Bus Inf Syst Eng 62(1):41–60 (2020)
https://doi.org/10.1007/s12599-019-00611-5

RESEARCH PAPER

Data Impact Analysis in Business Processes

Automatic Support and Practical Implications

Arava Tsoury · Pnina Soffer · Iris Reinhartz-Berger

Received: 11 September 2017 / Accepted: 27 May 2019 / Published online: 12 August 2019

Abstract Business processes and their outcomes rely on data whose values are changed during process execution. When unexpected changes occur, e.g., due to last minute changes of circumstances, human errors, or corrections of detected errors in data values, this may have consequences for various parts of the process. This challenges the process participants to understand the full impact of the changes and decide on responses or corrective actions. To tackle this challenge, the paper suggests a semi-automated approach for data impact analysis. The approach entails a trans-formation of business process models to a relational database representation, to which querying is applied, in order to retrieve process elements that are related to a given data change. Specifically, the proposed method receives a data item (an attribute or an object) and information about the current state of process execution (in the form of a trace upon which an unexpected change has occurred). It analyzes the impact of the change in terms of activities, other data items, and gateways that are affected. When evaluating the usefulness of the approach through a case study, it was found that it has the potential to assist experienced process participants, especially when the consequences of the change are extensive, and its locus is in the middle of the process. The approach contributes both to practice with tool-supported guidance on how to handle unexpected data changes, and to research with a set of impact analysis primitives and queries.

Keywords Business processes · Impact analysis · Data inaccuracy

Accepted after three revisions by Jörg Becker.

A. Tsoury (✉) · P. Soffer · I. Reinhartz-Berger
University of Haifa, Abba Khoushy Ave. 199, 3498838 Haifa, Israel
e-mail: atsoury@is.haifa.ac.il

P. Soffer
e-mail: spnina@is.haifa.ac.il

I. Reinhartz-Berger
e-mail: iris@is.haifa.ac.il

1 Introduction

Business process management has become a leading paradigm for managing and conducting business activities, providing organizations operational benefits of consistency, reduced cost, and increased speed and quality of both products and services (Hammer 2015). Business processes are supported by information systems and rely on data for the execution of activities and decision making. They also form a context for functions of the information systems, tying them together in a logical flow. While in the past process models mainly focused on control flows, in the last decade, data-centric approaches for process models gained recognition as necessary for supporting both control and data flows (Reijers et al. 2017). Thus, various studies, such as Trčka et al. (2009), Rodríguez et al. (2012) and Sun and Zhao (2013), consider analyzing and modeling data flows to ensure the complete and accurate design of business processes.

During business process execution, data values change, primarily due to activities that are performed as expected in the regular flow of the process. However, some value changes might be unexpected, due to last minute changes of circumstances (e.g., an urgent need to order an additional quantity to an order) or due to human errors or corrections of detected errors in data values (Soffer 2010). These changes may lead to modifications in the values of

additional data elements or to changes in the planned flow of the process. Moreover, unexpected changes in data values may have consequences for other parts of the process. It is even possible that already made decisions or already performed activities were based on inappropriate values and considerations. In such cases, it is important to understand the impact of the changes over the entire process, so that process participants can handle the change at runtime. In particular, it is necessary to figure out how the data that has changed may affect the execution path of the process, its activities and the outcomes these may have, so appropriate actions (or corrective actions) can be taken.

To illustrate these challenges, consider a course opening process in an academic institution. The process includes setting the details of the course (e.g., course code, course name, and number of academic credit points), determining the required resources (human and equipment) for the course, determining a test schedule, creating events planned for the course (e.g., lecture and laboratory slots), scheduling the timetable and allocating rooms for each event. Consider that an error occurred when typing the course capacity that is used for determining the resources required for the course, such as the size of the allocated room and the number of grading hours. Such an error requires not just the correction of the course capacity, but also affects decisions on compensatory actions, e.g., reallocation of rooms and redetermination of human resources. In contrast, this error has no effect on the assignment of teaching resources for the course, nor on the test dates allocation. Thus, an understanding of the scope of the change and the effort required for its application is needed.

Another example from a different domain concerns a make-to-order process, in which customers order products from a catalog, and the products are produced upon demand. Consider that the customer asks to change the shipping address to one that is different from what appears in the order. The value of this attribute is used in various activities, such as *handle delivery details*, and may influence additional data elements along the process, such as *approximate delivery date* and *packing type* (these attributes are determined based on the distance to the destination, directly derived from the shipping address).

The reaction to unexpected changes in business processes has been addressed in the context of service-oriented processes (Alam et al. 2015) where a change in one service or more may affect other parts of the process. To deal with this, methods for change propagation and change impact analysis in the context of business processes have been proposed, e.g., in Wang et al. (2010), Dam and Ghose (2015), and Dai et al. (2009). However, the focus of these approaches is mainly on the control flow of the process, with data only marginally considered. Furthermore, the granularity level in which data is typically addressed by these approaches is that of objects, rather than of their specific attributes. We consider this granularity as too coarse-grained to allow a detailed data-centric impact analysis.

To address these gaps, in a previous paper (Tsoury et al. 2016), we introduced the concept of data impact analysis in business processes, analyzing the effects of a single data item – an attribute or an object – on other business process elements. To explicitly address dependencies among business process elements, we suggested transforming a business process model to a relational database representation. We further suggested relation primitives and associated queries for populating this database and retrieving element dependencies from it. The queries can be used by different algorithms, as we exemplified with two proposed algorithms. One algorithm aimed at supporting design-time analysis, by retrieving all the impacts of a specific data item across the process. The other supported runtime changes, taking into consideration a specific process state when changes take place.

In the current paper we focus on supporting process participants in handling situations of unexpected changes in data values. Bearing in mind that such situations are exceptional rather than the typical course of the process, we consider the information a participant would need for handling changes at runtime. We extend and substantiate the approach presented in Tsoury et al. (2016) and develop an additional algorithm for runtime analysis that considers the impact of a data item on the execution path. Given a current state of an executed process (as a partial trace), the algorithm analyzes the impact of an (unexpected) change in a data item, mainly in terms of activities, other data items, and gateways. To evaluate the support provided by the approach to process participants, the paper presents an experimental case study which was performed in an organizational setting. The case study addresses different scenarios of unexpected data changes in the course of a process. We presented these scenarios to a group of experienced process participants and compared their understanding of the consequences of these changes with the analysis results obtained by the suggested approach. We then discuss the strengths and weaknesses of the proposed approach for different scenarios. Further, we describe the strategies the participants described for coping with unexpected changes.

The remainder of the paper is organized as follows. Section 2 presents preliminaries of the approach. Section 3 presents the extension of the approach and the newly added algorithm. Section 4 presents the case study and the evaluation results. Section 5 reviews the related work in view of our approach, and, finally, Sect. 6 concludes and discusses future research directions.

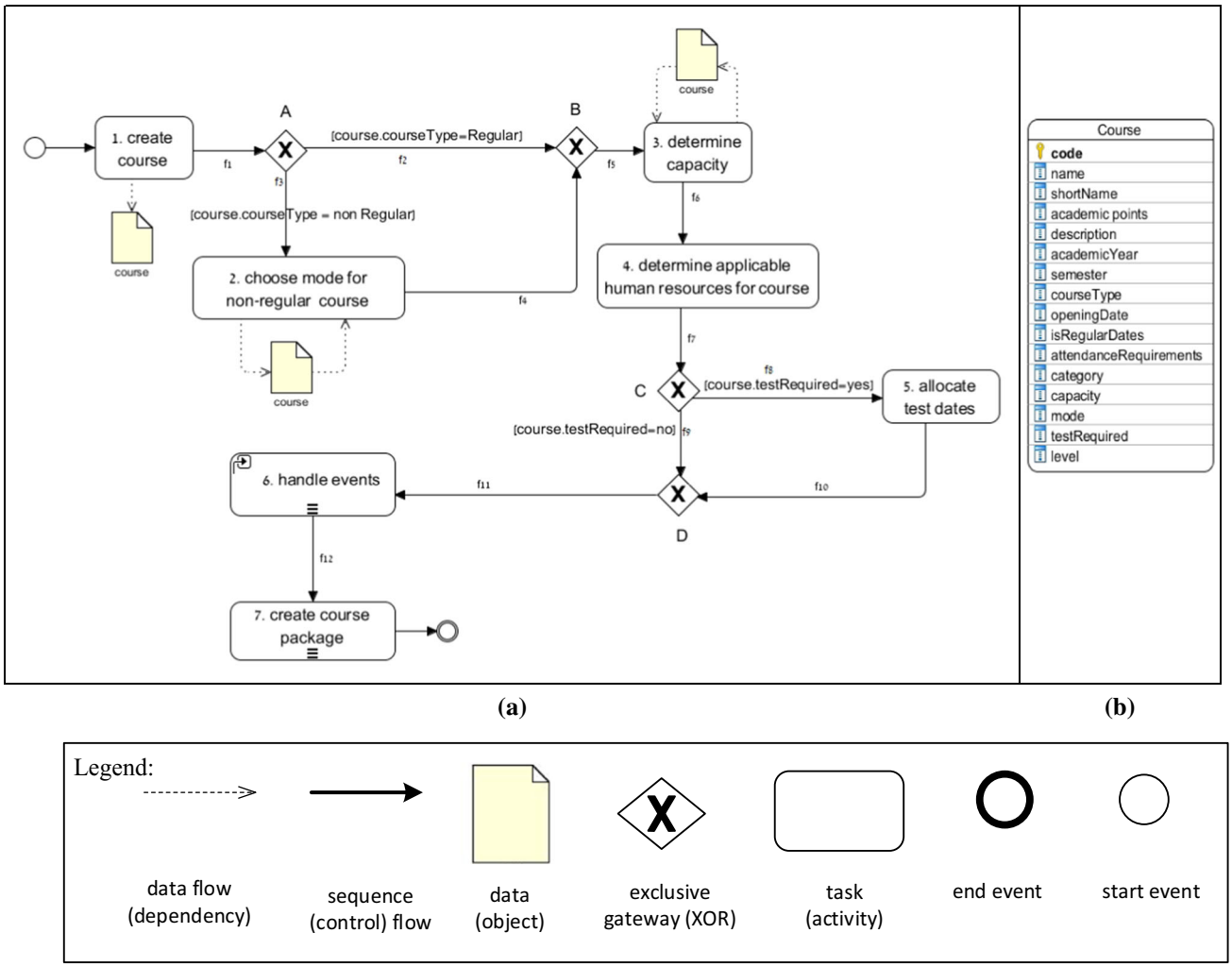

Fig. 1 **a** A high-level process model of course opening; **b** A partial data model of the course object

2 Preliminaries: From Process Model to Relational Database

2.1 Running Example

We start by introducing the course opening process that was mentioned in the introduction and will serve as a running example throughout the paper.

