Bus Inf Syst Eng 64(2):183–200 (2022)
https://doi.org/10.1007/s12599-021-00709-9

RESEARCH PAPER

Model-based Analysis of Data Inaccuracy Awareness in Business Processes

Yotam Evron · Pnina Soffer · Anna Zamansky

Received: 12 January 2020 / Accepted: 27 April 2021 / Published online: 28 July 2021

Abstract Problem definition: Data errors in business processes can be a source for exceptions and hamper business outcomes. Relevance: The paper proposes a method for analyzing data inaccuracy issues already at process design time, in order to support process designers by identifying process parts where data errors might remain unrecognized, so decisions could be taken based on inaccurate data. Methodology: The paper follows design science, developing a method as an artifact. The conceptual basis is the notion of data inaccuracy awareness – the ability to tell whether potential discrepancies between real and IS values may exist. Results: The method was implemented on top of a Petri net modeling tool and validated in a case study performed in a large manufacturing company of safety–critical systems. Managerial implications: Anticipating consequences of data inaccuracy already during process design can help avoiding them at runtime.

Keywords Business process management · Data quality · Model-based analysis

Accepted after two revisions by Jörg Becker.

Y. Evron (✉) · P. Soffer · A. Zamansky
University of Haifa, Mount Carmel, 3498838 Haifa, Israel
e-mail: yevron@is.haifa.ac.il

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

A. Zamansky
e-mail: annazam@is.haifa.ac.il

1 Introduction

Business processes form a core operational asset in organizations, as they coordinate the execution of activities which use and manipulate resources and data for achieving business goals. The design of business processes and their supporting information systems (IS) considers the data used by the process as an accurate reflection of reality. With this assumption, actors do not need to physically sense real values for deciding what activity to perform at a given moment and how to perform it. Rather, they can rely on the data stored in the IS. However, this assumption is not always realistic, and situations occur where data values differ from real values they should reflect; these bear substantial risks for the process and for business goals. Such situations, termed data inaccuracy situations, may lead to decisions that are based on incorrect information, imposing risks of not reaching the defined goals or other negative consequences. Since the values of data used in a process have such an impact on business goals, early detection of data errors is essential for avoiding negative consequences.

So far, data inaccuracy has mainly been considered and addressed in the business process management literature as a form of a *runtime exception*, to be dealt with by exception handling mechanisms (Russell et al. 2006a, b). This paper postulates that some of the negative consequences of data inaccuracy can be avoided if potential data inaccuracies are analyzed and anticipated already at *design time*. Indeed, the earlier we can identify potential data failures before executing the process, the better we can address and handle them, avoiding potential effects on crucial decisions in the process.

The main research question addressed in this paper is how to identify at design time process parts where the existence of data errors might remain unrecognized so decisions may be taken based on incorrect data and negative consequences are likely to materialize. Such analysis can guide process designers to modify the process design, to achieve early detection of data inaccuracy at runtime and thus to reduce its implications. To the best of our knowledge, only a few systematic design-time approaches (Marrella et al. 2018; Plebani et al. 2017), based on a process model, have been proposed so far for this purpose.

The paper proposes a tool-supported approach for a model-based analysis of potential data inaccuracy situations when a business process is executed. Its fundamental concept is *data inaccuracy awareness* (*DIA*), i.e., whether at a given moment in time one can be sure that the value of a data item in the IS matches the real-world value it should reflect. The idea is that when a data item is used without such awareness, this may negatively affect the execution of the process and the taken decisions, and lead to poorer business performance. Identifying the points where such awareness is not guaranteed holds the key to our design-time analysis.

The main contributions of this paper consist of proposing a new set of rigorously defined concepts which capture the manifestation of data inaccuracy in processes and form the basis for a design time automated analysis approach. This approach can help to identify potential consequences of data inaccuracy in early stages, thus aiding the process analysts in reducing potential data inaccuracy risks.

The remainder of the paper is organized as follows: Sect. 2 introduces preliminary notions and positions the problem addressed. A rigorously defined framework is presented in Sect. 3, as a basis for the analysis approach and algorithms, presented in Sect. 4, and its operationalization in business terms (Sect. 5). In Sect. 6, we evaluate the proposed approach by means of a case study in an industrial setting. Section 7 presents a discussion of our approach and its limitations. Section 8 discusses related works. Finally, Sect. 9 draws the conclusion and discusses future work.

2 Foundations

2.1 Background

This section presents the fundamental definitions of concepts and notations used later. Although we use a Petri net with data (DPN) formalism (de Leoni et al. 2018; Mannhardt et al. 2014) as our basic representation of process models, the definitions and concepts which are employed later are generic and can be adapted to any other process model (such as YAWL, CPN, BPMN, etc.).[1] Note that standard Petri-net and Workflow-net concepts, properties, and notations are not given here, and can be found in van der Aalst (1996, 1998) and Sidorova et al. (2010). A DPN is a Petri net enriched with conceptual *write* data operations that are assigned to transitions and transition guards represented by logical expressions over data, which monitor transition execution. More formally,

Definition 1 (*DPN*) (de Leoni et al. 2018; Mannhardt et al. 2014) A Petri net with data (DPN) N = (P, T, F, D, V, Val, W, G) consists of:

- a set of places P;
- a set of transitions T;
- a flow-relation $F \subseteq (P \times T) \cup (P \times T)$;
- a set D of data items;
- a set V of data item values;
- a function Val: $D \rightarrow 2^V$ defining the values admissible for each data item $d \in D$;
- a write function W: $T \rightarrow 2^D$ that labels each transition with a set of write operations;
- a guard function G: $T \rightarrow$ Formulas $(D \cup \{d' | d \in D\})$ that associates each transition with a guard formula.

Our approach targets DPNs that are well-structured workflow nets (Mannhardt et al. 2014), i.e., they have one initial place and one final place, and are composed of blocks whose entry and exit nodes are of the same type (either places or transitions). Well-structured workflow nets exist in a broad set of process models and hence this does not substantially limit the generality of the approach. Moreover, in many cases, it is possible to transform unstructured process models into well-structured ones (van der Aalst and Gunther 2007; Polyvyanyy et al. 2012).

To handle data inaccuracy, we complement DPN with concepts of the Generic Process Model (GPM) (Soffer et al. 2010; Soffer and Wand 2004). According to this, a process takes place in a domain, typically depicted as a set of state variables, whose values at a given moment in time reflect the domain state at that moment. A process is viewed as a sequence of state transitions of the domain, which are governed by a transformation law. IS, which typically support business processes, are considered as part of the process domain. They encompass data items, which correspond to state variables by reflecting their values.

Definition 2 (*Domain and Sub-domain representation*) A domain ***DM*** is a part of the world, represented by a set of state variables $\boldsymbol{SV} = \{x_1, x_2, \ldots, x_n\}$. A sub-domain is a subset of ***DM***.

[1] Note, while the abstract concepts are generic, the algorithms are specifically designed for DPN.

Definition 3 (*Corresponding couple*) Let ***DM*** be a domain, whose set of state variables is ***SV***. ***SV*** is reflected in the IS as a set of data items $\boldsymbol{D} = \{d_1, d_2, \ldots, d_n\}$, where $\boldsymbol{d_i}$ reflects $\boldsymbol{x_i}$. We call $\langle \boldsymbol{x_i}, \boldsymbol{d_i} \rangle$ a *corresponding couple*.

The common assumption when operating business processes is that data item values correctly reflect state variable values. Removing this assumption leads us to explore the phenomenon of data inaccuracy.

Definition 4 (*Data inaccuracy*) We say that a domain is accurately reflected by an IS at a given moment $\boldsymbol{t}$ if for every i the values of the corresponding couple $\langle \boldsymbol{x_i}, \boldsymbol{d_i} \rangle$ are equal. Any violation of this condition for $\boldsymbol{d_i}$ is termed data inaccuracy with respect to $\boldsymbol{d_i}$ (at $\boldsymbol{t}$).

Note that we assume the same granularity level for each corresponding couple (data items and state variables). Moreover, we assume the process model can handle all the expected execution paths (well-behaved cases) (Kiepuszewski et al. 2003; Russell et al. 2006a). In other words, in case a mismatch was discovered between x_i and d_i, this discrepancy can be handled according to the process owner policy as part of the process.

2.2 Running Example

To demonstrate the concepts related to data inaccuracy as well as its consequences, consider a company which provides technical service to equipment at the customer's site. When a customer requires service, a service order is created and a technician is sent to his address (Fig. 1).

Consider the following two scenarios: (a) the customer's address is recorded incorrectly, then the technician will not be able to provide the required service. (b) The serial number of the product for which service is required is incorrectly reported, then the technician might arrive at the right place but with the wrong test equipment and may provide service to the wrong product (assuming the customer has several products that can require service, the error might only be discovered later). Thus, different data inaccuracy situations may bear different consequences. These examples also raise essential issues such as the point at which the inaccuracy is discovered and where can the source of the mismatch be identified, assuming that the earlier we notice the discrepancy the earlier we can handle it and save valuable resources. Note that although all data items could be examined continuously as their values change, this may not be necessary, feasible or cost-effective (Bovee et al. 2003) (in terms of money, time, effort, etc.).

