Analyzing Process-Aware Information System Updates Using Digital Twins of Organizations[*]

Gyunam Park[1][0000−0001−9394−6513], Marco Comuzzi[2][0000−0002−6944−4705], and Wil M. P. van der Aalst[1][0000−0002−0955−6940]

[1] Process and Data Science Group (PADS), RWTH Aachen University, Germany
{gnpark,wvdaalst}@pads.rwth-aachen.de
[2] Department of Industrial Engineering, UNIST, Korea
mcomuzzi@unist.ac.kr

Abstract. Digital transformation often entails small-scale changes to information systems supporting the execution of business processes. These changes may increase the operational frictions in process execution, which decreases the process performance. The contributions in the literature providing support to the tracking and impact analysis of small-scale changes are limited in scope and functionality. In this paper, we use the recently developed Digital Twins of Organizations (DTOs) to assess the impact of (process-aware) information systems updates. More in detail, we model the updates using the configuration of DTOs and quantitatively assess different types of impacts of information system updates (structural, operational, and performance-related). We implemented a prototype of the proposed approach. Moreover, we discuss a case study involving a standard ERP procure-to-pay business process.

Keywords: Business Process · System Update · Impact · Digital Twin.

1 Introduction

Process-Aware Information Systems (PAIS), such as ERP and CRM, play a key role in modern organizations, underpinning the execution of business processes [8]. As the environment surrounding an organization dynamically changes and the competition becomes more intensive, a demand to accordingly change the implementation of PAIS may arise.

Modern digital transformation often involves frequent and small-scale changes, or *updates*, to information systems, which are vital to enable continuous adaptation to dynamic business environments [16]. The need for such capabilities is illustrated by the measures needed to handle the Covid-19 pandemic. For instance, organizations had to devise measures to adapt to varying degrees of home-working workforce, or to variations of sales and orders that deviate extensively and unexpectedly from the usual seasonality.

[*] This work is supported by the Alexander von Humboldt (AvH) Stiftung and the 0000 Project Fund (Project n. 1.220047.01) of UNIST (Ulsan National Institute of Science & Technology).

PAIS updates affect the operations of an organization. However, due to the complex design and deep operational pervasiveness of modern PAIS, it is challenging to assess the impact of PAIS updates. Contributions in the PAIS literature in this direction are limited. Existing approaches mainly take a model-driven view on impact analysis [19,17,6], and concrete implementations are missing. Taking a design science standpoint [21], we address this research gap by developing (software) artifacts supporting the impact analysis of PAIS updates.

In this work, we distinguish specifically among three types of impacts: *structural*, *operational*, and *performance* impacts. At the structural level, it is important to assess the magnitude of the scope of a proposed update, such as the number of business objects and business functions that it affects. At the operational level, the impact of a PAIS update concerns the running instances of business processes involving the business objects and functions subject to change. For instance, removing the need to collect some customer information for privacy reasons may affect those customers for whom the information has been collected already. Most importantly, PAIS updates may impact on diagnostic measures of business processes: for instance, decreasing the level of tolerated mismatch between invoices and payments may slow down the processing of procurement cases, which in turn decreases the throughput of the procurement process.

To analyze the structural/operational/performance impact, we rely on Digital Twins of Organizations (DTOs). A DTO is a digital replication of an organization's operations, providing a transparent view to the business processes of the organization and enabling process analysts to analyze existing operational frictions and identify improvement opportunities. Moreover, DTOs allow to monitor business processes and trigger management actions, e.g., adding more resources and configuring business rules if improvements are available.

Specifically, we use a *Digital-Twin Interface Model* (DT-IM) [18] that (i) represents PAISs based on Object-Centric Petri Nets (OCPNs) [1] and (ii) models PAIS updates with the notion of *actions*. We compute the impact of a PAIS update with the DT-IM by analyzing the elements of the OCPN that it modifies (structural impacts), current states affected by the update (operational impacts), and the new value of diagnostic measures after the update (performance impacts).

From a design science standpoint, the practical relevance of the problem that we address lies with the increasing relevance of digital transformation for modern enterprises [16]. The rigor of the design process is ensured by extending a sound conceptual model of DTOs already published in the literature, i.e., DT-IMs. The design search process is inspired by the existing literature on ERP post-implementation change management [17], from which we draw ideas to model system updates and evaluate their impact. In this work, we limit the evaluation of the artifact to the feasibility of the proposed approach. To this aim, we have built a Web-based impact analysis software artifact and conducted a case study based on a procure-to-pay process loosely modeled following the standard one of the SAP ERP system.

The remainder is organized as follows. We discuss the related work in Sec. 2. Then, we present the preliminaries in Sec. 3. Next, we introduce the DT-IM in Sec. 4 and an approach to impact analysis based on the DT-IM in Sec. 5. Afterward, Sec. 6 introduces the implementation of a web application and a case study using the web application. Finally, Sec. 7 concludes the paper.

2 Related Work

Digital twins enabled by increasingly powerful data modeling and analysis capabilities have been envisioned by Gelernter in the 1990s [10] and have found widespread adoption in engineering in the last few years [9]. The idea of digital twins of organizations has emerged recently [4] as a means to address the challenges of information processing in modern digital transformation.

Mendling et al. [16] recently have highlighted an ongoing tension between business process management and digital transformation (DT): processes must be flexible to adapt to the continuously changing requirements of DT, while DT must rely on some process compliance to avoid a continuous disruption of business operations.

Business processes and the information systems supporting them can evolve to support changing business requirements through configuration and adaptation [7,13]. The former entails that all the possible evolution options are known a priori and can be captured into configuration tables, whereas adaptation involves ad-hoc modification of the process models or systems. Configuration is obviously more agile, but often it cannot address unexpected situations.

Business process flexibility can also be achieved through run-time adaptation. Along this direction, van Beest et al. [2] have proposed an approach for repairing processes at run-time when they interfere, e.g., when inconsistent writing operations occur. Marrella [15] have proposed to use automated planning techniques to support process adaptation to changing environments.

An issue closely related to process and system adaptability is the tracking of their evolution over time, understanding, in particular, the impact of proposed updates. In the context of cross-organizational information systems [6] or multi-tenant cloud systems [12], the evolution of systems and processes can be tracked by the evolution of Service Level Agreements (SLAs), which are updated by changing business requirements.

In the context of ERP systems, Soffer et al. [19,20] have proposed an ERP modeling language and a related methodology to align the ERP system capabilities with the enterprise requirements. The modeling language allows to express dependencies between business processes and objects, but it does not allow to express changes of them and their impact. Parhizkar and Comuzzi [17,5] have proposed a methodology and initial tool support to characterize the impact of ERP post-implementation changes inspired by the engineering change management literature. Lin et al. [14] have proposed a method that supports users during the execution of ERP post-implementation changes but which does not support the evaluation of the impact of such changes.

In summary, methods and tools to holistically support the tracking of information systems updates, understanding, in particular, their impact, remain limited in scope, functionality, and degree of automation. This paper tackles this research gap by proposing to use DTOs to develop tools that can (semi-)automatically support the assessment of PAIS updates.