Figure 1 depicts a partial high-level model of the process, including a partial process model (a) and a partial data model only specifying the course class (b). The first activity – *create course* – sets the values of various attributes of the course, including its identifier (code), name, type, level, etc. This activity creates a new instance of a course, as depicted by the dependency between the activity and *course*. When creating the course, its type (e.g., regular or non-regular) is determined. Non-regular courses, which have no weekly physical meetings, can be in one of the following modes: *online* (virtual meetings, if any) or *special schedule* (not necessarily on a weekly basis). The second activity – *choose mode for non-regular course* - selects the *course mode*. This is followed by the activity *determine capacity*, which modifies a specific attribute of the course (capacity) according to the course type and mode (see additional details below). The next activity – *determine applicable human resources for course* – defines the list of staff members who can teach the course. The following activity – *allocate tests dates* – determines the tests dates for the course. Finally, the activity *handle events* is actually a subprocess in which a set of activities is performed for each event of the course, i.e., lecture, tutorial or lab. This set includes determination of event schedule, staff members, equipment, and rooms.

2.2 Data-Aware Process Model – Definitions

Process models can be specified in different languages (e.g., BPMN, EPC). In many of these representations, data

is a secondary element which is commonly under-specified. As a basis for conducting different types of analysis, we proposed a metamodel of process elements and their dependencies in Tsoury et al. (2016). This metamodel is generic and refers to a subset of elements common in most business process modeling languages. This includes activities, gateways, routing constraints, and flows. Note that different languages may include additional elements - such as events, resources, and states - which are currently out of the scope of this paper. Without the loss of generality, we demonstrate our approach by using BPMN, which is widely used in both industry and academia. Moreover, it has been claimed in Bhattacharya et al. (2007) that BPMN has a simple and understandable graphical notation, and even users who are not experts and unfamiliar with the full notation are able to understand these diagrams.

Many process modeling languages (including BPMN) use a construct of data objects to depict the data that is used and manipulated in the process. These objects typically relate to compound entities, such as orders and customers. In many cases, only specific attributes of these objects are relevant for particular process elements (e.g., activities). Thus, using data objects as a basic element will not allow for a detailed data-centric impact analysis. Following the course opening example, using the data object *course* in the process model does not explicitly capture cases where only one specific attribute is changed, and is hence too coarse-grained for our purpose. Generally, when high-level data objects appear in the model, it is difficult to analyze the exact impact of a specific data attribute. Hence, we introduced the lower-level concept of data items.

Definition 1 (*Data item*) Adata item is an attribute or an object whose impact is of interest. It is represented by a pair (n, t), where n is the data item name and t represents its type.

Examples of data items are (course name, string), (course type, {regular, non-regular}), (course level, int), and even course as a whole object (see Fig. 1b). Note that the data item type denotes the possible (finite or infinite) range of values the data item can assume. It can be simple, compound, or user-defined.

Data items may appear in different places in the business process model, including in routing constraints.

Definition 2 (*Routing constraint*) Arouting constraint is denoted as $r(d_1,\ldots, d_n)$, where r is any Boolean expression over the set of data items $\{d_1, d_2 \ldots d_n\}$ using arithmetic, as well as logical, operators.

[*Course.courseType = Regular*] and [*Course.courseType = nonRegular*] are examples of routing constraints in Fig. 1a, involving the data item *course type*.

To incorporate data items, rather than data objects, into business process models, we suggested the following definition of a data-aware process model.

Definition 3 (*Data-aware process model*) A data-aware process model is a tuple PM = (Act, G, F, DI, R, RF, IAO), where:

- Act is a finite (non-empty) set of activities
- G is a finite set of gateways
- $F \subseteq (Act \cup G) \times (Act \cup G)$ is a finite set of control flows
- DI is a finite (non-empty) set of data items
- R is a finite set of routing constraints over data items in DI. We denote by *Support* a mapping from R to the power set of DI, *Support:* $R \rightarrow \wp$ *(DI)*, which returns the data items involved in a certain routing constraint.
- $RF \subseteq R \times F$ associates routing constraints in R to control flows in F.
- $IAO \subseteq (DI \cup \{Ø\}) \times Act \times (DI \cup \{Ø\})$ is a set of data flows representing dependencies between activity inputs and outputs.[1]

The model depicted in Fig. 1a can be easily represented as a data-aware process model: Act = {create course, choose mode for non-regular course, determine capacity, determine applicable human resources for course, allocate test dates, handle events, create course package}, G = {A, B, C, D}, F = {f1, f2, …, f12}, D = {course, course.code, course.capacity, …}, and IAO = {(null, create course, course), (course.code, determine capacity, course.capacity), …}.

Generally, data items can be related to each other not only through process elements, but also through data dependencies, commonly specified as constraints and triggers in database management systems. Therefore, we assume a set of relations between data items.

Definition 4 (*Data item relation*) Adata item relation (DIR) is an ordered pair of the form (d_i, d_j), where d_i and d_j are data items and a change in d_i (potentially) implies a change in d_j irrespectively of the activities in which these data items are involved.[2]

In our example, non-regular courses, which have no weekly physical meetings, can be in one of the following modes: *online* or *special schedule*. If the mode is set to

[1] Note that the input or the output of an activity may be an empty set, when the output does not use any specific data input or the input is used without creating any output, respectively. Note in addition that if there are di, dj∈DI, a∈Act, such that (di, a, dj) ∈ IAO, then (null, a, dj) ∉ IAO and (di, a, null) ∉ IAO.

[2] Note that DIR is binary and uni-directional (dj depends on di). Ternary relations and relations of higher degrees are relaxed to binary relations. Bi-directional relations are specified as two uni-directional relations.

online, the course *capacity* is automatically assigned a default value of *unlimited;* otherwise, it is set to a predefined default numeric value. This means that (*mode*, *capacity*) ∈ DIR.

2.3 Data-Aware Process Model Dependencies and Relational Representation

Based on the above definitions, we designed a metamodel of a data-aware process model (see Fig. 2), using Entity-Relationship (ER) notation (Chen 1976), which is geared to describe relational databases. Note that, focusing on a single process, the metamodel includes the process elements and their relations, excluding the process model itself. Gateways and activities are abstracted as behavioral nodes, to enable jointly associating them to flows (as sources or destinations). The gateway types considered in this paper are the basic control flow patterns, namely, XOR and AND (van der Aalst et al. 2003), as our focus here is on data.

Four main differences exist between our metamodel and other data-aware process model definitions, such as those presented in Meyer and Weske (2013) and Meyer et al. (2013). First, we use the finer-grained notion of data items, as opposed to data objects. This allows us to analyze the impact of data changes at the item level. Second, we explicitly specify the (logical) relations between routing constraints and data items (the "used in" relation). This way we are able to consider changes in process paths as a result of data changes. Third, we capture indirect relations

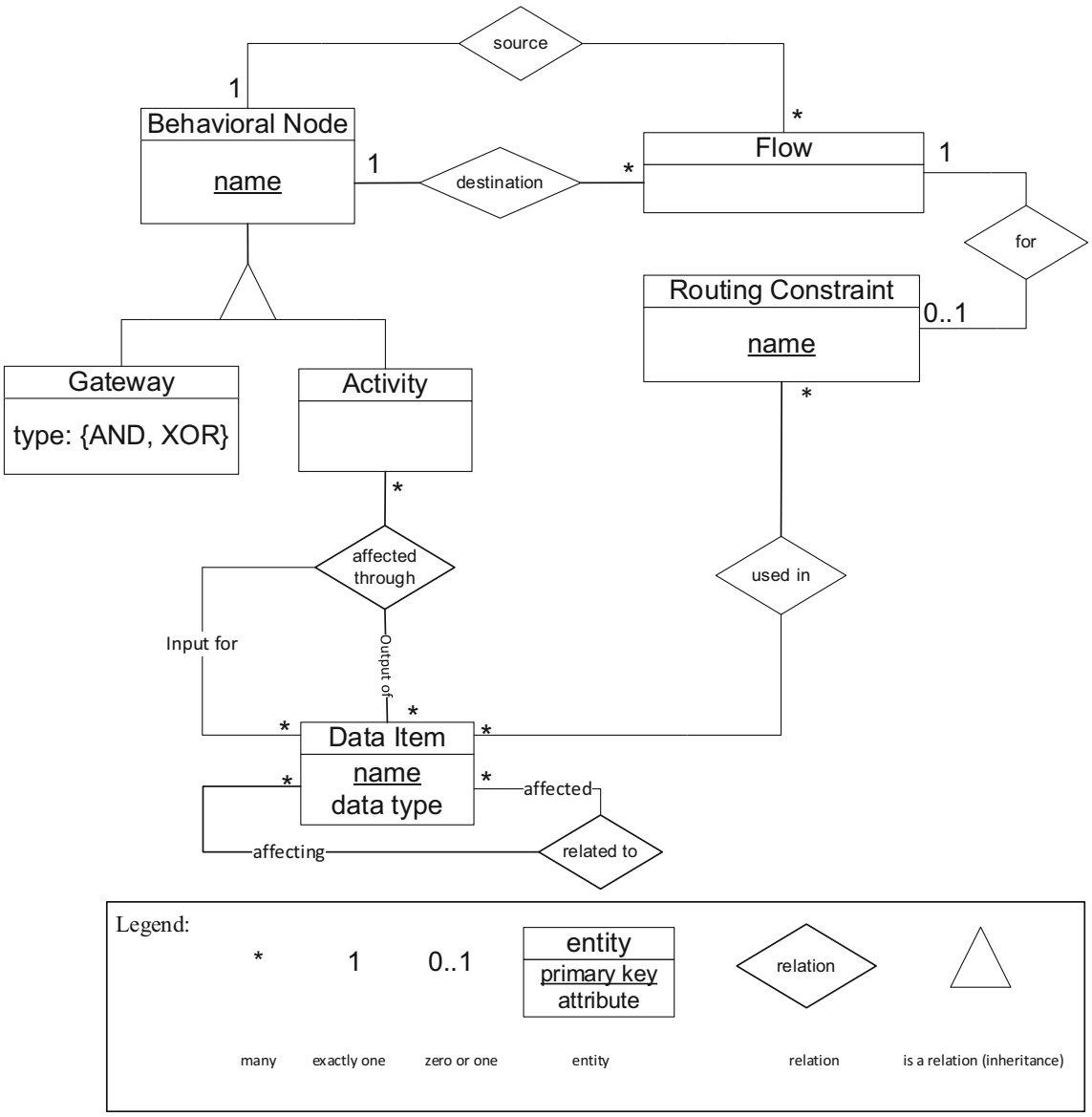

Fig. 2 A metamodel of a data-aware process model (using ER notation)

