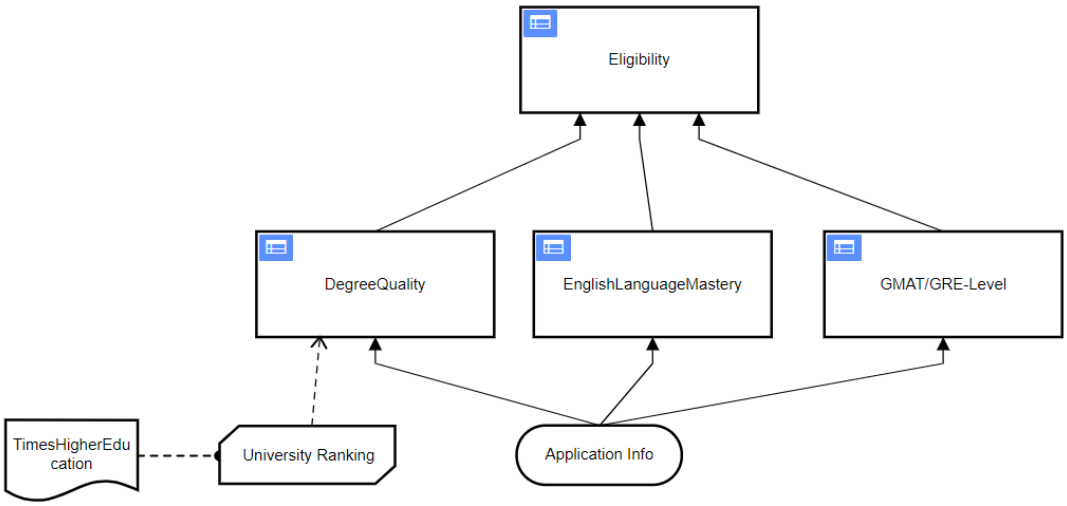

Fig. 6. DRD for deciding on the eligibility of an applicant

Eligibility | Hit Policy: Unique

	When	And	And	Then
	DegreeQuality	EnglishLangua…	GMAT/GRE-Level	Output
	"Insuffient","Cum Laude"	"Sufficient","Insufficient"…	"Sufficient","Insufficient"…	"Eligible","Ineligible"
1	"Insufficient"	-	-	"Ineligible"
2	"Cum Laude"	"Insufficient", "Missing"	-	"Ineligible"
3	"Cum Laude"	"Sufficient"	"Insufficient", "Missing"	"Ineligible"
4	"Cum Laude"	"Sufficient"	"Sufficient"	"Eligible"
5	-	-	-	
+	-	-	-	

Fig. 7. Decision Table for deciding on the eligibility of an applicant

EnglishLanguageMastery | Hit Policy: Unique

	When	And	And	And	And	Then
	Degree	TOEFL SCORE pape…	TOEFL Score Interne…	TOEFL Computer-Ba…	IETLS Score	Eligibility
	"Flemish","English-Spok…	"missing","<575",">=575"	"missing","<90",">=90"	"missing","<233",">=233"	"missing","<7",">=7"	"Sufficient","Missing","I…
1	"Flemish","English-Spoken"	-	-	-	-	"Sufficient"
2	"Other"	">=575"	-	-	-	"Sufficient"
3	"Other"	"missing","<575"	">=90"	-	-	"Sufficient"
4	"Other"	"missing","<575"	"missing","<90"	">=233"	-	"Sufficient"
5	"Other"	"missing","<575"	"missing","<90"	"missing","<233"	">=7"	"Sufficient"
6	"Other"	"missing","<575"	"missing","<90"	"missing","<233"	"missing","<7"	"Insufficient"
+	-	-	-	-	-	

Fig. 8. Decision Table for deciding on the applicant's mastery of the English language

