A migration framework for active BPMN processes in healthcare

Matteo Mantovani
Department of Computer Science
University of Verona - Italy
matteo.mantovani@univr.it

Emanuele Chini
Department of Computer Science
University of Verona - Italy
emanuele.chini@univr.it

Carlo Combi
Department of Computer Science
University of Verona - Italy
carlo.combi@univr.it

***Abstract*—The Business Process Modeling Notation (BPMN) is a diagrammatical notation to describe complex process models, and it can be used as a common language among stakeholders. These stakeholders could intervene in the development of the processes, also in an agile environment. Thus, the same process could evolve through different versions over time. BPMN has been already adopted in the healthcare domain, where designing healthcare processes is fundamental for delivering optimal and efficient services to patients without overburdening healthcare professionals. Adapting and updating BPMN processes within an agile development is paramount in the rapidly evolving healthcare domain. The challenges of migrating a process to its revised versions have been discussed since the introduction of BPMN. However, there is a lack of migration policies that consider compensation strategies when migrating to a revised version at runtime, combined with healthcare-related migration risk classes.**

In this paper, we propose a methodological framework that includes migration strategies to adopt when the user is already running the process in a previous version. We propose compensatory strategies to integrate the novelties of the revised process based on the actual users' completion status, with a specific focus on the addition, modification, and removal of tasks. Moreover, the proposed framework includes a color-coded risk classification system encapsulating migration risks and potential impact on patients. This system provides a visual and intuitive way to understand potential risks, enabling clinicians to make informed decisions about the migration strategy. Finally, we showed an application of the proposed framework in a real-world scenario, that is, through an ERAS-inspired prehabilitation program for pancreatic surgery currently developed at the Verona Pancreas Institute.

***Index Terms*—BPMN, BPMN framework, agile methodology, migration, BPMN migration, versioning**

I. Introduction

Nowadays, software applications are increasingly complex, and their realization involves stakeholders from heterogeneous fields of expertise. Therefore, it is fundamental to find a "common language" for them to communicate, and the Business Process Model and Notation (BPMN) is a standard used by many organizations to design their processes in a user-friendly way. BPMN allows the development of process-driven applications, which have shown to be particularly efficient in an agile environment, where continuous feedback at each iteration leads to a flexible and rapid response when a change in the requirements occurs. However, this dynamic environment needs the support of proper migration strategies to move from a process to its updated version. The first instances of process migration were discussed in [1] and [2]. Weber et al. extended these concepts and proposed a methodology to change patterns and support features in process-aware information systems [3], [4]. The agile environment's highly dynamic nature exacerbates the need for proper migration strategies because of their intrinsic frequent updates and revisions.

While BPMN was initially conceived to describe and define business processes, its application has rapidly expanded to the healthcare domain. In fact, with BPMN, we could define various kinds of clinical processes (e.g., treatment protocols) with the critical goal of delivering optimal and efficient services to patients without overburdening physicians, nurses, and the system in general. The study by Pufahl et al. [5] identified common challenges of process modeling in healthcare, and presented some BPMN best practices to guide the standard modeling of complex healthcare aspects.

Although some clinical processes could be well-established and subject to very few changes over the years, we may also find situations where fast prototyping is very common. For example, clinicians want to use a process-driven application to monitor patients in a clinical trial or for the digitalization of processes already in use. An example of this can be found in [6], where an agile process-driven application was developed with the continuous intervention of clinicians in order to digitalize an ERAS-inspired prehabilitation program. However, this solution lacked of migration strategies to allow patients already on the program to move to a revised part of it. Nevertheless, even if we have a technical methodology to perform a migration, we should always consider the patient's health status. Because of that, we have to introduce compensatory strategies to adjust for the actions already performed and the changes in their current path of the process to the new one. The concept of compensatory strategies was discussed in [7] for web applications, but it was not applied in the healthcare domain. Finally, clinicians could decide that there may be part of the process where the migration could be riskier for the patient, and this should be addressed when deciding the best migration strategy for a patient. This highlights the need for further research and development in this area.

In this paper, we propose a methodological framework that includes compensatory strategies to allow the migration while mitigating its effects on the patients conditions, and a color-coded risk classification system to migrate within BPMN

processes in healthcare. This system provides a visual and intuitive way to understand potential risks, enabling clinicians to make informed decisions about the migration strategy. The framework also considers the user completion status, as any change in the process can directly affect the quality of care and patient outcomes. Therefore, we propose various compensatory strategies to ensure that migration is possible and provides the highest benefit for the user. We show the potentiality of this framework using a real-world scenario, specifically the process-driven ERAS-inspired web application for pancreatic surgery initially used in [6].

This paper is structured as follows: Section II describes the background and the related literature; Section III presents the migration framework for BPMN processes, focusing on (i) compensatory strategies for the addition, modification, and removal of tasks in a process, and (ii) a color-coded risk classification system; Section IV shows a practical application of this framework, using a real-world example given by a process-driven application for pancreatic surgery; and finally, Section V outlines the final observations and highlights some future research directions.

II. Background and Related Work

In this section, we will describe the basic concepts used in the paper, such as the agile development methodology, BPMN, process migration, and the ERAS program. Then, the current works in the literature are exposed.

A. Agile Methodology

The principles for the *agile development methodology* were proposed in 2001 in the *Manifesto for the Agile Software Development* [8]. The 12 founding principles value collaboration, interactions among stakeholders, and responses to changes. The key idea consists of developing software through an iterative and incremental technique, where the software is created in small steps. In this iterative and incremental approach, each step improves and refines over the previous one. In each step, the stakeholders are involved, and their feedback is fundamental for the improvements in the next step. This approach provides endless feedback that increases the flexibility and the response to the new customer's requirements.

B. Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) is the OMG standard for representing business processes [9], and is important because it is readily understandable by all users. One of the main elements is the *task*, depicted as a rectangle, which represents an atomic activity.Different specialized tasks exist, such as the *User Task* (performed by a human) or the *Service Task* (performed by some script). Then, there are the *events*, designed as a circle, which represent things that happen instantaneously and can influence the flow of the process (e.g., processes *Start* and *End*). The *arcs* define the execution order of the process elements. They impose temporal constraints between flow objects and establish the sequence flow. The last main element is the *gateway*, defined as a diamond, that controls the process flow splitting or merging points. The most common gateways are the *Exclusive* gateways, that split the flow in different paths and choose one given a specific condition (i.e., an XOR), and the *Parallel* gateways, where all the outgoing flows are followed, and in the merging, all the activities of the incoming flows must be completed before continuing (i.e., an AND). Finally, the *token* concept is used to identify the progress and comprehend the state of the process. The *token* can be defined as a "theoretical concept that is used as an aid to define the behavior of a process that is being performed and it "traverses" its structure. A Start Event generates a token that MUST eventually be consumed at an End Event." [9].

C. Process Migration

Process instance migration is the act of moving a process instance from one version to another while it is still running. This migration, which can involve one or more instances, requires careful planning to prevent issues such as inactivity, data loss, duplication, missing I/O, and inconsistencies. Before initiating the migration, it is crucial to identify potential problem areas where unwanted effects are likely to occur and to determine the smoothest transition points to avoid errors. The ultimate goal is to seamlessly transfer one or more instances from the previous version to the next one without issues. In essence, migration is vital in enhancing process flexibility and ensuring its alignment with real-world scenarios. It allows for the continuous improvement and adaptation of processes, thereby increasing their efficiency and effectiveness.