2.3 The Problem Space

Data inaccuracy is a situation where a data item value does not reflect correctly the (real world) state variable value, i.e., $x_i \neq d_i$. As our aim is to analyze data inaccuracy at design time, we note that such an analysis is not feasible for every type of data item. Hence, we characterize the data items used by IS along the following dimensions (illustrated in Fig. 2). First – the context in which the value of the data item d_i can be updated at a given moment, which can range from an update that can occur: within a single process instance to one in different parallel instances of the same process or in different parallel processes and instances. Second – the stability degree of the value of the state variable x_i – which can range from completely stable and controlled values (e.g., customer ID) to values that are stable and uncontrolled (e.g., customer's address) and lastly to unstable and uncontrolled values which means

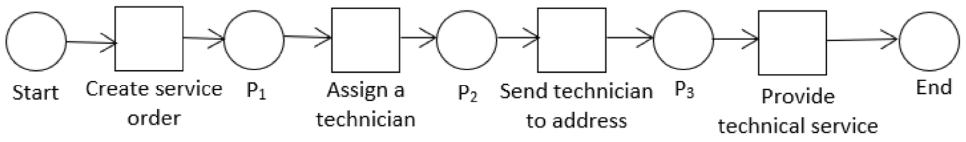

Fig. 1 A DPN (excluding data operations for the sake of simplicity) of the service provider's sub-domain

Fig. 2 Scoping the problem space

Stability of the state variable (x_i) / Updating context of the data item (d_i) at a given moment	Stable and controlled	Stable and uncontrolled	Unstable and uncontrolled
In different parallel processes and instances (global)	■	■	■
In different parallel instances of the same process	░	▓	■
Within a single process instance (local)		░	■

they change constantly (e.g., temperature, blood pressure). Stable and controlled values (Soffer et al. 2010) mean that any change in the value is a result of a suitable action in the process. Figure 2 also provides an overview of all the discussed dimensions. Note that such a classification is not absolute and should be adapted to the specific domain.

Figure 2 contains three main types of areas, each one indicating a different case: the "white" area is where the state variable value is relatively stable and the data update is performed within a single process instance. The "grey" area has two shades: the brighter one means there is a higher certainty of the value of the data item at runtime. We further elaborate on where in these areas our analysis can be applied. The "black" area is where the state variable value is unstable and uncontrolled, or where the data can be updated by parallel processes and instances. In such cases, we cannot assume that the data value is known as a basis for decision making.

Our analysis relies on the assumption that once a correspondence in the values of $\langle x_i, d_i \rangle$ is established, it can be expected to hold, unless changes are made to the values within the analyzed process. This assumption is reasonable for $\langle x_i, d_i \rangle$ couples in the "white" area in Fig. 2. An example would be the ordered price or the ordered product ID in an order fulfillment process.

Considering the different "grey" areas, an analysis can still be feasible under certain conditions. First, consider stable and controlled variables whose data items are updated globally. If the number of updates that can take place in parallel is restricted, then the data item can be treated as locally updated. For example, take quantity in stock, which can generally be updated by many process instances that use the same product. Still, if products are uniquely related to customers and orders are placed periodically, no two process instances that run in parallel can change the quantity in stock of the same product, so it in fact behaves like a locally updated data item.

Second, consider a state variable whose value is not controlled, namely, it can change in a manner which is uncontrolled by the process, but in practice changes do not take place often, which means that it is relatively stable. For example, a customer's address can change and is uncontrolled by the process. However, the customer is expected to notify the company about such a change if it happens during an ordering process, so in fact the data item value is supposed to reflect the real value.

In summary, the proposed analysis targets all $\langle x_i, d_i \rangle$ in the white area as well as parts of the grey area in Fig. 2. Furthermore, for any given data item it is possible to establish whether the analysis is applicable or not.

3 A Formal Framework for Data Inaccuracy

The Generic Process Model (GPM) views a process as occurring in a domain, captured by a set of state variables, whose values at a given moment reflect the domain state at that moment. A process can be thought of as a sequence of state transitions of the domain. These transitions rely on a subset of state variables as a decision base, and thus we may relate to subsequences of decision bases. Given two subsequences, if there is no overlap between the decision bases in both sequences, then the sub-domains that hold these subsequences are independent.

While there are (infinitely) many possible choices of how to split a domain into sub-domains, in our approach we focus on a specific decomposition into two sub-domains. One sub-domain, which we call process sub-domain, is where an IS-managed process takes place and decisions are made relying on the data items in the IS (i.e., the transformations in this sub-domain are IS-dependent while assuming the state of the world based on its reflection in IS data). The second sub-domain, which we call the external (or environment) sub-domain, holds and acts upon values of real-world state variables.

Consider the running example described above and assume that the company received a message and created a service order. From this point on, it can assign and send a technician on its own, in its process sub-domain based on the IS data, without relying on any information from the external world in order to proceed. In fact, the process, as modeled in Fig. 1, is based on a *closed world* assumption, namely, the environment is assumed to behave fairly (van der Aalst 2000). Figure 3 shows a broader scope "bird-view" of the process, decomposed into two sub-domains (process and environment sub-domains). None of the sub-domains has access to this "bird view" and each one of them can only see its own inner activities, data items and values. Note there is an unmodeled part of the process where the customer uses and maintains the equipment without interacting with the service provider. In this part of the process, the sub-domains employ independent (and concurrent) threads. More specifically, the environment contains all parts which are left outside of the process scope. For example, the department inserts a description of the equipment's faults. If there is an error in the description, the company will discover this error only when technical service should be provided.

Note that the two sub-domains shown in Fig. 3 operate concurrently and synchronize at 'Provide technical service'. Unlike the standard notion of *synchronization* (where several threads in a *model* converge), here *synchronization* refers to an unmodeled thread (sub-domain), outside of the process control. When a synchronization involves an external, uncontrolled sub-domain and a sub-domain

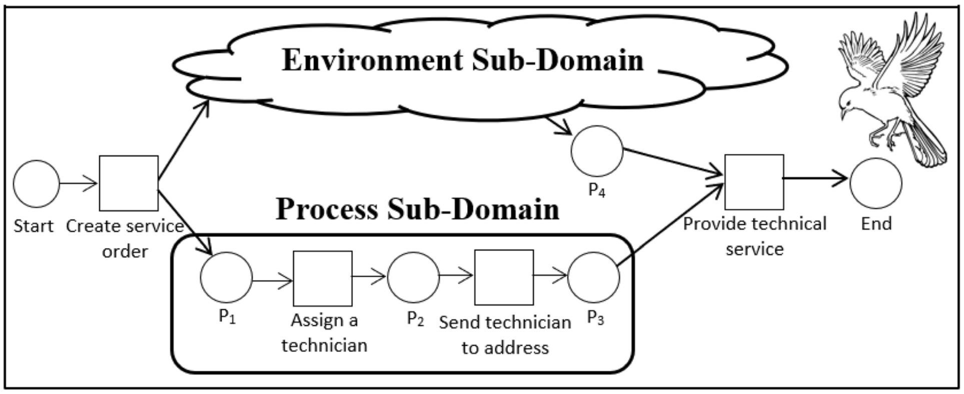

Fig. 3 A higher-level view of the two independent sub-domains

controlled by the process, an important concern is to ensure that the values of the IS data items and their corresponding state variables are equal.

Definition 5 (*External Synchronization Point* Evron et al. 2017a, b) Let $\langle \boldsymbol{x_i}, \boldsymbol{d_i} \rangle$ be a corresponding couple. An external synchronization point with respect to a data item $\boldsymbol{d_i}$ is a transition where two independent sub-domains synchronize, one of which includes $\boldsymbol{x_i}$ (in other words, it is a transition whose enactment relies on $\boldsymbol{x_i}$ – the real-life state variable).

Note that according to Soffer et al. (2010), when two subdomains synchronize (externally or internally to the process) they are no longer separate, and together they form a merged subdomain. They can separate again after synchronization. Since none of the sub-domains can take the "bird view", the only spot where they are exposed to each other's values is at external synchronization points. The external synchronization points in the process are inherent in the model. In this work we focus on notifying the process designer where there might be potential data inaccuracy problems according to a current process design (which may not take such situations into account).

For brevity, we refer to external synchronization points as *synchronization points*. Note that we do not aim to model the real world, we only assume that the domain expert has knowledge about the location of synchronizations (as the transition *provide technical service* in the running example).

We now turn to an operational view of the process using the DPN representation and extend it by adding a *read* label function and a synchronization function, allowing to mark read operations and a synchronization point with respect to a data item $d \in \boldsymbol{D}$ within a transition (Δ_D).

Definition 6 (*S-DPN*) A Synchronizing Petri net with data (S-DPN) is a net N (P, T, F, D, V, Val, W, G, R, S), consisting of:

- A DPN (P, T, F, D, V, Val, W, G)
- a read function R: $T \rightarrow 2^D$ that labels each transition with a set of read operations of data items $d \in \boldsymbol{D}$
- A synchronizing data labelling function S: $T \rightarrow 2^{|\Delta D|}$ assigning synchronization points to transitions, where Δ_D is a set of synchronization points of the form $\Delta_D(d)$, $d \in \boldsymbol{D}$.