between data items through activities (the "affected through" relation). Previous works capture this kind of dependencies through two independent relations – inputs and outputs – challenging the accurate impact of a certain input data item. Fourth, we explicitly describe dependencies between data items that are not represented in the business process model but may exist in the database as constraints or triggers (the "related to" relation). These dependencies have also impact on data changes.

The suggested metamodel depicts dependencies between the process elements. For example, gateways and routing constraints are related through flows; particularly, routing constraints are related to the outgoing flows of gateways of type XOR. Similarly, data items and gateways are also related through the notions of routing constraints and flows.

3 The Proposed Approach

The proposed approach follows the relations among process elements of the metamodel. We first discuss types of dependencies between model elements. Then, we present a set of data impact primitives. Finally, we introduce the new algorithm for analyzing the impact of a data item at process execution.

3.1 Data Dependencies

When a process is executed, its activities are enabled and activated based on the order determined by the control flow in the process model. The execution involves data dependencies and data flows.

According to Sun et al. (2006), activities can perform read or write operations on data items; this includes creation and modification of data items (write operations) or access to them (read operations). In general, each dependency between an activity and a data item represents that the activity has made changes to the data item, whereas a dependency in the opsite direction represents a read operation. An activity can write a data item without reading it first. For example, when creating the course, the activity *create course* creates an instance of a course from scratch, inserting a new record to the database without reading existing values. Accordingly, we refer to two possible effects of data on activities: (1) the data item serves as an input for the activity, for creating output or for affecting the way the activity will be performed. (2) The data value sets a precondition for the execution of the activity through a routing constraint attached to a control flow edge. Only if the routing constraint is evaluated as true the activity is enabled. For example, only if [*courseType* = *nonRegular*] is true, the activity *choose mode for non-regular courses* can be executed.

Following this, we refer to three types of dependencies in a process model: control flow dependency (CF), enabling constraint dependency (EC), and data flow dependency (DF).

Control flow dependencies link behavioral nodes (activities and gateways) through flow edges. In our running example, the flow f1 which connects the activity *create course* and the XOR gateway *A* exemplify a control flow dependency.

Enabling constraint dependencies are affected by routing constraints attached to flows and control the execution of the behavioral nodes at their destination. As an example, consider the dependency between the routing constraint [*courseType* = *nonRegular*], associated to the flow f2, and the activity *Choose mode for non-regular course*: the routing constraint enables the activity. Another example is the dependency between the routing constraint [*courseType* = *Regular*], associated to the flow f3, and the XOR gateway B. Here the routing constraint enables the gateway.

Together control flow and enabling constraint dependencies are used to determine the traversed paths of process executions.

Finally, data flow dependencies are the means by which data values affect and are affected by process elements, specifically activities and routing constraints. Activities are affected by input data items and affect output data items. For example, the data item *course type* is an input for the activity *determine capacity,* and the activity *create course* writes the data item *course*. These perform two data flow dependencies. In addition, routing constraints are evaluated based on values of data items. For example, the routing constraint [*courseType* = *non Regular*] uses the value of *course type*, creating another data flow dependency. Last, data dependencies may also exist directly as structural relations among data items, referred to as DIR. In our example, the data item *course.mode* affects the data item *course.capacity*, creating yet another data dependency.

Table 1 summarizes the dependencies among process model elements as a basis for the data impact primitives we present in the next section. The corresponding roles of the elements (in brackets) are taken from the metamodel in Fig. 2.

3.2 Impact Primitives

Taking a data impact perspective, the six proposed primitives are given in Table 2: the first one represents data item relations (DIR), the next four relate to data flow (DF) dependencies and the last one stands for enabling constraint (EC) dependencies. Note that control flow (CF) dependencies are not related to data, and hence are not regarded as part of the primitives. The table provides for

Table 1 Impact relations between process elements

	Activity	Gateway	Flow	Data item	Routing constraint
Activity	–	–	CF	DF (output of)	–
Gateway	–	–	CF	–	–
Flow	CF	CF	–	–	–
Data item	DF (input for)	–	–	DIR (related to)	DF (used in)
Routing constraint	EC (destination)	EC (destination)	–	–	–

CF control flow, *DF* data flow, *EC* enabling constraint, *DIR* data item relations

Table 2 Data impact primitives

Effect type	#	Name	Informal description	Formal representation	Example
DIR	P1	Data on data	When the value of d_i changes, the value of d_j may also be changed due to a structural constraint, irrespectively of a specific activity	$(d_i, d_j) \in$ DIR	(mode, capacity)[a] $\in$ DIR
DF	P2	Activity on data (output)	An activity a_x affects (creates, deletes, or updates) the value of a data item d_j independently of any other (input) data item	$(null, a_x, d_j) \in$ IAO	(null, create-course, course) $\in$ IAO
	P3	Data on activity (input)	An activity a_x uses the value of the data item d_i, but d_i does not determine any of a_x outputs	$(d_i, a_x, null) \in$ IAO	(courseType, determine-capacity, null) $\in$ IAO
	P4	Data on data through activity	An activity a_x uses the value of d_i for determining the value of d_j	$(d_i, a_x, d_j) \in$ IAO	(numRegistrants, register-to-course numRegistrants[b]) $\in$ IAO
	P5	Data on routing constraint	A data item d_i is used in the logical expression of the routing constraint r_x	$d_i \in$ Support(r_x)[c]	courseType $\in$ support (courseType = nonRegular)
EC	P6	Routing constraint on activity	The activity a_x is enabled only if r_x is evaluated to true	$\exists f \in F, b \in Act \cup G$ such that $(r_x, f) \in RF$ and $f = (b, a_x)$	(courseType = nonRegular, f3) $\in$ RF and f3 = (A, choose-mode-for-non-regular-courses)

[a]For the sake of simplicity, assume that course capacity is set by default to infinity if the course mode is online and to a finite value N otherwise

[b]After each execution of *register-to-course*, the number of registrants is increased by one

[c]Recall *Support* is a mapping from R to the power set of DI (see Definition 3)

each primitive an informal description, a formal representation, and an example related to the course opening process.

In order to retrieve impacts using the aforementioned primitives, the approach uses a relational database derived from the metamodel in Fig. 2. Each table stores certain process elements (e.g., data items, gateways, activities, or flows) or relations between process elements (e.g., RelatedTo, AffectedThrough or UsedIn). Due to primary key restrictions, the ternary relation – affected through – is realized through three tables capturing the following three cases:

1. IsOutputOf – holding pairs (a, d), where the data item *d* is an output of the activity *a* regardless of its input. This corresponds to tuples (null, a, d) $\in$ IAO.
2. IsInputOf – holding pairs (d, a), where the data item *d* is an input of the activity *a* that does not affect its output. This corresponds to tuples (d, a, null) $\in$ IAO.
3. AffectedThrough – holding triples (d_i, a, d_j) where the d_i is an input of the activity *a* used to produce or modify d_j. This corresponds to tuples $(d_i, a, d_j) \in$ IAO.

Seven queries are defined on top of the relational database (see Table 3). Each query filters the rows of a certain table and is recursively used by the approach for analyzing the impact of value change. Each query returns a list of elements that are affected by a given element (@key) along with their types ('d' for data items, 'a' for activities, and 'r' for routing constraints). The element type is used for percolating the impact on relevant process elements (e.g., change in data items may impact routing constraints, activities to which the data items serve as inputs, and other data items).