D. ERAS

The Enhanced Recovery after Surgery (ERAS) Programs, also known as enhanced recovery programs (ERPs), were developed to improve patient care, reduce complication rates, and shorten hospital stays following surgery [10]. Since its first application, the ERAS programs have been widespread in different surgeries, and pancreatic is no exception. It was demonstrated that these protocols help to reduce the hospital's length of stay and to increase the patient's quality of life without compromising morbidity and mortality. This was proven true for distal and total pancreatectomy. Furthermore, pancreaticoduodenectomy morbidity rates were significantly lower for patients managed according to ERAS principles [11].

E. Related Work

In the literature, versioning and migration are not new terms. For example, Weber et al. [3], [4] explored the different types of process instances and schema modifications, called adaptation patterns. Processes can be modified at runtime together with the version controls in process-aware information systems. Furthermore, versioning has been studied in [12], where the authors examined an approach to facilitate version control for business process schema evolution, with an emphasis on version compatibility, dynamic version shifts, and co-existence of multiple versions thanks to a preserving

graph model. A framework for adaptive process management is proposed in [13], which avoids semantic errors (e.g., drug incompatibilities) in real-world scenarios preventing conflicts before applying changes. Barkhordarian et al. [14] have elaborated a migration module capable of stopping a process instance in such a way as to be able to switch from one version to another. They have particularly focused on event cases (throw and catch), proposing a three-level classification that is neither deepened nor formalized. Rosemann et al. [15] argue for an explicit consideration of contextual factors in the design of business processes to make them more adaptive and propose a framework for classifying relevant contexts. They explore context conceptualization, classification, and integration with established business process modeling methodologies using a goal-oriented process modeling approach that can identify relevant context elements. Furthermore, a compensatory approach in Web service driven by BPEL processes is proposed in [7]. This work applies a particular adaptation approach since processes in production need to compensate for strategies that are included as a result of an adaptation step. Since Business Process Management (BPM) still requires a full integration with the agile development methodology, a framework for agile BPM is introduced in [16].

Tomaskova et al. [17] showed that, in the last decade, the adoption of BPMN in healthcare increased, and processes were used to describe heterogeneous aspects of the discipline. In fact, BPMN is also used for process-driven applications in healthcare, such as the one proposed in [6]. Here, the authors translated an ERAS-inspired program for pancreatic surgery into BPMN and used it to guide a web application that helps patients prepare for surgery. The result is an agile process-driven application, named *Fit to Fight*, developed in collaboration with the *Unit of Pancreatic Surgery – Verona Pancreas Institute* [18], the first interdisciplinary center of excellence for diagnosing and treating pancreatic diseases in Italy with more than 3000 pancreaticoduodenectomies since 1990. Although the agile development methodology [8] has been used to develop *Fit to Fight*, there were no migration strategies to change the process model at runtime.

Ultimately, the migration strategies currently proposed may be incomplete for the healthcare domain. Indeed, the migration could lead to consequences for the patient's health status. For this reason, it is mandatory to detect proper migration points decided not only for technical reasons but also for clinical ones. Finally, compensatory strategies are also required to allow the migration while mitigating its effects on the patient's conditions.

In this paper, we propose a migration framework for BPMN healthcare processes already in use within an agile development, thus subject to continuous changes. Because of that, we propose migration strategies to perform when the user is already running the process in a previous version. We propose compensatory methods to integrate the novelties of the revised process based on the current user completion status. In detail, we show different compensatory strategies based on the combination of completion status and the changed BPMN elements, such as new tasks. Moreover, we focus on healthcare processes where we must consider not only the technical challenges of the migration but also the impact of such migration for the users (i.e., patients) involved. This issue is addressed by also proposing a color-coded risk class that combines both aspects. Finally, we show an application of the proposed framework in a real-world scenario with an ERAS-inspired prehabilitation program used for pancreatic surgery.

III. Design Considerations - The Migration Framework

To motivate our proposal, in Figure 1 we show an example of a basic BPMN process model in healthcare. The patient starts with the *heart rate measurement* before the *first visit* with the clinician. Then, the clinician decides the proper therapy for the patient (either *Therapy 1* or *Therapy 2*). Finally, the clinician performs a *follow-up visit* to the patient to evaluate the success of the therapy. However, within the agile approach, clinicians could re-evaluate the current process model and decide that it needs some revision for many different reasons, such as new scientific evidence or new needs. For example, it may be necessary to also perform a *temperature measurement* before the *first visit*. *Therapy 1* has now two more cycles, and a new therapy, *Therapy 3*, is available. In this scenario, we could imagine that many patients are already running the first - not revised - process and at different stages of completion. It is unreasonable to have them start the whole revised process from the beginning again, but it also may be undesirable to have the patients end with the old process.

Here, we propose a methodological framework that includes different strategies to migrate patients to a new process model based on their current state within the current process. The framework considers that certain elements could be added, others could be removed or modified. Therefore, the framework is based on these key elements:

- Compensatory strategies - These are necessary to migrate the current process to its new adapted version. These strategies help to adjust the process based on the actual user instance.
- Interpretation of atomic set of tasks - When multiple tasks must be executed as a unique task that cannot be divided within the process execution, we should differentiate their migration strategies.
- Migration clinical risk - The migration involves both the process schema and its instances. We use color-coded risk classes to identify when it is recommended to migrate or not.

Hereinafter, we generalize the example illustrated in Figure 1 at a higher level of abstraction, as shown in Figure 2, where each specified task is mapped into a generic letter. This will allow us to better explain all the different strategies proposed by our migration framework.

A. Compensatory strategies

When a process model is revised, all the elements from start to end could be subject to any modification. This could include

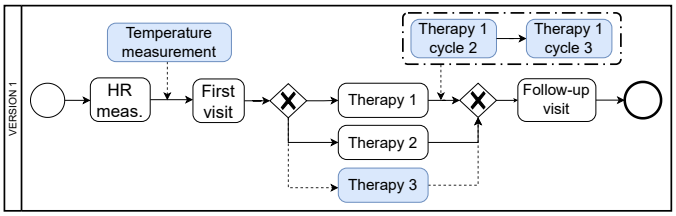


Fig. 1: Example of a BPMN process model in healthcare. Here, we have four different tasks - in blue - that have been added in a revised version of this process.

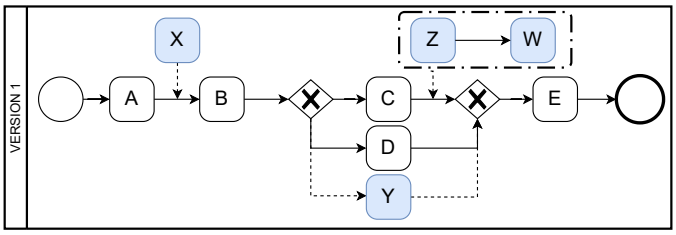


Fig. 2: A generalization of Figure 1 used to better explain the different aspects of the proposed framework.

