



Seamless conceptual modeling of processes with transactional and analytical data

Carlo Combi^a, Barbara Oliboni^a, Mathias Weske^b, Francesca Zerbato^{c,*}

^a Department of Computer Science, University of Verona, Italy

^b Hasso Plattner Institute, University of Potsdam, Germany

^c Institute of Computer Science, University of St.Gallen, Switzerland

ARTICLE INFO

Keywords:

Conceptual modeling
Business process modeling
BPMN
Data modeling
Data warehouse
Decision support

ABSTRACT

In the field of Business Process Management (BPM), modeling business processes and related data is a critical issue since process activities need to manage data stored in databases. The connection between processes and data is usually handled at the implementation level, even if modeling both processes and data at the conceptual level should help designers in improving business process models and identifying requirements for implementation. Especially in data- and decision-intensive contexts, business process activities need to access data stored both in databases and data warehouses. In this paper, we complete our approach for defining a novel conceptual view that bridges process activities and data. The proposed approach allows the designer to model the connection between business processes and database models and define the operations to perform, providing interesting insights on the overall connected perspective and hints for identifying activities that are crucial for decision support.

1. Introduction

In this period, during which we are experiencing a global pandemic, it is becoming more and more evident that many organizational and socially-relevant processes rely on sound data, which need to be suitably collected and managed. Such data are the basis for decision-making activities that would have significant consequences in the considered organization/community. Even though processes and data are known to be “two sides of the same coin” [1], the conventional techniques for the conceptual modeling of such “two sides” are usually distinct and have been studied by the Business Process Management (BPM) and database communities often from different points of views and with different approaches.

For some years, these two research communities have considered from different perspectives the key and intertwined roles of data and business process design, implementation, and execution [2,3]. Both the database point of view, focusing on persistent data supporting different business processes, and the BPM point of view, dealing with *data-aware* process modeling, have been considered in the literature, according to either the data-centric modeling paradigms [4,5] or the activity-centric one [6–9].

In this paper, we propose an approach that allows the designer to model business processes and related data in a seamless way, bridging the gap between process modeling languages and conceptual data models.

Process modeling languages and approaches traditionally focus on modeling the control flow of a process by emphasizing the role of activities and their dynamic behavior. Various approaches have been proposed both to support software/business process development – we mention here the well-established Business Process Model and Notation (BPMN) standard [10], UML

* Corresponding author.

E-mail addresses: carlo.combi@univr.it (C. Combi), barbara.oliboni@univr.it (B. Oliboni), mathias.weske@hpi.de (M. Weske), francesca.zerbato@unisg.ch (F. Zerbato).

<https://doi.org/10.1016/j.datak.2021.101895>

Received 12 February 2021; Accepted 10 May 2021

Available online 29 May 2021

0169-023X/© 2021 Elsevier B.V. All rights reserved.

activity diagrams [11,12], the Workflow Management Initiative [13,14] – and to provide some theoretical foundation – it is worth mentioning here formal approaches based on Petri nets and their extensions [6,15–17] and on constraint networks [18–20].

As for the conceptual data modeling, the database community widely acknowledges that it is the first phase of the whole database design, followed by the logical design, and by the physical one [21–23]. At each design phase, different data models may be adopted, allowing the suitable specification of the required details. At the highest level, conceptual data models are used to create conceptual schemata that represent at a high level of abstraction how the information of interest for a specific domain is structured and what is the semantics associated with such information. In the conceptual design phase, diagram-based models are often used, such as the Entity-Relationship (E-R) model and Object-Oriented data models [11,22,24]. A further, widely acknowledged distinction exists between transactional (operational) databases and analytical (decision-support) data warehouses. While transactional databases are related to operational tasks involving frequent and fast read and write operations, data warehouses support decision-based tasks where there is the need for complex read-only analyses of huge amounts of aggregate data [25].

The business process life-cycle lacks such a sound “vertical” distinction between the conceptual, logical, and physical design and the related models. For example, BPMN process models may represent the business and organizational domain at different abstraction levels: from high business process level to low operation level description of process activities. According to this scenario, it is quite common that the cross-relationships between processes and data are managed at the end of the design process, during the implementation of the overall system (or of their single parts). Thereby, there is a risk of discovering in this late phase possible conceptual gaps or mismatches between the representation of a process and the related data [2,7].

As a first step to bridge this conceptual gap, in this paper we provide a unified view of conceptual process and data models. We will discuss how to connect activities to data during the conceptual design by introducing and exemplifying the concept of *Activity View*, proposed in [26], both through a formal representation and some graphical notation. In detail, we will use BPMN process models to represent processes at the conceptual level (thus, disregarding possible implementation details) and UML class diagrams to model the conceptual database schema (even in this case, not representing implementation details), as they are both standards widely used in real organizational domains.

The main novelty of this paper is the focus on the conceptual modeling of processes using both transactional data and analytical data, together with their connection to different process activities realized through Activity Views, which we extend to explicitly represent some specific features of multidimensional data modeling and analysis.

The remainder of the paper is structured as follows. In Section 2, we will present a simple example to motivate our proposal and describe the considered context. In Section 3, we will briefly summarize our proposal for connecting processes and data [26], and in Section 4, we will extend it to consider also data stored in data warehouses. In Section 5, we recall and extend the conceptual insights that can be derived by using the Activity View on databases and data warehouses. In Section 6, we provide an overview of the literature related to the modeling of process and data, mainly focusing on conceptual approaches. In Section 7, we summarize our contribution and sketch possible directions for future work.

2. Motivating scenario

In this section, we introduce a (simplified) real-world process for motivating our approach, using the BPMN [10] and UML [12] standards for modeling business processes and conceptual database schemata.

Fig. 1 shows a BPMN process model related to the management of the emergency caused by the COVID-19 pandemic. This process is executed with a periodicity that depends on the overall situation, from once a day in some periods of fast diffusion of the virus to once per week, when the diffusion of the virus is, at least partially, under control. It is worth noting that, as it often happens for decision-support processes, such periodicity is not strictly predefined and may suddenly change.

Activities in business processes may need to manage information in order to be properly executed. Useful data can be stored both in databases and data warehouses. Thereby, business activities may need to read and write data in transactional databases and, in the case of decision activities, also analyze aggregate data.

The first activity of the BPMN process in Fig. 1, after the *start event* *s*, is *task* Situation assessment. To assess the situation, the regional authorities need to access both data related to (a) hospitals, divisions, and the related personnel, and (b) information describing coronavirus infection rates.

In Fig. 1, the first data source is represented by the *data store* labeled DB1, and the second one is represented by the *data store* labeled DW. The DW data store is depicted with a small cube to highlight that it is a data mart of a possibly larger data warehouse, i.e., it contains aggregate data that support complex analyses [25]. Data stores are connected to one or more activities through directed *data associations*, representing that data are either read or written.

Once the regional governments have evaluated the situation by analyzing data in the database and the data mart, different activities can be performed based on the infection trend. In the process, this decision is represented by exclusive gateway *Infection trend?* that splits the flow into three process branches. In case of a *Stable* trend, activity *Measure confirmation* is executed. In case of a *Decreasing* trend, activity *Measure mitigation* is executed. Otherwise, if the trend is *Increasing*, both activities *School and University shutdown* and *Public events cancellation* are performed in parallel, as shown by the enclosing *parallel gateways*. Finally, activity *Notification to media* is executed and *end event* *e* concludes the process.

In this scenario, we suppose to model data stored in a data mart by using the Dimensional Fact Model (DFM) [27], a graphical formalism for supporting the conceptual modeling of a data warehouse. In a data warehouse composed of different data marts, data are organized according to the multidimensional model, which is based on the concepts of fact, dimensions, and measures.

In Fig. 2, we show the DFM schema for the *Infection* fact, which is the focus of interest for the considered decision-making process. Dimensions are properties describing coordinates for analyzing the fact. In the considered example of Fig. 2, Time is one

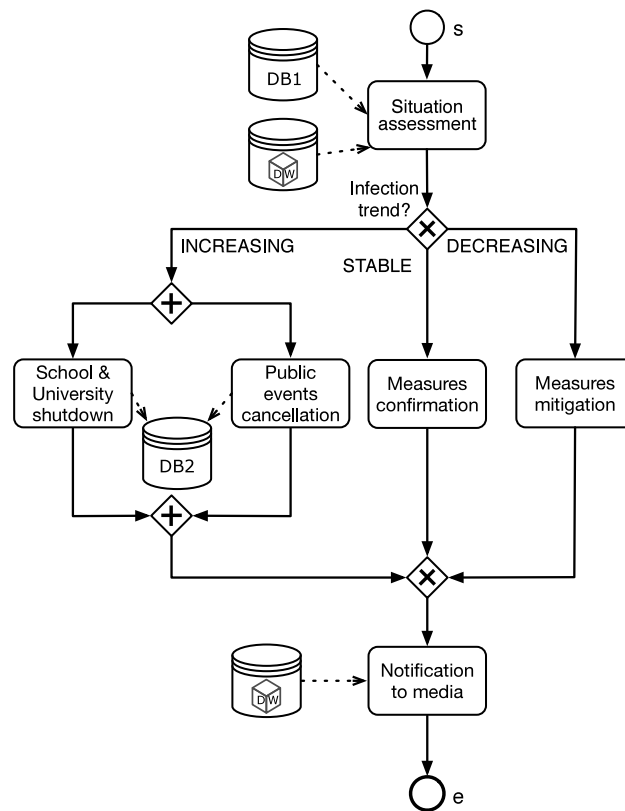


Fig. 1. A BPMN process model representing simplified governmental measures to face epidemic and pandemic events.

of the possible dimensions, having Date, Week, Day, Month, and Year as dimensional attributes. Dimensions are structured in hierarchies, which determine how facts can be aggregated. For example, Date is the root of the hierarchy tree related to the Time dimension. A hierarchy tree is a directed tree with nodes that represent dimensional attributes and arcs that model many-to-one associations between pairs of dimensional attributes. In Fig. 2, the Time dimension is included in the hierarchy rooted in Date, which represents its finest aggregation granularity, and ending with Year, which represents its coarser aggregation granularity. The fact of Fig. 2 is described through a set of numerical measures, for example Number of Infected, Number of Deaths, and Number of Recovered.