3.3 Impact Analysis of Data Change

Focusing on runtime, we wish to retrieve all process elements impacted by unexpected changes in data values that

Table 3 Queries for extracting impacts of process elements

Effect Type	Primitive name	Query output	Query	Example
DIR	P1 – Data on data (through value)	Returns all data items affected by a given data item through structural relations	Q1: Select affected.name, 'd' From RelatedTo Where affecting.name = @key	@key ← 'mode'; all data items affected by *mode* change, e.g., *capacity*
DF	P2 – Activity on data (output)	Returns all data items modified by a given activity, independently of its input	Q2: Select dataItem.name, 'd' From IsOutputOf Where activity. name = @key	@key ←'create course'; all data items which are the output of *create course*, e.g., all attributes of *course*
	P3 – Data on activity (input)	Returns all activities using a given data item without utilizing it for generating outputs	Q3: Select activity.name, 'a' From IsInputFor Where dataItem.name = @key	@key ←'capacity'; all activities which use (read) *capacity* without modifying data items, e.g., *allocate rooms*
	P4 – Data on data through activity	Returns all output data items affected by a given input data item through activities	Q4.1: Select outputOf, 'd,' From AffectedThrough Where inputFor = @key	@key ←' numRegistrants'; all data items which are modified based on 'numRegistrants', e.g., numRegistrants itself (which is increased by 1 due to a new registration)
	P4 – Data on data through activity	Returns all activities using a given input data item to create or modify an output data item	Q4.2: Select activity.name, 'a' From AffectedThrough Where inputFor = @key	@key ← 'numRegistrants'; all activities which uses this data item to modify a data item, e.g., *register-to-course* (the activity which increases numRegistrants by 1)
	P5 – Data on routing constraint	Returns all routing constraints using a given data item in their logical expressions	Q5: Select routingConstraint. name, 'r' From UsedIn Where dataItem. name = @key	@key ← courseType; all routing constraints that uses the value of courseType, e.g., [course.coureType = regular]
EC	P6 – Routing constraint on activity	Returns all activities executed only if a given routing constraint is evaluated to true	Q6: Select activity.name, 'a' From Flow Where routingConstraint. name = @key	@key ← [Course.courseType = nonRegular]; all activities that are executed if the given routing constraint is evaluated to true, e.g., *choose mode for non-regular courses*

may occur while the process is executed. As an example, assume that while determining the course *capacity* (see Fig. 1a), an unexpected change in the *course type* is required since the lecturer decided that the course should be *regular*, instead of *non-regular* as previously announced. Assuming the *course type* was *non-regular*, the secretary previously selected the path of *create course* → *A* → *choose mode for non-regular courses* → *B* → *determine capacity*. With the updated value, the relative path should be: *create course* → *A* → *B* → *determine capacity*. As a consequence, the activity *choose mode for non-regular courses* should be undone and the course capacity should be determined. To this end, the decision point represented by gateway *A* should be revisited so the corrected path can be taken.

To tackle such scenarios, a clear definition of the process state when a change takes place is required. This information is given in the form of a process trace.

Definition 5 (*Process trace*) A*process trace* is a sequence $pt = <n_1, \ldots, n_m>$, where n_i is a behavioral node (an activity or a gateway) and the order of n_i reflects the temporal order of a single process execution (instance).

$pt_1 = <$*create course, A, choose mode for non-regular courses, B, determine capacity*$>$, $pt_2 = <$*create course, A, choose mode for non-regular courses*$>$ and $pt_3 = <$*create course, A, B, determine capacity*$>$ are examples of possible traces in Fig. 1a. We consider the partial trace performed for a specific process instance until the occurrence of an

unexpected data change as denoting the process state upon which the change occurred. Yet, this refers to the behavioral nodes only, without considering the associated data items. To explicitly refer to the data items relevant for a certain process trace, an extraction of the data flow in the trace is required. Therefore, we define next a closure of a trace ($\overline{pt}$).

uses the queries defined in Sect. 3.2 to retrieve process elements in the closure of the given trace that are affected by the given data item. This is repeated for each element that was retrieved, using the queries that fit the element type. Note that any element is checked only once, and thus the algorithm stops when all impacts (linked elements) have been retrieved and no elements are left to check.

Inputs: *di* – a data item, *pt* – a process trace
Used structure: S is a list of pairs *(name, type)*, where type can be d (data), a (activity), or r (routing constraint).
Used functions:
pt.closure () – returns a list of elements in the closure of pt ($\overline{\boldsymbol{pt}}$). Each element is describes as a pair (name, type), where type ∈ {d, a, r}
S.*getFirstUnchecked()* – returns the first element in the list S that was not checked by the algorithm (maintains a local list of checked elements); returns null if all elements in S were checked.

```
// Initialization; S holds the given data item
pt = pt.closure()
S = new List();
S.add (di, 'd')

// Loop on the elements in S according to the order of insertion
e = S.getFirstUnchecked()
Do until e=null
        switch e.type
                case (d)  // For data items, primitives P1, P3, P4 (handled by Q4.1, Q4,2), and P5 are relevant
                          S = S ∪ (∪_{i∈{1,3,4.1,4.2,5}}(Q_i ∩ pt))
                case (a)  // For activities, primitive P2 is relevant.
                          S = S ∪ (Q_2 ∩ pt)
                case (r)  // For routing constraints, primitive P6 is relevant
                          S = S ∪ (Q_6 ∩ pt)
        e = S.getFirstUnchecked()
End Do
return S
```

Listing 1. Pseudo code of the impact analysis algorithm of value change

Definition 6 (*Closure of trace*) The closure of a trace includes the trace itself and all data items and routing constraints related to behavioral nodes in the trace. Formally expressed, given a trace pt = <n_1, …, n_m>, its closure $\overline{\boldsymbol{pt}}$ is defined as $\{n_i\}_{i=1\ldots m} \cup \{d \in DI \mid \exists n \in \{n_i\}_{i=1\ldots m}, d' \in DI \cup \{\emptyset\}$ such that (d, n, d′) ∈ IAO or (d′, n, d) ∈ IAO$\} \cup \{r \in \mid \exists f \in, n, n' \in \{n_i\}_{i=1\ldots m}$ such that f = (n, n′) and (r, f) ∈ RF$\}$.

In our example, $\overline{\boldsymbol{pt}_2}$ = {create course, A, choose mode for non-regular course, mode, [courseType = nonRegular], determine capacity, course, capacity}.

The algorithm, whose pseudo code is provided in Listing 1, retrieves potential impacts of an unexpected change in data value at runtime. It receives a partial process trace (depicting the partial execution of the process until the occurrence of the change) and a data item to which the change applies. It returns a set S holding the process elements (activities, data items, and routing constraints) that might be affected by the change. To this end, the algorithm

Considering our example of an unexpected change in the course type, for the inputs d_i = *courseType*, *pt* =< *create course, A, B, determine capacity*>, the algorithm will retrieve the following process elements: {*(courseType, 'd'), (capacity, 'd'), ([courseType = Regular], 'r'), (determine capacity, 'a')}*. This would imply that the activity *determine capacity* should be revisited. The value of course *capacity* may also change, as it is the output of the revisited activity *determine capacity*. The gateway leading to the routing constraint *[courseType = Regular]* need to be revisited in order to decide whether the value change leads to a different execution path.

Note that the process participants need to get *all* retrieved activities and data items as these may require some correction. In particular, when the process model is not fully complied with, the full list of activities to be revisited is an important guidance. The complete set of traversed routing constraints is less relevant, since the previously selected execution path may no longer be valid

due to the value change. Instead, the process participants need to be directed to points in the control flow where decisions could be modified as a result of the value change. Such points are modelled as gateways of type XOR. Hence, on top of the retrieved process elements, S, the output given to the process participants by our method includes three sets: the set of affected activities (S_A), the set of affected data items (S_{DI}) and the set of affected gateways (S_{DG}) which are defined next.

Definition 7 (*Set of affected activities*) All the activities that have been performed and may need correction due to the value change form the *set of affected activities* (S_A). Formally expressed, $S_A = \{e \in S \mid e \text{ is an activity}\}$.

Definition 8 (*Set of affected data items*) All data items whose values have been modified or may need to be modified form the *set of affected data items* (S_{DI}). Formally expressed, $S_{DI} = \{e \in S \mid e \text{ is a data item}\}$.

In order to define the set of affected gateways, we need to define first the notion of gateway precedence.

Definition 9 (*Gateway precedence*) Given a data-aware process model PM = (Act, G, F, DI, R, RF, IAO), a gateway $g' \in G$ precedes a gateway $g \in G$ iff there are $n_k \in G \cup$ Act, $k = 1_{...}N$, such that $n_1 = g' \wedge n_N = g \wedge (n_i, n_{i+1}) \in F$ for $i = 1_{...}N - 1$.

Definition 10 (*Set of affected gateways*) The set of affected gateways (S_{DG}) contains the set of XOR gateways included in the process trace pt, for which (a) at least one of the outgoing routing constraints is affected by the value change, and (b) there are no preceding XOR gateways according to the process model PM that satisfy (a). Formally expressed, given a data-aware process model PM = (Act, G, F, DI, R, RF, IAO) and a process trace pt, and the set of elements returned from the algorithm S, $S_{DG} = \{g \in G \mid g.type='XOR' \wedge \exists n_i, n_{i+1} \in pt, r \in S \cap R$ such that $n_i = g \wedge (r, (n_i, n_{i+1})) \in RF \wedge \neg\exists g' \in G, n_j, n_{j+1} \in pt, r' \in S \cap R$ such that $g.type='XOR' \wedge n_j = g' \wedge (r', (n_j, n_{j+1})) \in RF \wedge g'$ precedes $g\}$.

In our example, S_{DG} contains only one gateway (A), deciding whether to execute *choose mode for non-regular course*. Since this is a simple example, the number of retrieved elements is small. However, in large process models, the number of affected elements may be large and complicated to handle by process participants.

The complexity of the algorithm is $O(n^2)$, where n is the number of elements in the process model. The outer loop runs |S| times which is at most n. For each element, five queries run at the worst case (for data items). Each one of the defined queries runs on a single table, namely its complexity is O(n). Note that the creation of the trace closure ($\overline{pt}$) is also $O(n^2)$.

4 Evaluation

The evaluation of our approach aims to examine change impacts obtained by the algorithm in comparison to (a) change impacts obtained through the existing common practice, which is the manual handling of unexpected changes by experienced process participants, and (b) a 'ground truth' baseline of change impacts. To this end, we conducted an experimental case study concerning a detailed course opening process at a university. The process was operated using an SAP information system, which was deployed at the university in 2006.

We examined four scenarios where modifications to data values were required during process execution. The impacts of these changes were identified in three independent ways: first, we established the 'ground truth' baseline with the help of a process expert. Second, we applied our algorithm to obtain a list of change impacts. Third, we conducted a series of interviews with experienced process participants to obtain the change impacts as identified when manually handling the scenarios. Below, we elaborate on the settings, procedure, analysis, and results of the evaluation.