We can now enrich the running example in Fig. 1 using DPN as presented in Fig. 4. For example, *Provide technical service* is an (external) synchronization point with respect to the data item *address* (marked Δ(*address*)) since at this point the actual location of the customer is sensed.

Synchronization points are crucial for our analysis approach: if data inaccuracy exists at run-time, these are the only points where it is certain to be detected. In the running example, when receiving the message ordering technical service, the IS data item of address is written. The process sub-domain and the external sub-domain diverge, converging again (synchronizing) at the point of providing technical service. It is assumed that the real address of the customer (x) equals the IS data value (d). In case of an incorrectly recorded address, service provision (which will not be possible) is the first point in the process where the error is certain to be discovered. As long as the external and the process sub-domains progress independently, there is no way of knowing whether the data item correctly reflects the state variable value.

4 Data Inaccuracy Awareness-based Analysis

The above leads to the observation that at a given state in the process at runtime it may not be known whether $\boldsymbol{x_i} = \boldsymbol{d_i}$ holds for a corresponding couple $\langle \boldsymbol{x_i}, \boldsymbol{d_i} \rangle$. Clearly, at a place which follows an external synchronization point, this is not the case – at this point, it is always known whether these values are equal. However, as new write operations with respect to $\boldsymbol{d_i}$ occur, this may no longer be the case.

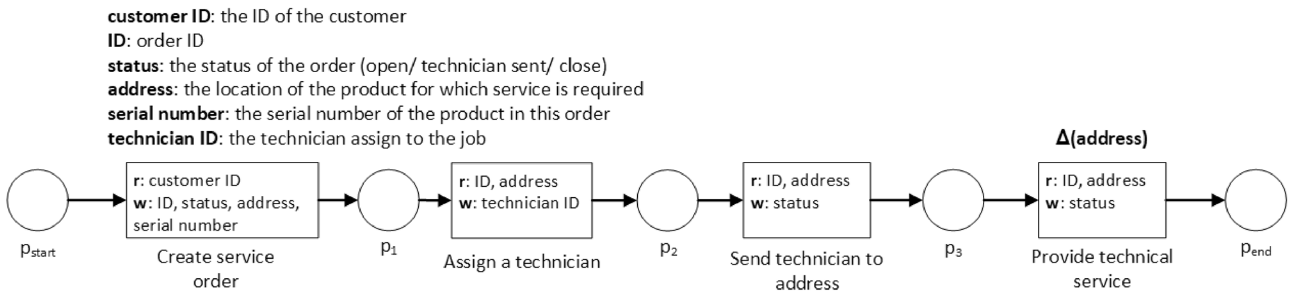
customer ID: the ID of the customer
ID: order ID
status: the status of the order (open/ technician sent/ close)
address: the location of the product for which service is required
serial number: the serial number of the product in this order
technician ID: the technician assign to the job
Δ(address)
r: customer ID
w: ID, status, address, serial number
r: ID, address
w: technician ID
r: ID, address
w: status
r: ID, address
w: status
Pstart
Create service order
P1
Assign a technician
P2
Send technician to address
P3
Provide technical service
Pend

Fig. 4 The running example modeled using S-DPN

Definition 7 (*Data Inaccuracy Awareness at a Place*) Let N be a S-DPN and $\langle \boldsymbol{x_i}, \boldsymbol{d_i} \rangle$ a corresponding couple. A place p of N is *Data Inaccuracy Aware* (DIA) with respect to $\boldsymbol{d_i}$ iff during the enactment of N, at the sub-domain state represented by a token in p, it is known whether $\boldsymbol{x_i} = \boldsymbol{d_i}$. p is not *Data Inaccuracy Aware* (non-DIA) otherwise.

In our running example, the place P_{end} (see Fig. 4) after the synchronization point (*provide technical service*) is DIA with respect to the *address* data item.

We formalize this using the standard Petri net notation (van der Aalst 1998): $\bullet t$ and $t\bullet$ ($\bullet p$ and $p\bullet$) denote the sets of input and output places of a transition t (a place p) respectively.

Observation For a data item d, if there exists a synchronization point in a transition t then for each place $p \in t\bullet$ if $|\bullet p| = 1$, then p is DIA with respect to d.

Justification: let t be a transition that has a synchronization point with respect to d, $p_1 \in t\bullet$, $|\bullet p_1| = 1$. Then the only transition that precedes it is where synchronization takes place. At a synchronization point, the synchronized subdomains are merged into one subdomain where the values of d and the corresponding state variable x can be observed. Hence, the immediately following p_1 is DIA with respect to d.

Data inaccuracy is a concept which applies to an enacted process at runtime. The above definition establishes the concept of DIA and relates it to a specific element (place) in a process model (currently a S-DPN). Thus, for a given data item, we can determine the DIA value for each place already at process design, enabling the prediction of potential consequences of data inaccuracy situations. For example, a *read* operation of a data item d following a non-DIA place (with respect to d), signifies potential use of inaccurate data.

Definition 8 (*Potential Use of Inaccurate Data*) A potential use of inaccurate data (UID), with respect to data item $\boldsymbol{d}$ is one of the cases:

1. Transition – A transition t with a *read* operation of $\boldsymbol{d}$ such that the aggregation of DIA values of all its input places ($\bullet t$) is non-DIA,
2. Guard – A guard expression which is a function of $\boldsymbol{d}$, and is assigned to one of the output transitions of a non-DIA place where $|p\bullet| > 1$.
3. The final place of the process is non-DIA.

The final place is considered an UID since if it is *non-DIA*, then the last value recorded in the database when the process ends is not certain to reflect the real value. Any future use of d in another instance of the process or in another process may hence use a (potentially) inaccurate value.

In the following we propose an algorithm which, given a S-DPN, classifies each place as DIA/non-DIA for a data item $\boldsymbol{d_i}$, and identifies its set of UIDs. We start by presenting the main premises of our approach.

Premise 1 When changes are made to the value of a data item through a write operation, errors are possible, so the new value does not necessarily match the corresponding state variable. Hence, a write operation induces a non-DIA state.

Premise 2 We will know at runtime whether the value of the data item matches the corresponding state variable if a synchronization point is reached (although the real value is not necessarily known at that time). Hence, a synchronization point induces a DIA state.

We further limit our analysis to processes whose S-DPN representation N satisfies the following conditions: (1) The DPN represented by N is sound (van der Aalst 1996). (2) The S-DPN N is well-structured (van der Aalst 1998). (3) For a given data item, each transition can have either a *write* or Δ but not both. In cases where both operations take place, we assume they are represented by different transitions; (this does not reduce generality, and is conclusive about the order in which these data operations take place – crucial information for our analysis). (4) For two or more parallel sequences of *N*, only one can[2] include a *write* operation of a given data item *d*. As a required modeling convention, when the permission to update changes between sub domains, there should be a merge between them.

We denote by EP_b the set of elementary paths of a block $b = (x, y)$ where *x* is the entry node and *y* the exit node. For a place *p* in a block *b* that does not contain any other block, we write *Block(p) = b*. We denote by *LastOpOnPath(ep, b)* the last data operation for *d* occurring on an elementary path *ep* in *b*. If no data operation for *d* occurs in *ep*, we let *LastOpOnPath(ep, b) = none*. For $p \in y\bullet$, where *y* is a transition, we define $MaxLastOp(p,b) = max^* \{ LastOpOnPath(ep,b) | \; ep \in EP_b \}$ where *max** is a maximum function induced by the total order *none* < *synch* < *write* (the minimum function induced by the total order *false* < *true* < *NULL*). Note that in case y is a place (XOR-join) we do not need to use the function *MaxLastOp*, and we only need to take the DIA values of the $\bullet y$ and calculate the DIA value of *y* using the function max*.

The DIA algorithm uses an auxiliary function *Propagate* (see Listing 1) for setting $DIA_d(p)$ for each place *p* according to the premises described above.

```
Function: Propagate
input: a data item d, a place p, a transition t such that p∈t•, a truth value v ∈ {true, false} and an operation
maxLastOp ∈ { none, synch, write}
begin
  if (wt(t)=d), then    // t has a write operation
  {
     p.lastOp = write
     p.dia = false
  }
  else if (sp(t)=d), then    // t has a synch operation
  {
     p.dia = min( p.dia, true )
     if ( p.dia = true ), then
        p.lastOp = synch
  }
  else
  {
     temp = p.dia
     p.dia = min( p.dia, v )
     if (p.dia = temp), then     // if the DIA value was not changed
        p.lastOp = max*{ p.lastOp, maxLastOp }
     else
        p.lastOp = maxLastOp
     if (p.lastOp=sync) p.dia = true
  }
  if ( |t•| > 1 ), then  p.lastOp = none    // in case the transition t starts a new block
end
```

Listing 1. Function propagate

The input of the main algorithm is a S-DPN $N = \langle P, T, F, D, V, Val, W, G, R, S \rangle$ and a data item $d \in D$. Its output is the net *N* decorated with DIA values for each place.