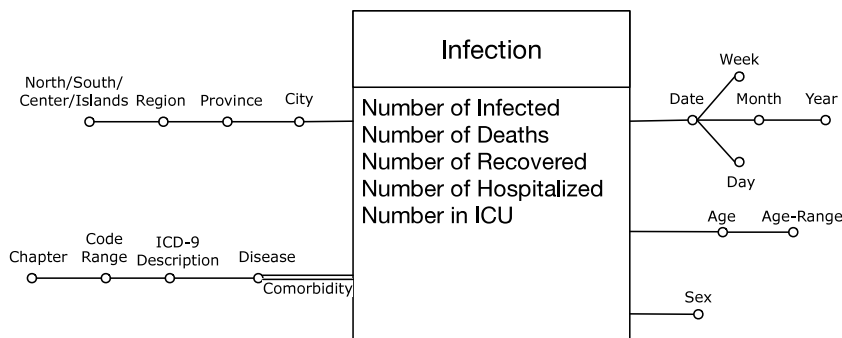


Fig. 2. Dimensional Fact Model representing the data warehouse DW containing data about infections.

In Fig. 3, we depict the considered process diagram (Fig. 3(a)) and the data schemata accessed by the process itself. In Fig. 3(b), Fig. 3(c), and Fig. 3(d) we report the UML class diagrams representing data used by the considered process. In particular, the database in Fig. 3(b) is accessed by the two activities School and University shutdown and Public events cancellation. The database in Fig. 3(c) is accessed by the first activity of the process, together with data stored in the data mart, whose UML class diagram is shown in Fig. 3(d).

This latter schema, which is related to the DFM schema of Fig. 2, consists of a fact class, represented through the stereotype «fact», and many dimension classes, represented through the stereotype «dimension». Fact classes allow one to store metrics corresponding

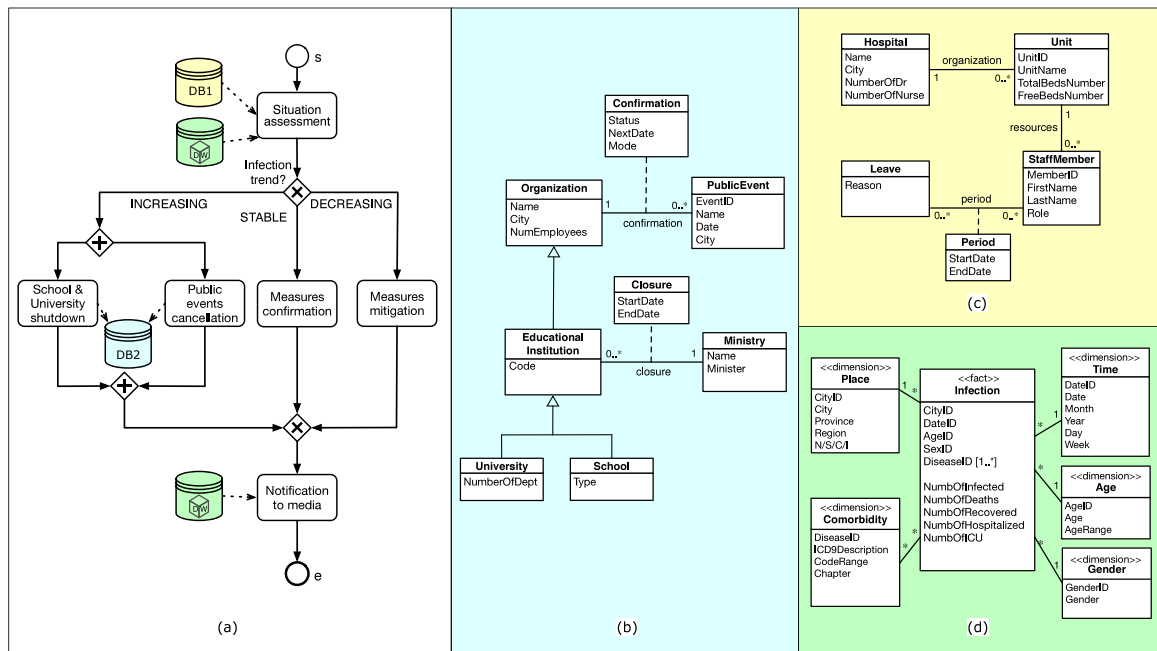


Fig. 3. (a) BPMN process for coronavirus emergency management using databases DB1 and DB2, and data warehouse DW; (b) UML class diagram representing the conceptual schema of database DB2; (c) UML class diagram representing the conceptual schema of database DB1; (d) UML schema of data warehouse DW.

to specific event measures, while dimension classes can store values for attributes describing the fact data. A fact class is thus composed of metrics and references to dimensional classes. Although such a representation resembles and is quite similar to a star schema, representing data marts in a relational setting [28], in this case, the conceptual class diagram provides a higher-level representation, as it supports, for example, a many-to-many association between the fact class and a dimension class, that would not be representable in a star schema.

The introduced motivating example illustrates the need for connecting processes and data at the conceptual level to improve the process design. Indeed, although BPMN allows the designer to specify the informational perspective of the process through data objects and data stores, it does not provide details about the internal organization of the accessed data and the operations performed by process activities on persistent data sources.

These limitations of BPMN process models raise some relevant issues that we aim to tackle in this paper.

- (i) BPMN enables the specification of database accesses through directed data associations and data stores [10]. These connections are defined at a very high level, and there is no way of specifying and detailing the considered database schema. Although directed data associations can be used to represent either the reading or writing operations performed by an activity on the database, the detailed specification of which data are accessed and which operations are performed on them is not supported. We tackled these aspects in [26] through the concept of *Activity View*.
- (ii) BPMN data stores represent generic databases without providing details about the type of database. In the process of Fig. 3(b), data accesses are directed both to databases and data marts of big data warehouses. In [26], we defined an approach to model the connection between a BPMN process model [10] and UML class diagram [12] related to a database. In this paper, we extend our approach to also capture data access operations performed on a data mart and show the conceptual insights that can be enabled on this kind of data access.
- (iii) The connection of different tasks to some parts of databases and data marts during the conceptual design allows identifying decision activities with respect to operational ones. Moreover, activities accessing both operational databases and data marts could be highlighted, and possible (hidden) dependencies between activities may be identified. In this paper, we will provide some examples of such kind of analysis that can be performed during the conceptual design.

3. Activity views for connecting processes and data

In this section, we briefly summarize the *Activity View*, a solution proposed in [26] for capturing the conceptual connection between a process model, represented in BPMN, and the conceptual schema of a database, represented as a UML class diagram.

The Activity View allows the designer to connect activities in process models and database schemata by defining data access operations at the conceptual level. In particular, the Activity View shows the portion of a conceptual database schema (i.e., the conceptual view) that is accessed by a given process activity and provides details about the operations performed on the database when the activity is executed.

Definition 1 (Activity View). Given an activity ac in a process, its Activity View av_{ac} is a set of tuples $av_{ac} = \{t_1, \dots, t_n\}$, where each tuple t_i denotes a particular data access operation performed by ac on one or more classes of a database schema. Each tuple $t_i \in av_{ac}$ is defined as

$$t_i = \langle C_{set_i}, A_{set_i}, AccessType_i, AccessTime_i, NumInstances_i \rangle$$

where:

- C_{set_i} represents the set of classes and association classes accessed by activity ac .
- A_{set_i} represents the set of reflexive or binary associations that directly link any two classes of C_{set_i} .
- $AccessType_i$ defines the type of access to the related information, i.e., read (R), insertion (I), deletion (D), or update (U) operations.
- $AccessTime_i$ denotes the moment the operation is performed with respect to activity execution, i.e., *start*, *during* or *end*, hereby defining a partial order among Activity View tuples.
- $NumInstances_i$ represents the minimum and the maximum number of objects involved in the considered operation as a pair of values (*min*, *max*). Briefly, *min* can take value '1', when the operation uses for sure at least one object of any class, or value '0' if the operation could not use any object of a class. Instead, *max* can take value '1' if the operation selects at most a single object of (at least) a class or value '*' if the operation may select many objects of any class.

Some relevant aspects to write consistent Activity Views connecting process and data need to be considered. We summarize some of them below and refer the reader to [26] for more details about the definition of the Activity View and its consistency constraints.

