Addressing Business Process Deviations through the Evaluation of Alternative Pattern-Based Models

Charbel Kady, Khaled Jalloul, François Trousset, Charles Yaacoub, Adib Akl, Nicolas Daclin, Grégory Zacharewicz

applied sciences

MDPI

Article

Addressing Business Process Deviations through the Evaluation of Alternative Pattern-Based Models

Charbel Kady [1,2], Khaled Jalloul [2], François Trousset [1], Charles Yaacoub [3], Adib Akl [4], Nicolas Daclin [1] and Gregory Zacharewicz [1,*]

[1] Laboratory for the Science of Risks, IMT-Mines Ales, 6 Avenue de Clavières, 30100 Alès, France; charbel.kady@mines-ales.fr (C.K.); francois.trousset@mines-ales.fr (F.T.); nicolas.daclin@mines-ales.fr (N.D.)
[2] School of Engineering, Lebanese American University (LAU), P.O. Box 36, Byblos 1401, Lebanon; khaled.jalloul@lau.edu
[3] Université Catholique de Lille, 60 Bd Vauban, 59800 Lille, France; charles.yaacoub@univ-catholille.fr
[4] School of Engineering, Holy Spirit University of Kaslik (USEK), P.O. Box 446, Jounieh 1200, Lebanon; adibakl@usek.edu.lb
* Correspondence: gregory.zacharewicz@mines-ales.fr; Tel.: +33-4-6678-5000

Abstract: Business processes (BPs) have become extremely complex with thousands of tasks to be completed. These processes should be modelled to manage their complexity, but deviations from these models can occur during execution. To account for the potential outcomes, models are often over-specified, but in fact, it is impossible to anticipate every scenario. Therefore, the issue of how to model the response to these problems arises. The approach put forward in this study entails the creation of a primary model with the aid of domain experts. If a deviation occurs during execution, the system searches for compatible patterns in a collaborative library of workflows, rather than relying on dedicated solutions for specific problems. Nevertheless, the main challenges lie in selecting the most appropriate set of candidates from a vast number of patterns, as well as identifying the optimal injection points on the primary model to correct the deviation. One main objective is to rapidly eliminate incompatible patterns by employing simple mathematical techniques to narrow down the pool of candidate solutions. This serves to minimize the number of patterns that should be considered where more sophisticated methods can then be utilized to rank and select the best solution from this smaller set.

Keywords: BPMN elements characterization; workflow deviations handling; pattern-based solutions; interoperability; process model repair

Citation: Kady, C.; Jalloul, K.; Trousset, F.; Yaacoub, C.; Akl, A.; Daclin, N.; Zacharewicz, G. Addressing Business Process Deviations through the Evaluation of Alternative Pattern-Based Models. *Appl. Sci.* **2023**, *13*, 7722. https://doi.org/10.3390/app13137722

Academic Editor: Krzysztof Koszela

Received: 2 June 2023
Revised: 21 June 2023
Accepted: 27 June 2023
Published: 29 June 2023

1. Introduction

Business processes have evolved to become increasingly complex, often consisting of a multitude of tasks that need to be executed. While these processes are typically modelled to manage their inherent complexity, deviations from these models can occur during their execution. The ability to anticipate every possible scenario and incorporate it into the models is practically impossible. Consequently, the challenge arises of how to effectively model the response to these unforeseen problems.

The approach outlined in this study involves the creation of a primary model in collaboration with domain experts. In the event of a deviation during execution, the system searches for compatible patterns within a collaborative library of workflows, rather than relying on specific solutions tailored for individual problems. Selecting the most appropriate candidates from a vast number of patterns and identifying optimal injection points in the primary model to rectify the deviation constitute the main challenges in this approach.

The proposed method introduces three basic types of repairs, which differ from conventional approaches that typically focus on predicting and solving locally identified deviations or exceptions. By modifying the model dynamically through the insertion

or deletion of parts of the workflow, the proposed method offers a more flexible and comprehensive approach to addressing unexpected situations. Additionally, the compatible solutions are evaluated and ranked based on performance indicators, enabling decision makers to select the best solution automatically or manually.

Central to the proposed approach is a collaborative library of random patterns, which stores patterns that may serve as potential candidates for unexpected deviations in workflow execution. Unlike conventional approaches that store solutions for each problem, this library leverages the expertise of domain experts and external sources to identify compatible solutions based on the current state of the system. The searching mechanism employed in this approach efficiently narrows down the pool of candidate solutions by rapidly eliminating non-compatible patterns using simple algorithms. Subsequently, more sophisticated methods are utilized to refine and rank the remaining patterns, leading to improved workflow performance.

In order to accurately calculate the impact of candidates on the system and determine the most effective course of action, this study introduces new characterizations and notations that enrich BPMN elements. These characterizations provide a valuable framework for accurately assessing the impact of repairs on the workflow. The ease of implementation is another significant advantage of the proposed methodology, as demonstrated through the utilization of Python to implement the entire approach, enabling practitioners to readily incorporate it into existing systems and workflows.

Furthermore, the proposed iterative solution injection approach allows the system to continuously analyse historical data and suggest optimizations at the main model level. By learning from past experiences and adapting to new circumstances, the system can continually enhance its performance. This opens up possibilities for future studies in artificial intelligence, where historical databases can be leveraged to further refine and improve the design of the model.

The remainder of this paper is organized as follows: Section 2 presents a comprehensive review of the related work, Section 3 provides a comprehensive overview of the materials and methods employed in this study. In Section 4, a concise case study is presented as a compelling proof of concept, illustrating the ease with which this method can be implemented in real-world scenarios. Section 5 delves into a comprehensive discussion of the findings, exploring their implications, limitations, and potential applications. Lastly, Section 6 presents a conclusion summarizing the key insights and contributions of this approach. It also proposes future directions and potential research avenues to explore, emphasizing the significance and potential impact of this work.

2. Related Work

As stated in the introduction, in recent years, there is growing attention towards process-aware information systems, which are designed to aid in managing, supervising, and overseeing business processes. [1]. Business process management (BPM) is a well-known method that encompasses a set of organized and interconnected tasks or activities aimed at delivering a distinct service or product to one or multiple customers. It is commonly represented as a flowchart illustrating the sequential progression of its activities [2].