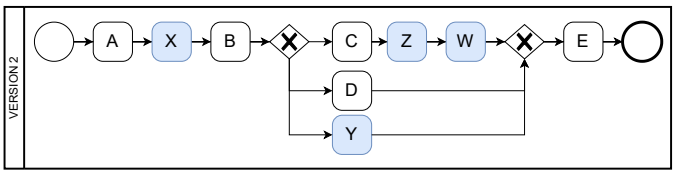


Fig. 3: The revised process from Figure 2, where the new tasks are inserted in the process model.

adding, modifying, or removing any task from the process. However, if the users already run the previous process, we may want to migrate them (i.e., their instances of the process model) to the new one. Nevertheless, the users' migration depends heavily on their process completion status, and compensatory strategies could be required to integrate changes made in previous parts of the process. In the following, we introduce a set of compensatory strategies that one should use to migrate when tasks are added, changed, or removed. These strategies also evaluate the patient/user completion status and the BPMN elements where the change occurs. Here, we consider changes to tasks that also involve exclusive and parallel gateways.

The first set of compensatory strategies involves all the tasks that are *added* in the revised process model. In Figure 2, all the tasks with a blue background (i.e., tasks X, Y, X, and W) need to be added in the revised process model. In Figure 3, those tasks are added to their desired position. In our compensatory strategies, the general idea is that all the added tasks after the current completion status (i.e., represented by a token in the token-based semantics of process execution [19]) do not represent a particular issue, as we still have to complete that part. However, we also need to integrate and adapt the tasks that are added to the parts of the process model that have been already completed, as they may be necessary for the proper execution of the remaining part of it.

In the healthcare domain, compensatory actions can be used to provide training and/or knowledge and to give or collect the information the patient or clinician needs to conclude the process positively. In the following, we will use Figures 2, 3, and some of their variations to provide an example of a process with all the different scenarios in which the compensatory actions are needed.

1) Plain sequence of tasks:

This scenario, depicted in Figure 4, is the simplest one but useful to introduce the concept of "compensatory strategies". In fact, here, we only have a set of tasks subsequent to their previous one. In the migration process to the revised version, shown in Figure 5, the users have to complete first the new tasks added before their token and only then they can proceed to complete the rest of the process (Figure 6). As seen in Figure 6b, in a situation where the token is already on task B, we need a compensatory strategy to execute task X as it may be necessary to have all the elements to complete the remaining part of the process. However, task B can be completed without additional knowledge coming from X, as B can be done after A in the first version of the process without compromising its outcome. If X had added new data required for B, then B would have been changed to a new task named, e.g., B'. The relevance of task order will become more evident with complex process models that include, for example, gateways.

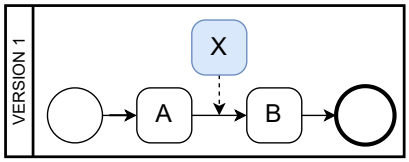


Fig. 4: Plain sequence of tasks, where task X is the task added in the revised version.

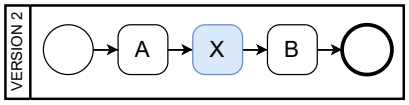
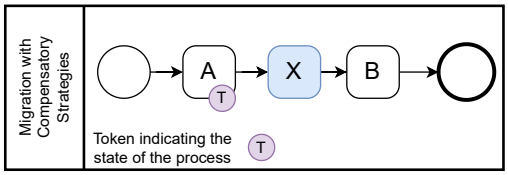
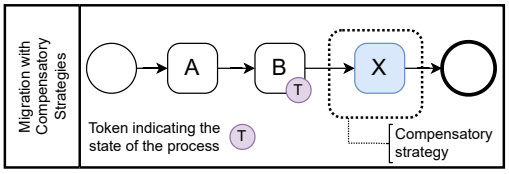


Fig. 5: The revised process from Figure 4, where the new tasks are added in the process.



(a) No actions needed if the token is in A, as X would already have been its natural successor in the execution of the process.



(b) After the completion of task B, we want to perform task X.

Fig. 6: Compensatory strategy for the migration to the revised process shown in Figure 4, based on the position of the token.

2) Exclusive gateway:
A more complex scenario is represented by a series of tasks with the addition of an exclusive gateway. This is our reference case, shown before in Figures 2 and 3.

In the following, we present different cases based on the completion status of the process (i.e., the position of the token):

(A) As shown in Figure 7, when the token is at the beginning of the process on A, the migration is flawless as we are at the beginning of the process, and no compensatory actions have to be done.

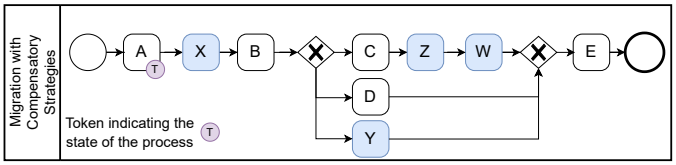


Fig. 7: Compensatory strategy are not needed when the token is in task A.

(B) If the token is in B, we need to complete the new task X before proceeding any further, as shown in Figure 8.
In fact, in the revised process the split gateway's conditions have been changed to add another possible pathway. Because of that, the process may also need the data from task X to send the user to the right path correctly. This behavior stresses the concepts expressed before; that is, to migrate to the revised version, the user advances only by completing the tasks added before its token.

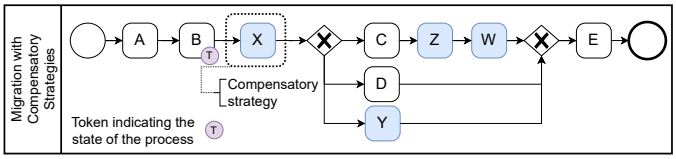


Fig. 8: Compensatory strategy when the token is in B.

(C) The token is in C. Here, the compensatory strategy is more complex. In fact, in the revised version tasks Z and W are executed right after task C. However, task X may also be needed for the following tasks. In this context, as depicted in Figure 9, we add a parallel gateway to allow the execution of tasks Z and W together with task X.

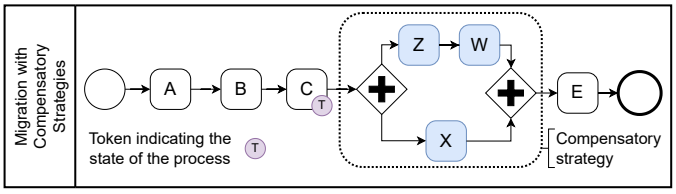


Fig. 9: Compensatory strategy when the token is in C.

(D) The token is on D. As shown in Figure 10, Once D is completed, the user has to execute X before proceeding to the next steps.

When the token is either in C or D, we consider only the situations where the new conditions in the revised gateway are still met for the token. That is, the token is in C or D and a path has already been taken in the gateway. Therefore, the user cannot migrate between branches of the same exclusive gateway even if another option (i.e., task Y) is added.