3 Background and Preliminaries

This section introduces a conceptual model of PAISs (Subsec. 3.1) and the OCPNs, which we use later to formally model PAISs (Subsec. 3.2).

3.1 Process-Aware Information Systems (PAISs)

Fig. 1 introduces a meta-model of PAIS entities, updates, and impacts of PAIS updates considered in this work. This is inspired mainly by the work of Parhizkar and Comuzzi [17] in the context of ERP systems.

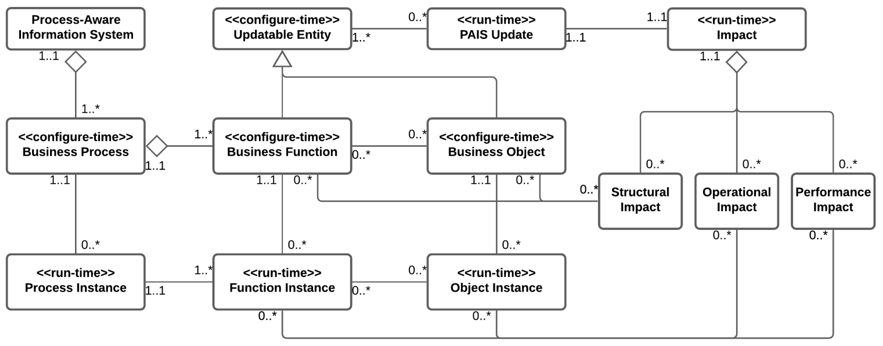

Process-Aware Information System
<<configure-time>> Updatable Entity
<<run-time>> PAIS Update
<<run-time>> Impact
<<configure-time>> Business Process
<<configure-time>> Business Function
<<configure-time>> Business Object
Structural Impact
Operational Impact
Performance Impact
<<run-time>> Process Instance
<<run-time>> Function Instance
<<run-time>> Object Instance
1..1
1..*
0..*

Fig. 1: A meta model of PAISs as a *UML 2.0 class diagram*

PAIS Entities. The meta model separates entities concerned with the configure-time from entities related with the run-time. At configure-time, a PAIS is constituted by a set of *business processes*, which in turn orchestrate a set of *business functions* that manipulate, using CRUD operations, *business objects*. At run-time, a PAIS instantiates instances of processes, functions, and objects.

PAIS Updates. A PAIS may be modified by creating, updating, or deleting an existing entity, i.e., an object, function or process. In this work, we focus on updates to business functions and objects. Creating a new entity has no impact on the business operations since it extends the functionality of a system and it only applies to new instances of business processes, functions, or objects. Deleting an entity can be seen as a special case of updating it. Updates to business processes, such as modifications of their control flow, are left out of scope.

A PAIS update is associated with an updatable entity, i.e., business functions and objects. First, an *update of a business function* is achieved by changing business rules and functionalities defining it. For instance, a *place order* function in an ERP system can be updated by changing the minimum price to create orders (i.e., a business rule) or changing a set of attributes that the function will update in a business object (i.e., a functionality). Such updates not only affect the business function, but also instances of the business function.

Second, an *update of a business object* is achieved by adding or removing attributes to it. For instance, the *order* business object can be updated by adding *payment terms* to specify the conditions of the payment, e.g., received by the end of the month, within seven days or through monthly installments. The update affects both business objects and their instances.

Impacts of PAIS Updates. A PAIS update results in an impact to the system. Below are the three types of impacts that we consider.

- *Structural impacts* concern configure-time entities, i.e., business objects and functions. They include:
  - *structural object impact*: the impact on business objects, e.g., the number of impacted business objects, and
  - *structural function impact*: the impact on business functions, e.g., the number of impacted business functions.
- *Operational impacts* concern run-time entities, i.e., the instances of business objects and functions. They include:
  - *operational object impact*: the impact on object instances of business objects, e.g., the number of object instances of impacted business objects, and
  - *operational function impact*: the impact on object instances of business functions, e.g., the number of objects of impacted business functions.
- *Performance impacts* concern changes in diagnostic measures of business objects and functions. They include:
  - *object performance impact*: the performance impact on business objects, e.g., difference in the avg. service time for a business object before/after updates, and
  - *function performance impact*: the performance impact on business functions, e.g., difference in the avg. waiting time for a business function before/after updates.

3.2 Object-Centric Petri Nets (OCPNs)

In this work, we model a PAIS introduced in Subsec. 3.1 with a DT-IM that uses an OCPN as its core formalism. This subsection introduces OCPNs. First, a Petri net is a directed bipartite graph of places and transitions. A labeled Petri net is a Petri net with the transitions labeled.

Definition 1 (Labeled Petri Net). *Let $\mathbb{U}_{act}$ be the universe of activity names. A labeled Petri net is a tuple $N{=}(P, T, F, l)$ with P the set of places, T the set of transitions, $P \cap T{=}\emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ the flow relation, and $l \in T \not\rightarrow \mathbb{U}_{act}$ a labeling function.*

A marking $M_N \in \mathcal{B}(P)$ is a multiset of places. A transition $tr \in T$ is *enabled* in marking M_N if its input places contain at least one token. The enabled transition may *fire* by removing one token from each of the input places and producing one token in each of the output places.

Each place in an OCPN is associated with an object type to represent interactions among different object types. The variable arcs represent the consumption/production of a variable amount of tokens in one step.

Definition 2 (Object-Centric Petri Net). *Let $\mathbb{U}_{ot}$ be the universe of object types. An* object-centric Petri net *is a tuple $ON{=}(N, pt, F_{var})$ where $N{=}(P, T, F, l)$ is a labeled Petri net, $pt \in P \rightarrow \mathbb{U}_{ot}$ maps places to object types, and $F_{var} \subseteq F$ is the subset of variable arcs.*

Fig. 2 shows an OCPN describing a part of the peer review process for an academic conference. There are two types of places associated with two object types, i.e., $conf$ (denoted in red) and $subm$ (cyan). For instance, $pt(p1){=}pt(p2){=}conf$, $pt(p3){=}pt(p5) = subm$, and $(p3, t2)$, $(t2, p5)$, and $(p7, t5)$ are variable arcs.

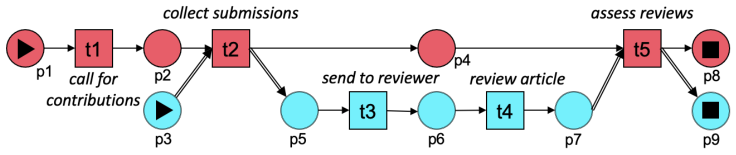


Fig. 2: An example of object-centric Petri nets

Definition 3 (Marking). *Let $\mathbb{U}_{oi}$ be the universe of object identifiers. Let $ON{=}(N, pt, F_{var})$ be an object-centric Petri net, where $N{=}(P, T, F, l)$. $Q_{ON}{=}\{(p, oi) \in P \times \mathbb{U}_{oi} \mid type(oi){=}pt(p)\}$ is the set of possible tokens. A marking M of ON is a multiset of tokens, i.e., $M \in \mathcal{B}(Q_{ON})$.*

For instance, marking $M_1{=}[(p4, c1), (p7, s1), (p7, s2)]$ denotes three tokens where place *p4* has one token referring to object *c1* of type *conf* and *p7* has two tokens referring to objects *s1* and *s2* of type *subm*.