The Integrated Computer-Aided Manufacturing (ICAM) program, starting in the 1970s, recognizes the need for enhanced analysis and communication techniques among individuals working to improve manufacturing productivity. As a result, the program introduced a set of modelling approaches known as the ICAM definition (IDEF) methods [3]. Since the 1980s, scholars have strived to develop standardized and cohesive methods to represent and quantify processes [4] such as the structured analysis and design technique (SADT) (top-down approach) [5].

The Business Process Model and Notation (BPMN) was first developed by the Business Process Management Initiative (BPMI), which made version 1.0 available to the public in May 2004. Following the merger of BPMI with the Object Management Group (OMG) in

June 2005 [6], the OMG published a BPMN specification document in February 2006 [7]. The strength of the BPMN resides in two important aspects: (i) simplicity, which is due to the abstraction level provided by the standard; and (ii) the possibility of being automatically translated into a business execution language and, then, to generate a machine-readable prototype of business processes [8]. BPMN was created with the explicit purpose of bridging the gap between the business and technical perspectives regarding processes [9,10]. In BPMN, tokens serve as a theoretical concept to define a process's behaviour, with elements interacting as the token moves through the process. However, the specific details of what a token contains are not described in the official BPMN specification [11]. It is common to associate tokens with details such as process instance ID, data, flow object ID, timestamp, and state information. The data held by a token can differ depending on the specific case and implementation [12].

In contemporary organizations, as BPs are becoming increasingly complex, often involving thousands of activities that need to be modelled to control their complexity. Over-specifying the models to handle all foreseen situations and repair anomalies has been common practice in industry to overcome unexpected malfunctions. However, this approach is expensive and still cannot account for all possible situations, as evidenced by the COVID-19 pandemic. Therefore, there is a pressing need to repair the model to sustain business operations. Bendraou et al. [13] addressed the recurring issue of developer deviations from the process model in the multi-viewpoint-based development of complex systems. These deviations, which can be behavioural or structural in nature, arise without proper methodological support. The authors proposed a solution to overcome these issues through a viewpoint-based development process.

Earlier research has addressed four key areas related to the correction of deviations in business models, namely: problem identification, solution evaluation, model validation and model reparation.

Although, detecting deviations is not the subject of this study, and in this paper, it is supposed that an anomaly detection has been flagged and a repairing mechanism is proposed, numerous studies, and organizations such as Airbus have focused on the aspect of problem identification in the One-Way project [14], which established a Numerical Twin to identify any deviations in a system from its intended behaviour, whereas other studies have considered artificial intelligence (AI) to improve the system's ability to identify and describe systematic deviations present in data and models [15,16]. Hodge et al. [17] outlined numerous methods for detecting deviations. More over TariQ et al. [18] proposed a novel technique for detecting abnormalities in BP execution by extending conformance analysis methods. It utilizes event logs to identify correlations and discrepancies, filters logs into successful and failed instances, and classifies abnormalities based on conformance dimensions. The approach includes the concept of a conformance lifeline for early predictions and has been applied to a real-world event log, providing process-specific improvement measures. While Weijian et al. [19] addressed the issue by presenting an innovative method for identifying abnormalities in BP execution by extending conformance analysis techniques. By analysing event logs, correlations and discrepancies are identified using non-traditional conformance analysis approaches. The approach involves filtering event logs into successful and failed instances, utilizing the former to derive an optimal process model, and classifying abnormalities in the latter based on conformance dimensions. While it is impractical to mention all of them in this article, it is essential to note that this paper assumes deviation has already been detected, serving as a starting point to identify an appropriate repair for the problem. Nonetheless, the tools and notations presented in this manuscript can aid future studies in detecting model deviations.

Solution evaluation was also addressed by many studies. Ducq et al. [20] defined different methods of aggregations of key performance indicators (KPIs). These method will be use in this paper especially when it comes to estimating the KPI for multiple blocks on the model. More, these indicators were used to evaluate the interoperability of systems, as seen in Heguy's study [21], and to aggregate KPIs computed on single

BPMN elements to parts of the model, as described in Ougaabal's work [22]. Numerous theories have been introduced for model evaluation to validate models. One example is the research by Kherbouche [23], proposing a technique to validate workflow modifications resulting from the deletion or insertion of tasks based on KPIs. Another study by Mallek-Daclin [24] suggested verification techniques to validate interoperability in a collaborative process model based on data quality and time. In the evaluation of the model or the candidate set of solutions, it was noted that some approaches used in the literature exhibit similarities or can be complemented by the method employed in this manuscript. A fitness function to optimize candidate solutions with respect to their overall quality of service (QoS) attributes was suggested by Da Silva et al. [25]. This approach handles composition dimensions simultaneously, resulting in solutions that are fully executable, adhere to conditional constraints, and are optimized in line with QoS requirements. Importantly, this method could be utilized in future work for the implementation of the objective function. In the selection dimension, the labelling of candidate services based on their computing ability is accomplished using a user utility function, as proposed by Zhang et al. [26]. The appropriate label types are chosen, and only services with good labels are selected by genetic algorithms (GA) to determine the best service composition. Although efficiency in computation time was demonstrated by Zhang's method, our approach relies on the fast elimination of non-compatible candidates and reserves more complex algorithms for later stages when the candidate set is narrowed down.

There are two primary categories of model reparation: overspecified models (OSM) and underspecified models (USM).

The OSM category places emphasis on the variability approach, which predefines all possible solutions and determines the system's response at runtime based on a variation point and certain conditions. Certain techniques utilized object-oriented programming languages to create workflows and manage potential deviations [27]. Svendsen [28] used the common variability language (CVL) to describe and generate variants of the same model to achieve application reconfigurations, while Honghao et al. [29] discussed the workflow reconfiguration. In their investigation, La Rosa et al. [30] categorized four diverse approaches to variability, where each approach considered a distinct variation point. The first approach, node configuration, considers the node as the variation point, and for each node, various paths exist [31]. The second approach, element annotation assigns domain properties through Boolean expressions, and the selection can be made either manually or with the aid of a model. It involves testing a specific condition at the task level, such as a lower cost [32]. The third approach, activity specialization, emphasizes the customization of a business model based on activity specialization [33]. Finally, the fourth approach, fragment customization, depends on constraints: an activity is added if it satisfies a given constraint or rejected otherwise [34]. In these OSM approaches, whether in deviation or reparation models, rely on the concept that all paths must be predetermined. However, in most real-life situations, it is impossible to anticipate all scenarios [35].