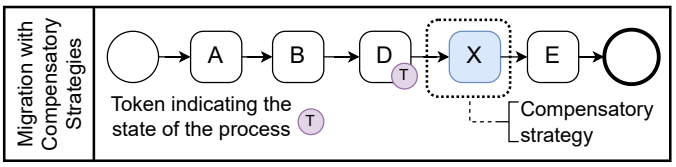
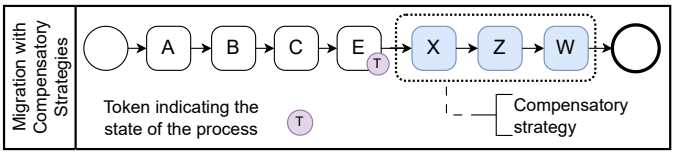
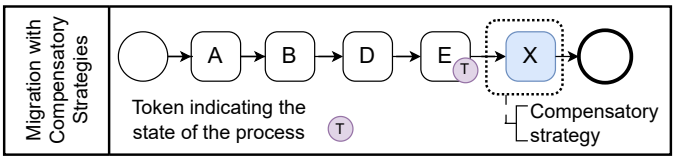


Fig. 10: Compensatory strategy when the token is in D.

(E) The token is on E.
The user is at the end of the process and the compensatory actions establish that after E, X has to be completed, as visible in Figure 11. Then, as shown in Figure 11, two different scenarios emerge. In fact, if the users' path has passed through task C, they also have to execute tasks Z and W as they are part of the revised process right after task C. On the other hand, if the users' path has passed through D, there are no additional tasks.



(a) Comp. strategy when the path of the user involved task C.



(b) Comp. strategy when the path of the user involved task D.

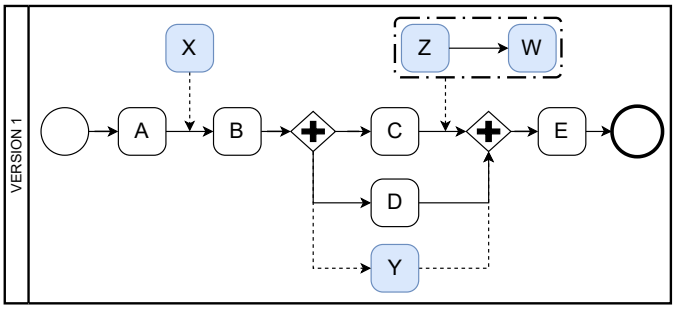
Fig. 11: Compensatory strategy when the token is in E.

3) Parallel Gateway:
An alternative scenario to the process described in the previous subsection is with a parallel gateway instead of an exclusive gateway, as illustrated in Figure 12.

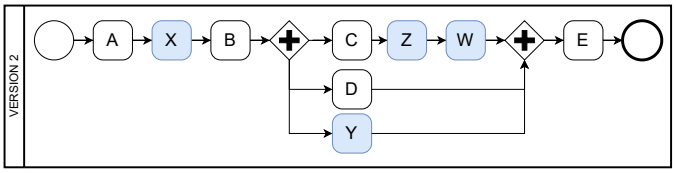
Here, the compensatory strategies based on the token position are as follows:

(A) The token is at the beginning of the process on task A. As shown in Figure 13, no compensatory strategies are required.

(B) The token is in B, and the compensatory strategy requires that the new task X has to be completed before going to the next steps of the process, as seen in Figure 14.



(a) Variation of our reference example shown in Figure 1 with a parallel gateway.



(b) Revised version of the process above.

Fig. 12: Process with a parallel gateway in its revised version, where the blue tasks are added.

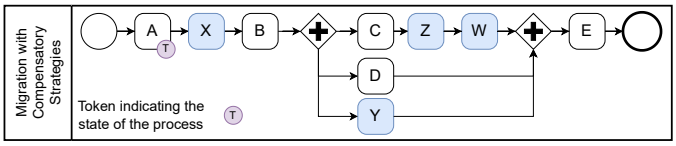


Fig. 13: Compensatory strategy when the token is in A.

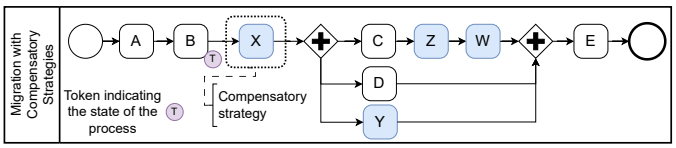


Fig. 14: Compensatory strategy when the token is in B.

(C|D) The token is inside the parallel gateway, with at least one task between C and D to be completed. As shown in Figure 15, after all the tasks in the parallel gateway are completed, the compensatory strategy intervenes. Here, it is necessary to complete task X first and then another parallel gateway is opened, where the user has to execute tasks Y, Z, and W. In the revised process, all these four tasks were added before the next task (i.e., task E), thus we had to complete them first, where task X is always the initial one.

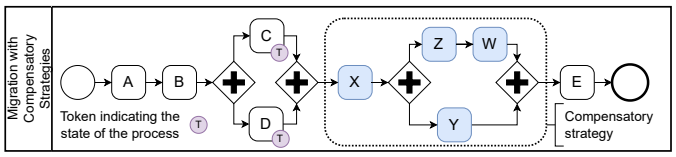


Fig. 15: Compensatory strategy when the token is in C, D, or both of them.

(E) The user is at the end of the process (i.e., the token is on E).
Once E is completed, task X is the first one to be executed in the compensatory strategy, and then a parallel gateway with all the remaining tasks (Y, Z, W) is opened, as shown in Figure 16.

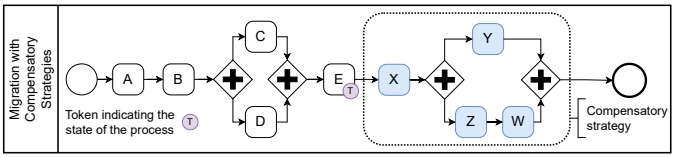


Fig. 16: Compensatory strategy when the token is in E.

4) Removed tasks:

In the previous part, we showed different compensatory strategies to adopt when a task is *added* in the revised process. Here, we want to show the compensatory strategies when a task is *removed* in the revised process. Even in this scenario, all the tasks removed after the token's position do not need any compensatory strategy. However, if a task has already been executed and it is deleted in the revised version, we have to introduce a proper compensatory strategy. The idea is to add a "negation" sub-process that contains all the elements needed to compensate for what the task did. For example, if *Therapy 1* is no longer suitable for the considered treatment, we should allow a step where the effects of the therapy are voided/mitigated (e.g., the administration of an antidote) for all the patients that already received such therapy. Here are some possible scenarios:

(B) The token is in task B and A is removed. As shown in Figure 17, we add the sub-process ¬A as a compensatory strategy right after B.

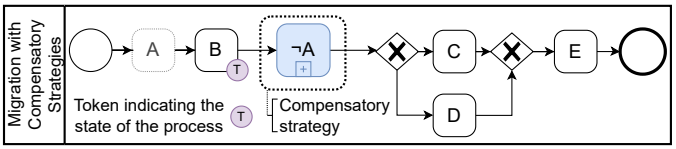


Fig. 17: Compensatory strategy when task A is removed.

(X) The token is in task E, and tasks C and D are removed from the exclusive gateway. Here, we should remember that a user can only take a single path within an exclusive gateway. Thus, as shown in Figure 18, we add the sub-process only to void the executed task.

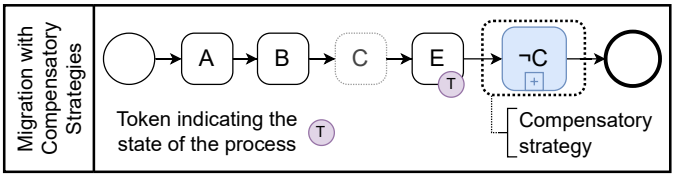


Fig. 18: Compensatory strategy when tasks C and D are removed within an exclusive gateway, and the user already executed C.