DegreeQuality | Hit Policy: Unique

	When	And	And	And	Then
	Level	Country Of Origin	University Rank…	Average%	DegreeQuality
	"Academic Master","Aca…	"Belgium","Other"	"Excellent","Good","Low"	integer	"Cum Laude","Insufficie…
1	"Academic Bachelor","non-Academic"	-	-	-	"Insufficient"
2	"Academic Master"	"Belgium"	-	>= 68	"Cum Laude"
3	"Academic Master"	"Belgium"	-	< 68	"Insufficient"
4	"Academic Master"	"Other"	"Low"	-	"Insufficient"
5	"Academic Master"	"Other"	"Good"	< 76	"Insufficient"
6	"Academic Master"	"Other"	"Good"	>= 76	"Cum Laude"
7	"Academic Master"	"Other"	"Excellent"	< 70	"Insufficient"
8	"Academic Master"	"Other"	"Excellent"	>= 70	"Cum Laude"
+	-	-	-	-	

Fig. 9. Decision Table for deciding on the quality of the applicant's degree

GMAT/GRE-Level | Hit Policy: First

	When	And	And	And	Then
	GRE Quantitative (integer)	GRE Verbal (integer)	GMAT Quantitative (integer)	GMAT Overall (string)	GMAT/GRE-level ("Sufficient","Insufficient","Missing")
1	>175	>= 160	-	-	"Sufficient"
2	<=175	-	missing	missing	"Insufficient"
3	-	<160	missing	missing	"Insufficient"
4	-	-	>50	>=650	"Sufficient"
5	missing	missing	<=50	-	"Insufficient"
6	missing	missing	-	<650	"Insufficient"
7	missing	missing	missing	missing	"Missing"

Fig. 10. Decision Table for deciding on the sufficient level of the applicant's GRE or GMAT score

Organize Interview

Fig. 11. Shows the diagram for the Organize Interview Process Process and **Table 4** defines its mapping to business object types and business event types.

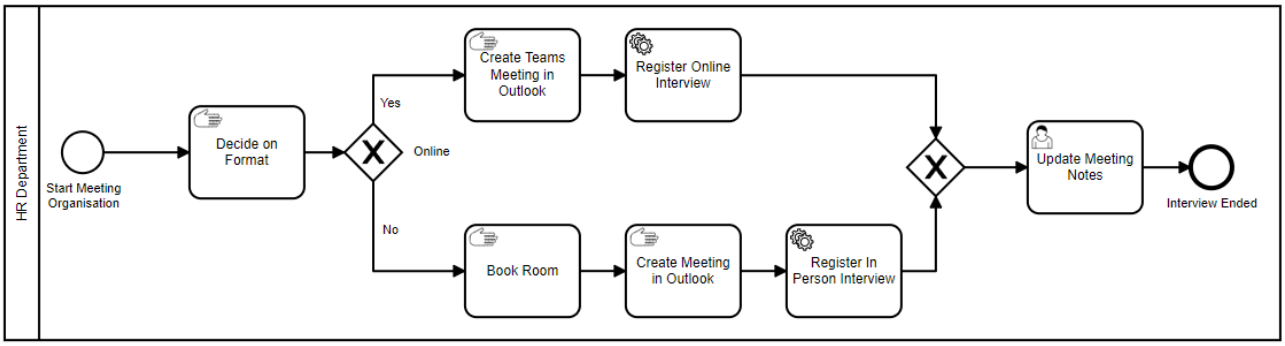

Fig. 11. Organize Interview Process

Table 4. Process/Tasks mapped to invoked Business Event Types and required Business Object Types for the Organize Interview Process

Process / DT	Task	Invoked Business Event	Information Needs
Organize Interview	Decide on Format		
	Create Teams Meeting in Outlook		
	Register Online Interview	EVcrOnlineInterview	Application
	Book Room		
	Create Meeting in Outlook		
	Register in Person Interview	EVcrInPersonInterview	Application
	Update Meeting Notes	EVupdateInterview	Interview

Data −Domain model

In order to ensure integration from a data perspective, classes and attributes need to be defined as needed per process. **Fig. 12** represents the class diagram.