A binding describes the execution of a transition consuming objects from its input places and producing objects for its output places. A binding (tr, b) is a tuple containing a transition tr and a function b mapping the object types of the input/output places to sets of object identifiers. For instance, $(t5, b1)$ describes the execution of transition $t5$ with $b1$ where $b1(\mathit{conf}){=}\{c1\}$ and $b1(\mathit{subm}){=}\{s1, s2\}$.

A binding (tr, b) is *enabled* in marking M if all the objects specified by b exist in the input places of tr. For instance, $(t5, b1)$ is enabled in marking M_1 since $c1$, $s1$, and $s2$ exist in its input places, i.e., $p4$ and $p7$.

A new marking M' is reached by executing an enabled binding (tr, b) at marking M, denoted by $M \xrightarrow{(tr,b)} M'$. For instance, by executing $(t5, b1)$, $c1$ is removed from $p4$ and added to $p8$, while $s1$ and $s2$ are removed from $p7$ and added to $p9$, leading to the new marking $M'{=}[(p8, c1), (p9, s1), (p9, s2)]$.

Finally, a relation function $rel \in T \rightarrow \mathcal{P}(P)$ maps a transition to a set of places associated with the transition. It is defined recursively: for any $tr \in T$,

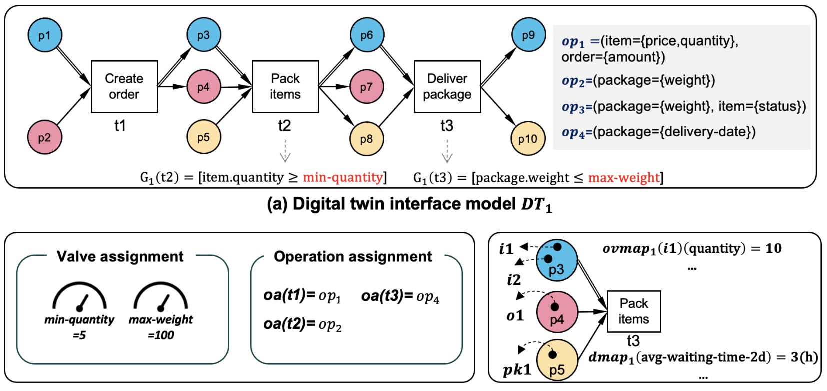
(1) $rel(tr){=}\emptyset$ if $\bullet tr{=}\emptyset$ and (2) $rel(tr){=}\bullet\, tr \cup \bigcup_{p \in \bullet tr, tr' \in \bullet p} rel(tr')$, where $\bullet tr$ is a set of input places of tr, i.e., $\bullet tr{=}\{p \in P \mid (p, tr) \in F\}$. For instance, $rel(t2){=}\{p1, p2, p3\}$ and $rel(t5){=}\{p1, \ldots, p7\}$.

4 Modeling PAISs: Digital Twin Interface Model

In this work, we use a *digital twin interface model* to model PAIS entities along with PAIS updates introduced in Subsec. 3.1.

4.1 Modeling PAIS Entities

A digital twin interface model consists of 1) an OCPN, 2) valves, 3) guards, and 4) operations. A *valve* is a system configuration, e.g., minimum required quantity to place orders. A *guard* is a formula composed of attributes, including valves, with relational operators (e.g., $\leq$,$\geq$,$=$) and logical operators (e.g., conjunction $\wedge$, disjunction $\vee$, and negation $\neg$). $F(X)$ denotes the set of such formulas defined over a set of attributes X. An *operation* describes a business operation, e.g., updating the quantity and price of an order.


p1
p2
Create order
t1
p3
p4
p5
Pack items
t2
p6
p7
p8
Deliver package
t3
p9
p10
op_1 =(item={price,quantity}, order={amount})
op_2=(package={weight})
op_3=(package={weight}, item={status})
op_4=(package={delivery-date})
$G_1(t2)$ = [item.quantity ≥ min-quantity]
$G_1(t3)$ = [package.weight ≤ max-weight]
Valve assignment
min-quantity =5
max-weight =100
Operation assignment
oa(t1)= op_1
oa(t3)= op_4
oa(t2)= op_2
i1
i2
o1
pk1
p3
p4
p5
Pack items
t3
$ovmap_1(i1)$(quantity) = 10
...
$dmap_1$(avg-waiting-time-2d) = 3(h)
...

(a) Digital twin interface model DT_1

(b) Configuration of DT_1 at time τ_1

(c) Operational state of DT_1 at time τ_1

Fig. 3: An example of digital twin interface models

Definition 4 (Digital Twin Interface Model (DT-IM)). *Let $\mathbb{U}_{valve}$ be the universe of valve names. Let $\mathbb{U}_{attr}$ be the universe of attribute names. A digital twin interface model, denoted as DT, is a tuple (ON, V, A, G, O) where*

- *$ON{=}(N, pt, F_{var})$ is an object-centric Petri net, where $N{=}(P, T, F, l)$,*
- *$V \subseteq \mathbb{U}_{valve}$ is a set of valve names,*
- *$A \subseteq \mathbb{U}_{attr}$ is a set of attribute names,*
- *$G \in T \to (F(V \cup A) \cup \{true\})$ associates transitions with guards, and*
- *$O \subseteq OT \nrightarrow \mathcal{P}(A)$ is a set of operations associating an object type to a set of attributes to be updated, where $OT{=}\{pt(p) \mid p \in P\}$.*

The transitions in the OCPN of a DT-IM represent *business functions*, whereas the object types associated with places indicate *business objects*. Guards and operations represent the business rule(s) and functionality (writing operations) of *business functions*, respectively. Fig. 3(a) shows an DT-IM, $DT_1{=}(ON_1, V_1, A_1, G_1, O_1)$, representing a PAIS supporting a simple order management process. It involves three object types: *item*, *order*, and *package*. First, an order is created with multiple items. Next, the item is packed and the packaged is delivered to the customer. Valves are depicted using red italic fonts, and guards are described in squared brackets. Operations are described in the gray box. For instance, $op_2 \in O_1$ describes a business operation writing *weight* of *package*, i.e., $op_2(package){=}\{weight\}$, $op_2(item){=}\perp$, and $op_2(order)={\perp}$.

A configuration defines the semantics of a DT-IM by determining the value of valves and the assignment of operations to transitions.