(+) The token is in task E, and tasks C and D are removed from the parallel gateway. In this scenario, we know that the users have to execute all the tasks in the parallel gateway before proceeding to the next step, but the order in which they execute them is not relevant. Thus, as shown in Figure 19, we need to add a new parallel gateway right after E with ¬C and ¬D sub-processes to void what tasks C and D did, regardless the order of execution.

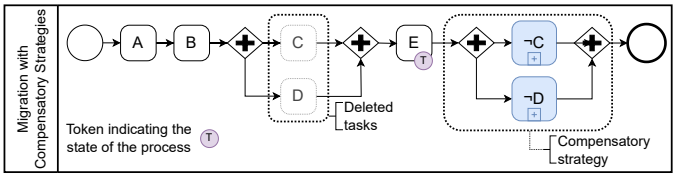


Fig. 19: Compensatory strategy when tasks C and D are removed within a parallel gateway.

5) Modified tasks:
Compensatory strategies are also needed when an already executed task is *modified* in the revised process. Similar to the deletion strategy, for each modified task we want to introduce a sub-process containing all the elements needed to compensate for the modification. For example, suppose we modify a task containing a questionnaire for evaluating the patient's physical condition. In that case, the compensatory strategy consists of a sub-process containing a task with the modified/added questions. The compensatory strategies for modified tasks are:

(B) The token is in task B and A is modified. As shown in Figure 20, we add the sub-process A' as a compensatory strategy right after B.

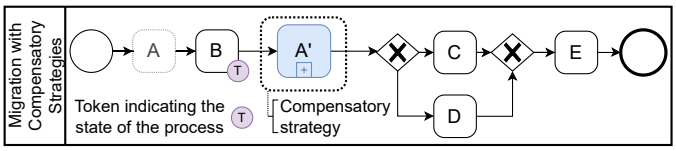


Fig. 20: Compensatory strategy when task A is modified.

(X) The token is in task E, and tasks C and D are modified in the exclusive gateway. Because a user can only take a single path within an exclusive gateway, we add the sub-process only to reflect the changes in the executed task, as shown in Figure 21.

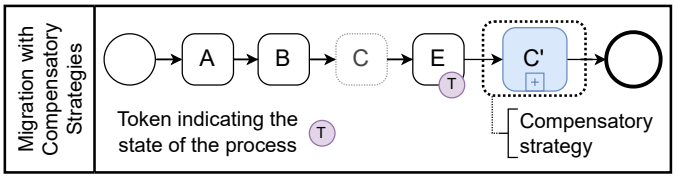


Fig. 21: Compensatory strategy when tasks C and D are modified within an exclusive gateway, and the user already executed C.

(+) The token is in task E, and tasks C and D are modified in the parallel gateway. Because the users had to execute all the tasks in the parallel gateway - without a specific order - before proceeding to the next step, we need to add a new parallel gateway right after E with C' and D' sub-processes to compensate for the changes in tasks C and D, as shown in Figure 22.

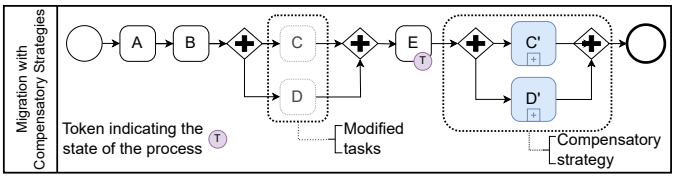


Fig. 22: Compensatory strategy when tasks C and D are modified within a parallel gateway.

6) Atomic transactions:
A series of tasks may need to be executed all together in a precise order, and we could only advance to the next series of tasks when they all have been completed successfully. The failure to complete just one of them corresponds to the failure of the whole set of tasks. In the clinical domain, for example, a set of drug administration procedures must be completed together to fulfill the therapy successfully, or a complete set of exams is needed to have the correct diagnosis. In this context, we embed this set of tasks within a *transaction sub-process*. We then execute a transaction that is either completed, not initiated, or interrupted (which implies starting over all the transaction's tasks). Therefore, a transaction sub-process consists of two or more tasks that, despite being different (both in type and content), cannot be divided within the process execution.

With transactions, the compensatory strategies seen before have to adapt accordingly, and Figure 23 shows an example of this. In fact, tasks A and B represent a transaction, and even if the token is in task A, we have to complete task B before migrating to the revised version, thus using the compensatory strategy that includes the execution of task X after the completion of task B. In other words, with atomic actions we can increase the process's abstraction level, which helps to better model the patients' healthcare needs and improve their quality of care.

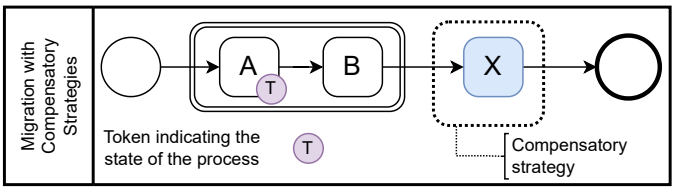


Fig. 23: In this figure, tasks A and B represent a transaction sub-process enclosed in a BPMN transaction.

7) Sub-Processes: In a process, it is common to have one or more sub-processes to enhance the readability, reuse, and provide better logic separation. Therefore, it is fundamental to

manage the sub-process migration and its repercussions on the process. Here, we want to consider only the sub-processes that represent a *logical* division of the tasks. That is, a process that could be rewritten without sub-processes but with the inclusion of all the logic they include. Because of that, there is no need to propose different compensatory strategies dedicated to this kind of sub-processes. In the case of reusable (standalone) sub-processes, we could have changes in them that influence the parent process, such as a variation in the number of variables returned. In that situation, it could be necessary to introduce a compensatory strategy also for the parent process, but it is something that this paper will not address.

B. Migration clinical risks

Healthcare-related applications should always prioritize the patients' well-being. Thus, it is only natural to consider the repercussions that one action could provoke on the health of a patient (e.g., a fragile cancer subject). In order to manage the migration with the lowest impact possible on the patient, we introduce three classes of risk to identify the best part of the process where the migration has a minimum impact. The class of risk is assigned by the clinicians, and to facilitate their use they are constructed to imitate the scoring system used in guidelines. In this context, the benefits are given by the revised versions of the process, as it should be more accurate/appropriate than the previous one. On the other hand, the migration could be riskier or not recommended during some parts of the process model. These risks are generally valid for all the patients (schema level), but there could be situations where a single patient's condition changes these risks (instance level). Let us use again Figure 1 as an example. A clinician could highly recommend a migration after *Therapy 1* in order to perform the added two cycles of therapy. However, if a patient had complications or adverse drug reactions after *Therapy 1*, the clinicians could decide that it is not recommended to migrate and add two more cycles. In the following, we evaluate both scenarios.

1) Class of risk:

We define the class of risk, which is the class associated with one particular element of the process (e.g., a task) and has three possible levels depending on whether it is safe, or not, to migrate. As shown in Table I, we also decide to color-code these classes as follows:

- Green: safe to migrate, the overall risk is very low;
- Yellow: possible, but with warnings, the overall risk is medium;
- Red: migration is discouraged, and the overall risk is high.