Adaptive case management (ACM) [36] is a software-driven approach that supports knowledge work and unpredictable business processes by offering flexibility and informed decision making based on real-time data. In contrast, traditional BPM focuses on structured workflows and optimizing efficiency for repetitive tasks. By combining ACM and BPM, organizations can effectively manage both structured and unstructured workflows, enhancing efficiency and decision making. While this method has many advantages demonstrated in the literature [37], it is categorized under the OSM category as previously discussed. The proposed approach in this study differs from ACM by avoiding predefined paths and instead utilizing a library of patterns to iteratively select the most compatible pattern as a candidate solution.

Dynamic BPM [38] refers to the capability of supporting process modifications by any role with minimal delay. It encompasses a range of practices and technologies that empower individuals and systems to make timely adjustments in response to both obvious and underlying process requirements [39,40]. This enables the simulation of changes without

disrupting the current operational processes, systems, or applications. Once more, as noted, this method can also be categorized under OSMs in contrast with the proposed approach.

The second category, the USMs, employs a management by exception approach. In this approach, a primary model is established for standard operations while an exception handling mechanism is developed to address any unforeseen issues that may occur in the model during runtime by utilizing a library of sub-models. Russel et al. [41] classified different types of exceptions that may arise during the execution of a process and suggested techniques to manage them. Similarly, Adams et al. [42] maintained a database of exception handling procedures known as exlets to deal with predefined categories of exceptions. The database may be built during the design stage or dynamically during runtime. In a recent study, Jasinski et al. [43] presented a workflow management system that controls the environment using dynamically generated workflows. Exception detection and handling in the process creation provides solutions for possible occurrences. This approach enables rapid development of new tasks, known or unknown, and assesses the quality of the recommendations created through feedback from the managed environment. Taken a step further by Kerstin et al. [44], they introduced an additional layer of fragments that can be reused in workflows in case of failure. However, one notable drawback of USM approaches is their limited scope when it comes to repairs. These approaches typically focus on addressing issues within a specific area, and the solutions available are often pre-defined and stored in databases. If a malfunction occurs, the primary workflow will halt, a solution will be selected from a library, and the work will continue from the point where the issue was detected.

Górski et al. [45] conducted a systematic literature review on the optimization of BP execution in a service architecture, the authors categorized relevant methods into three stages: resource allocation [46,47], service composition, and service scheduling [48,49]. While some of these aspects may not directly fall within the scope of the current paper, the proposed method can complement the aforementioned stages. It should be noted that resource allocation and service scheduling will not be the focus of this paper. Regarding service composition, there are similarities that can be identified, and certain algorithms used in the reviewed papers, such as GAs [50–52], can be beneficial for future work, particularly in refining pattern selection. In the study by Kalasapur et al. [53], the authors proposed defining a BP using basic services (building blocks) and searching for service instances that realize one or several blocks, contrasting with our approach where patterns in the library serve as potential repair solutions and are not explicitly linked to specific services. Another study by Ukor et al. [54] stated that BPs can have multiple execution paths, with each path having a distinct set of optimal service instances. The authors suggested performing service composition for each execution path and selecting one path at the beginning based on initial data. This approach differs from the proposed method in this paper, as it requires over specification of multiple paths.

3. Materials and Methods

In modern BPs it has become common practice to create OSMs to accommodate all possible scenarios. However, anticipating every potential failure is a costly and challenging task. For instance, the recent COVID-19 pandemic demonstrated how quickly unforeseeable situations can arise and disrupt even the most well-designed processes. Knowing that it is critical to fix the deviation quickly to maintain business operations, this paper proposes a new approach that builds a schema process with the aid of domain experts, and if a deviation is detected during execution, the system searches a collaborative library of workflows to find compatible solutions that can correct the deviation. One possible method to address deviations in processes is the utilization of simulation. However, it is important to note that this method can be time-consuming due to the requirement of simulating numerous combinations for each pattern at every injection point pair of entry and exit points. Instead, in this manuscript, a more efficient approach is proposed, involving basic mathematical techniques to quickly eliminate non-compatible solutions and reduce

the number of candidate patterns to a limited set. Furthermore, advanced mathematical approaches, such as fuzzy set theory or simulation time management, can be used to evaluate and rank the remaining solutions from which the best solution can then be selected and applied to correct the deviated process. It is important to mention that the proposed instance correction may not only affect the deviated block at the point of failure P, but may also replace all blocks between P and any point Q on that instance. Therefore, the solution workflow must satisfy both local compatibility at the injection points and global compatibility at the process instance level. In other words, the solution workflow should not endanger the entire process leading to the degradation of the schema process.

3.1. Element Characterization and Notations

Element characterization and annotations are used in this manuscript to help with the explanation of the underlying concepts and methodologies. The benefit of using mathematical expressions is that they offer a succinct and accurate representation of complicated ideas. This is extremely helpful in understanding and communicating scientific findings. In the remaining sections of this manuscript, resources can be more precisely defined and discussed using mathematical notations, state tokens, and constraints.

3.1.1. State Token

This paper presents a novel version of the BPMN token, called the state token, which is enriched with information related to KPIs, resources, products, and other variables used in testing throughout the model execution. The state token provides a more detailed understanding of the process dynamics, which facilitates distinguishing normal from abnormal behaviour during deviation execution.
The state token proposed in this study is expressed as a tuple consisting of various types of information that start circulating right after the process instance begins and continues propagating throughout the model. This token is denoted as ST, which is a tuple defined as:

$$ST =< K, P, R, V >$$

- K is the set of KPIs.
- P is the set of products.
- R is the set of the available resources.
- V represents a set of system variables that are distinct from the variables included in the KPIs, products, and resources sets. It is important to note that V is a limited set of system variables that are specifically determined during the design phase of the model, and their selection is contingent upon the particular business domain. For instance, in the apiculture domain examples could include temperature, weight, and weather conditions, while other domains may feature a distinct set of variables along the execution.

The annotations $ST_{\underline{A}}$ and $ST_{\overline{A}}$ are used to differentiate the state token before and after the execution of a task A, respectively. In this case, $ST_{\underline{A}}$ denotes the actual state token observed before entry of task A, and $ST_{\overline{A}}$ denotes the actual state token observed after exit of task A. Furthermore, we can define $\widetilde{ST}_A$ as the estimated state token (computed) after exiting task A. $\widetilde{ST}_A$ will be detailed in the foloowing. Table 1 summarizes the latter annotations:

Table 1. State token notations.