Definition 5 (Configuration). *Let $\mathbb{U}_{val}$ be the universe of attribute values. Let $DT{=}(ON, V, A, G, O)$ be a DT-IM with $ON{=}(N, pt, F_{var})$ and $N{=}(P, T, F, l)$. A configuration $conf_{DT} \in (V \rightarrow \mathbb{U}_{val}) \times (T \rightarrow O)$ is a tuple of a valve assignment va and an operation assignment oa. We denote $\Sigma_{DT}{=}(V \rightarrow \mathbb{U}_{val}) \times (T \rightarrow O)$ to be the set of all possible configurations of DT.*

Given $conf_{DT}{=}(va, oa){\in}\Sigma_{DT}$, $\pi_{va}(conf){=}va$ and $\pi_{oa}(conf){=}oa$. Moreover, we denote the configuration of digital twin interface model DT at $\tau \in \mathbb{U}_{time}$ as $conf_{DT,\tau}$. Fig. 3(b) describes a configuration of DT_1 at $\tau_1 \in \mathbb{U}_{time}$, where $\pi_{va}(conf_{DT_1,\tau_1})(min\text{-}quantity){=}5$, $\pi_{oa}(conf_{DT_1,\tau_1})(t2){=}op_2$, etc.

An operational state describes the current status of a business process, i.e., which objects reside in which parts of the process, using the marking of OCPNs. Moreover, it represents the various diagnostics about the performance of the process, e.g., the average waiting time of activity in the last seven days. We denote Δ_{DT} to be the set of all possible diagnostics of the digital twin interface model DT.

Definition 6 (Operational State of A DT-IM). *Let $\mathbb{U}_{vmap}{=}\mathbb{U}_{attr} \not\rightarrow \mathbb{U}_{val}$ be the set of all partial functions mapping a subset of attribute names to values. Let $DT{=}(ON, V, A, G, O)$ be a DT-IM with $ON{=}(N, pt, F_{var})$ and $N{=}(P, T, F, l)$. An operational state of DT is a tuple $os_{DT}{=}(M, ovmap, dmap)$ where*

- *$M \in \mathcal{B}(Q_{ON})$ is a marking of ON,*
- *$ovmap \in OI \rightarrow \mathbb{U}_{vmap}$ is an object value assignment where $OI{=}\{oi \in \mathbb{U}_{oi} \mid \exists_{p \in P}\ (p, oi) \in M\}$, and*
- *$dmap \in \Delta_{DT} \not\rightarrow \mathbb{R}$ is a diagnostics assignment such that, for any $diag \in \Delta_{DT}$, $dmap(diag) ={\perp}$ if $diag \notin dom(dmap)$.*

Given $os_{DT}{=}(M, ovmap, dmap)$, $\pi_M(os_{DT}){=}M$, $\pi_{ovmap}(os_{DT}){=}ovmap$, and $\pi_{dmap}(os_{DT}){=}dmap$. Fig. 3(c) describes an operational state of DT_1 at $\tau_1 \in \mathbb{U}_{time}$, i.e., $os^1_{DT_1}{=}(M_1, ovmap_1, dmap_1)$. In the remainder, we denote the operational state of the digital twin interface model DT at time τ as $os_{DT,\tau}$.

We define the semantics of DT-IM by extending the semantics of OCPNs with configurations and operational states. To this end, we use the notion of digital twin bindings.

Definition 7 (Digital Twin Binding). *Let $DT{=}(ON, V, A, G, O)$ be a DT-IM with $ON{=}(N, pt, F_{var})$. A digital twin binding of DT is a tuple $((tr, b), w, \tau)$ where (tr, b) is a binding of ON, $w \in \mathbb{U}_{ot} \to \mathcal{P}(\mathbb{U}_{attr})$ is a write function, and $\tau \in \mathbb{U}_{time}$ is a timestamp. A digital twin binding $((tr, b), w, \tau)$ is enabled with $conf_{DT,\tau}$ and $os_{DT,\tau}$ if the following conditions are satisfied:*

- *(tr, b) is enabled at $\pi_M(os_{DT,\tau})$,*
- *guard $G(tr)$ evaluates to true w.r.t. valve assignment $\pi_{va}(conf_{DT,\tau})$ and object value assignment $\pi_{ovmap}(os_{DT,\tau})$, and*
- *w corresponds to the assigned operation of tr, i.e., $w{=}\pi_{oa}(conf_{DT,\tau})(tr)$.*

Digital twin binding $((t2, b), w, \tau_1)$, where $b(item){=}\{i1, i2\}$, $b(order){=}\{o1\}$, $b(package){=}\{pk1\}$, and $w(package){=}\{delivery\text{-}date\}$, is enabled with the configuration of Fig. 3(b) and the operational state of Fig. 3(c) since $(t2, b)$ is enabled at $\pi_M(os_{DT_1,\tau_1})$, $G_1(t2)$ evaluates to true, and w corresponds to $\pi_{oa}(conf_{DT_1,\tau_1})(t2)$.

4.2 Modeling PAIS Updates

Next, we model the PAIS updates introduced in Subsec. 3.1 using the notion of *actions* in DT-IMs. An action updates the configuration of a DT-IM. First, updating valve assignments of the configuration corresponds to the update of business functions, e.g., updating the value of *min-quantity* changes the business rule of the function *pack items*. Second, updating operation assignments of the configuration corresponds to both 1) the update of business functions, e.g., updating the operation of *pack items* from op_2 to op_3 changes the functionality of it (by updating the attribute *status* of items in addition to *weight* of packages), and 2) the update of business objects (by modifying items to have a new attribute *status*).

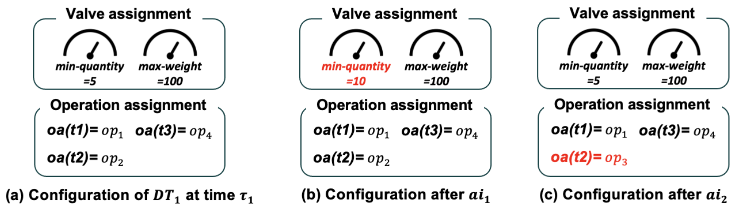


Fig. 4: A configuration of DT_1 at τ_1 and new configurations after applying ai_1 and ai_2

Definition 8 (Action). *Let DT be a DT-IM. An action $act \in \Sigma_{DT} \to \Sigma_{DT}$ updates the configuration. A_{DT} is the set of all possible actions defined over DT.*

An action instance describes the application of an action. An action is applied at a certain start time and, in principle, the configuration change that it entails can remain in place for the foreseeable future until a condition, e.g., on performance metrics, is met or until a specific end time. For simplicity, in this work, we consider only the latter case.

Definition 9 (Action Instance). *Let DT be a DT-IM. An action instance $ai \in A_{DT} \times \mathbb{U}_{time} \times \mathbb{U}_{time}$ is a tuple of an action, start timestamp, and end timestamp. AI_{DT} is the set of all possible action instances defined over DT.*

For instance, $ai_1{=}(act_1, \tau_1, \tau_2) \in AI_{DT_1}$ describes the application of $act_1 \in A_{DT_1}$ to the configuration of DT_1 at τ_1 (Fig. 4(a)) leading to the configuration depicted in Fig. 4(b) until τ_2. $ai_2 = (act_2, \tau_1, \tau_3) \in AI_{DT_1}$ describes the application of $act_2 \in A_{DT_1}$ to to the configuration of DT_1 at τ_1 (Fig. 4(a)), producing the configuration shown in Fig. 4(c) until τ_3.