Again, the class of risk is given by the clinical experts, which evaluates the risks to benefits ratio. To be precise, they assign a generic class of risk for each task in the process, but they could also assign a different class of risk for some tasks related to a specific patient.

2) Class assessments:

When preparing a migration, we consider the two levels: schema (generic class of risk assigned) and instance (related to a specific patient). Therefore, each task in the process has two separate classes of risk that can be equal or different. Then, we have to merge these classes for each task to have a final class of risk that the clinician should consider for the migration. Because the clinicians give these classes, we could think the process-related risk classes are used as a reference. However, the clinician could override these classes for a specific patient condition. As shown in Table II, in case of different classes of risk between schema and instance, the final class follows the patient-related class.

TABLE I: Table indicating the classes of risk.

Class of risk	Migration	Definition
I	Low risk	Safe to migrate, the overall risk is very low.
II	Medium risk	Migration is possible but with warnings, the overall risk is medium.
III	High risk	Migration is dangerous, the overall risk is very high.

TABLE II: Managing the distinct cases of the class assignment when they differ.

Process-related classes of risk	Patient-related classes of risk	Final class of risk
I	II	II
I	III	III
II	I	I
II	III	III
III	I	I
III	II	II

Figure 24 shows an example of the class assignment at the schema level. Here, the migration risk starting from tasks A and B is low, while migration from tasks C and D is at medium risk as they could represent a more delicate part of a treatment. Finally, task E is at high risk, potentially because the task itself is severely delicate in that phase of the process or just because it is the last task and the clinician prefers to end the whole process instead of migrating. Figure 25 focuses on a single instance, that is, a specific patient using the process. As mentioned before, in this example we want to recreate a situation where a patient suffered from the administration of a specific treatment (e.g., task C). So, the clinician evaluated this specific scenario and decided to assign a higher risk than the default one. However, we could have another instance where the patient responded better than expected for task C, and in that situation the clinician could have lowered the risk class. Finally, the same Figure 25 also subsumes the previous two assessments and represents the final class of risk.

IV. System description

In the previous section, we outlined the basic principles of our approach. Now, we will illustrate its application to a real-world scenario. This framework could be perfectly adopted in the processes used in our previous work [6]. In that paper, we developed a process-driven web application, namely *Fit to Fight*, using the agile methodology. This web-app was specifically developed to integrate an ERAS-inspired

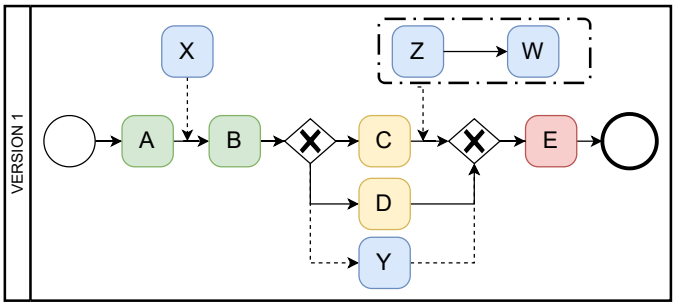


Fig. 24: General assigned classes of risk for a process with an exclusive gateway.

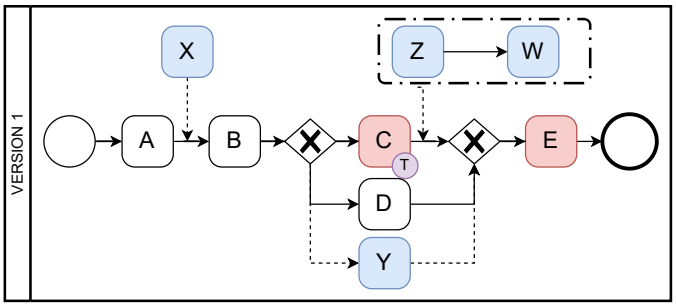


Fig. 25: Patient-related classes of risk for the same process above. This also represents the final classes of risk, as we always consider the highest risk between the process-related and the patient-related.

prehabilitation program for patients with pancreatic cancer. This program, currently in use by the Verona Pancreas Institute (the leading center in Italy for pancreatic surgery), contains a set of multidisciplinary activities to perform in the 30-40 days before hospitalization for pancreatic surgery to reduce post-operative complications. The processes of *Fit to Fight* were continuously refined, but there were no migration strategies when patients had already started the ERAS-inspired program. The main BPMN process underlying this program, depicted in Figure 26, is called *Fit to Fight Complete Process* (also referred to as the *Pancreatic Surgery Preparation Process*), and it is structured as follows: at the beginning, there is a psychological screening made by five surveys, then there is the main program divided into four sections (surgery, nutrition, psychology, and physical activity). Once the four main areas are completed, the patient has a visit with the psychologist. Finally, the patient completes the last screening, which is composed of the same questionnaires as the initial one.
In this section, we will discuss how the changes have impacted the migration process and how classes are allocated to different cases. We will examine various scenarios to understand how to migrate a process instance in different states (represented by a patient, with his/her process position indicated by his/her token). This comprehensive exploration will provide a thorough understanding of the migration process under different circumstances. Here, we provide a concise overview of the change implemented from version 1 (Figure 26) to version 2 (Figure 27), which concerns a *modified* and an *added* task. In the revised process, the task *Final Psychological Visit* was modified into *Final Visit*, as it now includes a visit with all the clinical specialists. Moreover, a task for the Customer Satisfaction (*CSAT*) survey was added in the final questionnaires, because *Fit to Fight* is developed in an agile environment where the patients are one of the stakeholders. Thus, their feedback is useful to improve the application iteratively, too.

Now, we will proceed to assess and allocate the classes of risk to the two previously described levels: the schema (process-related) and the instance (patient-related). This evaluation will allow us to better understand the potential challenges and devise strategies to mitigate them effectively.

A. Process-related Classes of Risk

Here, the clinician assigns the risk classes as illustrated in Figure 28, producing a BPMN process where each task is highlighted with a color representing the risk. In this case, the initial questionnaires are marked as green as no modifications involve this part of the process, thus migrating here represents no risks for the patients. The following part, which involves *Nutrition*, *Psychology*, and *Physical activity* sub-processes, is marked in yellow. They contain a 30 to 40-day set of activities for the patients, and although it may be possible to migrate, we should proceed with care depending on the patient's completion status. The *Surgery* sub-process is marked as green because it is composed only of informative videos about pancreatic surgery, so it has a lower risk (class I).
The *Final Psychological Visit* and the final questionnaires are marked as class I because, although modified, they can be easily migrated thanks to the compensatory strategies discussed in the following. Lastly, the final task, which is a service task, is marked as the highest class of risk because the patient has no other tasks to complete, and the process itself is almost finished. Therefore, it is unreasonable to migrate at this point in the process.