The DIA analysis algorithm (Listing 2) searches the S-DPN in a Breadth-First Search (BFS)-like manner, setting $DIA_d(p)$ for each place *p* using the *Propagate* function, using also an auxiliary function for identifying UIDs (see Listing 3). Each place has a variable *p.dia* indicating its current DIA classification (*true/false*). When addressing a branch in a block, the *lastOp* variable represents the last

[2] This is because parallel threads imply independence of the relevant sub-domains irrespectively of the order in which transition takes place. It is hence not possible for parallel threads to update the same data item (Soffer et al. 2010). Moreover, if both write to the same data item, there must be a synchronization (even at the database level). Whether or not this synchronization appears as a merge in the model, is a matter of granularity and modeling decision.

operation in that branch (*none*/*synch*/*write*). At every step, *CURRENT_P* is a set which contains the remaining places to classify. The algorithm starts with a DIA value of p_{start} set as *false* (Premise 2). Then, for each output transition of p_{start}, output places are added to the set *CURRENT_P*. For each place in *CURRENT_P*, the propagate function determines the DIA value. In case of concurrency merge (a transition with more than one input places), we use the maximal data operation (max*) of the last ones on the merging branches (function *MaxLastOp*).

The UID identification algorithm searches for: (1) *read* operations in transitions whose aggregation of input DIA values is non-DIA; (2) *guard* expressions for transitions whose input places DIA value is non-DIA; (3) a non-DIA final place. As an output, the algorithm provides a list of UIDs (elements – transitions/places).

```
Algorithm: DIA algorithm for a data item d
input: A S-DPN net $N = <P,T,F,D,V,Val,W,G,R,S>$ for $D$ and a data item $d \in D$
output: A S-DPN net $N$, with places $P$ classified as DIA or non-DIA
init
  $\forall p \in P$: p.dia = NULL        // the dia variable is initialize to NULL for all places in P
  $p_{start}$.dia = false                // the initial place is non-DIA by Premise 2 above.
  $\forall p \in P$: p.lastOp = none     // the lastOp variable is initialize to none for all places in P
  CURRENT_P $\Leftarrow$ { $p_{start}$ }   // places to classify, initialize with $p_{start}$
begin
  while (CURRENT_P != $\emptyset$)
  {
    curr $\in$ CURRENT_P
    CURRENT_P $\Leftarrow$ CURRENT_P \ {curr}

    foreach transition t $\in$ curr$\bullet$
    {
      foreach place p $\in$ t$\bullet$
      {
        op = $|\bullet t| > 1$ ? max*{MaxLastOp(p,b)| $\forall b \in$Block(p)} : curr.lastOp   //  there exist a block or not
        Propagate(d, p, t, curr.dia, op )
        CURRENT_P $\Leftarrow$ {p} $\cup$ CURRENT_P
      }
    }
  }
  UIDs_identification(N, d)
end
```

Listing 2. DIA analysis algorithm

```
Algorithm: UIDs_identification
input: A S-DPN net $N = < P,T,F,D,V,Val,W,G,R,S >$ for $D$ and a data item $d \in D$, with a set holding Boolean
variables $DIA_d(p)$
output: a set of $UID \subseteq T \cup P$
init
  $UID\_ELEMENTS \Leftarrow \{ \}$      // An empty set of elements to return
begin
  foreach transition $t \in T$
  {
    if ( $|\bullet t| = 1$ ), then      // transition t is sequential
      if ($rd(t) == d \,||\, grd(t) == d$), then
      {
        place $p \in \bullet t$
        if ($p.dia == false$), then $UID\_ELEMENTS \Leftarrow \{t\} \cup UID\_ELEMENTS$
      }
    else      // transition t has more than one input
    {
      if ($rd(t) == d \,||\, grd(t) == d$), then
      {
        bool isNonDIA = $true$
        foreach place $p \in \bullet t$      // look for a synchronization point as last operation in the block
          if ($p.lastOp == synch$), then isNonDIA = $false$;
        if (isNonDIA == $true$), then $UID\_ELEMENTS \Leftarrow \{t\} \cup UID\_ELEMENTS$
      }
    }
  }
  if ($p_{end}.dia == false$), then $UID\_ELEMENTS \Leftarrow \{ p_{end} \} \cup UID\_ELEMENTS$   // check the final place
end
```

Listing 3. An algorithm for identifying UIDs

As an example, consider the S-DPN provided in Fig. 5, that has one data item *d*. For the initial place p_{start}, the DIA value is *false* and the *lastOp* is *none*. Transition t_1 has only a *read* operation for *d*, and transition t_2 does not have any data operation, thus, we propagate the DIA and *lastOp* values to p_3 and to p_1, then to p_2. Transition t_3 has a synchronization point for *d* (recall that it means that at this point at runtime we will realize whether the value of *d* is accurate or not). Thus, the DIA and *lastOp* values are set to *true* and *synch* in accordance. For t_4, evaluating its DIA value requires comparing the *lastOp* of its input places with the function max*. Since the last operation in this block is

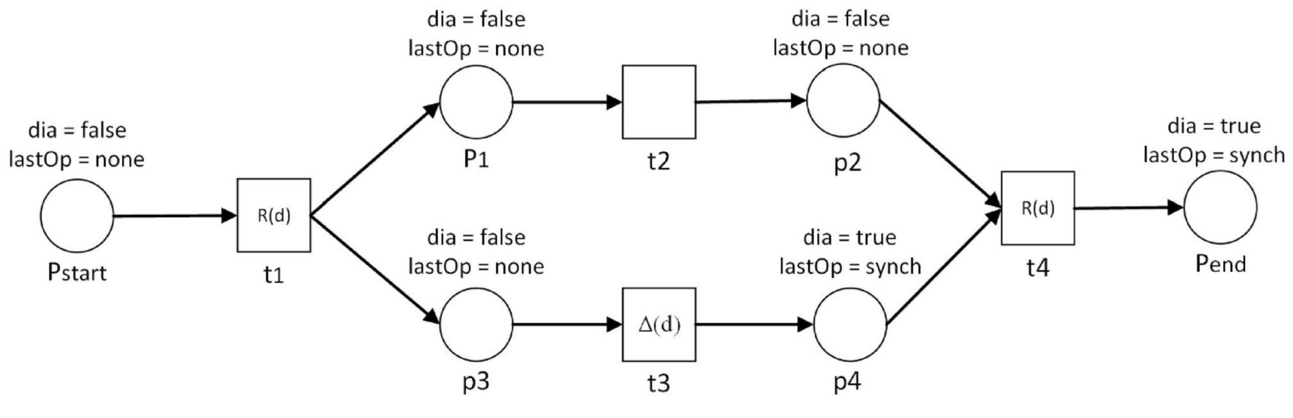

Fig. 5 A S-DPN example of DIA classification using the DIA analysis algorithm

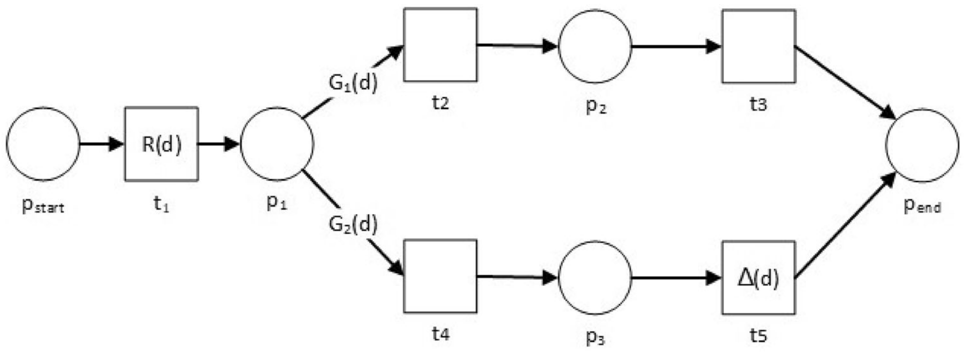

Fig. 6 UID and DE example

the synchronization point in t_3, *lastOp* value of p_4 is propagated to p_{end} and thus, since the *lastOp* is *synch*, the DIA value of p_{end} is *true*.

Next, for UID identification, the model includes two *read* operations (at t_1 and t_4), where t_1 is an UID since its input place p_{start} is non-DIA. t_4 is not an UID since the aggregated DIA value of its input places is *true.* Last, the final place p_{end} is not an UID since its DIA value is *true*.

5 Operationalizing the DIA Analysis

The DIA analysis presented above produces an S-DPN, where it is marked for each place *p* and data item *d* whether *p* is *DIA* for *d*, using this marking to create a list of UIDs. Next, an operationalization in business terms is needed which tracks for each possible UID the exact place where potential consequences of this use will be discovered.

Definition 9 (*Potential data Discrepancy Exposure*) Given a UID *u*, a transition *t* is a potential data discrepancy exposure (DE) with respect to data item *d* and *u* iff in case an inaccurate value of *d* is used in *u*, this will necessarily be recognized in *t*.

In our running example (see Fig. 4), in transition *send technician to address* the data item *address* is read without any validation that it holds the accurate value. Thus, we can read a false value and send the technician who will arrive at the wrong address; hence, this is a UID. We can realize how accurate this value is and recognize that a false value was used only at *provide technical service,* where the data value of *address* will have to match the real customer's address for the transition to be executed; therefore, this is a DE.