4.1 Settings

4.1.1 Experimental Material

The selection of the process for the evaluation study followed six criteria: (1) The process should be structured and possess a well-defined flow. (2) The process should be rich in terms of relationships among its model elements. In particular, the process should demonstrate the different types of relations of the metamodel (Fig. 2). (3) The process should be executed by humans using an information system (as opposed to a fully automated or a manual one). (4) Unexpected changes in data values should be part of the routine operation of the process so that participants were used to handle them. (5) Experts (IT people) should be available to provide details and support the creation of an initial process model, change scenarios and a 'ground truth' baseline for change impacts. With these criteria, we selected the university course opening process, performed by the administrative staff of the academic departments. The process is introduced as the running example in Sect. 2.1 and depicted at a high level in Fig. 1. A detailed version of the process, along with its data flows, can be found online.[3] Overall, in its detailed level, the process includes 15 activities, 14 gateways, 10 routing constraints, and 66 data items (which relate to 25 distinctive data objects).

[3] https://sites.google.com/view/dataimpactanalysis/home.

Table 4 The examined scenarios

#	Name	Description	Locus of change	Span of effect	Prevalence
1	TA change (due to parental leave)	A teaching assistant assigned to a course announces that he/she will be absent due to parental leave and will not be able to teach the course	Middle	Medium	Common
2	Change in course type (from regular to non-regular)	A lecturer who was assigned to teach a regular (standard schedule) course announces that she/he will be away during the semester. A special schedule will be needed for the course to be completed	Beginning	High	Exceptional
3	Additional resources needed (timetable and rooms)	A lecturer was assigned to teach an online course (infinite capacity) with a defined schedule for online sessions. The lecturer wishes to change the course type to a regular course	Beginning	High	Rare
4	Unavailability of rooms	After all the courses have been opened, the maintenance division announces that due to renovations an entire floor in a certain building will not be available in the first semester	End	Medium	Common

We created an initial BPMN model of the process based on the local SAP user manual. Next, we interviewed two managers who oversaw the process to validate the model. Last, we interviewed the Head of the Students and Teaching division in the IT department of the university, who deals with the academic administration processes and was also involved in the incorporation of the SAP system at the university and the accompanied process design. For our purposes, she was considered as the top expert of the process (hereafter, we refer to her as the process expert) who could provide a final fine-tuning and validation of the process model.

4.1.2 The Examined Scenarios

We designed four modification scenarios (see Table 4) that may occur during the process, following several considerations. First, the scenarios should be specific in order to communicate them to the administrative staff. However, they should represent a class of cases from the participants' world. For example, the scenario of parental leave (scenario #1) represents a situation where an already assigned member of the course staff needs to be replaced (we chose this scenario rather than any other reason for absence since the time of child birth cannot be negotiated, thus this is a firm constraint).

Second, these scenarios need to systematically cover a spectrum of change behaviors. For this purpose, we looked for a possible basis over which systematic coverage could be established. First, we reviewed studies on flexibility of processes (Weber et al. 2008), exception handling in processes (Rinderle and Reichert 2006), and impacts of changes in processes (Soffer 2005). However, these mostly focus on changes in the control flow, such as inserting or deleting activities, and changing their order. Although such changes may lead to data changes and may potentially lead to data anomalies such as missing, redundant, conflicting or lost data, we found them not suitable as a basis for a set of scenarios of unexpected value changes at runtime. Similarly, data-centered redesign changes suggested in (Tsoury et al. 2016) mainly refer to change operations that apply to the schema of the process rather than to runtime values, and are thus not a suitable basis for the scenarios. We hence used our proposed metamodel as a basis and designed the scenarios to include the data dependencies discussed in Sect. 3.1. Particularly, the *related-to* dependency, in which the changed data item affects another data item without an activity, is used in scenarios #3 and #4; the *affected-through* dependency, in which the changed data item is used in an activity to produce an output, is considered in scenarios #1, #2 and #4; and the *used-in* dependency, in which the changed data item is used in a routing constraint and in which therefore potential changes in the flow are expected, is used in scenario #2.

Third, the scenarios should exhibit a variety of behaviors concerning (1) the locus of the change, i.e., where in the process the change takes place (beginning, middle, [towards the] end), (2) the span of the effect, which could be substantial (changes at least 50% of the process elements) or medium (changes of up to 50%), and (3) prevalence of the scenario in the regular work of the subjects, which can indicate the familiarity of the administrative staff with the change. For each scenario Table 4 presents, besides its name and description, the locus of the effected change, the span of the effect, and the prevalence of such change.

4.2 Procedure

After selecting the scenarios and specifying them, we introduced them to the process expert and asked her to indicate the expected impacts of each scenario. For this task, we showed her the process model and asked her to indicate for each scenario the set of affected data items, activities, and decision points (gateways). According to her answers, we built the 'ground-truth' baseline.

Table 5 The ground truth baseline, the algorithm's results and an example of participant's responses before and after guidance for scenario #1 (TA change)

Ground truth baseline—indicated element	Algorithm's results	Participant's results (initial responses)	Participant results (after guidance)
Activities			
Determine applicable human resources for course (in case that the new TA is not in this list already)	√	X	X
Define sub-event schedule (in case the schedule does not fit to the new TA's schedule)	√	X	√
Define sub-event human resources (for the actual replacement)	√	√	√
Allocate rooms (if the schedule has changed)	√	X	√
Allocate computer labs (if the schedule has changed and computer labs are required)	√	X	√
Create course package (to hold the new data as one package)	√	X	X
Data items			
Staff member for course	√	X	
Staff member for sub-event	√	√	√
Sub-event schedule	√	X	√
Sub-event schedule_computerLab	√	X	√
Sub-event schedule_lectureRoom	√	X	√
Package	√	X	X
Academic year catalog	X	X	X
Schedule shifts	X	X	√

We ran the proposed algorithm for the same scenarios. To this end, we have developed a prototype tool implemented in MS-SQL environment. The database tables were derived from the metamodel in Fig. 2. The tool further supports an automated population of the database by extracting the elements of a BPMN process model in an XML format. Additional relations among elements (data item relations – DIR – and input-activity-output – IAO) were manually inserted.

In parallel to the tool development and algorithm execution, we performed a series of interviews with experienced process participants to understand how they would handle such scenarios. The participants were eleven administrative staff members at the university who operate the course opening process. They belonged to eight different departments. Seven of them were chief administrative staff while four were administrative assistants. Most of them (nine out of eleven) had worked at the university for more than 10 years and were familiar with the SAP information system since its deployment at the university. The other two participants had worked at the university for about 5 years. The participants operated the *course opening* process on a yearly basis for all the courses that were offered by their departments.

We introduced the process to each participant using a short presentation to ensure a shared perception of the process and its underlying concepts. In addition, the participants received handouts of the process model and a list of process elements along with their descriptions.

The interviews were separately conducted with each of the eleven participants. For each scenario, we presented (a) the current state of the process, upon which the scenario begins, (b) the trigger for the change, and (c) the (unexpected) value change in a specific data item. The participants were requested to indicate the operations that would be required to handle the changes implied by the scenario. It was expected that as a result, a chain of implied changes should take place, and these would form the impact of the (initial) value change. After the participants provided their answers, we verbally summarized them and asked 'what-if' questions, allowing additions or modifications to their answers. We refer to the answers given at this step as 'after guidance.' The interviews were recorded and transcribed.

Table 5 exemplifies the outcomes of the analysis for scenario #1 (TA change).

4.3 Data Analysis

Overall, we had 44 responses (11 participants and 4 scenarios). However, we excluded five answers for the following reasons: (1) for Scenario #2, two of the participants avoided giving specific answers, claiming that the entire process, including all steps and elements, would be affected, and they would have to update "everything." They did not change their answers after guidance. (2) In Scenario #3,

three participants answered that they would not change anything in the process. Instead, they would handle the changes manually, or would not allow such scenario at all. Eventually, 39 responses were analyzed.

The interview transcriptions were coded to match the exact terms used in the process model. To avoid bias, a sample of the coding was checked by two independent, non-involved researchers. Each of the responses was compared to the 'ground truth' baseline, and its elements were classified as true positives (indicated elements that match the baseline indications), true negatives (not indicated elements that match the baseline lack of indications), false positives (indicated elements that do not match the baseline), and false negatives (not indicated elements that do not match the baseline).

We then calculated for each scenario and participant the metrics of precision, recall, and F-measure. Precision measures the fraction of elements indicated by a participant that are indeed relevant according to the baseline (true positives out of all positives). Recall measures the fraction of elements that were indicated by the participant from the relevant ones (true positives out of all "true"). F-measure is the harmonic mean of precision and recall. We calculated the average precision, recall, and F-measure for all participants, separately for their initial answers and for the answers 'after guidance'. Similarly, we analyzed the results of the algorithm and calculated the related metrics. The results are reported in Sect. 4.4.1

We note that the interviews also included a lot of free discussions, where participants expressed their opinions about the scenarios and described the way they handle changes. These parts were qualitatively analyzed and the related insights about change practices are reported in Sect. 4.4.2.

4.4 Results and Discussion

4.4.1 Quantitative Results

Table 6 summarizes the quantitative results of the study. As can be seen, in all the scenarios the approach scored much higher in the recall and F-measure than the participant's scores, before and after guidance.

For the participants, in most scenarios the precision was high (91–100%). These results show that the participants are familiar with the dependencies among elements and do not presume dependencies unless they exist. However, the values of the recall, and consequently of the F-measure, were relatively low. In particular, the recall of the initial answers (44% on average) was much lower than that achieved after guidance (60% on average).

These results suggest that despite their high level of experience, the participants could not immediately recognize the full extent of the impacts of the changes. Even after guidance, their perception of the impacts was still partial.

The precision of our approach in all the scenarios was of 100%, implying that no irrelevant elements were indicated by the algorithm. Recall, however, was lower – 79% overall, but still higher than that of the participants. The lowest recall score of Scenario #3 *(Additional resources needed)* (60%) can be explained by the fact that this specific scenario uses information external to the information system: for setting the course schedule, the participants consider constraints which are informally known but are not handled as part of the formal process and hence cannot be considered by our approach. In Sect. 4.5 we discuss the gaps between the baseline, the participants' answers and the outcomes of the approach.

From the quantitative results, we can conclude that tool support is required for such tasks and that our approach could improve the performance of experienced process participants.