State Token	Annotation
Actual state token before entry of task A	$ST_{\underline{A}}$
Actual state token after exit of task A	$ST_{\overline{A}}$
Estimated state token after exit of task A	$\widetilde{ST}_A$

3.1.2. Task/Activity

Task Characterization

The BPMN is a well-established standard in BP modelling, as evident in previous studies. Moreover, the BPMN standard offers an XML representation of each element in the model, enabling the enrichment of tasks or activities in a manner similar to the SADT annotation, a widely discussed modelling technique in the literature. By incorporating constraints (C_A), resources (R_A), inputs (I_A), and outputs (R_A) into each task or activity (A) in the BPMN, it is feasible to comprehensively define each element and monitor the model's performance. All these conditions (C_A, R_A, I_A, O_A) can be expressed as Boolean expressions.

- Constraints C_A: The constraint defines a condition expressed by a Boolean expression C_A that must be true at entry, during, and at exit of task A. The constraints mostly apply to elements K and V of the state token ST. For example:
$$(Cost < 30K€) \wedge (T^{\circ} > 20^{\circ})$$
where $Cost$ is a KPI belonging to K and T° is a state variable belonging to V.
- Resources R_A: Resources can be either local or global, and specifying them for each task or activity helps manage them more efficiently. It is important to mention at this point that resources management is not the object of this study. However, it is crucial to ensure normal execution of the workflow. As reported in [55], resources can be sorted in various ways. In this study, the main focus would be the behaviour of a resource rather than its type, namely allocation, utilization, and consumption. One can determine the availability of a resource r belonging to type T by verifying if r is present in the set of resources of type T available for the process. R_A represents the availability of resources needed by A as a Boolean expression. R_A must be true to enter task A. This expression mostly applies to element R and P of the state token ST. This expresses which resources are allocated by task A and will consequently be no longer available for other tasks during the execution of task A.
- Inputs I_A: I_A stands for the state token's necessary condition which is true at task entry. These conditions are mainly applied to K and V.
- Outputs O_A: O_A stands for the state token's necessary condition which is true at task exit. These conditions are applied to K, P, V and R. O_A represents the transformation applied to the state token prior to entry, resulting in the state token after exit. This expresses the changes in KPIs, such as duration and cost, as well as the impact on resources and the product resulting from the task. In fact, R_A indicates which resources are kept by A during its execution but does not provide information about which will be available again at the exit of task A. The expression of the consumption of resources by task A is then expressed in O_A.

This manuscript describes the use of the aforementioned characterizations represented in Figure 1 to repair any potential deviations that may arise during model execution. However, these characterizations may also be utilized in future studies for the detection of deviations, as well as for model optimization and improvement.

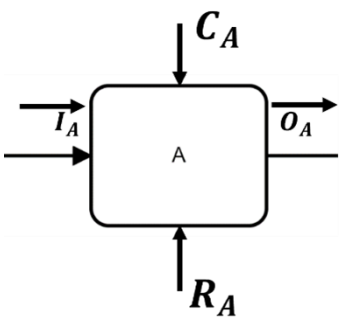

Figure 1. General model of a task.

Table 2 summarizes the new elements added to enrich the BPMN task and activity characterization, including constraints, resources, input, and output, to enable monitoring and control of model performance. Incorporating these elements can enhance the BPMN's representation of real-world processes and provide a more comprehensive framework for process modelling and management. A detailed discussion of these elements will be presented in the following.

Table 2. Task element characterization.

Task Characterization	Annotation	Task Entry	During Execution	Task Exit
Resources	R_A	True	True	...
Constraints	C_A	True	True	True
Input	I_A	True	...	...
Output	O_A	...	...	True

Output State Token Estimation

The output state token represents the set of valid state tokens when exiting task A. This information can be derived by applying transformations to the input state token $ST =< K, P, R, V >$ as described by O_A (cf. Section Task Characterization in Section 3.1.2) on KPIs, resources, products and variables. Instead of explicitly computing the complete set of valid state tokens, for computational issues, only a simpler superset of the valid state token is computed called an estimated state token ($\widetilde{ST}_A$). This estimated state token ($\widetilde{ST}_A$) is a set of state tokens containing, at minimum, all the valid states resulting from the actual transformation caused by task A. By extension, if the task's input consists of an estimated state token $\widetilde{ST}_{\underline{A}}$ instead of a single state token ($ST_{\underline{A}}$) the estimated output state token $\widetilde{ST}_{\overline{A}}$ is obtained by computing a superset of the union of all estimated output state tokens for each state token in $\widetilde{ST}_{\underline{A}}$.

As previously mentioned, the estimated state token is a set of potential state tokens. However, when the model is executed only one of the state tokens within this set will be realized. For example, in the case where the duration of a task falls within the range of [10 min, 20 min], it becomes impossible to ascertain the precise duration of this task during the estimation of the exit state token. Therefore, all possible values within the duration range must be taken into account in the estimated state token.

The following describes how the state token is estimated when tasks are connected to each other in a BPMN model. This paper focus on three main configurations: sequential execution (in series), parallel execution (AND) and exclusive execution (XOR).

- Sequential execution

  Considering a sequential configuration, as shown in Figure 2, $\widetilde{ST}_{\overline{A}}$ is estimated and becomes $\widetilde{ST}_{\underline{B}}$. Then, as describe above, $\widetilde{ST}_{\overline{B}}$ can be estimated (cd Section Output State Token Estimation of Section 3.1.2). This can be easily generalized to a sequence of n tasks.

Figure 2. Sequential execution of tasks.

- Parallel execution 

  In the parallel execution configuration, as illustrated in Figure 3, all paths are considered. Subsequently, each input state token is divided into several parts (one for each branch) such that the recombination of all parts reconstitutes the original input state token. Next, the estimated output state token for each branch is computed, and their recombination yields the corresponding estimated output state token for the parallel execution of $W_1, \ldots, W_n$. Furthermore, the estimated state token for the entire parallel execution is obtained by computing a simple superset of the union of all the estimated

state tokens. Instead of individually evaluating the transformation of each input state token, an estimation is made for the set of state tokens associated with each part (input of the branches). Nevertheless, this does not change the rest of the process.