The application of actions results in different *effective changes*, depending on the configuration at the start of the action instance. An effective change of an action instance denotes the valves and transitions whose values and activity assignments are changed due to the action.

Definition 10 (Effective Change). *Let DT be a DT-IM and $ai{=}(act, st, ct) \in AI_{DT}$ an action instance. An effective change of ai is a tuple of a set of valves and a set of transitions, i.e., $\delta_{ai}{=}(V_c, T_c)$ with $V_c{=}\{v{\in}V \mid \pi_{va}(conf_{DT,st})(v){\neq}\pi_{va}(act(conf_{DT,st}))(v)\}$ and $T_c{=}\{tr{\in}T \mid \pi_{oa}(conf_{DT,st})(tr){\neq}\pi_{oa}(act(conf_{DT,st}))(tr)\}$).*

As noted in Fig. 4(b) and Fig. 4(c) with red fonts, the effective change by ai_1 is valve *min-quantity*, i.e., $\delta_{ai_1}{=}(\{\textit{min-quantity}\}, \emptyset)$, and the effective change by ai_2 is the operation assignment of *t2*, i.e., $\delta_{ai_2}{=}(\emptyset, \{\textit{t2}\})$.

5 Impact Analysis

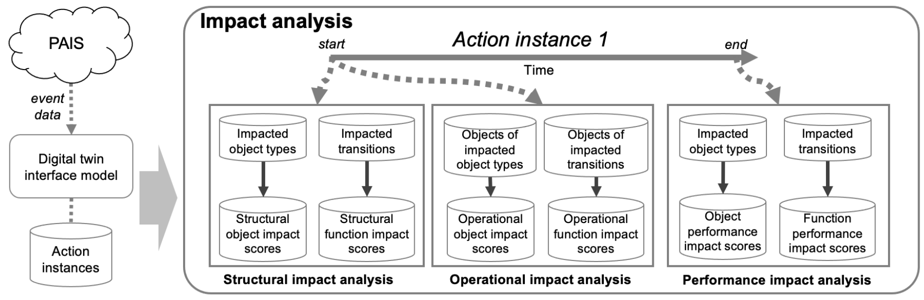

PAIS
event data
Digital twin interface model
Action instances
Impact analysis
start
Action instance 1
end
Time
Impacted object types
Impacted transitions
Structural object impact scores
Structural function impact scores
Objects of impacted object types
Objects of impacted transitions
Operational object impact scores
Operational function impact scores
Impacted object types
Impacted transitions
Object performance impact scores
Function performance impact scores
Structural impact analysis
Operational impact analysis
Performance impact analysis

Fig. 5: An overview of the proposed impact analysis using digital twin interface models

This section introduces an approach to impact analysis of PAIS updates based on digital twin interface models. Fig. 5 shows an overview of the proposed approach consisting of three components: structural/operational/performance impact analysis. Using a DT-IM representing a target PAIS, we analyze structural and operational impacts of an action instance, i.e., PAIS updates, at its execution and performance impacts at its completion.

5.1 Structural Impact Analysis

Structural impact analysis identifies the *structural object/function impacts* of an action instance. To this end, we identify the object types, i.e., business objects, and transitions, i.e., business functions, affected by an action instance. First, an object type is considered to be impacted if the operations newly assigned by an action instance introduce new attributes to update or remove existing attributes for the object type.

Definition 11 (Impacted Object Types). *Let DT be a DT-IM and $ai{=}(act, st, ct) \in AI_{DT}$ an action instance. $IOT_{DT}(ai) \subseteq \mathbb{U}_{ot}$ denotes the set of the object types impacted by ai, i.e., $IOT_{DT}(ai){=}\{ot \in \mathbb{U}_{ot} \mid \delta_{ai}{=}(V_c, T_c) \wedge tr \in T_c \wedge \pi_{oa}(conf_{DT,st})(tr)(ot) \triangle \pi_{oa}(act(conf_{DT,st}))(tr)(ot) \neq \emptyset\}\}$, where $\triangle$ denotes a symmetric difference of sets.*

For instance, $IOT_{DT_1}(ai_2){=}\{item\}$ since the effective change of ai_2 (i.e., $(\emptyset, \{t2\})$) in the operation assignment of *t2* introduces the new attribute *status* of *item*, i.e., $\pi_{oa}(conf_{DT_1,\tau_1})(t2)(item) \triangle \pi_{oa}(act_2(conf_{DT_1,\tau_1}))(t2)(item) \neq \emptyset$ (i.e., $\{status\} \triangle \emptyset \neq \emptyset$).

Next, the transitions of a DT-IM are considered to be impacted if they are associated with the effective change in valve assignments and operation assignments. First, changes in the valve assignment influence transitions by changing the meaning of the guard associated with them, e.g., changing the valve *min-quality* affects the guard of *pack items*. Second, changes in the operation assignment affect transitions by changing their functionality.

Definition 12 (Impacted Transitions). *Let $DT{=}(ON, V, A, G, O)$ be a DT-IM with $ON{=}(N, pt, F_{var})$. Let $ai \in AI_{DT}$ be an action instance. $IT_{DT}(ai) \subseteq T$ denotes the set of the transitions impacted by ai, i.e., $IT_{DT}(ai){=}\{tr \in T \mid \delta_{ai}{=}(V_c, T_c) \wedge ((\exists_{v \in V_c}\ v \in G(tr)) \vee tr \in T_c)\}$.*

For instance, $IT_{DT_1}(ai_1){=}\{t2\}$ since the effective change of ai_1 (i.e., $(\{min\text{-}quantity\}, \emptyset)$) in valve *min-quantity* affects the guard associated to *t2*. Moreover, $IT_{DT_1}(ai_2){=}\{t2\}$ since the effective change by ai_2 (i.e., $\delta_{ai_2}{=}(\emptyset, \{t2\})$) change the functionality of *t2*.

Once the impacted object types/transitions are identified, various structural object/function impact scores can be measured. In this work, we focus on basic count-based measures, e.g., how many object types and transitions are impacted. These measures can be absolute or relative, i.e., normalized by the total number of respective entities.

Additional measures can be obtained by applying filtering or prioritizing to count-based measures. For instance, by filtering *financial* objects, such as *invoice*, we can measure the absolute/relative impact on objects that are relevant for the finance department. Prioritizing refers to weighting differently the impact on different types of entities. For instance, a higher weight can be given to the impacts on *verification*-related activities in a process.