Classes and attributes. The C_{set} includes classes and their attributes, which constitute the *conceptual objects* of interest needed by a process activity to be executed. When defining the Activity View, designers can specify the list of attributes of a class that are involved in the data operation or use the shorthand '*' to denote that all attributes are affected by the operation. In this way, designers can specify data operations leaving out attributes that are not related to the process, for example, in cases where the data schema represents a database that has not been purposely designed for process support. Moreover, this allows for a more precise definition of update operations performed by multiple activities on a particular object but addressing different attributes [29]. Finally, when taking process roles and data access privileges into account, it is plausible that specific attributes may have restricted access (e.g., certain data are subjected to privacy) and, thus, a class may not always be accessible as a whole. Clearly, the C_{set} cannot be empty in a consistent Activity View.

Data associations and association classes. Associations are included in the A_{set} . Having the association set allows designers to describe the accessed data at a fine level of detail, especially when dealing with data schemata that include multiple associations between any two classes or reflexive associations. Indeed, if associations were not specified, it is not immediately clear how any two classes of C_{set} are connected. As an example, let us consider DB2 in Fig. 3 and imagine to draw another association RecordedMedia with the related association class between Organization and PublicEvent, representing that the organization owns the media recorded at the public event. Such material may be subjected to copyright or data protection laws and, therefore, should not be accessible to the regional government employee in charge of rescheduling the event. Similarly, let us assume to have a reflexive association manager defined on class StaffMember of DB1 capturing staff managers. To distinguish between a data operation reading all staff members, e.g.,

$$\langle \{StaffMember(*)\}, \emptyset, R, start, (1, *) \rangle$$

and one retrieving only managers, e.g.,

$$\langle \{StaffMember(*)\}, manager, R, start, (1, *) \rangle$$

it is necessary to explicitly specify whether association manager is accessed or not.

For an Activity View to be consistent, the A_{set} must observe the following constraints:

- The set of associations can be empty if and only if the set of classes is a singleton and it is not connected to reflexive associations accessed by the process;
- All the classes being at the extremes of each association included in the set of associations must be in the set of classes.
- If the set of classes includes more than one element, then at least one of associations connecting any of those classes must be in the association set. The association classes related to associations in A_{set} must be in the set of classes.

Access type. Information related to the type of access is essential from both the process and data perspectives. For example, BPMN processes make use of directed data associations to distinguish when a piece of information is read or written by a certain activity [10]. From a database perspective, read and write operations have different semantics and, thus, we decided to maintain this distinction and rely on the well-known CRUD (Create, Read, Update, and Delete) operations, but labeling object creation as an "insertion" operation. For an Activity View to be consistent, $AccessType$ cannot be empty.

Access time. By definition, the tuples of an Activity View are not ordered, as different orders of data operations may produce the same result on the underlying database, e.g., in case of reading operations. However, process activities have specific data requirements that need to be observed, such as input data that must be read for the activity to begin [8]. Therefore, we opted for adding information related to when data operations are performed during activity execution. $AccessTime$ defines a partial order among the tuples of the Activity View, which becomes useful to distinguish data input/output requirements and for process abstraction purposes [30], e.g., when combining multiple activities into a high-level one. For an Activity View to be consistent, $AccessTime$ cannot be empty.

Number of accessed objects. The number of objects involved in a data operation is captured by *NumInstances*. Being able to define how many instances of a certain data class are accessed by a process activity improves the understanding of how certain activities are executed. On the one hand, accessing a set of objects one element at a time or accessing the same set of objects with a single operation is different under a database perspective. For an Activity View to be consistent, *NumInstances* cannot be empty.

An Activity View provides a clear representation of the conceptual data schema part used by one activity as well as of the data operations performed on it. For example, let consider the Situation assessment activity of our motivating example. During Situation assessment, the regional government must retrieve data regarding the total number of free beds in each hospital, the total number of free beds in ICU, and the number of staff members currently not on leave. To this end, during the execution of task Situation assessment (SA), three different reading operations are performed on the database. These operations are represented by three distinct tuples in the Activity View av_{SA} , which is formalized as follows:

$$av_{SA} = \{ \{ \{ Hospital(*), Unit(TotalBedsNumber) \}, \{ organization \}, R, start, (1, *) \}, \\ \{ \{ Unit(UnitID, UnitName, FreeBedsNumber) \}, \emptyset, R, during, (1, *) \}, \\ \{ \{ Leave(*), StaffMember(memberID, role), Period(*) \}, \{ period \}, R, end, (1, *) \} \}.$$

The first tuple captures the fact that all the attributes of class Hospital, denoted by the * symbol, and attribute TotalBedsNumber of class Unit, which are connected through association organization are read (*R*) at the beginning (*start*) of the task. The operation may select at least one and possibly many objects of any class, as denoted by the values '1' and '*' at the end of the tuple.

For readability reasons, the tuple of the considered Activity View can be represented in a tabular form. In Table 1 we show the tabular representation of the Activity View av_{SA} .

Table 1
Tabular representation of the Activity View related to task Situation assessment.

C_{set}	A_{set}	AccessType	AccessTime	NumInstances
$\{Hospital(*), Unit(TotalBedsNumber)\}$	$\{organization\}$	<i>R</i>	<i>start</i>	(1, *)
$\{Unit(UnitID, UnitName, FreeBedsNumber)\}$	$\emptyset$	<i>R</i>	<i>during</i>	(1, *)
$\{Leave(*), StaffMember(memberID, role), Period(*)\}$	$\{period\}$	<i>R</i>	<i>end</i>	(1, *)

Similarly, let us consider task Public events cancellation (PEC), which requires the government to cancel all scheduled events from DB1 momentarily, if any. Rescheduling an event implies checking all events that have been confirmed and generating a new confirmation with an updated status, e.g., "rescheduled", a future date, and an updated planned mode, e.g., "remote". The Activity View av_{PEC} of this task, will be formalized as follows:

$$av_{PEC} = \{ \{ \{ Organization(Name), Confirmation(*), PublicEvent(*) \}, \{ confirmation \}, R, start, (1, *) \}, \\ \{ \{ Confirmation(Status, NextDate, Mode) \}, \emptyset, U, during, (0, *) \} \}.$$

The tabular representation of the Activity View av_{PEC} is reported in Table 2.

Table 2
Tabular representation for the Activity View related to task Public events cancellation.

C_{set}	A_{set}	AccessType	AccessTime	NumInstances
$\{Organization(Name), Confirmation(*), PublicEvent(*)\}$	$\{confirmation\}$	<i>R</i>	<i>start</i>	(1, *)
$\{Confirmation(Status, NextDate, Mode)\}$	$\emptyset$	<i>U</i>	<i>during</i>	(0, *)

In parallel, the regional government must take care of the School & university shutdown (SUS). To this end, all educational institutions that are still open must be closed, and, in the end, the closure date of institutes that are already closed should be updated. Hereby, the Activity View av_{SUS} can be formalized as follows:

$$av_{SUS} = \{ \{ \{ EducationalInstitution(*), Closure(*), Ministry(Name) \}, \{ closure \}, R, start, (1, *) \}, \\ \{ \{ Closure(*) \}, \emptyset, I, during, (0, *) \}, \\ \{ \{ Closure(*) \}, \emptyset, U, during, (0, *) \} \}.$$

In Table 3 we show the tabular representation of the Activity View av_{SUS} .

Table 3
Tabular representation for the Activity View related to task School & university shutdown.

C_{set}	A_{set}	AccessType	AccessTime	NumInstances
$\{EducationalInstitution(*), Closure(*), Ministry(Name)\}$	$\{closure\}$	<i>R</i>	<i>start</i>	(1, *)
$\{Closure(*)\}$	$\emptyset$	<i>I</i>	<i>during</i>	(0, *)
$\{Closure(*)\}$	$\emptyset$	<i>U</i>	<i>during</i>	(0, *)

Fig. 4 provides the graphical representation of the Activity View of task Situation Assessment, by using dashed arrows that connect the activity to different areas of the database schema corresponding to the class and association sets specified in each operation (tuple). Connecting arrows are labeled with the information related to the access type, access time, and the number of objects involved in the operation, while dashed lines of the same color frame the area of the database involved in each operation.

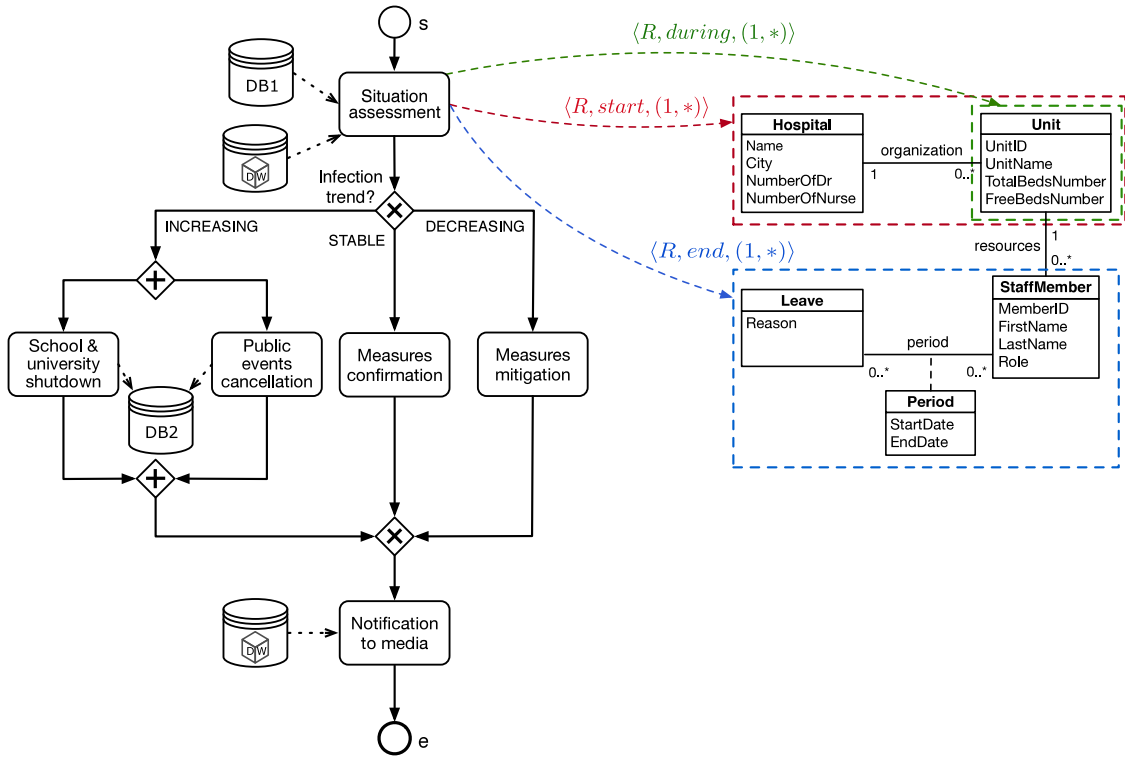


Fig. 4. Graphical representation of the data operations captured by the Activity View of task Situation assessment, connecting the process and database DB1.