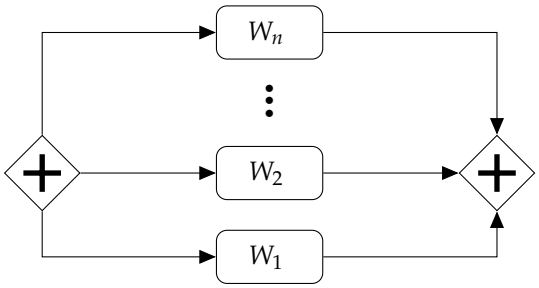

Figure 3. Parallel execution of workflows (AND).

- Exclusive execution
  In the case of exclusive execution, as shown in Figure 4, each branch is executed with the subset of the input state tokens that fit the entry condition of the branch. At the end, a simple superset of the union of all the estimated output tokens obtained from each branches is computed.

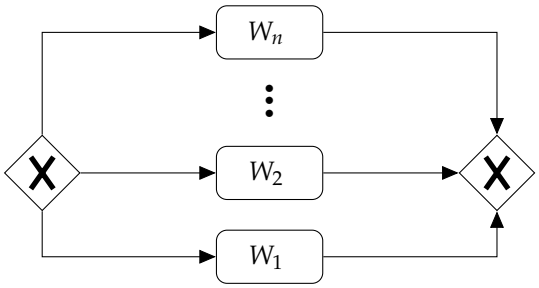

Figure 4. Exclusive execution of workflows (XOR).

3.1.3. Evaluation of BPMN Models

In practice, when evaluating the outputs of a BPMN model (or a portion thereof), the computation of estimated state tokens is performed throughout the model from an initial state token. The computation of the superset of state tokens varies depending on the phases of the entire process, as depicted in figure in Section 3.1.6. The further the phase, the more intricate the computations become, and the more time it consumes. The primary objective at each step is to eliminate as many impossible state tokens as possible. To optimize the computational efficiency, initial steps employ straightforward computations, requiring minimal time to process a huge number of candidates and effectively eliminate many of them (cf. Section 3.1.6).

3.1.4. Library of Patterns

The library of patterns is a database of patterns containing both atomic activities and sets of activities or a workflow. Unlike previous literature this library serves as a resource for storing fragments or full workflows without explicit mapping to specific deviations in process execution. The library of patterns can be viewed as a resource that can begin with no content and gradually expand to encompass thousands of workflows, ultimately serving to resolve deviations in process execution in an implicit manner. Different users contribute through collaboration. Moreover, it can be characterized by its input and output, required resources, and associated constraints, similar to how activities were previously characterized in Section Task Characterization in Section 3.1.2. Figure 5 illustrates the general model of the library of patterns. Hence, the library of patterns can be a valuable resource for all users in resolving deviations during process execution.

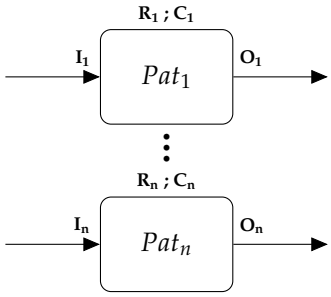

Figure 5. General model of a library.

3.1.5. Repairing Typology

In this paper, three main types of basic repairs are proposed to restore a BP model to an "executable state". The first type is called correction and involves correcting the system state at the start or end of a process. The second type is restore, which aims to fix a specific failing task A. The third type is bypass, which involves replacing one or more activities within the process. These repairs are illustrated in Figure 6.

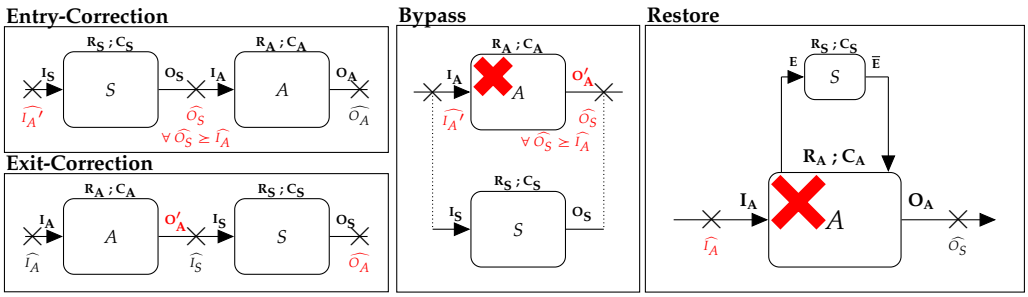

Figure 6. Repairing typology.

At this stage, it should be emphasized that the bypass option can be customized to replace one or multiple blocks as required, depending on where a library pattern can be injected as a solution in the main model. This approach is innovative when compared to previous methods described in the literature.

3.1.6. Repairing Mechanism

The proposed repairing mechanism can benefit from task characterization, state tokens, and a library of patterns. These concepts provide a better understanding of a task's role and expected outcome, aiding in identifying process deviations. In fact, monitoring task input, output, constraints, and resource requirements can quickly identify deviations in process execution, while, repairing deviations in a model can be a complex and challenging task, especially when dealing with large and complex systems. The proposed method employs pattern matching to identify candidates for deviation repairing. It utilizes a library of patterns to search for suitable solutions to the problem. An efficient search is achieved by using simple methods to quickly remove incompatible patterns as simple as comparing intervals and mathematical operations, reducing the set of candidates to a more manageable size. Afterwards, more complex analyses and methods, such as fuzzy sets and probabilistic approaches or even simulation, can be employed to further refine the search, and rank and choose the best solution.

Each phases of the method uses a different approach to process the evaluated estimated states throughout the model to compute the estimated state tokens as mention in Section 3.1.3. In each phases it is assumed that the set of valid state tokens is included in the estimated state tokens computed by the method. While using more complex solutions, the number of states in the estimated state tokens is narrowed down, converging towards the real set of state tokens. However it is important to mention that the more complex the methods, the longer the computation time. Thus, due to enormous number of potential

reparations, it is necessary to start with very simple computations that allow as many candidates as possible to be eliminated with minimum computations. The remaining candidates are then reduced using more and more complex methods; however, these are also more costly in terms of time (cf. Section Output State Token Estimation of Section 3.1.2). As shown in Figure 7, the first step of computation comprises using a method based on valid interval computation without considering resources. It can easily be proven that the obtained estimated state tokens follow the definition in Section Output State Token Estimation of Section 3.1.2, meaning that all real state tokens are included in any of the computed estimated state tokens. The temporal management of resources is one of the most time-consuming processes and is executed in the last steps of computation on a very limited set of alternatives (couple $(pattern, returning - point)$).