5.2 Operational Impact Analysis

Operational impact analysis aims to analyze *operational object/function impacts* of an action instance. To that end, we identify the existing objects of the impacted object types (for the former) and the objects related to the impacted transitions (for the latter) in a DT-IM. First, to identify the existing objects of the impacted object types, we use markings from the operational states of the DT-IM.

Definition 13 (Objects of Impacted Object Types). *Let* $DT{=}(ON, V, A, G, O)$ *be a DT-IM with* $ON{=}(N, pt, F_{var})$*. Let* $ai{=}(act, st, ct) \in AI_{DT}$ *be an action instance and* $IOT_{DT}(ai)$ *the impacted object types by ai.* $\widehat{IOT}_{DT}(ai) \subseteq \mathbb{U}_{oi}$ *denotes the set of objects of* $IOT_{DT}(ai)$*, i.e.,* $\widehat{IOT}_{DT}(ai){=}\{oi \in \mathbb{U}_{oi} \mid p \in dom(pt) \wedge pt(p) \in IOT_{DT}(ai) \wedge (p, oi) \in \pi_M(os_{DT,st})\}$*.*

For instance, $\widehat{IOT}_{DT_1}(ai_2)$ is a set of objects associated with all tokens in *item* places, i.e., $i1, i2, \ldots$ of marking $\pi_M(os_{DT_1,\tau_1})$.

Next, we identify objects related to impacted transitions. An object is related to a transition if it may perform the transition in the future.

Definition 14 (Objects of Impacted Transitions). *Let DT be a DT-IM. Let* $ai{=}(act, st, ct) \in AI_{DT}$ *be an action instance and* $IT_{DT}(ai)$ *the impacted transitions by ai.* $\widehat{IT}_{DT}(ai) \subseteq \mathbb{U}_{oi}$ *denotes the set of objects of* $IT_{DT}(ai)$*, i.e.,* $\widehat{IT}_{DT}(ai){=}\{oi \in \mathbb{U}_{oi} \mid tr \in IT_{DT}(ai) \wedge p \in rel(tr) \wedge (p, oi) \in \pi_M(os_{DT,st})\}$*.*

For instance, $\widehat{IT}_{DT_1}(ai_1)$ is a set of objects associated with all tokens in $rel(t2){=}\{i1, i2, o1, pk1\}$ of marking $\pi_M(os_{DT_1,\tau_1})$.

Based on the objects of impacted object types/transitions, we measure operational object/function impact scores. As for the structural impact analysis, in this work, we focus on basic count-based measures, e.g., how many objects of impacted object types/transitions are impacted by an update.

Also in this case, we can apply filtering or prioritizing to define new measures. For instance, objects can be filtered based on the value of specific attributes or the stage of their lifecycle, e.g., orders higher than a certain amount or from premium customers, or objects for which payments have been cleared. Regarding objects of impacted transitions, we can filter or prioritize objects that lie directly in the queue for the impacted transition, e.g., giving a higher weight to objects currently waiting for the impacted transition.

5.3 Performance Impact Analysis

Performance impact analysis aims at analyzing *object/function performance impacts*. First, to analyze the former, we compare diagnostics related to impacted object types before and after applying an action instance.

We define the object performance impact analysis as follows. Let $ai{=}(act, st, ct)$ be an action instance and $iot \in IOT_{DT}(ai)$ an impacted object type. Let $diag_{iot} \in \mathit{\Delta}_{DT}$ be a diagnostics relevant to iot, e.g., the average total service time of iot. We measure the performance impact of ai on iot w.r.t. $diag_{iot}$ as follows: $\pi_{dmap}(os_{DT,ct})(diag_{iot}) - \pi_{dmap}(os_{DT,st})(diag_{iot})$

Next, we analyze function performance impacts by comparing diagnostics associated with impacted transitions before and after applying an action instance. Examples of relevant diagnostics are the average service time of the transition, or the average waiting time of the transition.

We formally define the function performance impact analysis as follows. Let $ai{=}(act, st, ct)$ be an action instance and $it \in IT_{DT}(ai)$ an impacted transition. Let $diag_{it} \in \Delta_{DT}$ be a diagnostics relevant to it, e.g., the average service time of it. We measure the performance impact of ai on it w.r.t. $diag_{it}$ as follows: $\pi_{dmap}(os_{DT,ct})(diag_{it}) - \pi_{dmap}(os_{DT,st})(diag_{it})$.

In this work we consider general purpose diagnostics, such as the average total service time of impacted object types, or the average total waiting time of impacted object types. Other diagnostics can be defined applying the filtering and prioritizing principles introduced earlier, e.g., considering the average total waiting time of objects of a certain type, or giving more weight to the waiting time of certain types of objects. Diagnostics can also be defined on a domain-specific basis, e.g., process-specific KPIs.

6 Evaluation

This section presents the implementation of the approach presented in this paper and evaluates its feasibility by applying it to a simulated PAIS.

6.1 Implementation

We have implemented a cloud-based Web service to support the impact analysis with a dedicated user interface. Sources, manuals, and a demo video are available at `https://github.com/gyunamister/impacta`. The service comprises the following four functional components:

Designing DT-IMs. This component supports the design of DT-IMs based on event data and domain knowledge. The input is event data of the standard OCEL [11], valves, guards, and operations in a JSON-based format. The event data are used to discover an OCPN using the technique introduced in [1], and valves, guards, and operations enhance the discovered OCPN, completing the design of a DT-IM.

Updating Configurations and Operational States. This component updates the configuration and operational state of a DT-IM in sync with the updates in a target PAIS. To this end, it is connected with the PAIS, specifying: 1) the source of the current setting of the PAIS and 2) the source of the streaming event data from the PAIS. Using the current setting, the configuration of the DT-IM is updated. Then, the operational state is updated by replaying the streaming event data using the token-based replay technique described in [3].

Defining and Executing Actions. The goal of this component is to 1) define actions based on the available valves and operations and 2) instantiate them as action instances by specifying start and completion times. To this end, the service provides visual information to support the definition of actions and action instances. Once executed, an action instance changes the configuration setting of the system and, accordingly, the configuration of the DT-IM.