4. Modeling the connection between processes and data warehouses

In this section, we start from the Activity View introduced [26], which we summarized and exemplified in the previous section, and provide a new definition for allowing also the specification of data access operations related to a data mart of a data warehouse.

To give this novel definition of Activity View, we rely on the well-known concepts of specialization and inheritance in object-oriented modeling [12]. The following definition is obtained starting from the definition we proposed in [26], and by using specialization both by restriction of values and by extending tuples with new attributes.

Definition 2 (DW — Activity View). Given an activity ac in a process model, its Activity View accessing a (data mart of a) data warehouse $av-dw_{ac} = \{t_1, \dots, t_n\}$ is a set of tuples, where each tuple t_i denotes a (reading) data access operation performed by activity ac on classes of a given data schema. The schema is composed of a set of classes Cl and a set of associations As . Each tuple t_i is defined as follows:

$$t_i = \langle C_{set_i}, A_{set_i}, AccessType_i, AccessTime_i, NumInstances_i, Op_{set_i} \rangle$$

where:

- $C_{set_i} = \{c_1, \dots, c_j\} \subseteq Cl$, is the set of connected classes accessed by process activity ac . Each class $c_j(attr_1, \dots, attr_n) \in C_{set_i}$ is characterized by a unique name c_j and a set of attributes $\{attr_1, \dots, attr_n\}$. If all the attributes of c_j are involved in the data operation, we write $c_j(*)$. Instead, if only a subset of attributes of c_j is accessed, we explicitly specify it by $c_j(attr_g, \dots, attr_m)$ with $1 \leq g < m \leq n$. When considering a data mart of a data warehouse, at least one of the classes in C_{set_i} must be a fact class within one or more classes representing dimensions. The specification of a hierarchy attribute allows the definition of the minimum considered level in the dimension hierarchy.
- $A_{set_i} = \{a_1, \dots, a_r\} \subseteq C_{set_i} \times C_{set_i} \subseteq As$ is a set of binary associations that directly link any two classes of C_{set_i} . When considering data mart associations, A_{set_i} represents connections between fact classes and dimension classes. The A_{set_i} set contains either one or more associations between classes or it is empty. For each association in A_{set_i} , both classes participating to the association must be included to C_{set_i} .
- $AccessType_i = \{R\}$ denotes the *read* operation for accessing the related information. In case of data warehouse accessing, only read operations are considered.
- $AccessTime_i \in \{start, end, during\}$ denotes when a data operation is performed with respect to activity execution, and defines a partial order among tuples in Activity View.

- $NumInstances_i = (min, max)$, where $min \in \{*, 0, 1\}$, and $max \in \{*, 1\}$, denotes the number of objects involved in the considered operation. min can take value '1', when the operation will use for sure at least one object of any class, or value '0' if the operation could not use any object of a class. Instead, max can take value '1' if the operation selects at most a single object of (at least) a class or value '*' if the operation may select many objects of any class.
- $Op_{set_i} = \{op_1, \dots, op_j\}$ where $op_i \in \{count, sum, mean, min, max\}$ is the set of allowed aggregation operators. $\diamond$

The Situation assessment activity of our motivating example accesses both database DB1 and data warehouse DW. When executing Situation assessment, the regional government must retrieve some aggregate data. At the beginning, the required information is represented by the number of infected and the number of dead people, grouped by date. During the activity, the government may need to know the number of infected, the number of dead people, and the number of patients in ICU, grouped by date and region. At the end of activity, the number of dead people grouped by date, region, and range of age are required. Tuples in $av - dw_{SA}$ represent the described operations.

$$\begin{aligned}
 av - dw_{SA} = & \{ \langle \{ Infection(NumOf Infected, NumOf Death), Time(Date) \}, \{ InfTime \}, R, \\
 & \quad start, (1, *), sum \rangle, \\
 & \langle \{ Infection(NumOf Infected, NumOf Death, NumOf ICU), Time(Date), \\
 & \quad Place(City, Region) \}, \{ InfTime, InfPlace \}, R, during, (1, *), sum \rangle, \\
 & \langle \{ Infection(NumOf Death), Time(Date), Place(City, Region), Age(AgeRange) \}, \\
 & \quad \{ InfTime, InfPlace, InfAge \}, R, end, (1, *), sum \rangle \}.
 \end{aligned}$$

The tabular form for tuples in $av - dw_{SA}$ is reported in Table 4.

Table 4

Tabular representation for the DW — Activity View related to task Situation assessment.

C_{set}	A_{set}	AccessType	AccessTime	NumInstances	Op
$\{ Infection(NumOf Infected, NumOf Death), Time(Date) \}$	$\{ InfTime \}$	R	start	(1, *)	sum
$\{ Infection(NumOf Infected, NumOf Death, NumOf ICU), Time(Date), Place(City, Region) \}$	$\{ InfTime, InfPlace \}$	R	during	(1, *)	sum
$\{ Infection(NumOf Death), Time(Date), Place(City, Region), Age(AgeRange) \}$	$\{ InfTime, InfPlace, InfAge \}$	R	end	(1, *)	sum

Fig. 5 shows the graphical representation of operations in the Activity View related to Situation assessment when the activity accesses the data mart. For representation purposes, the names of the associations in the data warehouse are not depicted, but can be read in the provided Activity Views.

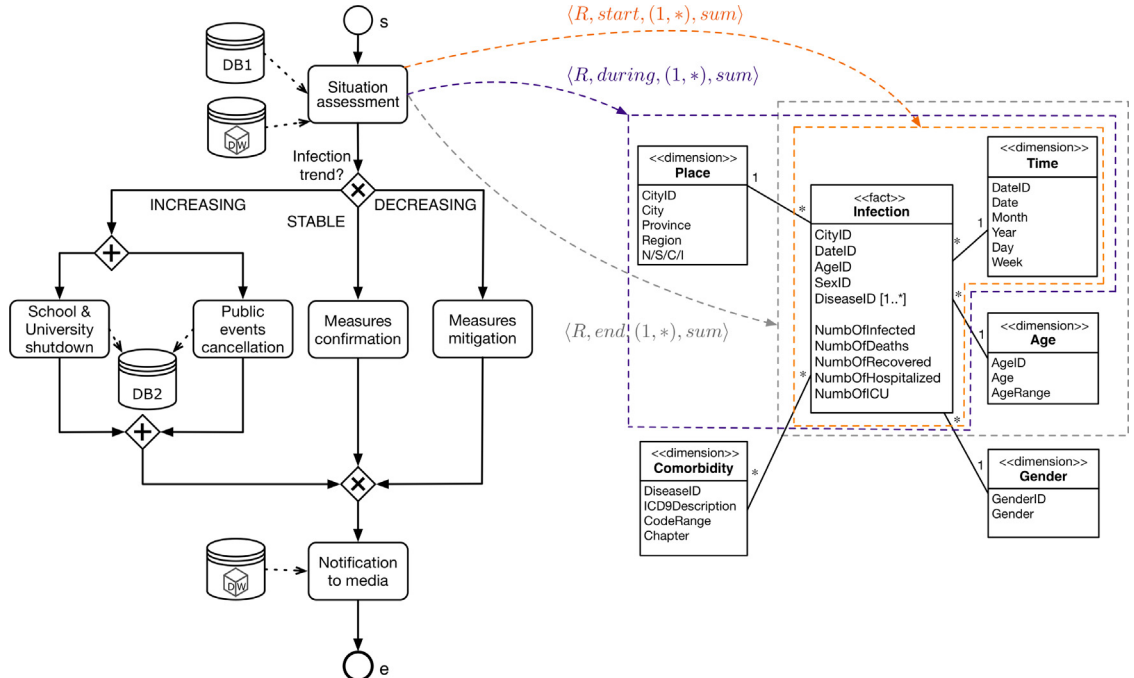


Fig. 5. Graphical representation of the data warehouse operations in the Activity View of task Situation assessment, connecting the process and database DW.