To characterize DE identification in the following Lemma, we say that a synchronization point *P* is *closest* to a UID *U* if no path from *P* to *U* contains other synchronization points.

Lemma *Given a UID U with respect to a data item d, all the transitions which are in the set C of closest synchronization points to U (with respect to d) are DE.*

Proof Let $t \in$ C. Since *t* is a synchronization point with respect to *d*, it is known at *t* whether *d* is accurate. Since *t* is closest to U, no path from t to U contains other synchronization points, thus there are no earlier transitions on a path to U where it is known whether *d* is accurate. Hence, *t* is a DE.

We illustrate the relation between UID and DE by means of the example depicted in Fig. 6.

A possible error due to reading *d* in t_1 or an erroneous selection of t_4 (following the *guard* G_2) will be identified at the synchronization point in t_5. If, however, data inaccuracy causes the selection of t_2 (through the *guard* G_1), there is no DE in the path to the final place, and the potential error will not be detected until the end of the process (p_{end}).

Listing 4 specifies the DE identification algorithm, which identifies the first synchronization points in all the following paths from a specific UID.

```
Input: a S-DPN net N = <P,T,F,D,V,Val,W,G,R,S> for D, a data item d∈D model and a UID u.
Output: mappingSet – a set containing pairs (u, de) where u is a UID and de is the closest DE
reachable from u.
Used methods:
reachableTransitionInBlock(t) – returns the set of DE (transition) in this block. In case there is no
DE in this block, it returns NULL.

Begin
mappingSet ← { }
// t is the transition belong to u
CURRENT ← {t} // the set which point out on the next transition to handle
while (CURRENT != ∅)
{
    temp ∈ CURRENT
    CURRENT ← CURRENT \ {temp}
    if ( |temp•|>1 && de = reachableTransitionInBlock(temp) )
        for each x∈de
            mappingSet ← mappingSet ∪{(u,x)}
    else
        place p∈temp•
        for each transition t∈p•
            if(sp(t)=d)
                mappingSet ← mappingSet ∪{(u,t)}
            else
                CURRENT ← {t} ∪ CURRENT
}
return mappingSet
End
```

Listing 4. An algorithm for identifying DEs for a UID

To summarize, Fig. 7 provides an overview of the method for a DIA-based analysis as described. First, the DIA analysis algorithm classifies each place in the S-DPN as DIA or non-DIA and reveals a list of UIDs. Then, the UIDs' aim is to detect DEs. Note that a DE may manifest itself at runtime as an actual error event. The main goal of the approach is the detection at design time of unaware decisions (UID) that may arise due to potentially inaccurate values of data items.

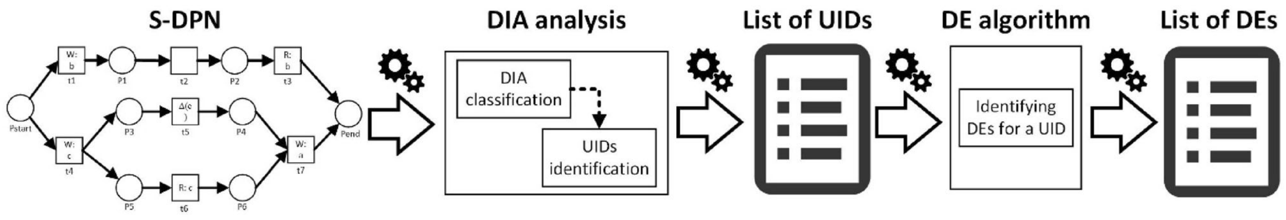

Fig. 7 The process of the DIA based analysis

6 Evaluation

In this section, we evaluate the approach by testing its capability to foresee negative consequences that might emerge in real-life data inaccuracy situations. Note that controlled experiments concerning the scalability of the DIA analysis algorithm and its applicability to non-trivial processes are reported in Evron et al. (2017a). Here we report on an experimental case study investigating the validity of the analysis results. According to Tellis (1997), a single case study can be useful for revelatory purposes. Our study aims to assess whether conclusions drawn using the DIA analysis are consistent with real-life observations.

6.1 Organizational Setting

The case study focused on a real-life process in a large manufacturing company (over five thousand employees) of complex and safety–critical systems in Israel. Due to confidentiality reasons, most details about the process and the organization cannot be disclosed.

The studied process is considered crucial for the quality of the products. It deals with providing services to equipment and machinery owned by the organization. In particular, this includes three main types of service designed to ensure proper usage of the equipment: repair, calibration, and acceptance testing for new equipment. The process model, as constructed by the researchers based on the domain expert's description and approved by the domain expert, consists of 25 transitions, 17 places, 23 data items, 14 guards, 30 synchronization points, and involves 4 different organizational units.

6.2 Study Procedure

Our goal was to compare potential data discrepancy exposure (DE) points indicated by the proposed approach against real error events as described by a domain expert responsible for the process.

The procedure for validating the analysis results included the following steps (Fig. 8):

(1) *Conduct an introduction meeting* – a first interview with the domain expert to examine potential processes and obtain the organization's permission to analyze their process. We selected the process using the following criteria: importance for the organization, sufficient information on the data items and data operations, impact on other processes, and the number of building blocks (to ensure a non-trivial process).

(2) *Elicit process details* – through three rounds of semi-structured interviews (Myers and Newman 2007) with the domain expert,[3] we examined each activity in the process, focusing on its resources and the data operations. Each interview lasted about 90 min; the interviews were recorded and transcribed.

(3) *Model the process* – based on the information obtained from the expert, the process was modeled using S-DPN. The initial model was created by one researcher, and then reviewed by the other two researchers; a final version was reached through a discussion.

(4) *Validate the model* – we validated the process model through another interview with the domain expert. Where needed, we made corrections to the model as suggested by the expert.

(5) *Elicit the error events* – we elicited a list of error events that might occur. These error events are related to data inaccuracy as recalled by the domain expert. Note that this list cannot be considered exhaustive, as it is subject to the cognitive limitations and biases (Tversky and Kahneman 1974) of the domain expert. Our main goal here was to overcome the possible cognitive biases which might arise when relying on the domain expert's memory. In order to cope with these biases, we examined each activity in the process with the domain expert and investigated each data item that was involved in that activity.

(6) *Perform DIA and UID analysis* – the DIA analysis algorithm was executed using the process model for each data item separately, marking each place as DIA or non-DIA for that data item, and yielding sets of UID cases for each data item.

(7) *Inference of DEs* – the DE analysis algorithm was executed, providing a list of potential data discrepancy exposures (DEs).

(8) *Confirm the DEs* – we confirmed the DEs through an interview with the domain expert. For each DE she indicated whether this situation could potentially occur in the process or not. The result was a list of confirmed DEs.

(9) *Map between the confirmed DEs and the initially provided error events* – the *confirmed* DEs were mapped to the *error events* recalled by the domain expert as follows: given an error event and a confirmed DE related to the same data item, they correspond if: (a) they relate to the same transition, (b) they stem from the same UID. A DE stems from a UID if it is identified as one of its closest synchronization points. An error event stems from a UID if the error event originates in the UID. The

[3] The semi-structured interview guide is available in https://sites.google.com/view/model-basedanalysisofdatainacc/home

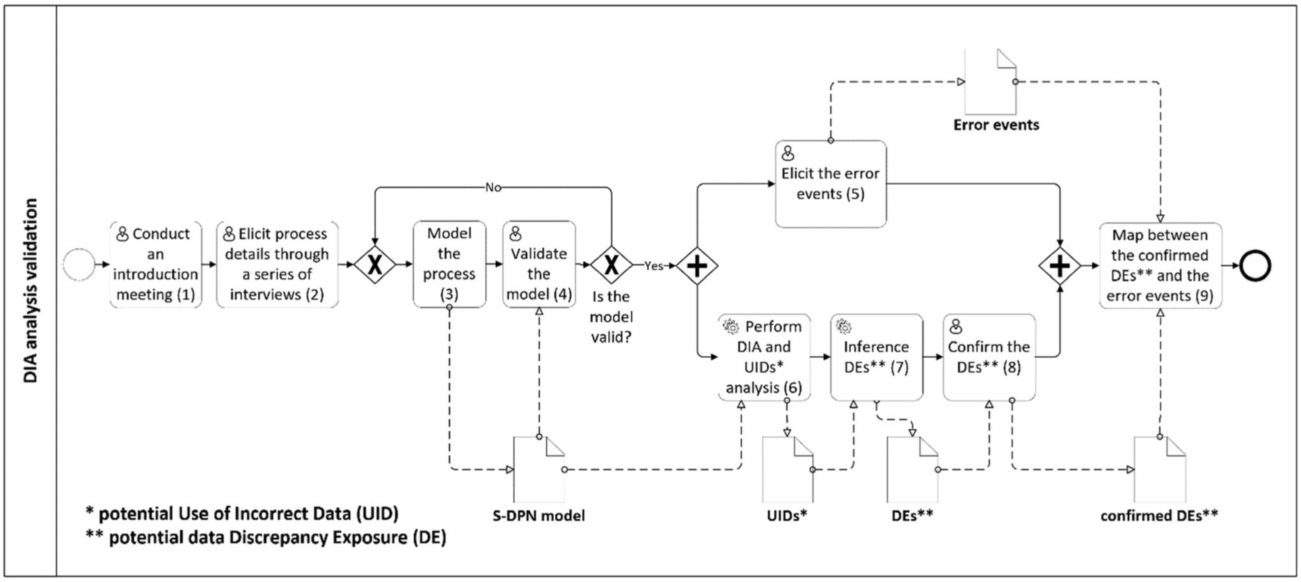
DIA analysis validation
Conduct an introduction meeting (1)
Elicit process details through a series of interviews (2)
Model the process (3)
Validate the model (4)
Is the model valid?
No
Yes
Elicit the error events (5)
Error events
Perform DIA and UIDs* analysis (6)
Inference DEs** (7)
Confirm the DEs** (8)
Map between the confirmed DEs** and the error events (9)
S-DPN model
UIDs*
DEs**
confirmed DEs**
* potential Use of Incorrect Data (UID)
** potential data Discrepancy Exposure (DE)

Fig. 8 Validation procedure

mapping was first performed from confirmed DEs to error events and then backwards (from error events to confirmed DEs).