4.4.2 Qualitative Results

As noted, the interviews yielded additional observations, mainly related to organizational and human aspects dealing with changes. Analysis of the answers revealed seven strategies the participants described for coping with unexpected changes: (1) reliance on the information system guidance, (2) managing the changes manually, (3) taking preventive steps to avoid future changes, (4) data

Table 6 Evaluation results

Scenario	Participants (initial answers)			Participants (after guidance)			Proposed approach		
	Prec. (%)	Recall (%)	F (%)	Prec. (%)	Recall (%)	F (%)	Prec. (%)	Recall (%)	F (%)
1. TA change	100	33	34	100	62	73	100	83	91
2. Change in course type	100	32	44	100	48	61	100	85	92
3. Additional resources needed	91	54	63	91	58	68	100	60	75
4. Unavailability of rooms	97	56	69	95	71	80	100	86	92
Average	97	44	53	97	60	71	100	79	87

Table 7 Strategies described by participants for dealing with changes

No	Strategy	Quote	Consequences
1	Reliance on the information system guidance	"*I am not sure what to do, but the system will guide me how to make the changes that need to be made. I will start with the most obvious tasks, and then I will wait for the system instructions*"	Participants do not take responsibility for addressing the change The change might be handled partly and inappropriately
2	Managing the changes manually	"*I would write the changes in a booklet, outside the system, and publish the exact dates for the students. I would not touch the system*" "*I will manage the change elsewhere. This is my kind of conservatism. I do not trust the computer. I only trust myself*"	Some changes may not be documented in the system, negatively affecting decision making in general Data quality may suffer poor integrity, leading to difficulties in operational tasks
3	Taking preventive steps to avoid future changes	"*I usually ask the lecturers to sign detailed forms, approving the agreed upon schedule before it is entered into the system; only after receiving these signed documents, we update the system. These steps are intended to prevent unexpected changes.*"	Preventive operations are time-consuming. Life may raise unexpected situations to which we cannot get ready in advance
4	Data manipulation for gaining control over the process	"*I always ask for a room, even if the course is defined as an online course*" "*When opening a course, I increase the capacity to be able to reserve larger classrooms. When students' registration is opened, I decrease the capacity to the original value. This ensures that the assigned classrooms are not too small.*" "*I will open more events than needed to reserve rooms throughout the semester in case unexpected needs are raised*"	Data quality may suffer poor integrity leading to difficulties in operational tasks Some manipulations may consume unneeded resources, negatively affecting the efficiency and effectiveness of the organization
5	Forbidding changes after a certain point in time	"*I do not approve last-minute changes unless there are substantial reasons such as illness. Otherwise, I will not approve the change*" "*I will not agree…, the lecturer will have to find someone to replace him/her while being away. Travel plans are not my concern. If the lecturer doesn't find anyone to replace him/her, I will not approve this request*"	This reduces flexibility in the process Life may raise unexpected situations
6	Waving responsibility	"*The university will have to find a solution for this. It is not my problem*" "*I won't do anything. I will ask other units to take care of this*"	Participants expect other units to handle the change, not considering the whole organization perspective
7	Deleting the entire process instance and starting again	"*How will I make the change in the system? It is simple; I would delete the course and open a new one*"	Deleting operations are time-consuming and may be error-prone

manipulation for gaining control over the process, (5) forbidding changes after a certain point in time, (6) waving responsibility, and (7) deleting the entire process instance and starting again.

Table 7 lists these strategies and provides typical statements (literally translated from the interview transcripts) in the context of these strategies. In summary, all the strategies that are taken have negative consequences in terms of data integrity, process flexibility, and overall control, and may lead to additional errors. Strategy #7 – deleting the entire process instance and starting again – was the most popular among the interviewees, whose immediate response to the different scenarios was to delete the instance. It seemed easier for participants to start all over again than to cope with the change, but this decreases productivity and may lead to additional errors.

4.5 Insights from the Quantitative and Qualitative Results

From the quantitative and qualitative analyses of the findings, we can conclude that handling unexpected changes is a difficult task even for experienced users. This task requires understanding the full extent of dependencies among process elements.

Although our participants were experienced and thoroughly familiar with the process, the results show that our

approach can significantly improve the way changes are handled. Moreover, we observed that:

(a) The less familiar the scenario is, the harder it is for participants to cope with it. In the exceptional scenario #2 (*change in course type)* and the rare scenario #3 (*additional resources needed*), the recall obtained was very low even after guidance (48% and 58%, respectively). The answers of the participants for these scenarios imply that the effects of such changes are not fully understood and hence they tend to resist and refuse to handle them. Most participants responded to these scenarios saying that they would try to avoid the change (by forbidding changes after a certain point of time or taking preventing steps to avoid future changes).

(b) When the changes take place early in the process, they are easier to handle, since commonly the number of affected elements is smaller. When a change occurs in advanced stages of the process, the participants tend to handle it by deleting the entire instance and starting again.

(c) When the corrective actions are complicated, it is harder to fully understand the consequences of the change and to handle them. Scenario #1 (*TA change*), for example, was quite a common one. However, the data change occurred in the middle of the process. Therefore, a chain of modifications was required which was not easy to fully grasp, resulting in unexpectedly low values of recall and F-measure, which improved only after guidance.

Our experience in this case study suggests that process participants tend to think in terms of activities (tasks) rather than in terms of data. For example, they stated that they need to allocate resources again, schedule rooms, and so on. Hence, we believe that providing them with an indication of activities and decision points that require their attention, as our approach does, is very important for supporting decision making regarding corrective actions. This outcome, in addition to the actual changes to the data, will guide the process participants' decision how (in terms of activities and decision points) to act for coping with the unexpected changes in data.

As to our suggested algorithm, while the recall obtained was generally high, the results, specifically for scenario #3 (*additional resources needed*), illustrate the sensitivity of the approach to the quality of the process model and especially to its completeness. Process elements that are not explicitly specified (e.g., organizational documents) negatively affect the recall and consequently the F-measure.

4.6 Threats to Validity

The threats to validity are discussed according to the categories in Wohlin et al. (2012). First, *conclusion validity,* which deals with the ability to draw a conclusion about relations between the treatment and the outcome of the evaluation, may seem limited by the small number of participants (eleven). However, these were experienced users from eight different departments, and we used four modification scenarios for each, resulting in 39 valid responses in total.

Second, with respect to *construct validity*, which concerns the ability of the measured constructs to express the relationship between theory and observation, a possible limitation is the form of the task. We asked the participants to verbalize their way of handling changes, as opposed to actually performing these actions. It can be argued that an actual performance, using the system and its forms, would be different. To address this concern, we discussed their responses with the participants and allowed modifications of the initial answers ('after guidance').

Third, considering *internal validity,* which addresses factors that may affect a dependent variable, we note that differences in performance among the scenarios may be due to a learning curve. Since the order in which the scenarios were presented was the same for all participants, their performance might have improved from scenario to scenario due to learning. Indeed, the average scores for the fourth scenario were better than for the first scenario, but the average recall and F-measure for the second and third scenarios were worse than for the first one. In any case, our focus was on the comparison to the baseline rather than between scenarios.

Fourth, for *external validity,* which concerns the possible generalization of the results, we note that we examined only one process in one specific organization. Yet, the participants were from different departments, exposed to variants of the process and different departmental policies. In addition, the selection of this process, described in Sect. 4.1.2, followed certain criteria that increase the external validity of the study, including the ability to recruit highly experienced participants.

5 Related Work

Here we review work in three related areas: (a) data-aware process modeling, which is the basis for any attempt to analyze the impact of data on process elements, (b) process modifications in design and run-time, and (c) change impact analysis in general, and particularly in business processes.

Data-aware process modeling Many attempts have been made to address the relations between processes and data, sometimes referred to as "duality of control flow and data flow" (Kumar 2018). The importance of integrating data and process flows, and the increasing number of studies on this topic, have led to several surveys that consolidate and compare studies in this area (Meyer et al. 2011; Reijers et al. 2017; Steinau et al. 2019). Meyer et al. (2011) claim that processes and data are equally important for business process management. They evaluated 12 process modeling languages that consider data, according to 23 criteria related to capabilities of flow modeling, data modeling, connection between control flows and data, and execution semantics. They conclude that none of the studied languages support all criteria. Moreover, they state that in most of the approaches only basic principles such as the representation of data objects are supported, while complex principles such as dependencies and interrelations between data objects are neglected.

The evaluation in Reijers et al. (2017) addresses 14 data-centric process approaches. The approaches are categorized into three main groups: (1) Data-driven process structures (e.g., Müller et al. 2007, 2008), in which the objects involved in the process are described using a data-model (each object has a lifecycle to define its states); (2) Product-based workflow design (e.g., Reijers et al. 2003; Vanderfeesten et al. 2011), in which a model describes the data elements involved in the process in form of tree-like structure (the top elements in the tree are the final data elements that represent the outcome of the process); and (3) Artifact-centric process modeling (e.g., Bhattacharya et al. 2009; Cohn and Hull 2009), which focuses on describing how business data (artifacts) are changed by a particular action, task, or service throughout the process (the states of an artifact are described by a lifecycle model, and an information model of the process describes all business artifacts and their relationships). The authors conclude that although there is a significant increase in developing data-centric process modelling approaches, not much attention is given to the individual needs (i.e., users' perspective usability).

When discussing and introducing data-centric process modeling approaches, Kumar (2018) mainly focuses on product-based and artifact centric approaches. These and others (e.g., Bhattacharya et al. 2009 and Sadiq et al. 2004) tie process modeling to analysis approaches that address the data perspective and take measures to avoid data flow problems. These approaches are fundamentally centered on data rather than on control flows. They support the design of process models with data but do not aim to analyze data impacts as we do. In most studies, the granularity level is of data objects, which we indicate to be too coarse to support a detailed data-centric analysis. Several studies, such as Reichert (2012), Ryndina et al. (2006), and Meyer and Weske (2013), further represent the state of data objects along the process flow, referring to a specific attribute (e.g., "status"), which changes its value during process execution to reflect the progress in the process. Meyer et al. (2013) refer to the structural relations between data objects, as specified in the database schema, to enable automated execution of process models. The approach extends BPMN and combines concepts of relational data modeling but does not refer to the individual attributes (data items).