Given a process activity, tuples in its Activity View can describe operations for accessing both a database and a data mart in a data warehouse. This means that in the Activity View we can represent accessing operations to different data sources. To specify the accessed data source, we use the dot notation in the set describing accessed classes by specifying the name of the data source, followed by a dot followed by the class name.

$$\begin{aligned}
 av_{SA} = & \{ \{ DB1.Hospital(*), DB1.Unit(TotalBedsNumber) \}, \{ organization \}, R, start, (1, *) \}, \\
 & \{ DW.Infection(NumbOfInfected, NumbOfDeath), DW.Time(Date) \}, \{ InfTime \}, \\
 & R, start, (1, *), sum \}, \\
 & \{ DB1.Unit(UnitID, UnitName, FreeBedsNumber) \}, \emptyset, R, during, (1, *) \}, \\
 & \{ DW.Infection(NumbOfInfected, NumbOfDeath, NumOfICU), DW.Time(Date), \\
 & DW.Place(City, Region) \}, \{ InfTime, InfPlace \}, R, during, (1, *), sum \}, \\
 & \{ DB1.Leave(*), DB1.StaffMember(memberID, role), DB1.Period(*) \}, \{ period \}, R, end, (1, *) \}, \\
 & \{ DW.Infection(NumbOfDeath), DW.Time(Date), DW.Place(City, Region), DW.Age(AgeRange) \}, \\
 & \{ InfTime, InfPlace, InfAge \}, R, end, (1, *), sum \}.
 \end{aligned}$$

The tabular form for all the tuples in av_{SA} , i.e. describing operations accessing both database and data warehouse, is reported in Table 5.

Table 5
Tabular representation for the DW — Activity View related to task Situation assessment.

C_{set}	A_{set}	AccessType	AccessTime	NumInstances	Op
$\{ DB1.Hospital(*), DB1.Unit(TotalBedsNumber) \}$	$\{ organization \}$	R	start	(1, *)	
$\{ DW.Infection(NumbOfInfected, NumbOfDeath), DW.Time(Date) \}$	$\{ InfTime \}$	R	start	(1, *)	sum
$\{ DB1.Unit(UnitID, UnitName, FreeBedsNumber) \}$	$\emptyset$	R	during	(1, *)	
$\{ DW.Infection(NumbOfInfected, NumbOfDeath, NumOfICU), DW.Time(Date), DW.Place(City, Region) \}$	$\{ InfTime, InfPlace \}$	R	during	(1, *)	sum
$\{ DB1.Leave(*), DB1.StaffMember(memberID, role), DB1.Period(*) \}$	$\{ period \}$	R	end	(1, *)	
$\{ DW.Infection(NumbOfDeath), DW.Time(Date), DW.Place(City, Region), DW.Age(AgeRange) \}$	$\{ InfTime, InfPlace, InfAge \}$	R	end	(1, *)	sum

5. Novel conceptual insights

Other than supporting the conceptual modeling of process and related data, the Activity View provides designers with novel perspectives on the connected system, that enable basic reasoning on the interplay between process and data models, useful for both process analysis and documentation purposes. In this section, we discuss how Activity Views can be used to capture interesting aspects of the connection between processes and conceptual data schemata, considering both databases and data marts.

Identifying the portion of a data schema accessed by a process activity. As described in Sections 3 and 4, the Activity View and the DW-Activity View allow one to identify which classes and associations of a data schema are accessed by a certain activity, thus providing a more precise way of specifying the kind of data source and its content than the “black-box” data stores available in BPMN [10]. To identify the view of the data schema accessed by an activity ac_k , all the tuples $t_{1,k}, \dots, t_{n,k}$ of av_{ac_k} must be combined as follows: the comprehensive set of classes and association classes of a data schema accessed by ac_i is $\bigcup_{j=1}^n C_{set_{j,k}}$, where j denotes the tuple and k the activity, while the set of all associations accessed by ac_k is $\bigcup_{j=1}^n A_{set_{j,k}}$. In Figs. 4–6, the view on the data schema is graphically rendered by framing classes with dashed lines.

Detecting the activities that operate on a certain data class. From a different viewpoint, Activity Views and DW-Activity Views allow designers to learn which process activities have access to objects of a particular class. This information is valuable for several reasons, including facilitating communication with domain experts and end-users during process modeling. In particular, end-users are often interested in knowing where specific data are used in the process to gain an overview of which is the information that drives certain activities and used to make decisions. This is also useful when considering the compliance of the process with business rules or regulations: indeed, in some cases, the quality of activity execution may drastically improve if proper information is available. Last but not least, from an engineering perspective, understanding how data are used during process execution provides hints for data management support and process re-engineering.

For example, class Infection of data warehouse DW (cf. Fig. 6) is accessed by activity Situation assessment and Notification to media, as highlighted by the filled background. By taking a look at the structure of the process, one can see that class Infection is needed both at the beginning and at the end of the process. The set of activities $ac_g, \dots, ac_l$ that perform read or write operations on a certain class c_i is defined as $\{ ac_k \mid \exists t_{j,k} (c_i \in C_{set_{j,k}}) \}$ and can be derived by scanning all the Activity Views of a process and checking whether c_i belongs to at least one class set $C_{set_{j,k}}$ of a tuple $t_{j,k} \in av_{ac_k}$.

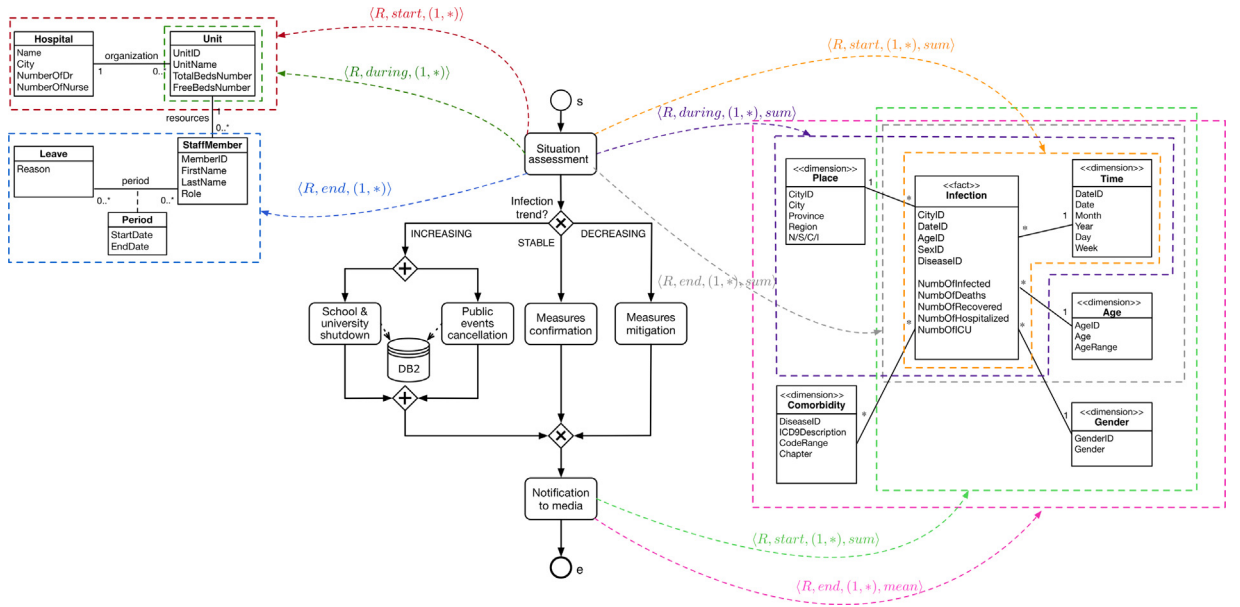


Fig. 6. Conceptual insights provided by using Activity Views on both database and data warehouse schemata.

Understanding which classes are either read or written by a process activity. The type of access to data allows designers to easily visualize whether classes have associated read or write operations and how these are distributed in the process. Besides, it is also possible to retrieve which classes of the data schema are associated only to read or write accesses. This is particularly useful when speaking about data integrity, as several activities of one or more processes may operate on the same data class concurrently and, thus, transactional properties must be considered at lower data modeling levels [5,6]. Moreover, when sequences of read and write operations are performed on the same classes, inconsistencies may arise and must be properly detected [9,31]. As an example, consider learning if the objects of a certain class c_k are only read by process activities. To this end, we must go through all the Activity Views and ensure that there exists no tuple having $c_k \in C_{set}$ and access type of kind I , U or D : $\{c_k \mid \nexists j, i ((C_{set_{j,i}} \ni c_k) \wedge AccessType_{j,i} \in \{I, U, D\})\}$. For example, class *Hospital* is only read by the process of Fig. 6. Clearly, since Definition 2 considers only read operations, this insight becomes trivial when considering data warehouse classes.

Identifying decision activities, i.e., activities accessing a data warehouse. In our proposal, we define as *decision activity*, an activity managing data stored in a data warehouse for supporting decision-making by using aggregate information. In the process of Fig. 5, activity *Notification to media* is a *decision activity* accessing data warehouse DW in order to support a decision by means of aggregate information.

BPMN provides the Business Rule Task [10], which allows the process to interact with a Business Rules Engine for executing one or more rules. Business rules can be specified by means of the Decision Model and Notation (DMN) [32], the OMG standard for modeling decisions.

The definition *decision activity* we propose is broader from a Business Rule Task since a *decision activity* is able to perform different types of operations and not only executing business rules. As an example, in the process of Fig. 5, activity *Situation assessment* can be considered as a *decision activity* even if it accesses both data stored in a transactional database and a data mart. Such a definition can support the identification of decision activities, which are often regarded as those preceding exclusive gateways or identified by stakeholders [33,34].