We now illustrate step 9 by zooming into the following part of the process shown in Fig. 9.

When operational equipment requires service, a job is issued and recorded in the IS. Then, if the equipment is in the warehouse, it is delivered to the *responsible department* (e.g., mechanical, electrical etc.). Otherwise, the *responsible department* takes care of the equipment in its permanent location (see activity *Exit to the equipment's location*). Then the job goes through to the *Calibration* or *Repair* activities according to the assigned *service code*. Note that an inaccurate value can be introduced in several ways. For example, an employee in the service department can type in a wrong service code which will send the equipment to the wrong department.

To illustrate the mapping, consider the activity *Deliver to responsible department*, where the value of *Responsible department*, which was set during *Open job for existing equipment*, is checked. This is manifested in the model as a *read* operation in *Deliver to responsible department*. However, since the DIA value of p_2 with respect to *Responsible department* is *false,* this is a UID. It will be discovered at activities *Calibration* or *Repair,* where synchronization with respect to the data item *Responsible department* takes place. Thus, there exist two DEs in the activities: (1) *Repair*, (2) *Calibration*.

The domain expert recalled several events where errors in the *Responsible department* were recognized. According to her, these events revealed when the technicians attempt to repair the equipment and realize that the value is inaccurate.

The matching between the recalled error event and the confirmed DE was based on two criteria: (a) relation to the same transition – the DE and the error event were on the *Repair* transition; (b) the source of both the DE and the recalled error event is a UID at the *Deliver to responsible department*. Hence the DE and the error event on *Repair* correspond to each other. However, according to our analysis, two potential DEs stem from the same UID. The second one, on the *Calibrate* activity, could not be matched to a corresponding error event recalled by the domain expert. When it was presented to the domain expert, she confirmed that such an event was possible, and hence it was classified as a confirmed DE which was not matched to an error event.

Finally, when all confirmed DEs and error events were mapped, we could see which of the confirmed DEs had also been indicated by the domain expert, which of them had not been indicated (although confirmed by the domain expert as valid), and which error events were not identified by the analysis.

6.3 Findings

In our validation procedure we consider four types of possible results (see Fig. 10): (1) error events which were raised by the domain expert and were not spotted as DEs by our DIA based analysis; (2) DEs which were identified using the DIA based analysis and corresponded to error events indicated by the domain expert; (3) Additional DEs which were detected by the DIA based analysis for which

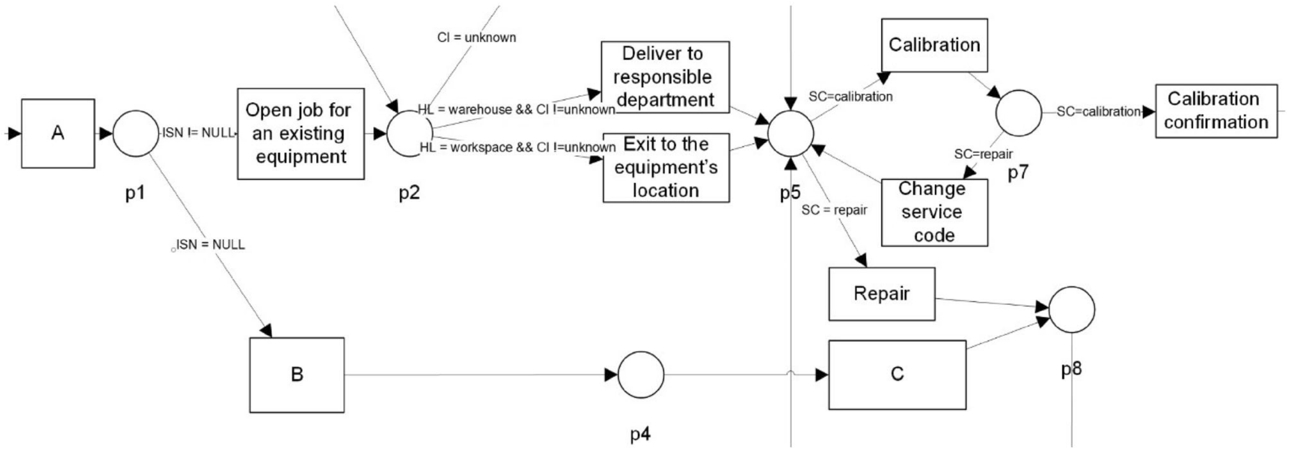
A
p1
ISN != NULL
ISN = NULL
Open job for an existing equipment
p2
CI = unknown
HL = warehouse && CI !=unknown
HL = workspace && CI !=unknown
Deliver to responsible department
Exit to the equipment's location
p5
SC=calibration
SC = repair
Calibration
Change service code
SC=repair
p7
SC=calibration
Calibration confirmation
Repair
p8
B
p4
C

Fig. 9 A part of the process

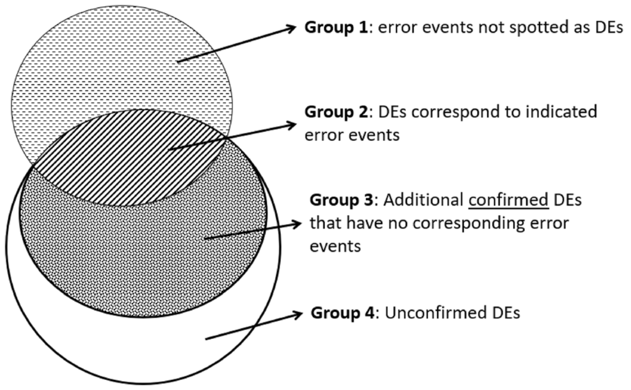
Group 1: error events not spotted as DEs
Group 2: DEs correspond to indicated error events
Group 3: Additional confirmed DEs that have no corresponding error events
Group 4: Unconfirmed DEs

Fig. 10 Illustration of the possible results

no corresponding error events were indicated initially, yet they were confirmed by the domain expert as likely to occur; (4) DEs which were detected by the DIA based analysis and were not confirmed as possible error events by the domain expert. Ideally, we would wish the sets of types (1) and (4) to be minimal to show the accuracy of our analysis, and (3) to be substantial to show its usefulness and value.

Clearly, it is not possible to establish a ground truth for a precise measurement of classification accuracy (e.g., using precision and recall metrics). We note, however, that group 1 can be regarded as identified false negatives, groups 2 and 3 together as true positives, and group 4 as false positives. According to this categorization, we created two metrics for assessing the accuracy of our results: (1) an approximate recall measure – the ratio between the number of confirmed DEs and the total number of confirmed error events: $\frac{G2+G3}{G1+G2+G3}$; (2) an approximate precision measure – the ratio between the number of confirmed DEs and the total number of DEs (confirmed and unconfirmed): $\frac{G2+G3}{G2+G3+G4}$.

For both metrics, we would like the results to be as close to one as possible.

In our study, the domain expert recalled 7 error events in the process (i.e., groups 1 and 2). All 7 error events were related to 7 different data items and were part of 6 different transitions. As a result of the analysis, 23 DEs were indicated (i.e., groups 2, 3 and 4). The 23 DEs were related to 13 different data items and were part of 10 different transitions. They were all confirmed by the domain expert as possible error events (namely, 0 DEs in group 4). One of the reasons for the differences between the results of the analysis (DEs) and the ones indicated by the domain expert (error events) is that the additional DEs do not necessarily materialize frequently. When we rely on a person's memory, limitations of memory and cognitive biases (Tversky and Kahneman 1974) might affect the recollection. In particular, some scenarios may occur very rarely, while others are more frequent or recent and are thus available in the expert's memory. Moreover, 4 out of the 16 new DEs are related to the same data items for which different associated transitions are included in the 7 error events identified by the domain expert. These may have masked other possible DEs associated with the same data items.