The proposed repair mechanism comprises several steps as outlined in the BPMN, Figure 7, and in the following.

1. Deviation detection:
   This paper assumes that a deviation has already been detected and focuses on the repair mechanism. Before calculating the state token, it is important to highlight three types of deviations: at the entry, during the execution and at the exit of a task (cf. Figure 6).
2. State token calculation:
   The subsequent step entails the computation of all estimated state tokens $\widetilde{ST}_{X_i}$ that would be correct if no deviation had been encountered. All $\widetilde{ST}_{X_i}$ (at the start and exit of tasks, and AND and XOR gates) between the deviation point D and the exit of the model are computed. In the following X_i is called nodes. The estimated state tokens ($\widetilde{ST}_{X_i}$) are compared with the estimated state token of the exit point for each pattern from the library to assess their compatibility and identify potential points of return for each pattern in the main model. (cf. matching output requirements below 3). This step is reprocessed for each advanced computational iteration. However, with each iteration, the number of computed nodes is reduced. As the computational method becomes more complex, it become more time consuming. Nevertheless, the number of elements to compute decreases. It is anticipated that the reduction in the number of computed elements surpasses the increase in time consumption per element computation.).
3. Library search:
   The stage is set for exploring the library of patterns containing workflows that may serve as solutions for the identified deviation. This process involves three phases: validating constraints, matching input requirements, and matching output requirements. At each phase, a significant number of incompatible patterns are swiftly discarded, resulting in a refined and condensed set of candidate solutions.

   - Validating constraints:
     This refers to the first step of the elimination process in which patterns that do not meet the required global constraints are swiftly removed, as defined in constraints in Section 3.1.2 and Table 2.
   - Matching input requirements:
     The state token is evaluated against the inputs of each pattern. Recalling Task Characterization in Section 3.1.2 and Table 2, the input requirements are constraints that must be verified with the state token. Two methods in matching the inputs requirements are distinguished:
     Method 1:
     Only patterns which strictly satisfy the constraints are kept in the potential solution set and all others are eliminated. This method is highly restrictive, and there is a possibility of overlooking candidates. Moreover, the selected cases may not necessarily represent valid solutions.

Method 2:
The proposed method involves retaining patterns that partially satisfy the constraints, while discarding those outside a set threshold value. The ratio of satisfaction can be expressed as a percentage and compared to a threshold value determined by the domain expert during the design phase. Patterns falling below the threshold value are discarded. This method is less restrictive, as it does not miss any solutions, but it requires more computation.
The difference between the two methods lies in the number of patterns retained in the potential set. Method 1 results in few selected patterns, with many discarded, while the second method retains more solutions in the solution set. In practice, one or a combination of the two methods can be applied to each constraint.
For example, exceeding a cost constraint by €4 in the car industry may be tolerable, whereas exceeding a €4 cost constraint in a electronic chip manufacturing may not be acceptable.

- Matching output requirements:
At this stage, the remaining workflow patterns are validated against the estimated state tokens from the deviation point to the end of the model, using the same methods described in Output State Token Estimation of Sections 3.1.2 and 3.1.3. However, at this stage, the set of candidates is comprises a combination of a workflow pattern with its potential injection point within the model (couple $(pattern, returning - point)$). In simpler terms, the same pattern can be injected (pattern's input and output) at the deviation point or it can bypass one or more blocks within the main model.

4. Advanced techniques:
At this stage, with a smaller set of candidates, more sophisticated computational techniques can be utilized to further refine the analysis. For instance, methods such as probability theory, fuzzy logic, and simulation tools can be considered as viable options. These techniques can help in the identification of the most effective solution patterns among the reduced set of candidates.
5. Solution ranking:
Once the set of candidates has been reduced to combinations of workflows with corresponding returning points, the next step is to evaluate each solution to rank and present them to the user. The evaluation process is critical in selecting the optimal solution(s) that meet the BP requirements while minimizing any potential negative effects on the system. In doing so, this aids in the decision-making process. To evaluate and rank candidates after narrowing them down to a collection of workflows with returning points, a BP specialist can set an objective function based on KPIs, such as quality, granularity, cost, and interoperability. The subjective nature of this function means that KPIs and their weights can differ across industries and specialists, highlighting the need for customized solutions. The objective function is denoted as $Obj(K)$, and can include KPIs such as granularity and overquality, providing an additional layer of evaluation. Granularity helps determine whether the proposed solutions are too general or too specific by analysing the number of blocks being bypassed on the main model; while overquality assesses whether a solution produces a higher output or quality than required, which may not be preferable over other solutions.

6. Solution selection:
   The solution selection process can either be manual or automatic, depending on the configuration. Automatic selection involves the system choosing the top-ranked solution based on the objective function, while manual selection involves presenting all the relevant performance indicators and the objective function to the decision maker for each solution. Regardless of the selection method, the chosen solution must not only solve the local problem but also align with the global objective.
7. Solution injection:
   After selecting a preferred solution, manually or automatically, it will be injected into the executed process. At this level there are two modes of injection: Non-recursive deviation and recursive deviation. The former is for isolated incidents while the latter involves an iterative approach where the system proposes an optimization at the main model level based on historical data. This enables the system to learn from past experiences, adapt to new circumstances, and continually improve its performance.