B. Patient-related Classes of Risk

In order to explain the patient-related risk, we provide three different completion statuses:

1) The patient has completed 2 out of 5 initial questionnaires. The clinician asserts that here it is highly recommended to migrate, as the patient is at the beginning of the process and no compensatory strategies are needed because all the changes in the revised process are in a later stage. However, the clinician also decided that, for this specific patient, the class risk for the physical activity sub-process is the highest possible due to the patient's physical conditions, as seen in Figure 29.
2) The patient has already initiated the task *Final Psychological visit*, that will be modified in the revised process. Here, the migration is still possible, but some compensation has to be done. As showed subsection III-A5, the compensatory strategy introduces a new sub-process to compensate for the modification. For example, suppose the modified task now comprehends a visit with

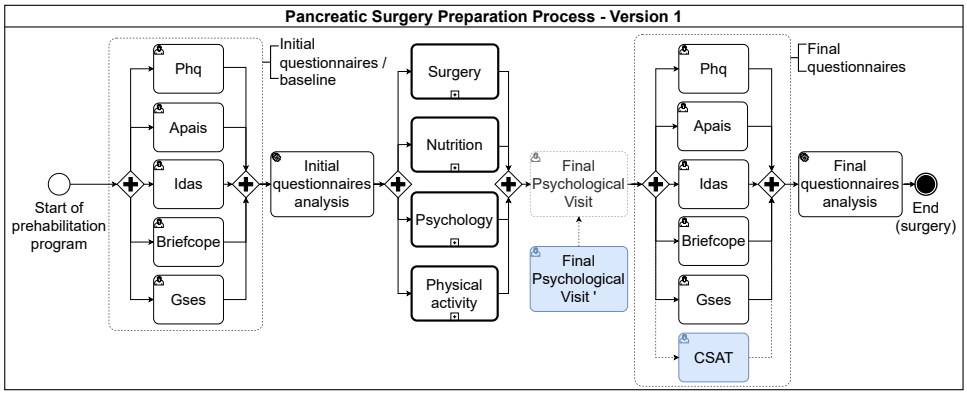


Fig. 26: Old design of the process *Pancreatic Surgery Preparation Process*.

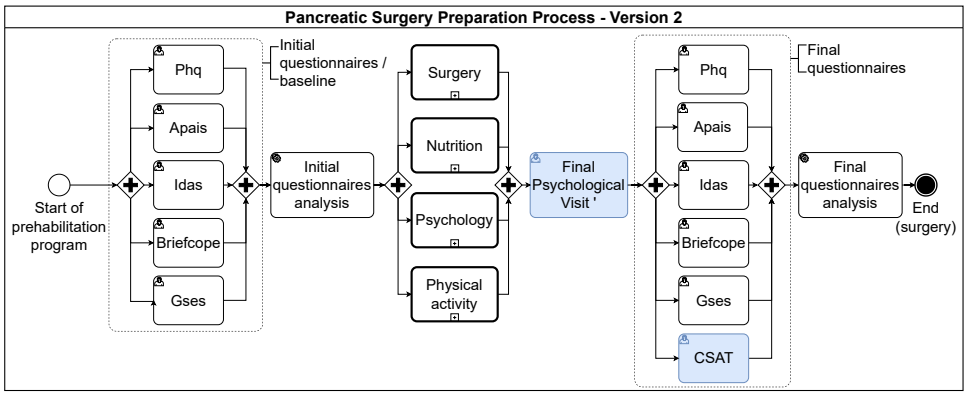


Fig. 27: New design of the process *Pancreatic Surgery Preparation Process*.

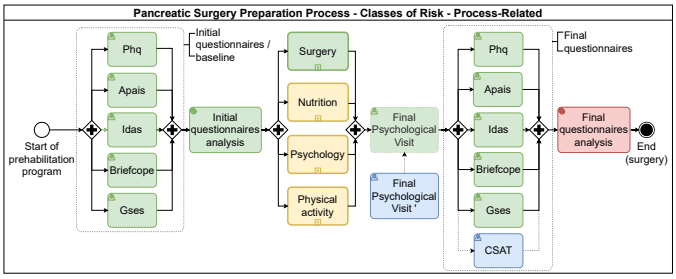


Fig. 28: Process-Related classes of risk for the process.

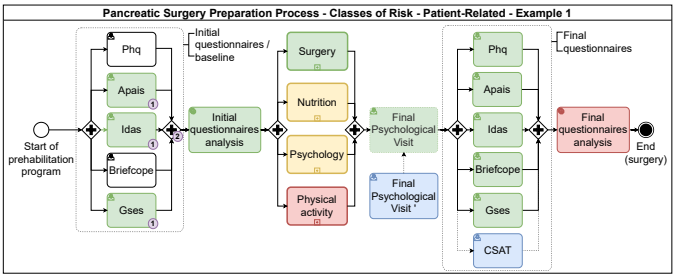


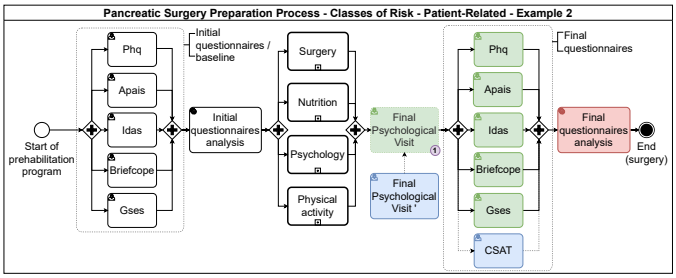
Fig. 29: Patient-related classes of risk.

multiple psychological specialists. In that case, the sub-process may include the tasks needed to obtain a visit from the specialists not already seen in the original task. This scenario is shown in Figure 30.

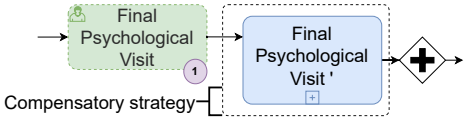
3) The patient is now near the end of the process, with only one survey left to complete. However, that task is marked in green and migration is possible. In that situation, a compensatory strategy is needed as visible in Figure 31. Here, we execute the revised tasks in sequence, as they are designed in the process, with the sub-process that compensates for the modified task, and then the addition of the last survey. Although the patient is almost at the end of the process, thanks to the compensatory strategy, the patient can migrate and take advantage of the revised process.

C. Final class assignment

As previously discussed, the final step involves merging the two risk levels, where the patient-related risk classes supersede the process-related ones. Here, the only exception is the *Physical Activity* sub-process, with a higher risk for the considered patient. Thus, the resulting class mapping is the one shown in Figure 29. This classification effectively captures the varying levels of risk associated with the migration process for different individuals, allowing for more targeted and efficient management strategies.

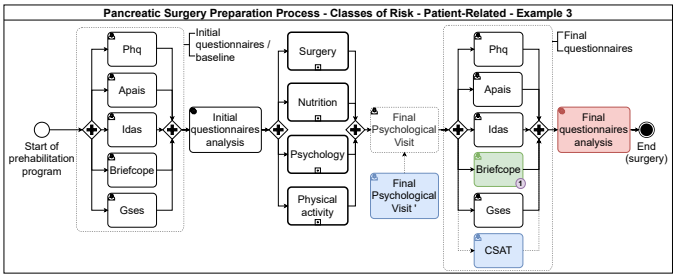


(a) Classes of risk for the patient doing task *Final Psychological visit* in *Pancreatic Surgery Preparation Process*.