In summary, for the four types of results shown in Fig. 10, group (1) and (4) had no observations (thus our list of DEs is fully accurate), group (3) had 16 observations and group (2) had 7 observations, showing the analysis provided valuable results. For both metrics, approximate recall and precision measures, the result is one. Furthermore, the high number of observations in group (3) shows that our approach yielded non-trivial DEs that are valid (namely, confirmed by the domain expert), yet could not be identified otherwise.

7 Discussion

We have presented a process analysis approach here which consists of an automated analysis in terms of DIA and UID, together with its operationalization in terms of DE. The UID analysis yields a list of process steps where data is used while its value is not certain to be accurate. The other part of our approach, namely the DE identification, explores the potential runtime consequences of UIDs.

The main goal of our approach is to aid process analysts to reduce data inaccuracy risks already at design time. As such, an important concern is its easy and systematic integration into the analysts' process of designing processes. We suggest that after creating an initial process model, the analyst can run our automatic DIA analysis for all the relevant data items, creating a list of UIDs for consideration. It would be beneficial if the model can be modified so that all UIDs are eliminated and every use of data in the process only happens with an affirmative DIA value. However, with the intensive use of data in processes, this may result in many additional control steps and a complex process. To enable a better-informed decision, UIDs are mapped to DEs so that their consequences become apparent and the course of action can be determined. One of the actions that can be taken is to insert additional synchronization points before a DE with a possible modification of the process model, if needed, at design time. In our future research we intend to develop a method that will help avoiding possible negative consequences in DEs by injecting synchronization points as control measures. For instance, considering the running example (Fig. 3), if a synchronization point was added before the technician visits the customer (to ensure that the details of the customer address and product serial number data items match the corresponding state variable values), the possible consequences of data inaccuracy would be avoided. The envisioned method should take into account relevant considerations and trade-offs for identifying the best place to inject synchronization points (with respect the process flow, additional costs etc.).

In this paper, we follow the rigorous steps proposed for design science research (Peffers et al. 2007):

1. *Problem identification and motivation* Data inaccuracy in business processes can be a source of exceptions or harm the business goals. It is crucial to identify vulnerability to a potential data inaccuracy as soon as possible, since the earlier we determine a potential data discrepancy, the easier it is to repair it or reduce the negative consequences
2. *Defining the objectives for a solution* The objective of our work is to develop a design-time method, an artifact, which can assist process designers in identifying potential data inaccuracies.
3. *Design and development* We developed a method which would be able to point out at design time data items and spots in the process where a potential data inaccuracy might affect the process in runtime. The method was implemented on top of a Petri net modeling tool. It is based on the notions of data inaccuracy, synchronization points, DIA, UID and DE.
4. *Demonstration of the use of the artifact to solve one or more instances of the problem* The method was demonstrated using a case study.
5. *Evaluation* Our evaluation was based on the case study mentioned above. First, we wanted to validate that our results are valid in real life based on the assessment of a domain expert. Second, to measure the accuracy of the results, we defined two metrics, approximate precision and recall. The obtained results, as measured by these metrics, show that the proposed method provides an appropriate solution to the problem, as it is capable of identifying, based on a process model, potential runtime problems that may arise due to data inaccuracy.

The proposed approach has a number of limitations which should be discussed. First, it builds on a model which includes a synchronization point construct whose identification heavily relies on domain knowledge. Evron et al. (2017b) we reported on an empirical study which explored the way analysts identify synchronization points in a given process model. We indicated that while novices were mostly able to correctly identify synchronization points, this is not a trivial task. In the current evaluation, we made similar observations while interviewing our domain expert, who easily identified some of the points, but needed time and some effort to correctly identify some others.

Second, we note that the applicability of the proposed method is limited by the assumptions depicted in the context of Fig. 2 and might be too simplistic for real enterprise environments. However, we consider the proposed approach as a basic building block, which can still be used in complex diverse enterprises. In an extended enterprise setting our proposed building block can be used in a broader context and applied to different units of analysis. Consider, for example, the customer address data item. Its limited stability and dependence on external (customer) behavior cannot be changed. However, its updating context (see Fig. 2) can be used for determining the units of analysis (e.g., using the entire customer management process instead of a single ordering case), so the updating context is confined to this process instance and not accessible by others. Furthermore, relations and dependencies among data items can be analyzed with a broad enterprise perspective to clearly identify how data updates can manifest themselves and be addressed in different process instances. Another possibility that may arise in reality is the existence of several systems, in which errors may be introduced in one, causing differences in the values of corresponding data items in the different systems, manifested as data inaccuracy. With our set of concepts, each IS has its own subdomain, and problems indicated with respect to synchronization between two subdomains can be extended to any number of subdomains and be addressed accordingly.

In summary, as the diversity in the settings increases, the decisions that are based on data items which are non-DIA may have a greater impact, hence the importance of the DIA concept and its analysis increases. Our proposed method is indeed based on a simplified model, but the principles highlighted here can be extended and made applicable to more complicated problems. Even in this setting, we have shown in our study that valuable indications can be obtained.

Third, as discussed in Sect. 2, the applicability of the approach is limited to variables and data items which are not subject to continuous and spontaneous value changes but are rather under control, and are not shared among multiple concurrent process instances. Nevertheless, a variety of processes in daily and common domains entail data items that can be addressed appropriately (as does our case study process).

8 Related Work

Data quality has been widely addressed in various contexts (Abedjan et al. 2016; Agmon and Ahituv 1987; Batini and Scannapieco 2016; Falge et al. 2012; Sadiq 2013; Yeganeh et al. 2009). A variety of quality dimensions has been proposed (Heravizadeh et al. 2009; Wand and Wang 1996), and its importance for IS design has been recognized. For example, Orr (1998) emphasized the importance of data quality, claiming that it could be improved by changing data usage. Wang and Strong (1996) defined data quality as the ability of the data to meet users' needs. They claimed that to enhance data quality, we must understand what the data means to those who consume it. Main dimensions of data quality, indicated in the literature (Heravizadeh et al. 2009; Razniewski et al. 2013; Soffer 2010), are accuracy (accurate reflection of real-life values), completeness (all relevant values are available) and timeliness (data values are updated at the same time as the real values). Considering poor data quality along these dimensions, incompleteness is a special case where $d_i = \text{NULL}$ (hence $x_i \neq d_i$), and a lack of timeliness implies inaccuracy for a certain time period.

In general, some studies emphasize the importance of addressing data elements during database design to guarantee data quality at runtime, while others discuss the quality of data simply as the ability to meet requirements. For example, Yeganeh et al. (2009) proposed an approach which takes into account user preferences (requirements) for data quality awareness in response to queries from multiple sources. In their work they allow user preferences regarding data quality to be modeled using Quality-aware Queries which is based on a multi-criteria decision-making technique. Another example by Cappiello and Pernici (2008) presented a design-time framework to assist in selecting the most suitable repair strategy to adopt. The selection is based on the influence of quality dimensions.

In the context of business processes, data quality has received limited attention thus far. Sadiq et al. (2004) provides a classification of various potential data flow problems, and stresses that data quality issues, if not detected prior to workflow deployment, may prevent the process from correct execution. Plebani et al. (2017; Marrella et al. 2018) take a data-centric approach for promoting process resilience to failure, proposing a modeling notation which allows the designer to model a business process (at design-time) whose results are easier to manage in case of failures at run-time. Yet the focus of this work is mostly on failure due to data unavailability. Moreover, the approach highly depends on the process designers' adoption of a new modeling notation, which is known to be challenging in practice.

Rodríguez et al. (2012) and Cappiello et al. (2013) introduce a BPMN-based process model, geared to represent data quality issues. The proposed approach is rather informal and aimed to serve as a basis for human considerations rather than to support a systematic or automated analysis of potential data quality issues in a business process. Gharib and Giorgini (2014) introduce a goal-oriented approach for modeling and analyzing information quality requirements in business processes from a socio-technical perspective. Automated soundness verification of the resulting models is possible in the proposed approach, yet it does not support an in-depth analysis of possible manifestations of data problems. Addressing data quality issues in process design is suggested by Bagchi et al. (2006), Bringel et al. (2004), and Gharib and Giorgini (2014), aiming to predict how changes in the business process would affect data quality. The goal is to support process designers during the (re)design of business processes, in consideration of data quality requirements. Their techniques require deep human involvement and do not include automated operations. Cappiello et al. (2008, 2014, 2018) propose a strategy for data quality enhancement by inserting Data Quality blocks (Shankaranarayanan et al. 2000), namely, designated monitoring points, to the process. Decision support and guidance is needed for designing the blocks and placing them along a process. To conclude, most of these works either require a high level of manual involvement by the process designer or do not support an analysis of the possible consequences of identified data quality problems as part of the process.