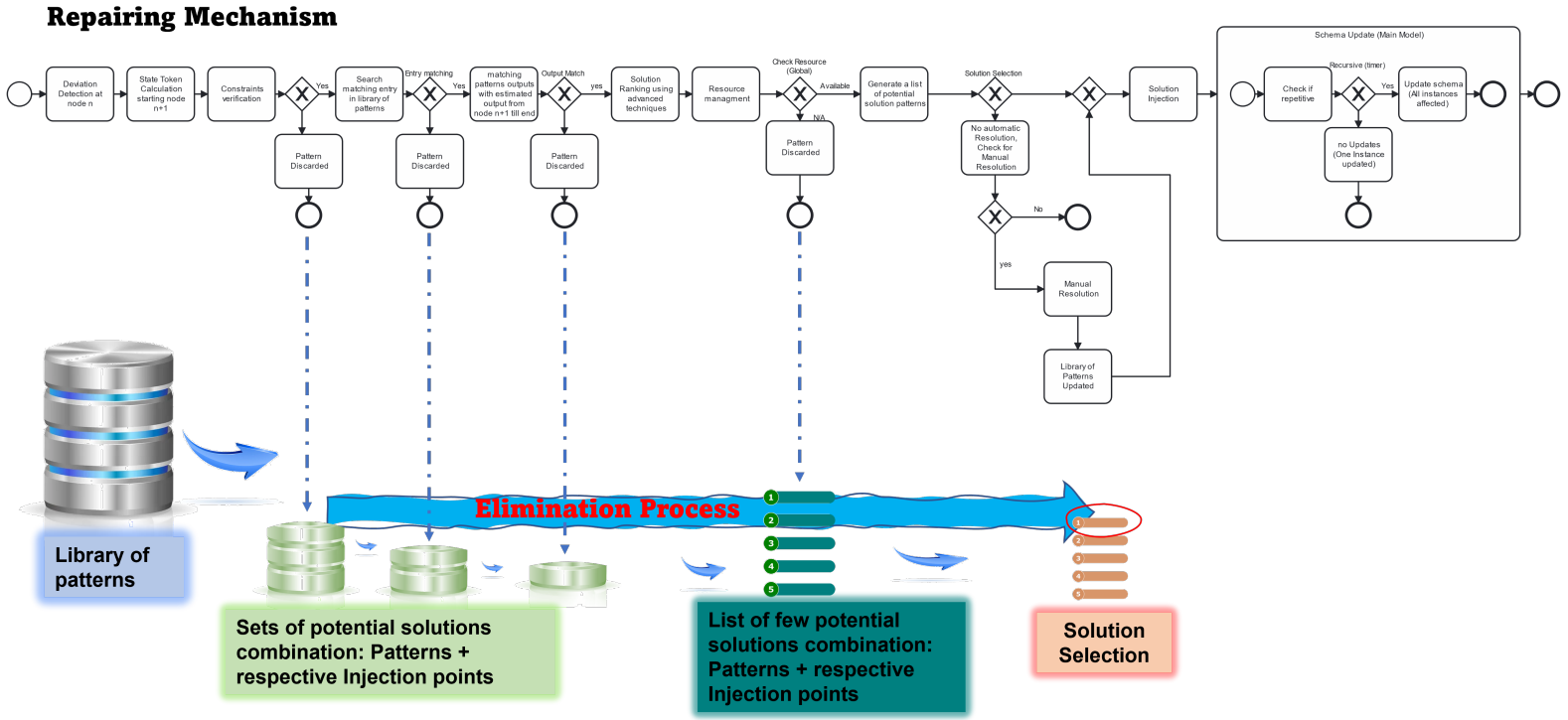

Figure 7. Repair mechanism.

4. Experiments and Results

In order to showcase the proposed approach and to demonstrate how fast the elimination of non-compatible patterns can be, the authors selected a piece of honeybee BPMN model developed with help of amateur and professional beekeepers in Lebanon and France, and well discussed in a previous publication from the same authors [16]. Before establishing the proof of concept for the bee hibernation model, the working environment was set up to interface with an existing BPMN model and process its tasks. Python 3.9.13 was chosen as the main scripting language as it includes a BPMN parsing package developed by Bocciarelli et al. [56] called py-bpmn-0.4.0, which allows for straightforward data extraction. The Tkinter Python package was used to create a GUI to display the final report. After setting up the environment, the first step was to extract all useful data from the given BPMN model in Figure 8 and access them inside the Python script. After specifying the path to the model's XML file, the py-bpmn-0.4.0 package scanned the file for its tags to map them to the respective tasks, gates, or conditions in the code.

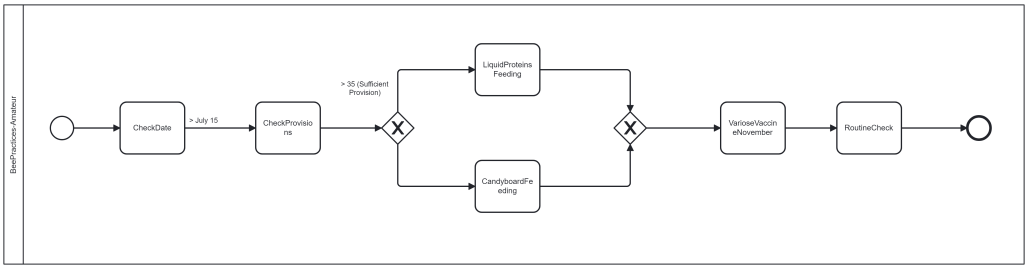

Figure 8. Part of the bee hibernation model.

After iterating over all the tasks, the script gains access to the methods named after each task, allowing the developer to simulate an action that takes place either before, during, or after the respective task, as shown in Figure 9.

```
43  <bpmn:exclusiveGateway id="Gateway_0d41f63">
44    <bpmn:incoming>Flow_0ckale9</bpmn:incoming>
45    <bpmn:incoming>Flow_1g1mzzk</bpmn:incoming>
46    <bpmn:outgoing>Flow_0dsg5yb</bpmn:outgoing>
47  </bpmn:exclusiveGateway>
48  <bpmn:serviceTask id="Activity_1i5d4a3" name="VarioseVaccineNovember">
49    <bpmn:extensionElements>
50      <zeebe:taskDefinition type="vaccine" />
51    </bpmn:extensionElements>
52    <bpmn:incoming>Flow_0dsg5yb</bpmn:incoming>
53    <bpmn:outgoing>Flow_0auzns6</bpmn:outgoing>
54  </bpmn:serviceTask>
55  <bpmn:serviceTask id="Activity_192xjq4" name="RoutineCheck">
56    <bpmn:extensionElements>
57      <zeebe:taskDefinition type="check" />
58    </bpmn:extensionElements>
59    <bpmn:incoming>Flow_0auzns6</bpmn:incoming>
60    <bpmn:outgoing>Flow_1sm7gsn</bpmn:outgoing>
61  </bpmn:serviceTask>
62  <bpmn:endEvent id="Event_05qmbam">
63    <bpmn:incoming>Flow_1sm7gsn</bpmn:incoming>
64  </bpmn:endEvent>
65  <bpmn:sequenceFlow id="Flow_0wi5uhq" sourceRef="Activity_0tm33n9" targetRef="Gateway_1emkcex" />
66  <bpmn:sequenceFlow id="Flow_1kt9pq1" name="&#62; 35 (Sufficient Provision)" sourceRef="Gateway_1emkcex" target
67    <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">payload.get("token_state").get("num_provisions")
68  </bpmn:sequenceFlow>
```

Figure 9. XML snippet of the BPMN hibernation process (a).