Data flow is discussed also in the context of workflow nets, e.g., in Sidorova et al. (2011) WFD-nets (workflow-nets with data) are suggested. The main purpose of adding the data flow to workflow-nets is to check the correctness of the data flow in the design phase. The model considers data elements at a granularity level that can be used in transitions' preconditions.

The gap between the business process models and underlying databases, especially when dealing with data integrity constraints that are not handled by the process model, is addressed by van der Aalst et al. (2005) and Lin and Sadiq (2010). This study proposes to use concepts of data integrity management through data dependency constraints in business process models. The paper introduces Conditional Data Dependency (CDD) which is used to define business rules to ensure data integrity through the process layer to the data layer. The data constraints can be modeled at the process model level. However, these dependencies are kept in a table form and translated into DBMS procedures. The constraints are checked in runtime in the database to ensure their enforcement.

Verification of process models has been widely addressed with respect to the control flow perspectives, and in some cases also extended to consider the data perspective, in combination with the control flow perspective or separately (Kumar 2018). Sun and Zhao (2013) and Sun et al. (2006) propose a workflow design approach based on dependencies among activities and their associated data. They specifically refer to data items and define a set of data dependencies. Their approach supports the design of workflow models but does not address impact analysis. The soundness property, which is commonly used for control flow, is extended in the context of workflow nets with data (WFD-nets) in Sidorova et al. (2011). The model considers data elements at a granularity level that can be used in in the preconditions of transitions. For the same modeling formalism, a set of "anti-patterns" supporting the detection of data flow errors in a process model are proposed by Trčka et al. (2009) and von Stackelberg et al. (2014). These patterns take dependencies among elements into account and allow verification of the control flow and the data flow at design time.

The metamodel we propose here refines data-related concepts and dependencies systematically, compared to existing models, and supports a finer-grained analysis of data impact in business processes.

Process modifications Over the years, several studies proposed techniques and heuristics for changing processes at design time (e.g., Reijers and Mansar 2005; Weber et al. 2008; Reichert and Weber 2012). Weber et al. (2008) propose 17 change patterns for processes. However, these patterns do not address data aspects and mainly refer to control flows. Reichert and Weber (2012) refer to related data problems, e.g., missing data, unnecessary data, and lost data, when changing a process. Their approach does not consider indirect impacts of the data and takes into account only local changes.

To handle process modifications at runtime, while considering data-centric models, case handling (van der Aalst et al. 2005) and product-based workflow design (Reijers et al. 2003; Vanderfeesten et al. 2011) were proposed. Case handling usually allows users to change the process at runtime and enables flexibility of the process while avoiding changes of the process model, e.g., by generating implicit alternative paths. A variety of mechanisms is defined to allow implicit deviations. Activities are described as forms in relation to atomic data elements. Data elements can either be free (not associated with particular activities), mandatory (required for completing the corresponding activity) or optional. An activity is considered as completed if all associated mandatory data elements have an assigned value. Free data elements are assigned to the process model and can be changed at any point in time by all users. However, only atomic data elements are provided. Data integration in terms of inter-relations is not considered, and the granularity of data objects is too coarse-grained for data impact analysis. Product-based workflow design is a data-centric approach to workflow specification and (re)design, which uses product data models that describe all data elements and their dependencies in a tree-like structure with nodes representing data elements and edges representing functional dependencies between them. Actions are located on edges between data nodes, generating new data values from the existing ones in a bottom-up manner. Production rules define which information of which elements should be combined to obtain other elements. The approach provides insights into essential aspects of the (re)designed process and its data, however, it does not consider the impact of data, especially not in terms of the business process.

In Russell et al. (2006) exception handling patterns in process-aware information systems were suggested to describe different ways for coping with exceptions that may occur during process execution. Patterns such as Rollback are defined to deal with exceptional situations by changing the state of a running process. However, they do not consider the full impact of the change.

A recent study (Andrews et al. 2018) presents concepts for supporting ad-hoc changes in object-aware processes; in particular, the paper presents seven requirements for handling such changes. The approach, supported by the PHILharmonicFlows framework and tool (Künzle and Reichert 2011), realizes an object and process aware information system. This approach refers to various components of processes, such as objects, relations, and coordination processes, and provides user assistance to cope with ad-hoc changes (including data values, but also changes in the process and its activities). However, this assistance depends on the existence of a data model and state definitions for each data object, which exist as part of the PHILharmonicFlows framework. In contrast, our approach is currently conceptual and not tied to a specific process execution framework. The study reported in Steinau et al. (2019) is intended to gain profound insights into the maturity of different data-centric approaches as well as their capabilities. The authors propose a framework for systematically evaluating and comparing data-centric approaches, throughout the whole business process lifecycle. The framework is applied to 38 studies belonging to three approaches: case handling, the artifact-centric and the object-aware approach. The authors further conclude that most data-centric approaches do not support ad hoc changes properly.

Change Impact analysis Change Impact Analysis (abbreviated as IA) has been addressed in the area of software engineering and is concerned with analyzing and assessing the consequences of a change in software systems (Alam et al. 2015). IA techniques are used to predict the impact of changes (to, e.g., architecture, source code, or requirements) before applying them, and to help evaluate the effect of the changes. IA has also been studied to some extent in the field of information systems, including business processes. In this context, IA has mainly been addressed for control flow changes (Zhou et al. 2008; Dai et al. 2009; Bouchaala et al. 2014; Kherbouche et al. 2013; Soffer 2005). Soffer (2005) discusses IA in business processes, referring to different kinds of changes including data values; however, no method for analyzing these impacts is provided.

A common technique in IA is dependency analysis (Alam et al. 2015), which is commonly performed on graphs or tables (Dai et al. 2009). The studies of Dai et al. (2009), Wang et al. (2010) and Bouchaala et al. (2014) define several dependency types between process elements for analyzing the impact of change while considering elements in the same process. Dependency graphs should be built for each process separately. While addressing data dependencies, these studies do not consider impacts of data

items on routing constraints or other data items. For a similar purpose, our approach suggests a generic representation and primitives that represent the dependencies in the process.

A resemblance exists between the database rollback mechanism (Elmasri 2008) and our approach. However, when a transaction fails and rollback occurs at the database level, changes are not saved in the database, and the rollback mechanism returns the database to the previous state. In business processes, (unexpected) changes may occur after writing values to the database and/or executing additional activities (e.g., by different actors). Furthermore, in our approach we do not rollback the process; we only retrieve the places in the process on which the change has an impact, leaving the decision whether to conduct corrective actions to the process participants.

Related approaches can also be found in relation to Complex Event Processing (CEP) (Hermosillo et al. 2010; Soffer et al. 2017), which deals with the real-time analysis of events using event streams. The study of Krumeich et al. (2014) which analyses the current progress in the area of CEP distinguishes six clusters of research in this area. One of the clusters is flexible process adaptation, including approaches which utilize CEP for detecting exceptional cases that require adaptations at runtime.

In Pufahl et al. (2017), a combination of CEP and BPMN is proposed to support re-evaluation of decision points at runtime using event processing techniques. A re-evaluation scope is formalized by using the BPMN semantics and can be interrupted by an event. Furthermore, queries are generated dynamically at run-time based on the actual decision output and the decision logic to consider only those events which trigger a different decision output. Thus, decisions can be re-evaluated only until a certain point. To support this idea, it is required to identify until which point the actions following a certain decision can be canceled or rolled back without severe consequences. This study considers only updates that lead to a different decision output.

Sid et al. (2019) observe that when flexibility is granted to process participants, it is typically not accompanied by appropriate user guidance in the respective supporting information systems. Therefore, they present a data centric approach that uses AI planning techniques in order to guide user decisions in highly flexible knowledge-intensive processes (KIP). The proposed approach allows capturing data inputs and outputs and relating them to process tasks, as a basis for planning, and thus bears similarity to our impact analysis approach. However, the process elements that are taken into account neglect routing constraints and structural data dependencies, which our approach considers.

In summary, the literature related to impact analysis does not focus directly on the impacts of data changes and does not consider all its possible forms. To fill this gap, our approach focuses on the impact analysis of data, attempting to provide a systematic approach for analyzing both direct and indirect data impacts and supporting users in handling them at runtime.

6 Conclusion and Future Work

This paper addresses a common, yet difficult-to-handle situation of unexpected changes in data values in business processes. Usually, such changes cannot be addressed locally, as due to dependencies their impact may stretch across the entire business process. In this paper, we consider the process participant perspective, extending a previous study and evaluating the approach by means of a real business process with highly experienced process participants. We analyzed the results using quantitative and qualitative analyses. We showed that the full impact of value changes is not grasped by experienced participants. Furthermore, attempting to reduce the required effort, participants typically employ informal strategies that may harm data integrity and decision making. The strategies were classified, and their possible consequences were described.

Implications of the approach are relevant to both practice and research. In practice, unexpected data changes can occur for various reasons, including data errors, last-minute changes, and exceptional situations. If addressed incorrectly or partly, this may lead to severe consequences for data integrity, decision making, and business achievements. Since unexpected data changes are mostly not addressed by formal process models, a tool-supported approach to guide the handling of such situations in a generic manner should be valuable. Using our approach at runtime when facing unexpected data changes, process participants will understand the consequences of the immediate data changes and be guided through the implied chain of changes if needed.

A contribution to research is the identification of the process elements affected by an unexpected change in data. For this, we refine existing business process metamodels by (a) emphasizing direct and indirect data relations, (b) incorporating data items rather than objects, (c) relating specific data item inputs to specific outputs through activities, and (d) referring to structural relations between data items. This refined metamodel can support future research of data-aware business processes. Furthermore, the concept of data impact analysis can promote research directions in the business process risk management and data quality areas. It can also explain unpredicted behavior observed in event logs in the area of process mining.