Model evolution note: In the expanded version, the object types DEPARTMENT, PERSON, DEPARTMENTMEMBERSHIP and HRADMINISTRATORDUTY have been added to cater for authorisations. Moreover, the INTERVIEW has been further specialized into ONLINEINTERVIEW and INPERSONINTERVIEW to demonstrate the use of inheritance.

In this diagram, whenever a person is linked to an object (e.g. as owner of a vacancy or as reviewer), this is done by choosing a person's role as member of a department, rather than linking to a person directly. This facilitates checking authorizations. For example, in this way, the model enforces automatically that one can only take a HR Admininistrator Duty if one is member of the HR department (an HRAdministratorDuty can only be created for a (living) DepartmentMembership object).

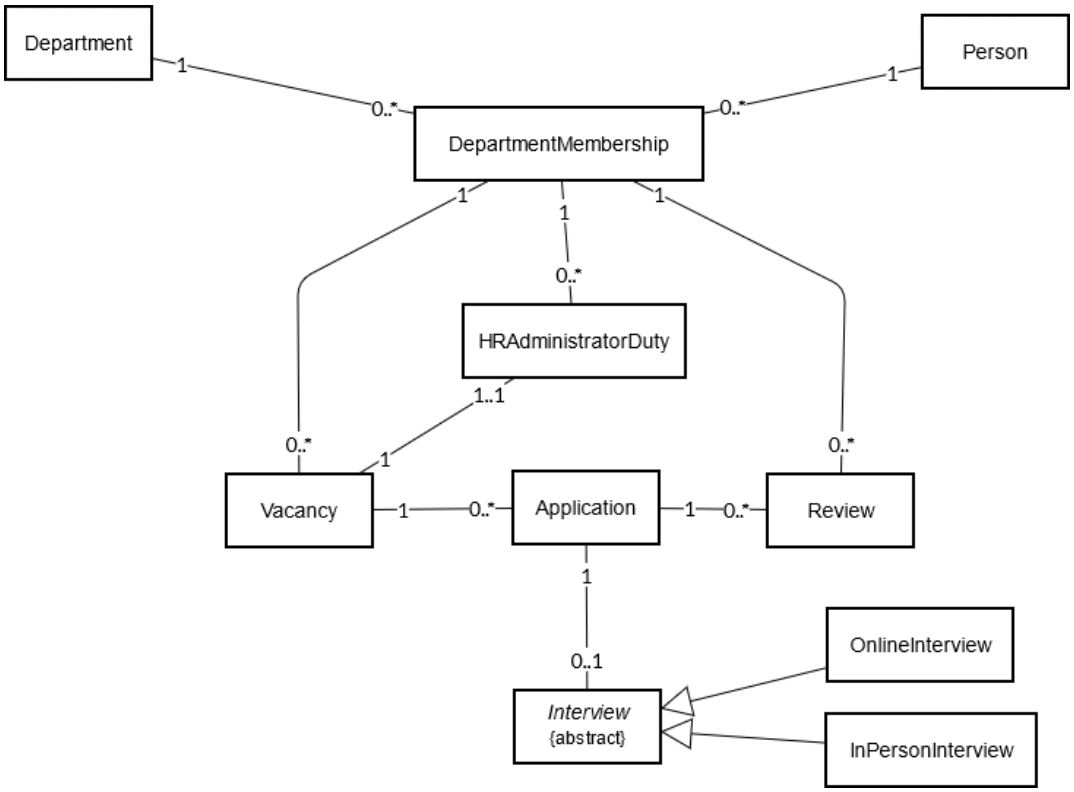

Fig. 12. Domain model (EDG)

MERLIN does not visualize attributes in the class diagram. They can however be inspected, added, modified, and deleted using the object type inspector (see Fig. 13). If no attributes have been defined, the code generator will add per default a "name" attribute to make simulation of the model possible. In addition, unique IDs and attributes that implement associations and states are generated as well. These need not to be defined manually.