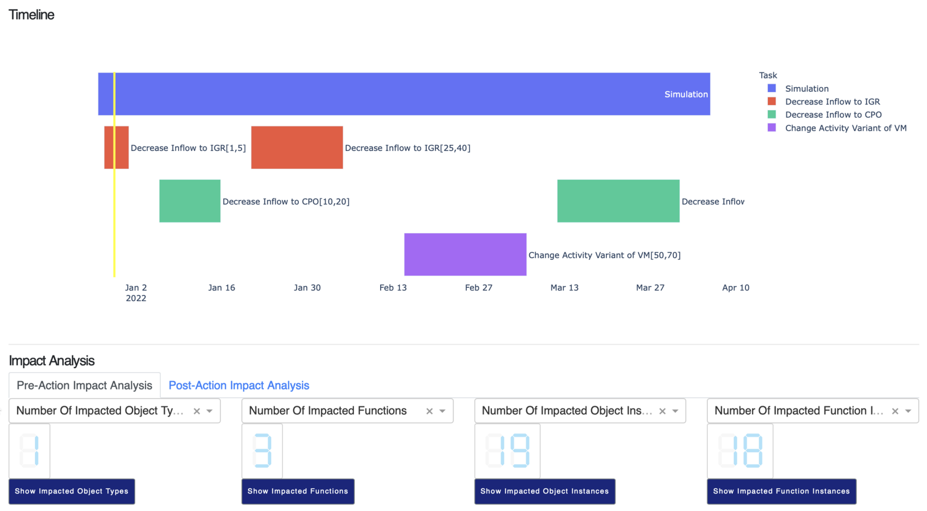

Timeline
Simulation
Decrease Inflow to IGR[1,5]
Decrease Inflow to IGR[25,40]
Decrease Inflow to CPO[10,20]
Decrease Inflov
Change Activity Variant of VM[50,70]
Jan 2
2022
Jan 16
Jan 30
Feb 13
Feb 27
Mar 13
Mar 27
Apr 10
Task
Simulation
Decrease Inflow to IGR
Decrease Inflow to CPO
Change Activity Variant of VM
Impact Analysis
Pre-Action Impact Analysis
Post-Action Impact Analysis
Number Of Impacted Object Ty…
Number Of Impacted Functions
Number Of Impacted Object Ins…
Number Of Impacted Function I…
1
3
19
18
Show Impacted Object Types
Show Impacted Functions
Show Impacted Object Instances
Show Impacted Function Instances

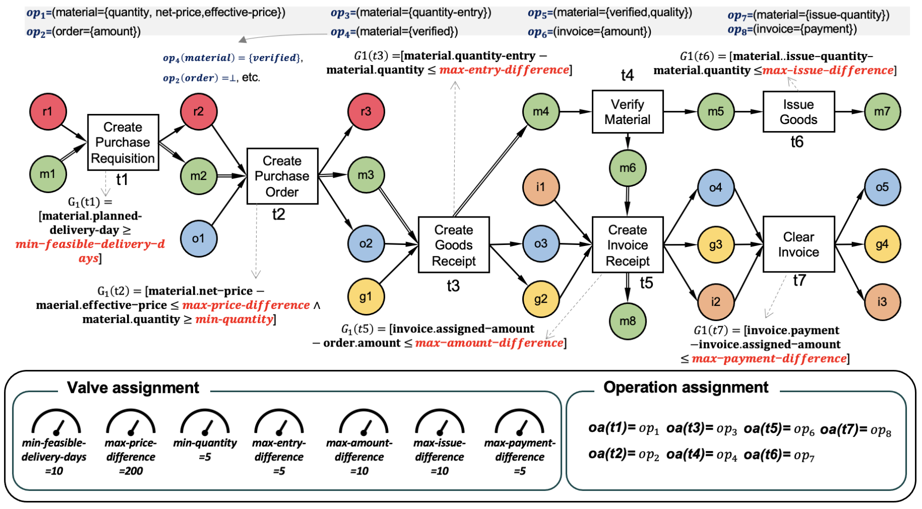
Fig. 6: Screenshot of the implementation

Analyzing Structural/Operational/Performance Impacts. This component evaluates the impact of action instances. Fig. 6 shows a screenshot of the Web service's interface. The timeline in the upper part shows the current status, including the current timestamp (yellow vertical line) and the overview of the action instances. Specifically, five action instances (horizontal bars) of three different actions (distinguished by colors) are scheduled to be executed. For instance, the first action instance named *Decrease Inflow to IGR* is effective from 26-12-2021 to 31-12-2021. Note that we compute structural/operational impacts at the start of action instances and performance impacts at the end of them.

6.2 Case Study

Using the implementation, we have conducted a case study on an artificial PAIS that supports a procure-to-pay process. The system is developed to reflect a real-life SAP ERP system supporting a procure-to-pay processes by using the same business objects, functions, and rules found in SAP. Using the artificial PAIS, we simulate the procure-to-pay process with 24 resources with different capacities and performance. Purchase orders are created by following the exponential distribution; the business hours are set as 9-17 from Monday to Friday; work assignments are scheduled using the *First-in-First-out* rule.

Fig. 7(a) shows the DT-IM representing the PAIS (DT_{p2p}). The process involves five object types. First, a purchase requisition is created with multiple materials. Next, a purchase order is created based on the purchase requisition and material. A goods receipt is produced after receiving the materials of the purchase order. Afterward, the material is verified and issued for various purposes, and concurrently the invoice for the purchase order is received. Finally, the invoice is cleared. Fig. 7(b) shows the default configuration of the system.



op_1=(material={quantity, net-price,effective-price})
op_2=(order={amount})
op_3=(material={quantity-entry})
op_4=(material={verified})
op_5=(material={verified,quality})
op_6=(invoice={amount})
op_7=(material={issue-quantity})
op_8=(invoice={payment})
$op_4(material) = \{verified\}$,
$op_2(order) = \perp$, etc.
$G1(t3)$ =[material.quantity-entry − material.quantity ≤ max-entry-difference]
$G1(t6)$ = [material..issue-quantity− material.quantity ≤max-issue-difference]
r1
m1
Create Purchase Requisition
t1
r2
m2
o1
Create Purchase Order
t2
r3
m3
o2
g1
Create Goods Receipt
t3
m4
Verify Material
t4
m5
Issue Goods
t6
m7
m6
i1
o3
g2
Create Invoice Receipt
t5
m8
o4
g3
i2
Clear Invoice
t7
o5
g4
i3
$G_1(t1)$ = [material.planned-delivery-day ≥ min-feasible-delivery-days]
$G_1(t2)$ = [material.net-price − maerial.effective-price ≤ max-price-difference ∧ material.quantity ≥ min-quantity]
$G_1(t5)$ = [invoice.assigned-amount − order.amount ≤ max-amount-difference]
$G1(t7)$ = [invoice.payment −invoice.assigned-amount ≤ max-payment-difference]
Valve assignment
min-feasible-delivery-days =10
max-price-difference =200
min-quantity =5
max-entry-difference =5
max-amount-difference =10
max-issue-difference =10
max-payment-difference =5
Operation assignment
oa(t1)= op_1 oa(t3)= op_3 oa(t5)= op_6 oa(t7)= op_8
oa(t2)= op_2 oa(t4)= op_4 oa(t6)= op_7

Fig. 7: A digital twin interface model of the PAIS supporting a procure-to-payment process and its configuration

Using the DT-IM and configuration, we define actions, A_1 and A_2, as follows:

- A_1 increases valve *min-quantity* to 10 to reduce the inflow to *create purchase order*, i.e., for any $conf_{DT_{p2p}} \in \Sigma_{DT_{p2p}}$, $A_1(conf_{DT_{p2p}})=conf^{A_1}_{DT_{p2p}}$ such that $\pi_{va}(conf^{A_1}_{DT_{p2p}})(min\text{-}quantity)=10$, and
- A_2 changes the operation of *verify materials* to op_5 to additionally update *quality* of materials, i.e., for any $conf_{DT_{p2p}} \in \Sigma_{DT_{p2p}}$, $A_2(conf_{DT_{p2p}})=conf^{A_2}_{DT_{p2p}}$ such that $\pi_{oa}(conf^{A_2}_{DT_{p2p}})(t4)=op_5$, where $op_5(material)=\{verified, quality\}$.