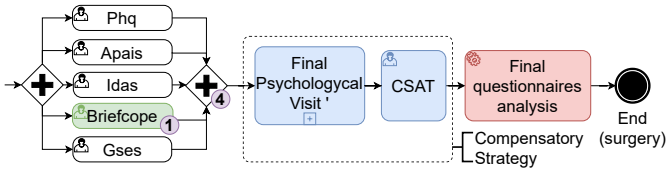


(b) Excerpt of the process, showing the compensatory strategy for a patient currently doing task *Final Psychological visit*.

Fig. 30: Compensatory strategy and classes of risk for the patient doing task *Final Psychological visit* in *Pancreatic Surgery Preparation Process*.



(a) Classes of risk for the patient doing the final surveys in *Pancreatic Surgery Preparation Process*.



(b) Detail of the compensatory strategy needed.

Fig. 31: Compensatory strategy and classes of risk for the patient doing final surveys in *Pancreatic Surgery Preparation Process*.

V. Conclusions

This paper introduced a novel migration framework explicitly designed for BPMN processes within a healthcare domain, focusing on the ones developed in an agile environment. The dynamic nature of this environment necessitates frequent process updates and revisions, presenting a challenge when users are operating on previous process model versions. On the other hand, in the healthcare domain the patient's health status is paramount. Here, migration challenges extend beyond technical aspects, as any changes in the process can directly impact the quality of care and patient outcomes. For this reason, we strove to ensure that our proposed migration strategies were technically robust and mindful of potential patient impact. This involved considering factors such as the complexity of new tasks, patients' abilities to adapt to changes (due to their condition), and potential migration risks. In the proposed framework, clinicians assign a color-coded class of risk for each task in the process, and then they can override some of these assignments for specific patients in non-standard health conditions. Then, based on the patients' completion status, we proposed a comprehensive set of compensatory strategies to ensure a seamless migration while mitigating the impact of its effects on the patient's conditions. In fact, the revised process should be better and more refined than the previous one, but some of its modifications could affect a part of the process the patient has already executed. Compensatory strategies help adapt the current path with a thorough set of cases for adding, modifying, or deleting a task. Then, we showed a real-world scenario and applied this framework to a process-driven application that embeds an ERAS-inspired prehabilitation program for pancreatic surgery developed using the agile methodology. This dynamic and continuously changing program served as an ideal testing ground for our framework, and we showed three different usage scenarios. In the future, we want to extend our migration strategies to include other BPMN elements, such as signals, timers, reusable sub-processes, and more sophisticated changes in the gateway conditions. This will involve developing new compensatory strategies tailored to these elements' unique characteristics and requirements. By doing so, we aim to enhance our framework's versatility and ensure its applicability across a broader range of process changes, further improving user experience and the operational efficiency of agile environments in healthcare.

Acknowledgment

The research leading to these results has received funding from the European Union—NextGenerationEU through the Italian Ministry of University and Research under PNRR—M4C2-I1.3 Project PE_00000019 "HEAL ITALIA" to Combi and Mantovani CUP B33C22001030006. The views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

This work has been carried out while Chini was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome in collaboration with the University of Verona.

References

[1] A. Cichocki and M. Rusinkiewicz, "Migrating workflows," in *Workflow Management Systems and Interoperability*. Springer, 1998, pp. 339–355.

[2] D. S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process migration," *ACM Computing Surveys (CSUR)*, vol. 32, no. 3, pp. 241–299, 2000.

[3] B. Weber, S. Rinderle, and M. Reichert, "Change patterns and change support features in process-aware information systems," in *Advanced Information Systems Engineering: 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007. Proceedings 19*. Springer, 2007, pp. 574–588.

[4] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features–enhancing flexibility in process-aware information systems," *Data & knowledge engineering*, vol. 66, no. 3, pp. 438–466, 2008.

[5] L. Pufahl, F. Zerbato, B. Weber, and I. Weber, "Bpmn in healthcare: Challenges and best practices," *Information Systems*, vol. 107, p. 102013, 2022.

[6] M. Mantovani, A. Caravati, G. Pozzi, C. Combi, and R. Salvia, "A BPMN- based framework to manage ERAS-inspired pathway for patients undergoing pancreatic surgery," in *2023 IEEE 11th International Conference on Healthcare Informatics (ICHI)*, 2023, pp. 21–31. [Online]. Available: https://doi.org/10.1109/ICHI57859.2023.00015

[7] M. Sonntag and D. Karastoyanova, "Compensation of adapted service orchestration logic in bpel'n'aspects," in *Business Process Management: 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30-September 2, 2011. Proceedings 9*. Springer, 2011, pp. 413–428.

[8] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," *https://agilemanifesto.org*, 2001.

[9] OMG, *Business Process Model and Notation (BPMN), Version 2.0*, Object Management Group Std., Rev. 2.0, January 2011, http://www.omg.org/spec/BPMN/2.0.

[10] C. Eskicioglu, S. S. Forbes, M.-A. Aarts, A. Okrainec, and R. S. McLeod, "Enhanced recovery after surgery (ERAS) programs for patients having colorectal surgery: a meta-analysis of randomized trials," *Journal of gastrointestinal surgery*, vol. 13, no. 12, pp. 2321–2329, 2009. [Online]. Available: https://doi.org/10.1007/s11605-009-0927-2

[11] M. E. Coolsen, R. Van Dam, A. Van Der Wilt, K. Slim, K. Lassen, and C. Dejong, "Systematic review and meta-analysis of enhanced recovery after pancreatic surgery with particular emphasis on pancreaticoduodenectomies," *World journal of surgery*, vol. 37, no. 8, pp. 1909–1918, 2013. [Online]. Available: https://doi.org/10.1007/s00268-013-2044-3

[12] X. Zhao and C. Liu, "Version management for business process schema evolution," *Information Systems*, vol. 38, no. 8, pp. 1046–1069, 2013.

[13] L. T. Ly, S. Rinderle, and P. Dadam, "Integration and verification of semantic constraints in adaptive process management systems," *Data & Knowledge Engineering*, vol. 64, no. 1, pp. 3–23, 2008.

[14] A. Barkhordarian, F. Demuth, K. Hamann, M. Hoang, S. Weichler, and S. Zaplata, "Migratability of bpmn 2.0 process instances," in *Service-Oriented Computing-ICSOC 2011 Workshops: ICSOC 2011, International Workshops WESOA, NFPSLAM-SOC, and Satellite Events, Paphos, Cyprus, December 5-8, 2011. Revised Selected Papers 9*. Springer, 2012, pp. 66–75.

[15] M. Rosemann, J. Recker, and C. Flender, "Contextualisation of business processes," *International Journal of Business Process Integration and Management*, vol. 3, no. 1, pp. 47–60, 2008.

[16] P. Badakhshan, K. Conboy, T. Grisold, and J. vom Brocke, "Agile business process management: A systematic literature review and an integrated framework," *Business Process Management Journal*, vol. 26, no. 6, pp. 1505–1523, 2019.

[17] H. Tomaskova and M. Kopecky, "Specialization of business process model and notation applications in medicine—a review," *Data*, vol. 5, no. 4, p. 99, 2020. [Online]. Available: https://doi.org/10.3390/data5040099

[18] Unit of Pancreatic Surgery, 2022, http://www.chirurgiapancreasverona.it/?lang=en.

[19] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*. Springer, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-33143-5