The approach has also some limitations. First, it examines the impact of data in a single instance of a single process, thus it is unable to trace impacts across instances of the same process or across processes. We consider it a starting point that should be further developed in this direction. Second, additional model elements such as events and advanced gateways types are not part of the current metamodel. However, it can be extended to support these and more. Third, the approach is sensitive to the quality and completeness of the model. This should be considered when building the infrastructure. Last, a set-up of transformation from a process model to the relational database has only a partial tool support, since not all the required information is usually available. However, it is a one-time effort that can be used through the lifecycle of the process. Maintenance of this database is only necessary as a part of redesign and not on a daily basis.

Future research directions would extend the analysis to more expressive process models, considering elements such as events, as well as to impacts across processes and process instances. Moreover, we would like to address additional perspectives such as resources and roles. Another future direction is to explore and propose a way to integrate our approach into existing Business Process Management tools and systems, such as Camunda[4] and ProcessMaker.[5] Such an integration will support the analysis and the exploration of the value change impact when defining and testing the workflow. Moreover, it will enable monitoring the process at runtime when the changes occur. Finally, it will be possible to utilize such an integrated tool for conducting extensive usability evaluations of the user support proposed here.

Acknowledgements This research is supported by the Israel Science Foundation under Grant 856/13.

References

Alam KA, Ahmad R, Akhunzada A, Nasir MHNM, Khan SU (2015) Impact analysis and change propagation in service-oriented enterprises: a systematic review. Inf Syst 54:43–73

Andrews K, Steinau S, Reichert M (2018) Enabling ad-hoc changes to object-aware processes. In: 2018 IEEE 22nd international enterprise distributed object computing conference, Stockholm. IEEE, pp 85–94

Bhattacharya K, Gerede C, Hull R, Liu R, Su J (2007) Towards formal analysis of artifact-centric business process models. In: International conference on business process management, Brisbane. Springer, Heidelberg, pp 288–304

Bhattacharya K, Hull R, Su J (2009) A data-centric design methodology for business processes. In: Handbook of research on business process modeling. IGI Global, pp 503–531

Bouchaala O, Yangui M, Tata S, Jmaiel M (2014) DAT: Dependency analysis tool for service based business processes. In: 28th international conference on advanced information networking and applications, Victoria. IEEE, pp 621–628

Chen PPS (1976) The entity-relationship model—toward a unified view of data. ACM Trans Database Syst 1(1):9–36

Cohn D, Hull R (2009) Business artifacts: a data-centric approach to modeling business operations and processes. IEEE Data Eng Bull 32(3):3–9

Dai W, Covvey D, Alencar P, Cowan D (2009) Lightweight query-based analysis of workflow process dependencies. J Syst Softw 82(6):915–931

Dam HK, Ghose A (2015) Mining version histories for change impact analysis in business process model repositories. Comput Ind 67:72–85

Elmasri R (2008) Fundamentals of database systems. Pearson Education India, Chennai

Hammer M (2015) What is business process management? In: vom Brocke J, Rosemann M (eds) Handbook on business process management 1. International handbooks on information systems. Springer, Berlin, Heidelberg

Hermosillo G, Seinturier L, Duchien L (2010) Using complex event processing for dynamic business process adaptation. In: IEEE international conference on services computing, Miami. IEEE, pp 466–473

Kherbouche OM, Ahmad A, Bouneffa M, Basson H (2013) Ontology-based change impact assessment in dynamic business processes. In: 11th international conference on frontiers of information technology, Islamabad. IEEE, pp 235–240

Krumeich J, Weis B, Werth D, Loos P (2014) Event-driven business process management: where are we now? A comprehensive synthesis and analysis of literature. Bus Process Manag J 20(4):615–633

Kumar A (2018) Business process management. Routledge, Abingdon

Künzle V, Reichert M (2011) PHILharmonicFlows: towards a framework for object-aware process management. J Softw Maint Evol Res Pract 23(4):205–244

Lin JYC, Sadiq S (2010) A business process driven approach to manage data dependency constraints. In: International conference on enterprise information systems, Funchal, Madeira. Springer, Heidelberg, pp 326–339

Meyer A, Pufahl L, Fahland D, Weske M (2013) Modeling and enacting complex data dependencies in business processes. In: Daniel F, Wang J, Weber B (eds) Business process management. Lecture notes in computer science, vol 8094. Springer, Berlin, Heidelberg

Meyer A, Smirnov S, Weske M (2011) Data in business processes (No. 50). Universitätsverlag, Potsdam

Meyer A, Pufahl L, Fahland D, Weske M (2013) Modeling and enacting complex data dependencies in business processes. In: Business process management. Springer, Heidelberg, pp 171–186

Müller D, Reichert M, Herbst J (2007) Data-driven modeling and coordination of large process structures. In: OTM confederated international conferences "on the move to meaningful internet systems", Vilamoura. Springer, Heidelberg, pp 131–149

Müller D, Reichert M, Herbst J (2008) A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: International conference on advanced information systems engineering, Montpellier. Springer, Heidelberg, pp 48–63

Pufahl L, Mandal S, Batoulis K, Weske M (2017) Re-evaluation of decisions based on events. In: Enterprise, business-process and information systems modeling. Springer, Cham, pp 68–84

Reichert M (2012) Process and data: two sides of the same coin? In: OTM confederated international conferences "on the move to

[4] https://camunda.com.

[5] https://www.processmaker.com.

meaningful internet systems", Rome. Springer, Heidelberg, pp 2–19
Reichert M, Weber B (2012) Enabling flexibility in process-aware information systems: challenges, methods, technologies. Springer, Heidelberg
Reijers HA, Mansar SL (2005) Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega 33(4):283–306
Reijers HA, Limam S, van der Aalst WM (2003) Product-based workflow design. J Manag Inf Syst 20(1):229–262
Reijers HA, Vanderfeesten I, Plomp MG, van Gorp P, Fahland D, van der Crommert WL, Garcia HDD (2017) Evaluating data-centric process approaches: does the human factor factor in? Softw Syst Model 16(3):649–662
Rinderle S, Reichert M (2006) Data–driven process control and exception handling in process management systems. In: International conference on advanced information systems engineering, Luxembourg. Springer, Heidelberg, pp 273–287
Rodríguez A, Caro A, Cappiello C, Caballero I (2012) A BPMN extension for including data quality requirements in business process modeling. In: International workshop on business process modeling notation, Vienna. Springer, Heidelberg, pp 116–125
Russell N, van der Aalst WM, ter Hofstede AH (2006) Exception handling patterns in process-aware information systems. BPM Center Report BPM-06-04, BPMcenter.org, 208
Ryndina K, Küster JM, Gall H (2006) Consistency of business process models and object life cycles. In: International conference on model driven engineering languages and systems, Genova. Springer, Heidelberg, pp 80–90
Sadiq S, Orlowska M, Sadiq W, Foulger C (2004) Data flow and validation in workflow modelling. In: Proceedings of the 15th Australasian database conference, vol 27, Dunedin. Australian Computer Society, pp 207–214
Sid I, Reichert M, Ghomari AR (2019) Enabling flexible task compositions, orders and granularities for knowledge-intensive business processes. Enterp Inf Syst 13(3):376–423
Sidorova N, Stahl C, Trčka N (2011) Soundness verification for conceptual workflow nets with data: early detection of errors with the most precision possible. Inf Syst 36(7):1026–1043
Soffer P (2005) Scope analysis: identifying the impact of changes in business process models. Softw Process Improv Pract 10(4):393–402
Soffer P (2010) Mirror, mirror on the wall, can I count on you at all? Exploring data inaccuracy in business processes. In: Gulden J et al (eds) Enterprise, business-process and information systems modeling, Hammamet. Springer, Heidelberg, pp 14–25
Soffer P, Hinze A, Koschmider A, Ziekow H, Di Ciccio C, Koldehofe B, Kopp O, Jacobsen A, Sürmeli J, Song W (2017) From event streams to process models and back: challenges and opportunities. Inf Syst 81:181–200. https://doi.org/10.1016/j.is.2017.11.002
Steinau S, Marrella A, Andrews K, Leotta F, Mecella M, Reichert M (2019) DALEC: a framework for the systematic evaluation of data-centric approaches to process management software. Softw Syst Model 18:2679–2716. https://doi.org/10.1007/s10270-018-0695-0
Sun SX, Zhao JL (2013) Formal workflow design analytics using data flow modeling. Decis Support Syst 55(1):270–283
Sun SX, Zhao JL, Nunamaker JF, Sheng ORL (2006) Formulating the data-flow perspective for business process management. Inf Syst Res 17(4):374–391
Trčka N, Van der Aalst WM, Sidorova N (2009) Data-flow anti-patterns: discovering data-flow errors in workflows. In: International conference on advanced information systems engineering, Amsterdam. Springer, Heidelberg, pp 425–439
Tsoury A, Soffer P, Reinhartz-Berger I (2016) Towards impact analysis of data in business processes. In: Schmidt R, Guédria W, Bider I, Guerreiro S (eds) Enterprise, business-process and information systems modeling. BPMDS 2016, EMMSAD 2016. Lecture Notes in Business Information Processing, vol 248. Springer, Cham
van der Aalst WM, ter Hofstede AH, Kiepuszewski B, Barros AP (2003) Workflow patterns 14(1):5–51
van der Aalst WM, Weske M, Grünbauer D (2005) Case handling: a new paradigm for business process support. Data Knowl Eng 53(2):129–162
Vanderfeesten I, Reijers HA, van der Aalst WM (2011) Product-based workflow support. Inf Syst 36(2):517–535
von Stackelberg S, Putze S, Mülle J, Böhm K (2014) Detecting data-flow errors in BPMN 2.0. Open J Inf Syst 1(2):1–19
Wang Y, Yang J, Zhao W (2010) Change impact analysis for service based business processes. In: IEEE international conference on service-oriented computing and applications, Perth. IEEE, pp 1–8
Weber B, Reichert M, Rinderle-Ma S (2008) Change patterns and change support features—enhancing flexibility in process-aware information systems. Data Knowl Eng 66(3):438–466
Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer, Heidelberg
Zhou Z, Bhiri S, Hauswirth M (2008) Control and data dependencies in business processes based on semantic business activities. In: Proceedings of the 10th international conference on information integration and web-based applications & services, Linz. ACM, pp 257–263