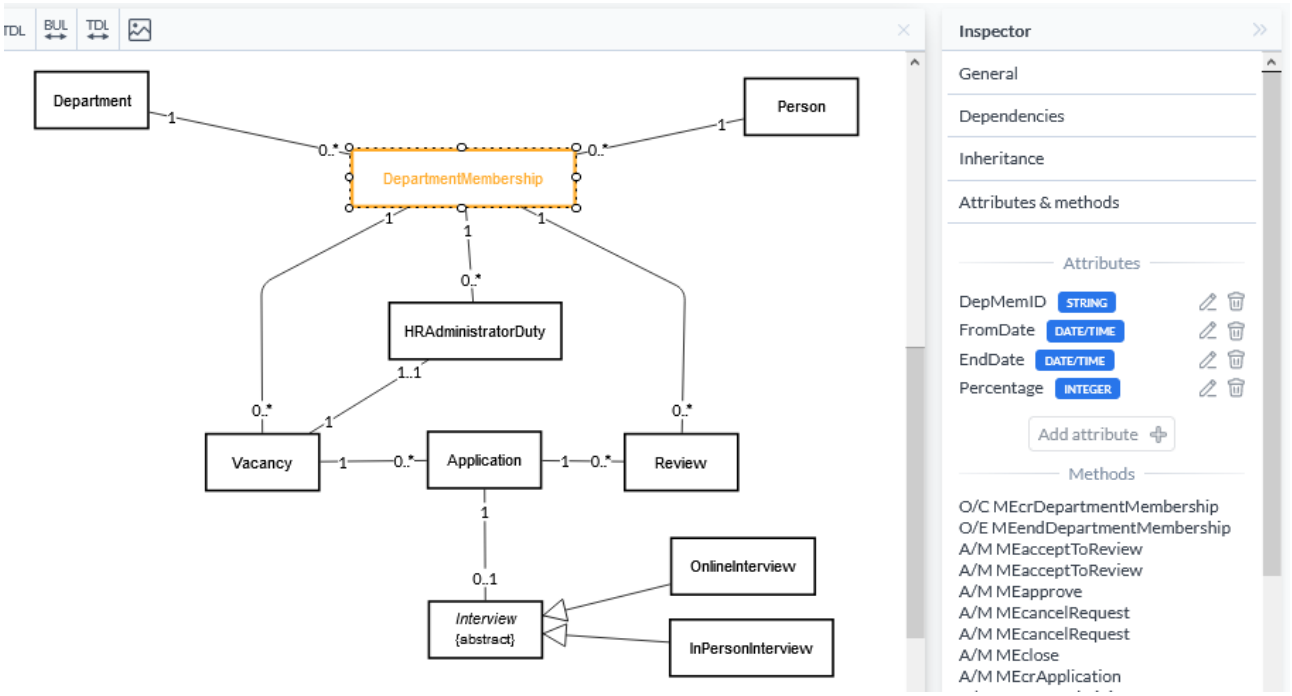

Fig. 13. Inspecting the attributes of DEPARTMENTMEMBERSHIP.

Behaviour − OET

The object-event table in **Fig. 14** lists all business events identified during the business process analysis, and maps them to the business object types:

	Department	Person	DepartmentMembership	Vacancy	Application	HRAdministratorDuty	Review	Interview	InPersonInterview	OnlineInterview
EVcrDepartment	O/C									
EVendDepartment	O/E									
EVcrPerson		O/C								
EVendPerson		O/E								
EVcrDepartmentMembership	A/M	A/M	O/C							
EVendDepartmentMembership	A/M	A/M	O/E							
EVcrVacancy	A/M	A/M	A/M	O/C						
EVapprove	A/M	A/M	A/M	O/M						
EVdisapprove	A/M	A/M	A/M	O/M						
EVmodVacancy	A/M	A/M	A/M	O/M						
EVpublish	A/M	A/M	A/M	O/M						
EVsubmitVacancy	A/M	A/M	A/M	O/M						
EVclose	A/M	A/M	A/M	O/E						
EVendVacancy	A/M	A/M	A/M	O/E						
EVcrApplication	A/M	A/M	A/M	A/M	O/C					
EVdecideNotToHire	A/M	A/M	A/M	A/M	O/M					
EVdecideToHire	A/M	A/M	A/M	A/M	O/M					
EVmodApplication	A/M	A/M	A/M	A/M	O/M					
EVsetEligible	A/M	A/M	A/M	A/M	O/M					
EVsubmitApplication	A/M	A/M	A/M	A/M	O/M					
EVendApplication	A/M	A/M	A/M	A/M	O/E					
EVsetIneligible	A/M	A/M	A/M	A/M	O/E					
EVcrHRAdministratorDuty	A/M A/M	A/M A/M	A/M A/M	A/M		O/C				
EVendHRAdministratorDuty	A/M A/M	A/M A/M	A/M A/M	A/M		O/E				
EVrequestReview	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/C			
EVacceptToReview	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/M			
EVsubmitReview	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/M			
EVupdateMotivation	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/M			
EVcancelRequest	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/E			
EVendReview	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/E			
EVrefuseToReview	A/M A/M	A/M A/M	A/M A/M	A/M	A/M		O/E			
EVcrInterview	A/M	A/M	A/M	A/M	A/M			O/C	I/C	I/C
→ *EVcrInPersonInterview*	A/M	A/M	A/M	A/M	A/M				S/C	
→ *EVcrOnlineInterview*	A/M	A/M	A/M	A/M	A/M					S/C
EVendInterview	A/M	A/M	A/M	A/M	A/M			O/E	I/E	I/E
EVendInPersonInterview	A/M	A/M	A/M	A/M	A/M				O/E	
EVendOnlineInterview	A/M	A/M	A/M	A/M	A/M					O/E
EVupdateInterview	A/M	A/M	A/M	A/M	A/M			O/M	I/M	I/M

Fig. 14. Object-Event Table

Behaviour − Lifecycles

The lifecycle of VACANCY (**Fig. 15**) reflects the states according to the global process. The states 'exists', 'submitted', 'approved' and 'published' will be passed during the execution of the subprocess 'Open Vacancy'. The subprocess 'Evaluate Applications' starts while a vacancy is in the state 'published', and may result in a transition to the state 'CandidateHired'. This demonstrates how an object lifecycle unifies object behaviour across the collection of BPs.

An additional intermediate state could be considered to ensure that applicants cannot submit a new application past the application deadline. However, this would make the system quite inflexible. For ensuring flexibility to receive applications even beyond the deadline, it is advised to manage such rules at the level of the business process layer.