This paper addresses data accuracy (or inaccuracy), which is a specific aspect of the broader notion of data quality, but proposes a *general* framework for its analysis at design time in a more systematic way than the works mentioned above. Taking a similar focus, analysis of potential data inaccuracy at design time has been suggested by Soffer (2010), who also provided a formalization of the problem and its underlying mechanism. Moreover, Soffer (2010) discussed the potential consequences of data inaccuracy in business processes and outlined possible scenarios of such. Following Soffer's ideas, we extended the formalism of DPN (De Leoni and van der Aalst 2013; de Leoni et al. 2018; Mannhardt et al. 2014) with the notion of *synchronization points*. In Evron et al. (2017a, b), the notions of synchronization points and DIA were first introduced and evaluated. This paper extends them (Evron et al. 2017a, b) by adding the notions of potential use of inaccurate data (UID) and potential data discrepancy exposure (DE) in order to enable an in-depth analysis to find out the consequences of potential data inaccuracy. Moreover, we identify where process decisions are made based on potentially inaccurate data. An evaluation using a real-life process has been presented.

9 Conclusion and Future Work

Data inaccuracy may manifest itself in business processes at runtime, with severe consequences. To the best of our knowledge, our approach is the first to suggest an automated analysis to address this problem at design time. We provide a rigorously defined framework for detecting process states at which potential discrepancies between real and runtime values may arise based on the notions of DIA, UIDs and DEs. We provide an empirical evaluation of the approach with an industrial case study.

While this paper relies on DPN for developing the analysis algorithms, the notions introduced in the paper are generic and independent of a particular modeling formalism. A similar analysis can be applied to different models (e.g., CPN, BPMN), provided they specify data operations at a sufficient level of detail. This would require an adaptation of the analysis algorithms, but not the development of completely different ones.

Future research can extend the approach in several directions. First, the approach can be extended to be applicable to globally updated data items or to unstable state variables. Second, this work is only a first step towards identifying all the potential data discrepancies. In order to improve this, we will incorporate data dependencies and take them into account as part of the analysis. The dependencies between data items might enable our approach to better identify the source of data inaccuracies, thereby improving the possibility to address these problems. Third, prioritizing the data items is a crucial challenge which can assist the process designer in focusing on the most influential data items. This can be done using the data dependency analysis which helps to determine the source of the discrepancy. Fourth, we will develop a mechanism that will use the approach to evaluate the best point in the process to inject additional synchronization points. Adding this can help resolve some of the data inaccuracy situations that our approach discovered.

Acknowledgements The research was supported by the Israel Science Foundation under Grant agreements 856/13 and 550/19.

References

Abedjan Z, Golab L, Naumann F (2016) Data profiling. In: IEEE 32nd international conference on data engineering (ICDE). IEEE, pp 1432–1435

Agmon N, Ahituv N (1987) Assessing data reliability in an information system. J Manag Inf Syst 4(2):34–44

Bagchi S, Bai X, Kalagnanam J (2006) Data quality management using business process modeling. In: IEEE international conference on services computing, pp 398–405

Batini C, Scannapieco M (2016) Data and information quality. Springer, Cham

Bovee M, Srivastava RP, Mak B (2003) A conceptual framework and belief-function approach to assessing overall information quality. Int J Intell Syst 18(1):51–74

Bringel H, Caetano A, Tribolet JM (2004) Business process modeling towards data quality: an organizational engineering approach. In: International conference on enterprise information systems, vol 4, pp 565–568

Cappiello C, Pernici B (2008) Quality-aware design of repairable processes. In: ICIQ, pp 382–396

Cappiello C, Caro A, Rodriguez A, Caballero I (2013) An approach to design business processes addressing data quality issues. In: ECIS 2013 proceedings. https://aisel.aisnet.org/ecis2013_cr/216

Cappiello C, Cerletti C, Fratto C, Pernici B (2018) Validating data quality actions in scoring processes. J Data Inf Qual 9(2):11

Cappiello C, Pernici B, Villani L (2014) Strategies for data quality monitoring in business processes. In: International conference on web information systems engineering. Springer, Berlin

De Leoni M, van der Aalst WMP (2013) Data-aware process mining: discovering decisions in processes using alignments. In: Proceedings of the 28th annual ACM symposium on applied computing, pp 1454–1461

de Leoni M, Felli P, Montali M (2018) A holistic approach for soundness verification of decision-aware process models. In:

Trujillo JC, et al (eds) Conceptual modeling, LNCS, vol 11157. Springer, Cham, pp 219–235

Evron Y, Soffer P, Zamansky A (2017a) Design-time analysis of data inaccuracy awareness at runtime. In: International conference on business process management. Springer, Cham, pp 600–612

Evron Y, Soffer P, Zamansky A (2017b) Incorporating data inaccuracy considerations in process models. In: Enterprise, business-process and information systems modelling. Springer, Cham, pp 305–318

Falge C, Otto B, Österle H (2012) Data quality requirements of collaborative business processes. In: 45th Hawaii Int conference on system science. IEEE, pp 4316–4325

Gharib M, Giorgini P (2014) Detecting conflicts in information quality requirements: the May 6, 2010 flash crash. Università di Trento. http://eprints.biblio.unitn.it/4379/

Heravizadeh M, Mendling J, Rosemann M (2009) Dimensions of business processes quality. In: Business process management workshops. LNBIP. https://doi.org/10.1007/978-3-642-00328-8_8

Kiepuszewski B, ter Hofstede AHM, van der Aalst WMP (2003) Fundamentals of control flow in workflows. Acta Informatica 39(3):143–209. https://doi.org/10.1007/s00236-002-0105-4

Mannhardt F, De Leoni M, Reijers HA (2014) Extending process logs with events from supplementary sources. International conference on business process management. Springer, Cham, pp 235–247

Marrella A, Mecella M, Pernici B, Plebani P (2018) A design-time data-centric maturity model for assessing resilience in multi-party business processes. Inf Syst 86:72–78

Myers MD, Newman M (2007) The qualitative interview in IS research: examining the craft. Inf Organ 17(1):2–26

Orr K (1998) Data quality and systems theory. Commun ACM 41(2):66–71

Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. J Manag Inf Syst 24(3):45–77

Plebani P, Marrella A, Mecella M, Mizmizi M, Pernici B (2017) Multi-party business process resilience by-design: a data-centric perspective. International conference on advanced information systems engineering. Springer, Cham, pp 110–124

Polyvyanyy A, García-Bañuelos L, Dumas M (2012) Structuring acyclic process models. Inf Syst 37(6):518–538

Razniewski S, Montali M, Nutt W (2013) Verification of query completeness over processes. Business process management. Springer, Heidelberg, pp 155–170

Rodríguez A, Caro A, Cappiello C, Caballero I (2012) A BPMN extension for including data quality requirements. business process model and notation (LNBIP 125). Springer, Heidelberg, pp 116–125

Russell N, van der Aalst WMP, ter Hofstede AHM (2006a) Exception handling patterns in process-aware information systems. BPM Center Report BPM-06-04, BPM Center Org, 208

Russell N, van der Aalst WMP, ter Hofstede AHM (2006b) Workflow exception patterns. In: Dubois E, Pohl K (eds) Advanced information systems engineering. LNCS, vol 4001. Springer, Heidelberg, pp 288–302

Sadiq S (ed) (2013) Handbook of data quality: research and practice. Springer, Heidelberg

Sadiq S, Orlowska M, Sadiq W, Foulger C (2004) Data flow and validation in workflow modeling. In: ADC'04 proceedings of the 15th Australasian database conference, pp 207–214

Shankaranarayanan G, Wang RY, Ziad M (2000) IP-MAP: representing the manufacture of an information product. In: Proceedings of the 2000 conference on information quality

Sidorova N, Stahl C, Trčka N (2010) Workflow soundness revisited: checking correctness in the presence of data while staying conceptual. In: Advanced information systems engineering. Springer, Heidelberg

Soffer P, Kaner M, Wand Y (2010) Assigning ontological meaning to workflow nets. J Database Manag 21(3):1–35

Soffer P (2010) Mirror, mirror on the wall, can I count on you at all? Exploring data inaccuracy in business processes. Enterprise, business-process and information systems modeling. Springer, Heidelberg, pp 14–25

Soffer P, Wand Y (2004) Goal-driven analysis of process model validity. Advanced information systems engineering. Springer, Heidelberg, pp 521–535

Tellis WM (1997) Application of a case study methodology. Qual Rep 3(3):1–19

Tversky A, Kahneman D (1974) Judgment under uncertainty: heuristics and biases. Sci 185(4157):1124–1131

van der Aalst WMP (1996) Structural characterizations of sound workflow nets. Comput Sci Rep 96(23):18–22

van der Aalst WMP (1998) The application of Petri nets to workflow management. J Circuits Syst Comput 8(01):21–66

van der Aalst WMP (2000) Workflow verification: finding control-flow errors using petri-net-based techniques. Business process management. Springer, Heidelberg, pp 161–183

van der Aalst WMP, Gunther CW (2007) Finding structure in unstructured processes: the case for process mining. In: Seventh international conference on application of concurrency to system design. IEEE, pp 3–12

Wand Y, Wang RY (1996) Anchoring data quality dimensions in ontological foundations. Commun ACM 39(11):86–95

Wang RY, Strong DM (1996) Beyond accuracy: what data quality means to data consumers. J Manag Inf Syst 12(4):5–33

Yeganeh NK, Sadiq S, Deng K, Zhou X (2009) Data quality aware queries in collaborative information systems. Advances in data and web management. Springer, Heidelberg, pp 39–50