The set of decision activities $ac_g, \dots, ac_i$ accessing a data warehouse can be derived by scanning all the Activity Views of a process and checking whether there exist at least one tuple $t_{j,k} \in av_{ac_k}$ whose C_{set} and A_{set} include classes and associations of the data warehouse.

Detecting which activities access data stored both in databases and in data warehouses. In supporting decision processes different information, stored in different data sources, could be useful. The quality of decisions improves when proper information is considered. Decisions should benefit from available information regardless of the type of the sources. Understanding which activities need information and are strongly related to data sources is a very important aspect to consider.

By combining the insights introduced above, designers can identify and visualize the critical information for supporting process execution, also with the help of domain experts. Such information can be captured by one or more data classes, which we refer to as *core classes* for a given process.

Informally, given a data schema of a database or a data warehouse, a *core class* is a class of the data schema that represents valuable process-related data and

- (i) it belongs to a considerable number of Activity Views related to the process (i.e., it is shared by multiple process activities);
- (ii) its objects are frequently accessed by the process, that is, it belong to a considerable number of Activity View tuples;
- (iii) its objects are used by the most important activities of the process (i.e., activities that are crucial for the chosen application domain or belong to the main execution path, if any);
- (iv) its objects are never deleted by the process;
- (v) it is mostly subjected to mandatory access [29], that is $\min$ of $NumInstances$ is never set to 0.

Starting from these different possibilities, we consider the first case (i) and introduce the following preliminary definition of *core class*.

Definition 3 (Core Class). Let us consider a set $Ac = \{ac_1, \dots, ac_n\}$ of activities in a process model, a set $Cl = \{c_1, \dots, c_m\}$ of classes in a data schema, and a set $AV = \{av_{ac_1}, \dots, av_{ac_n}\}$ of Activity Views corresponding to the activities of Ac . Given a threshold $0 \leq k \leq n$, a class $cc \in Cl$ is a core class if $|\{av | av \in AV \wedge cc \in C_{set_{av}}\}| \geq k$. $\diamond$

Concerning the process of Fig. 6, fact class Infection is a core class, as it is the most accessed in the process. Indeed it is used by both activities Situation Assessment and Notification to Media. Such activities, at the beginning and at the end of the process, respectively, are common to all the possible execution traces and represent the main decision activities of the process. While in such a motivating example the identification of important decision-support data is quite straightforward, the concept of core classes becomes useful in complex and highly branched processes, where identifying the key information to support (decision-intensive) process execution is not trivial. This kind of issue is similar to the open issue of identifying process artifacts in the field of data-centric process modeling [4,5].

6. Related work

Our work falls within the area of process modeling and, mainly, it is related to approaches considering the integrated modeling of activity-centric processes and data coming from persistent sources, e.g., databases or data warehouses, at the *conceptual level*. However, as demonstrated in [9], the Activity View can be specialized to consider process instances for checking data inconsistencies, thus touching the area of data-aware process verification [2], which we also discuss briefly. To wrap up, we recall some of the main contributions concerning business processes and data warehouses.

In the BPM field, several research efforts have tackled the relationship between data and processes, mostly focusing on the integrated modeling of data and activity-centric processes [7,8,35,36], including (high-level) Petri nets [6,37], and their verification [2,31,38–41], the design and verification of data- and object-centric processes [4,5,29,42], and the extraction of event data residing in databases for process mining [43–46].

Approaches based on activity-centric modeling paradigms, such as BPMN [7,8,35,36,41] and UML activity diagrams [47], have mostly focused on bridging the gap between process and data models, by considering the two of them as “independent entities” to be connected and, often, by attaching data to process models [42]. Despite their acknowledged limitations for modeling perspectives other than the control-flow, activity-centric modeling paradigms, and especially those based on BPMN, remain indeed the most used in practice [8].

Motivated by the need for supporting software development, Cruz et al. [35] propose an approach for deriving a data model from a BPMN process model, mapping swimlanes and data artifacts to data entities, associations, and attributes of the data schema. Meyer et al. [8] propose to automatically derive SQL queries from annotated BPMN data objects to check data requirements for activity execution. In a similar vein, De Giacomo et al. [7] rely on (Object Constraint Language) expressions to link BPMN diagrams to the information model of the process, represented as a UML class diagram, including a class *Artifact* representing all the process variables. BPMN activities are specified as OCL operation contracts, from which logic rules are derived and then mapped to SQL statements, which are run against the database during process execution. Despite acknowledging the need for a conceptual connection between processes and data, both works in [7,8] consider such connection at the logical level, i.e., defining variables, and focus on automating data manipulation. Along this line, Perez et al. [41] have recently proposed a methodology for verifying the consistency and completeness of BPMN process models annotated with data states, thereby tackling a more abstract modeling level compared to [8]. However, data objects are considered as simple instances of a database class; that is, the relations between multiple classes and potentially having different cardinalities are not considered. Similarly to the rule derivation in [7], Proietti et al. [36] introduce a framework based on constraint logic programming to define the behavioral semantics of a process, i.e., a BPMN diagram annotated with data manipulations, as a state transition system. Overall, a significant weakness of the Activity View lies in its graphical representation, which suffers from the rapid proliferation of connections. However, this issue seems to affect also the mappings and annotations in [8] and [41], as the number of data objects increases with higher numbers of data instances and operations.

As for approaches grounded in Petri nets, Montali et al. [6] propose *db-nets* as a novel three-layered model combining colored Petri Nets and relational databases through an intermediate data logic layer. With this approach, the Activity View shares the grounding idea of combining process and data models in a holistic view and the limitation of the graphical representation. However, the approach in [6] goes beyond conceptual modeling, as it also addresses the verification of the overall “connected system” [2]. Trčka et al. [31] defined workflow nets with data (WFD-nets) together with anti-patterns, expressed as temporal logic formulae, aimed to capture repeated mistakes during process analysis. Later, the same authors proposed a technique for verifying the soundness of workflow nets with data [38]. Tao et al. [39] suggest workflow nets with tables (WFT-nets), an extension to WFD-nets including

a table logic layer, specifying constraints and operations on data records, and a persistence layer, including a relational database. However, the main focus of the works in [31,38,39] is data-aware process verification, which is not considered in this paper, also because it would require taking the logical data level into account.

Data- and object-centric approaches promote data to a first citizen, modeling processes as a mirror of their manipulated data [42]. Artifact-centric approaches [4] are probably the best-known paradigm for conceptually integrating processes with business logic and data layers. Besides them, the umbrella of data-centric approaches includes a wealth of other research efforts. For instance, Sun et al. [5] focus on modeling the link between a set of business entities representing artifacts of interest and a relational database, defining the notions of updatability and isolation to ensure consistency between business entities and the database. Recently, Artale et al. [42] have formalized object-centric behavioral constraints (OCBC), a novel approach to process modeling, allowing one to integrate declarative constraints and data models and supporting formal design-time reasoning. However, data-centric approaches are grounded in different principles from the Activity View, as they consider the process as a subordinate of the data and do not allow one to model process activities that do not have data access. Thereby, we refrain from discussing them in detail.

The problem of providing a holistic view of processes and data was also tackled in the field of process mining, especially in the correlation and abstraction phases of log preparation [46]. Calvanese et al. [43] propose an approach grounded in conceptual modeling for supporting the extraction of event logs from legacy information systems. Their approach allows one to annotate the events in a log with the class of the conceptual model it refers to, to exploit such annotations for log extraction, building upon the ontology-based data access (OBDA) paradigm. The proposed annotations can be placed at the same level of Activity Views, but annotations connect one event in the log to one data class, i.e., activities are not considered, and events are not associated with multiple database classes. Murillas et al. [45] introduce a meta-model as a standard abstraction to support the creation of multi-perspective process logs from data coming from different sources. The meta-model includes three levels of process granularity, namely (i) processes, (ii) instances, and (iii) events, and three data abstraction levels, namely (i) data model, (ii) objects, and (i) versions, i.e., the instantiations of an object during a certain amount of time. The connection between the process and the data perspectives is realized between events and versions, i.e., at the lowest level. Tsoury et al. [44] design a conceptual framework merging information coming from an event log, a transaction log, and a relational database storing current values of data attributes, to support the in-depth exploration of business process behavior. However, both works in [45] and [44] link processes and data at the level of events and database instances, supporting process analysis at the level of case and object identifiers. Albeit this is not in the scope of this paper, it lays a fertile ground for future extensions of the Activity View, e.g., by considering different abstraction levels of process and data modeling.

The conceptual modeling of business process models and data warehouses is considered in the literature from different perspectives. One of the main streams of research deals with the modeling of Extraction, Transformation and Loading (ETL) processes using standard process modeling languages such as BPMN and UML [48–51]. Another one focuses on the design of (process-oriented) data warehouse systems for process mining [52]. Several papers are related to support process-related analytics and decision making [53,54], and to combine KPIs and decision variables with business processes [55,56].

Interestingly, the work by Santos et al. [54], which extends the proposal in [35], proposes an approach to derive a data warehouse schema from a set of BPMN processes and the conceptual model of the operational database used by these processes. In pursuing their aim, the authors discuss how different process aspects, particularly data stores, contribute to the analytical data model, which is derived from the processes and the operational data model. However, decision activities are not explicitly identified as those accessing the data warehouse, which is intended to support a set of high-level business decisions derived from organizational processes. Also, being derived from the processes as done in [53], the data warehouse schema is not “independent”. Instead, in our view, the process perspective and the data perspective are kept separate, i.e., data models can contain information that is not used by a specific process, and process activities do not necessarily have to access the database.