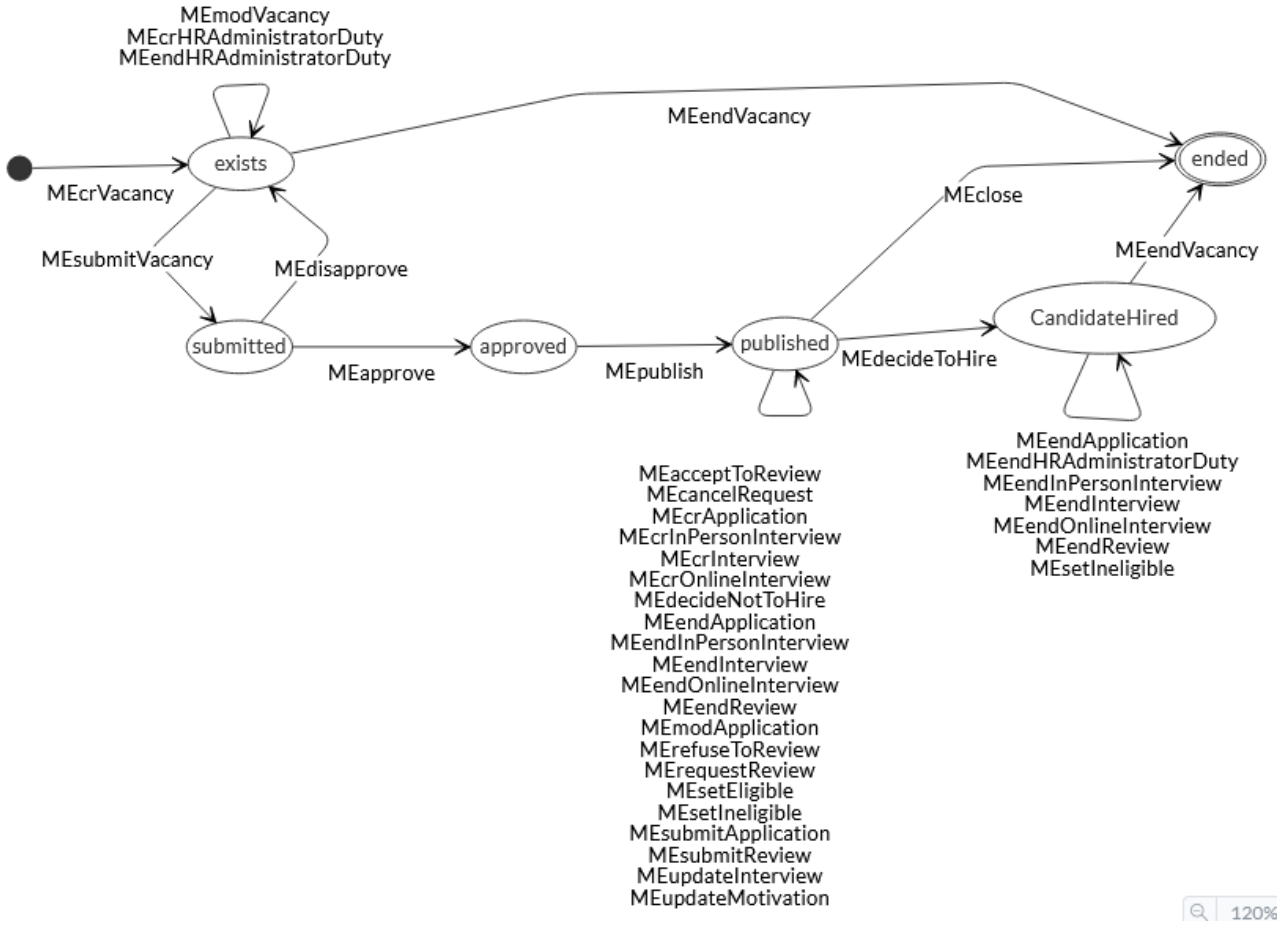

Fig. 15. State Chart representing the lifecycle of the business object type VACANCY

The lifecycle of an APPLICATION (**Fig. 16**) starts when a candidate submits a draft application. After submission, deciding on eligibility will route the application either to a final state or to the 'eligible' state where the review process can take place. Eventually the decision whether or not to hire the candidate will lead to a corresponding state. The lifecycle of the review object type (**Fig. 17**) has three intermediate states allowing to distinguish a requested, accepted, and submitted review. Three different end states allow distinguishing a cancelled request (due to non-response of the reviewer), a refused request, and one that has been accepted and submitted. All other object types have a default lifecycle.