Using the actions, we define the following action instances: $AI_1=(A_1, 1, 5)$ and $AI_2=(A_2, 10, 15)$. Note that, for the ease of the simulation, we abstract the timestamp to time steps each of which has the scale of 24 hours, starting from 09:00 20-12-2021. For instance, AI_1 is effective from 09:00 21-12-2021 to 09:00 25-12-2021.

Table 1: Results of the impact analysis on AI_1 and AI_2

Impact	Metric	AI 1	AI 2
Structural object impact	Total number of impacted business objects	0	1
Structural function impact	Total number of impacted business functions	1	1
Operational object impact	Total number of object instances of the impacted business objects	0	254
Operational function impact	Total number of object instances of impacted business functions	51	153
Object performance impact	Difference in avg. total service time for purchase orders	-4m	-
Object performance impact	Difference in avg. total service time for materials	-	1.4h
Function performance impact	Difference in avg. sojourn time of create purchase order	-9m	-
Function performance impact	Difference in total number of purchase orders	-13	-
Function performance impact	Difference in avg. sojourn time of verify material	-	1.6h

Table 1 shows the result of the impact analysis on AI_1 and AI_2. The action instance AI_1 affects one transition, i.e., *create purchase order* and 51 running objects related to it. As a result of the action, the average total service time of purchase orders has been improved by 4 minutes, while the average sojourn time of *create purchase order* has been reduced by 9 minutes. This is due to the decrease in the queue for the activity, resulting from the new business rule setting the higher minimum quantity for creating purchase orders. Moreover, the number of purchases has been reduced by 13 during the execution of the action.

The action instance AI_2 affects one object type and one transition, i.e., *material* and *verify material*. Besides, 254 running objects of *material* and 153 objects of *verify material* have been affected by the action. As a result of the action, the average total service time of purchase orders has increased of 1.4 hours, while the average sojourn time of *verify material* has increased of 1.6 hours.

7 Conclusions

In this paper, we proposed an approach to impact analysis of PAIS updates based on a DT-IM. PAIS updates are modeled as updates of the configuration in a DT-IM. Next, we identify PAIS entities impacted by PAIS updates and measure the structural/operational/performance impacts based on such entities. We have implemented the approach as a Web application and discussed a case study on a standard Procure-to-Pay process.

The proposed approach has several limitations. First, the identification of objects related to impacted business object types and transitions is limited to existing objects in the process and the future objects entering the process are not considered. Second, we identify objects related to impacted transitions, including all objects that potentially execute the transition. However, some objects may bypass the transition, e.g., a patient expected to perform surgery may die before it, or a doctor may decide for an emergency treatment at the last moment. Finally, the performance impact analysis does not isolate the objects subject to action instances to evaluate the performance impact, instead indirectly evaluating changes in diagnostics over all existing objects in the process.

Besides addressing the above limitations, as future work, we plan to extend the approach to predict the performance impact to provide timely and accurate information before the execution of any update. Another direction of future work is to improve the performance impact analysis such that it completely isolates the instances affected by changes to provide more realistic performance impact measures. Finally, we also plan to evaluate the proposed approach's ease of use and usefulness with business analysts in real-world situations.

References

1. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1-4), 1–40 (2020)

2. van Beest, N.R., Kaldeli, E., Bulanov, P., Wortmann, J.C., Lazovik, A.: Automated runtime repair of business processes. Information Systems **39**, 45–79 (2014)
3. Berti, A., van der Aalst, W.M.P.: A novel token-based replay technique to speed up conformance checking and process enhancement. Trans. Petri Nets Other Model. Concurr. **15**, 1–26 (2021)
4. Caporuscio, M., Edrisi, F., Hallberg, M., Johannesson, A., Kopf, C., Perez-Palacin, D.: Architectural concerns for digital twin of the organization. In: European Conference on Software Architecture. pp. 265–280. Springer (2020)
5. Comuzzi, M., Parhizkar, M.: A methodology for enterprise systems post-implementation change management. Industrial Management & Data Systems (2017)
6. Comuzzi, M., Vonk, J., Grefen, P.: Measures and mechanisms for process monitoring in evolving business networks. Data & knowledge engineering **71**(1), 1–28 (2012)
7. Döhring, M., Reijers, H.A., Smirnov, S.: Configuration vs. adaptation for business process variant maintenance: an empirical study. Information Systems **39**, 108–133 (2014)
8. Dumas, M., van der Aalst, W.M.P., Hofstede, A.H.T.: Process-aware information systems: bridging people and software through process technology. John Wiley & Sons (2005)
9. Eramo, R., Bordeleau, F., Combemale, B., van Den Brand, M., Wimmer, M., Wortmann, A.: Conceptualizing digital twins. IEEE Software (2021)
10. Gelernter, D.: Mirror worlds: Or the day software puts the universe in a shoebox... How it will happen and what it will mean. Oxford University Press (1993)
11. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: A standard for object-centric event logs. In: New Trends in Database and Information Systems. vol. 1450, pp. 169–175 (2021)
12. Kumara, I., Han, J., Colman, A., van den Heuvel, W.J., Tamburri, D.A.: Runtime evolution of multi-tenant service networks. In: European Conference on Service-Oriented and Cloud Computing. pp. 33–48 (2018)
13. La Rosa, M., Dumas, M., Ter Hofstede, A.H., Mendling, J.: Configurable multi-perspective business process models. Information Systems **36**(2), 313–340 (2011)
14. Lin, Y.Y., Nagai, Y., Chiang, T.H., Chiang, H.K.: Succerp: The design science based integration of ecs and erp in post-implementation stage. International Journal of Engineering Business Management **13**, 18479790211008812 (2021)
15. Marrella, A.: Automated planning for business process management. Journal on data semantics **8**(2), 79–98 (2019)
16. Mendling, J., Pentland, B.T., Recker, J.: Building a complementary agenda for business process management and digital innovation (2020)
17. Parhizkar, M., Comuzzi, M.: Impact analysis of erp post-implementation modifications: Design, tool support and evaluation. Computers in Industry **84**, 25–38 (2017)
18. Park, G., van der Aalst, W.M.P.: Realizing a digital twin of an organization using action-oriented process mining. In: ICPM 2021. pp. 104–111 (2021)
19. Soffer, P., Golany, B., Dori, D.: Erp modeling: a comprehensive approach. Information systems **28**(6), 673–690 (2003)
20. Soffer, P., Golany, B., Dori, D.: Aligning an erp system with enterprise requirements: An object-process based approach. Computers in industry **56**(6), 639–662 (2005)
21. Wieringa, R.J.: Design science methodology for information systems and software engineering. Springer (2014)