Overall, although the mentioned research lines are related to the Activity Views only partially, they tackle some critical challenges that are also central to the approach proposed in this paper, most and foremost, the need for supporting designer tasks and for identifying which data are required by process activities to be executed, including when and how these should be made available.

7. Discussion and conclusion

In this paper, we faced the issue of bridging the gap between process and data conceptual design by introducing and extending the concept of Activity View, which allows specifying the link between process activities and related data during the conceptual design. This issue is a very important topic in the BPM context, since process activities need to manage information stored in databases and data warehouses. The relation between processes and data is usually not realized at the conceptual level, even if connecting activities and data operations at the beginning of the modeling process is useful for supporting designers in having an overarching view of the modeled domain and detecting potential conceptual mismatches [9,31].

Building upon our initial proposal in [26], in this work we started from decision activities and related analytical data and suitably extended the concept of Activity View to allow the specification of the main aggregation operations on such multidimensional data. Then, we showed how Activity Views can support the discovery of interesting insights, feasible only when we have such a “connected perspective” in conceptual design. The concept of Activity View has been introduced both through a formal notation and a graphical one, which helps to improve the readability of conceptual diagrams. Moreover, different examples have been provided through a motivating example taken from the current worldwide situation.

For evaluating the Activity View, we conducted a controlled experiment with students and academics to assess whether the Activity View improves the comprehension of connected processes and data [26]. Our findings reported that the Activity Views

improve the comprehension of integrated processes and data. In detail, participants reported that the Activity View is rather easy to read, use, and understand, but had some difficulties in writing it and remarked the need to improve its graphical representation. Overall, this initial evaluation led to promising results and set the bases for future work.

In particular, since conceptual models are graphical models, their connection should be better described graphically. In the future, we plan to consider the graphical representation of Activity View in a more detailed way. Indeed, in its current form, the graphical representation of the Activity View could become very complex for large processes, and a modular approach based on different abstraction levels could improve its readability. Besides, we plan to conduct an exhaustive case study in real-world settings to understand the benefits and drawbacks of such a seamless approach to conceptual modeling and its implications for practice.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] M. Reichert, Process and data: Two sides of the same coin?, in: R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, I.F. Cruz (Eds.), 20th International Conference on Cooperative Information Systems, Springer, 2012, pp. 2–19, http://dx.doi.org/10.1007/978-3-642-33606-5_2.
- [2] R. De Masellis, C.D. Francescomarino, C. Ghidini, M. Montali, S. Tessaris, Add data into business process verification: Bridging the gap between theory and practice, in: 31st AAAI Conference on Artificial Intelligence, AAAI Press, 2017, pp. 1091–1099.
- [3] D. Calvanese, G. De Giacomo, M. Montali, Foundations of data-aware process analysis: a database theory perspective, in: 32nd ACM SIGMOD Symposium on Principles of Database Systems, PODS, ACM, 2013, pp. 1–12.
- [4] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, J. Su, Towards formal analysis of artifact-centric business process models, in: 5th International Conference on Business Process Management, BPM, in: LNCS, vol. 4714, Springer, 2007, pp. 288–304, http://dx.doi.org/10.1007/978-3-540-75183-0_21.
- [5] Y. Sun, J. Su, B. Wu, J. Yang, Modeling data for business processes, in: 30th International Conference on Data Engineering, ICDE, IEEE, 2014, pp. 1048–1059, <http://dx.doi.org/10.1109/ICDE.2014.6816722>.
- [6] M. Montali, A. Rivkin, DB-nets: On the marriage of colored Petri nets and relational databases, in: Transactions on Petri Nets and Other Models of Concurrency XII, in: LNCS, vol. 10470, Springer Berlin Heidelberg, 2017, pp. 91–118, http://dx.doi.org/10.1007/978-3-662-55862-1_5.
- [7] G. De Giacomo, X. Oriol, M. Estañol, E. Teniente, Linking data and BPMN processes to achieve executable models, in: 29th International Conference on Advanced Information Systems Engineering, CAiSE, in: LNCS, vol. 10253, Springer, 2017, pp. 612–628, http://dx.doi.org/10.1007/978-3-319-59536-8_38.
- [8] A. Meyer, L. Pufahl, D. Fahland, M. Weske, Modeling and enacting complex data dependencies in business processes, in: 11th International Conference on Business Process Management, BPM, in: LNCS, vol. 8094, Springer, 2013, pp. 171–186, http://dx.doi.org/10.1007/978-3-642-40176-3_14.
- [9] C. Combi, B. Oliboni, M. Weske, F. Zerbatto, Conceptual modeling of inter-dependencies between processes and data, in: ACM Symposium on Applied Computing, SAC, ACM, 2018, pp. 110–119, <http://dx.doi.org/10.1145/3167132.3167141>.
- [10] Object Management Group, Business process model and notation (BPMN), v2.0.2, 2014, Available at: <http://www.omg.org/spec/BPMN/2.0.2/>.
- [11] G. Booch, J. Rumbaugh, I. Jacobson, Unified Modeling Language User Guide, second ed., Addison-Wesley Professional, 2005.
- [12] Object Management Group, Unified modeling language, v2.5, 2015, Available at: <http://www.omg.org/spec/UML/2.5/>.
- [13] Workflow Manag. Coal., D. Hollingsworth, The workflow reference model, 1995, <http://www.wfmc.org>.
- [14] D. Hollingsworth, The workflow reference model: 10 years on, in: The Workflow Handbook 2004, Future Strategies Inc., Lighthouse Point, FL, USA, 2004, pp. 295–312.
- [15] J. Desel, J. Esparza, Free Choice Petri Nets, Cambridge University Press, USA, 1995.
- [16] W.M.P. van der Aalst, K. van Hee, Workflow Management: Models, Methods, and Systems, MIT Press, Cambridge, MA, USA, 2004.
- [17] W.M.P. van der Aalst, C. Stahl, Modeling Business Processes - a Petri Net-Oriented Approach, Cooperative Information Systems series, MIT Press, 2011, <http://mitpress.mit.edu/books/modeling-business-processes>.
- [18] G. De Giacomo, M. Dumas, F.M. Maggi, M. Montali, Declarative process modeling in BPMN, in: Advanced Information Systems Engineering, CAiSE '15, in: Lecture Notes in Computer Science, vol. 9097, Springer, 2015, pp. 84–100, http://dx.doi.org/10.1007/978-3-319-19069-3_6.
- [19] M. Montali, F.M. Maggi, F. Chesani, P. Mello, W.M.P. van der Aalst, Monitoring business constraints with the event calculus, ACM TIST 5 (1) (2013) 17:1–17:30, <http://dx.doi.org/10.1145/2542182.2542199>.
- [20] R. Posenato, F. Zerbatto, C. Combi, Managing decision tasks and events in time-aware business process models, in: BPM, in: Lecture Notes in Computer Science, vol. 11080, Springer, 2018, pp. 102–118.
- [21] R. Elmasri, S.B. Navathe, Fundamentals of Database Systems, Pearson, 2015.
- [22] B. Thalheim, Entity-Relationship Modeling - Foundations of Database Technology, Springer, 2000, <http://dx.doi.org/10.1007/978-3-662-04058-4>.
- [23] B. Thalheim, Conceptual modeling foundations: The notion of a model in conceptual modeling, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, second ed., Springer, 2017, pp. 1–3, http://dx.doi.org/10.1007/978-1-4899-7993-3_80780-1.
- [24] A. Olivé, A method for the definition of integrity constraints in object-oriented conceptual modeling languages, Data Knowl. Eng. 59 (3) (2006) 559–575.
- [25] A.A. Vaisman, E. Zimányi, Data Warehouse Systems - Design and Implementation, Data-Centric Systems and Applications, Springer, 2014.
- [26] C. Combi, B. Oliboni, M. Weske, F. Zerbatto, Conceptual modeling of processes and data: Connecting different perspectives, in: 37th International Conference on Conceptual Modeling, ER 2018, Xi'an, China, October 22–25, 2018, Proceedings, in: Lecture Notes in Computer Science, vol. 11157, Springer, 2018, pp. 236–250, http://dx.doi.org/10.1007/978-3-030-00847-5_18.
- [27] M. Golfarelli, D. Maio, S. Rizzi, The dimensional fact model: A conceptual model for data warehouses, Int. J. Cooper. Inform. Syst. 7 (02n03) (1998) 215–247.
- [28] R. Kimball, M. Ross, The Data Warehouse Toolkit: The Complete Guide To Dimensional Modeling, John Wiley & Sons, 2011.
- [29] V. Künzle, M. Reichert, Philharmonicflows: towards a framework for object-aware process management, J. Softw. Maint. Evol.: Res. Pract. 23 (4) (2011) 205–244, <http://dx.doi.org/10.1002/smr.524>.
- [30] S. Smirnov, H.A. Reijers, M. Weske, T. Nugteren, Business process model abstraction: a definition, catalog, and survey, Distrib. Parallel Databases 30 (1) (2012) 63–99, <http://dx.doi.org/10.1007/s10619-011-7088-5>.
- [31] N. Trčka, W.M. Van der Aalst, N. Sidorova, Data-flow anti-patterns: Discovering data-flow errors in workflows, in: 21st International Conference on Advanced Information Systems Engineering, CAiSE, Springer, 2009, pp. 425–439, http://dx.doi.org/10.1007/978-3-642-02144-2_34.
- [32] Object Management Group, Decision model and notation (DMN), v1.3, 2019.
- [33] K. Batoulis, A. Meyer, E. Bazhenova, G. Decker, M. Weske, Extracting decision logic from process models, in: International Conference on Advanced Information Systems Engineering, Springer, 2015, pp. 349–366, http://dx.doi.org/10.1007/978-3-319-19069-3_22.