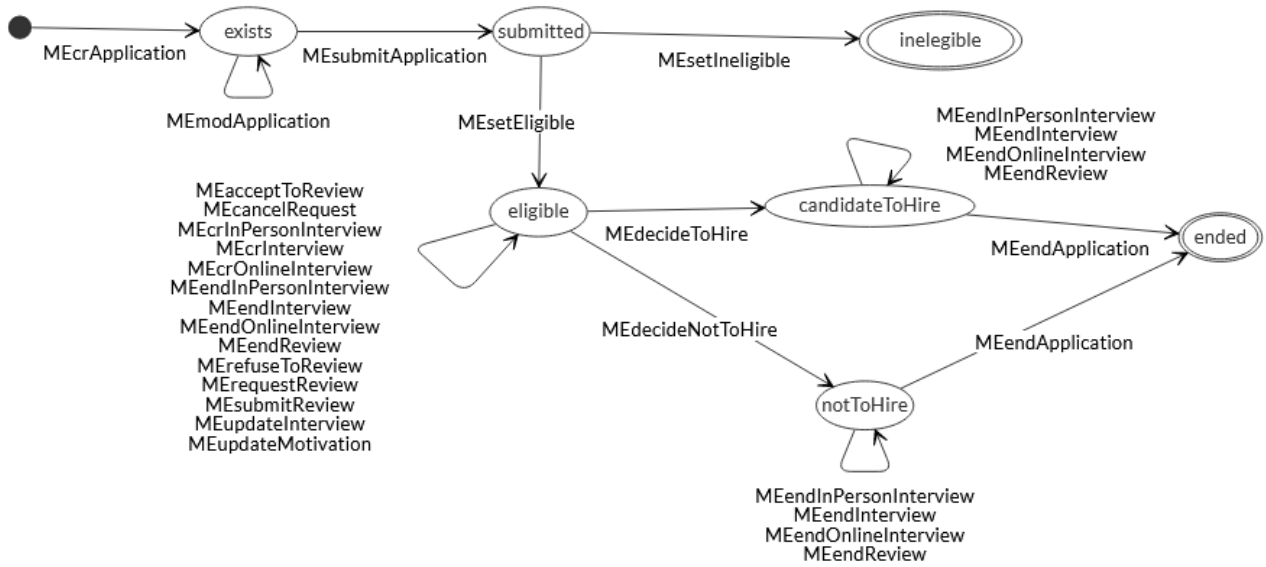

Fig. 16. State Chart representing the lifecycle of the business object type APPLICATION

Model evolution note: the lifecycle of the REVIEW was adjusted to create the review from the moment of the request rather than when the review is accepted. This leads to the extra states 'waitingForDecision', 'Cancelled' and 'Refused'.

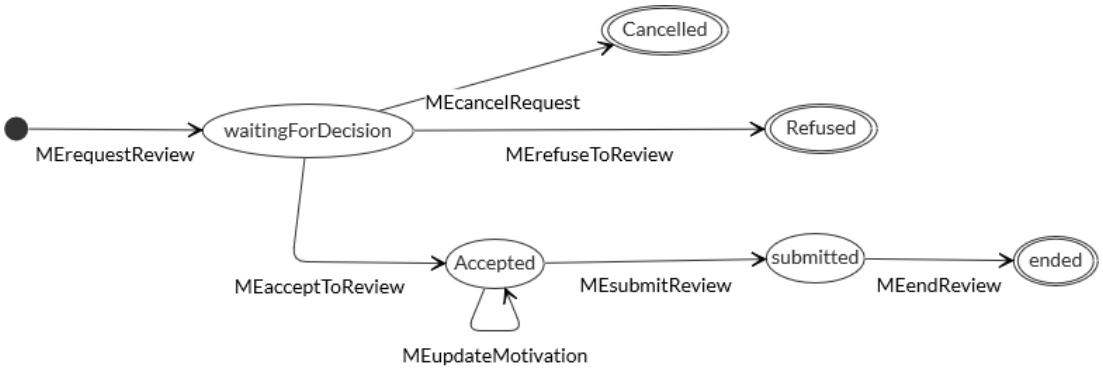

Fig. 17. State Chart representing the lifecycle of the business object type REVIEW

Constraints

Model evolution note: in the expanded version, we included a constraint to illustrate how this can be used.

The tool allows capturing path constraints for which code can be generated. As an example, we add a constraint that the person who is requested to write a review, should be different than the person who owns the vacancy (see **Fig. 18**).
Other constraints, such as the requirement that for a same person, all percentages of ongoing DepartmentMemberships should not surpass 100%, can be documented in the General Tab of the Inspector of an object type (see Fig. 19). Such constraints can be formulated e.g. in OCL, but as they are not (yet) considered by the code generator, any other format will do.