- [34] E. Bazhenova, F. Zerbato, B. Oliboni, M. Weske, From BPMN process models to DMN decision models, *Inf. Syst.* 83 (2019) 69–88, <http://dx.doi.org/10.1016/j.is.2019.02.001>.
- [35] E.F. Cruz, R.J. Machado, M.Y. Santos, From business process modeling to data model: A systematic approach, in: 2012 Eighth International Conference on the Quality of Information and Communications Technology, IEEE, 2012, pp. 205–210, <http://dx.doi.org/10.1109/QUATIC.2012.31>.
- [36] F. Smith, M. Proietti, Reasoning on data-aware business processes with constraint logic, in: 4th International Symposium on Data-Driven Process Discovery and Analysis, SIMPDA, 2014, pp. 60–75.
- [37] K. Lenz, A. Oberweis, *Inter-organizational business process management with XML nets*, in: *Petri Net Technology for Communication-Based Systems*, Springer, 2003, pp. 243–263.
- [38] N. Sidorova, C. Stahl, N. Trčka, Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible, *Inf. Syst.* 36 (7) (2011) 1026–1043, <http://dx.doi.org/10.1016/j.is.2011.04.004>, Special Issue: Advanced Information Systems Engineering (CAISE'10).
- [39] X. Tao, G. Liu, B. Yang, C. Yan, C. Jiang, Workflow nets with tables and their soundness, *IEEE Trans. Ind. Inf.* (2019) <http://dx.doi.org/10.1109/TII.2019.2949591>.
- [40] D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Formal modeling and SMT-based parameterized verification of data-aware BPMN, in: 17th International Conference on Business Process Management, BPM, in: LNCS, vol. 11675, Springer, 2019, pp. 157–175, http://dx.doi.org/10.1007/978-3-030-26619-6_12.
- [41] J.M. Pérez-Álvarez, M.T. Gómez-López, R. Eshuis, M. Montali, R.M. Gasca, Verifying the manipulation of data objects according to business process and data models, *Knowl. Inf. Syst.* (2020) 1–31, <http://dx.doi.org/10.1007/s10115-019-01431-5>.
- [42] A. Artale, A. Kovtunova, M. Montali, W.M.P. van der Aalst, Modeling and reasoning over declarative data-aware processes with object-centric behavioral constraints, in: T. Hildebrandt, B.F. van Dongen, M. Röglinger, J. Mendling (Eds.), *Business Process Management*, Springer International Publishing, Cham, 2019, pp. 139–156, http://dx.doi.org/10.1007/978-3-030-26619-6_11.
- [43] D. Calvanese, T.E. Kalayci, M. Montali, S. Tinella, Ontology-based data access for extracting event logs from legacy data: the onprom tool and methodology, in: International Conference on Business Information Systems, Springer, 2017, pp. 220–236, http://dx.doi.org/10.1007/978-3-319-59336-4_16.
- [44] A. Tsoury, P. Soffer, I. Reinhartz-Berger, A conceptual framework for supporting deep exploration of business process behavior, in: Proceedings of the 37th International Conference on Conceptual Modeling, ER, in: LNCS, vol. 11157, Springer, 2018, pp. 58–71, <http://dx.doi.org/10.1007/978-3-030-00847-6>.
- [45] E. González López de Murillas, H.A. Reijers, W.M.P. van der Aalst, Connecting databases with process mining: a meta model and toolset, *Softw. Syst. Model.* 18 (2) (2019) 1209–1247, <http://dx.doi.org/10.1007/s10270-018-0664-7>.
- [46] K. Diba, K. Batoulis, M. Weidlich, M. Weske, Extraction, correlation, and abstraction of event data for process mining, *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.* (2019) e1346, <http://dx.doi.org/10.1002/widm.1346>.
- [47] S.X. Sun, J.L. Zhao, J.F. Nunamaker, O.R.L. Sheng, Formulating the data-flow perspective for business process management, *Inf. Syst. Res.* 17 (4) (2006) 374–391, <http://dx.doi.org/10.1287/isre.1060.0105>.
- [48] J. Trujillo, S. Luján-Mora, A UML based approach for modeling ETL processes in data warehouses, in: I.-Y. Song, S.W. Liddle, T.-W. Ling, P. Scheuermann (Eds.), *Conceptual Modeling - ER 2003*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 307–320, http://dx.doi.org/10.1007/978-3-540-39648-2_25.
- [49] L. Muñoz, J.-N. Mazón, J. Trujillo, Automatic generation of ETL processes from conceptual models, in: Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP, in: DOLAP '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 33–40, <https://doi.org/10.1145/1651291.1651298>.
- [50] Z. El Akkaoui, J.-N. Mazón, A. Vaisman, E. Zimányi, BPMN-Based conceptual modeling of ETL processes, in: A. Cuzzocrea, U. Dayal (Eds.), *Data Warehousing and Knowledge Discovery*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 1–14, http://dx.doi.org/10.1007/978-3-642-32584-7_1.
- [51] J. Awiti, A. Vaisman, E. Zimányi, From conceptual to logical ETL design using BPMN and relational algebra, in: C. Ordóñez, I.-Y. Song, G. Anderst-Kotsis, A.M. Tjoa, I. Khalil (Eds.), *Big Data Analytics and Knowledge Discovery*, Springer International Publishing, Cham, 2019, pp. 299–309, http://dx.doi.org/10.1007/978-3-030-27520-4_21.
- [52] T. Vogelgesang, H.-J. Appelrath, Pmcube: A data-warehouse-based approach for multidimensional process mining, in: M. Reichert, H.A. Reijers (Eds.), *Business Process Management Workshops*, Springer International Publishing, Cham, 2016, pp. 167–178, http://dx.doi.org/10.1007/978-3-319-42887-1_14.
- [53] T. Benker, A generic process data warehouse schema for BPMN workflows, in: W. Abramowicz, R. Alt, B. Franczyk (Eds.), *Business Information Systems*, Springer International Publishing, Cham, 2016, pp. 222–234, http://dx.doi.org/10.1007/978-3-319-39426-8_18.
- [54] M.Y. Santos, J. Oliveira e Sá, A data warehouse model for business processes data analytics, in: O. Gervasi, B. Murgante, S. Misra, A.M.A. Rocha, C.M. Torre, D. Taniar, B.O. Apduhan, E. Stankova, S. Wang (Eds.), *Computational Science and Its Applications - ICCSA 2016*, Springer International Publishing, Cham, 2016, pp. 241–256, http://dx.doi.org/10.1007/978-3-319-42092-9_19.
- [55] J.M. Pérez-Álvarez, A. Maté, M. a Teresa Gómez-López, J. Trujillo, Tactical business-process-decision support based on KPIs monitoring and validation, *Comput. Ind.* 102 (2018) 23–39, <http://dx.doi.org/10.1016/j.compind.2018.08.001>.
- [56] A. Maté, J. Trujillo, J. Mylopoulos, Specification and derivation of key performance indicators for business analytics: A semantic approach, *Data Knowl. Eng.* 108 (2017) 30–49, <http://dx.doi.org/10.1016/j.datak.2016.12.004>.

Carlo Combi is full professor of Computer Science at the Dept. of Computer Science, University of Verona. In 1993, he received the Ph.D. degree in biomedical engineering from the Politecnico di Milan. From 2009 to 2013 he was chair of the Artificial Intelligence in Medicine Society (AIME). Since 2017 he is Editor-in-Chief of the journal *Artificial Intelligence in Medicine*. His main research interests are in the database and information systems field, with an emphasis on clinical data and processes.

Barbara Oliboni is associate professor at the Dept. of Computer Science of the University of Verona. She received the Ph.D. degree in Computer Engineering by the Politecnico di Milan. Her main research interests are in the database field, with an emphasis on semistructured data, temporal information, business processes management, and clinical information management. She is part of the Program Committee of International Conferences, and reviewer for International Journals.

Professor Dr. Mathias Weske is chair of the business process technology research group at Hasso Plattner Institute at the Digital Engineering Faculty, University of Potsdam, Germany. The research group aims at addressing real-world problems in business process management with formal approaches and engineering useful prototypes. His research focuses on the engineering of process oriented information systems, process mining, and event processing. The BPT research group has a track record in engineered prototypes with a significant impact on research and practice, including projects like Oryx and jBPT. He co-founded Signavio and he is business angel at Synfioo. Dr. Weske is author of the first textbook on business process management and he held the first massive open online course on the topic in 2013. With Matthias Kunze, he published a textbook on behavioral models. He is on the Editorial Board of Springer's *Computing journal*, and he is a founding member of the steering committee of the BPM conference series and, since September 2017, chair of the steering committee.

Francesca Zerbato is a post-doc at the Institute of Computer Science, University of St. Gallen, Switzerland. She obtained her Ph.D. degree in Computer Science from the University of Verona in 2019. Her main research interests are in information systems and BPM, with an emphasis on process and data modeling, and healthcare applications. She is a member of the program committee of the BPM and SAC conferences, and managing editor of the journal *Artificial Intelligence in Medicine*.