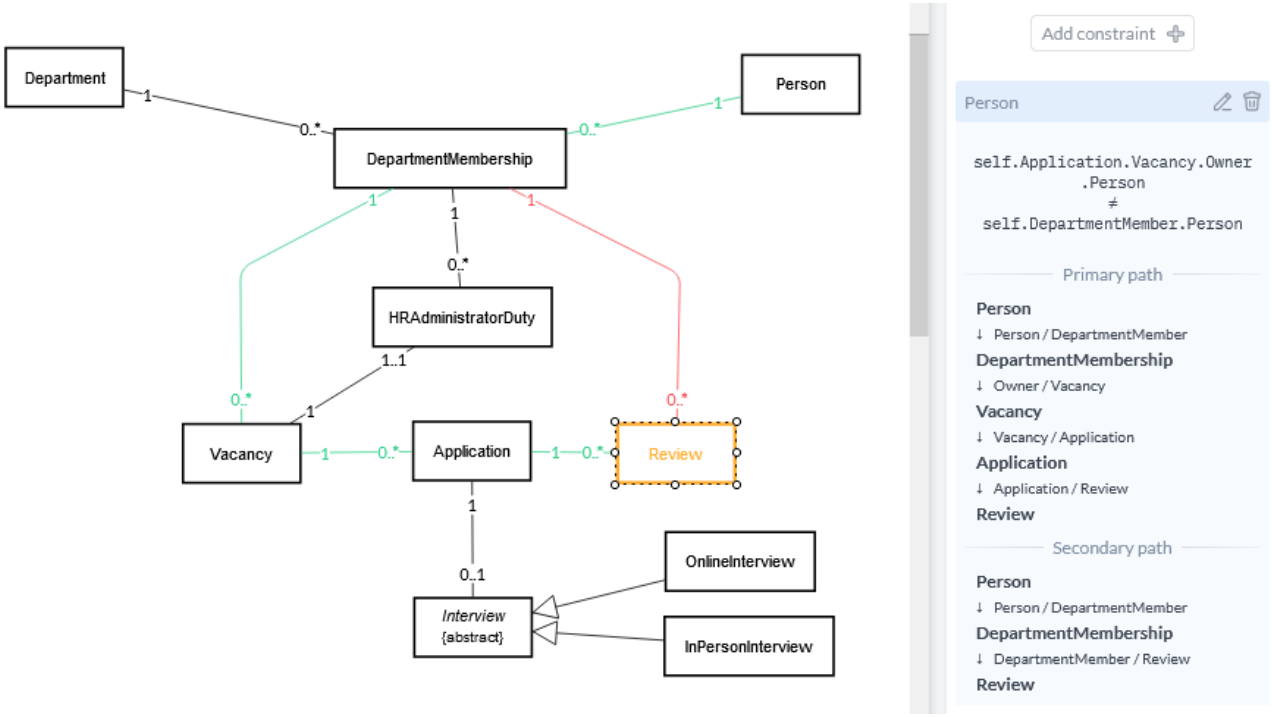

Fig. 18. Path constraint on a review, demanding that the person who is requested to do the review (red path), is different that the person who is owner of the vacancy (green path).

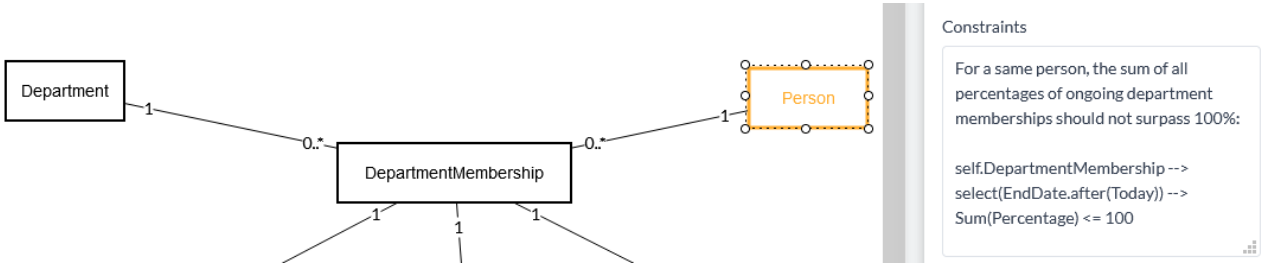

Fig. 19. Documenting additional constraints via the General Tab in the Inspector of an object type.