Once the XML file is parsed, every task in the process gains access to a shared variable called the 'payload' (represents the state token, see Section 3.1.1), which is automatically updated after every executed task, as discussed in Section Task Characterization in Section 3.1.2, where O_i was set to denote the transformation applied to the state token before entry. In addition, it can be updated manually by the user. This provides the opportunity to manually insert constraints and variables into the 'payload' variable to create a flow between the tasks representing task characterization, as discussed in Section Task

Characterization in Section 3.1.2. After setting custom constraints before and after some of the tasks, deviations can be simulated by, for instance, requiring a variable to be within a certain range at the end of a task, leading to the state token after exit. Upon completion, the 'payload' holds the location where the error took place based on the content of the state token, as shown in Figures 10 and 11.

```
class header();
def on_entertask ( self, **kargs ): ...
def on_exit_task ( self, **kargs ): ...
def on_CheckDate ( self, **kargs ): ...
def on_CheckProvisions ( self, **kargs ): ...
def on_CheckProteinFeeding ( self, **kargs ): ...
def on_VarioseVaccineNovember ( self, **kargs ): ...

def test_process:
instance = BpmnProcess()
instance.start_process ( open  ( "model/simple_bee.xml" , "r").read() , Handler() )
display_interface ( instance.payload )

if __name__ = '__main__':
test_process()
```

Figure 10. Parsed task methods, payload variables, and constraints (a).

```
def on_CheckDate ( self, **kargs ):
kargs['payload']['token_state'] = {}
kargs['payload']['deviations'] = {}
kargs['payload']['token_state']['date'] = 20230701
kargs['payload']['token_state']['temperature'] = 18
kargs['payload']['token_state']['weather'] = 'Sunny'
kargs['payload']['token_state']['cost'] = 100

def on_CheckProvisions ( self, **kargs ):
CheckConstraints ( "CheckProvisions", "enter", kargs )
kargs['payload']['token_state']['num_provisions'] = 20
kargs['payload']['token_state']['weight'] = 10
kargs['payload']['token_state']['cost'] += 40
CheckConstraints ( "CheckProvisions", "exit", kargs )
```

Figure 11. Parsed task methods, payload variables, and constraints (b).

Once the ability to access and modify the BPMN information was verified, the focus was shifted to simulating deviations and writing alternative search algorithms. Given the lack of a meaningful BPMN model to hand, the alternative tasks were generated by another Python script that includes a list of possible constraints and their range of values. Upon running the script, a number of alternatives are generated based on a random selection of random constraints.

To test the limits of the script, the chosen number of alternative tasks varied starting from 1000 and incrementally increased up to 2,000,000. This allowed the measurement of the different elapsed times required to load and search through the tasks. The generated number of alternatives was usually slightly less than desired due to discarding duplicates.

For the automated search, the main Python script was set to ensure that any constraint found within all the tasks was validated. Whenever a deviation or a mismatched constraint, is found, a method is triggered to search for alternative tasks among the generated pool that can fix the deviation. All alternatives are evaluated based on their entry constraints and their predicted exit values, depending on the number of remaining tasks in the process. For every alternative task, the entry constraints are first compared with the values in the current state token of the system. The number of mismatched constraints are recorded, and the alternatives are subject to a second round of filtering. The second stage of the search algorithm evaluates whether the alternatives' predicted outputs fall within the range of the entry constraints of the remaining tasks in the process, since it is possible for the same alternative task to connect to more than one remaining task. Once all possible alternatives are processed, a report is

generated describing the entire process, beginning with the original executed process and the state token across all the tasks, as shown in Figure 12.

The time taken for the program to load all data related to the alternative tasks is also displayed at the beginning of the report. This is used to evaluate how long the script takes to handle loading the alternatives depending on the size of the generated pool.

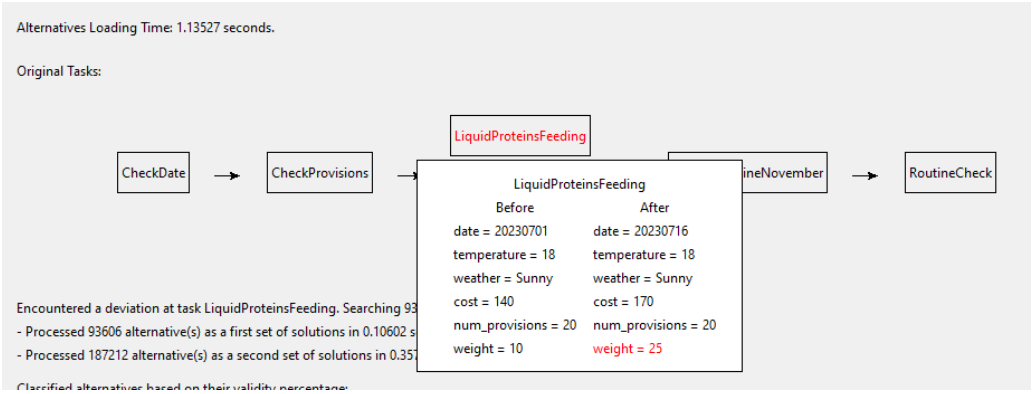

Figure 12. Display of the original process with a detected deviation.

The alternative tasks are classified in ranges depending on their percentage validity, which is the ratio of validated constraints over the task's total number of constraints. For alternative tasks that achieved total validity, i.e., when entry constraints and exit values satisfy all conditions in the state token, the solution is displayed in its suggested location in the updated process. The constraints of the original and new tasks in the updated process are also available to be viewed by the user, as shown in Figure 13.

It is noted that for alternative tasks that are able to be linked to more than one remaining task, multiple solutions are displayed showing the different possibilities that can be implemented.

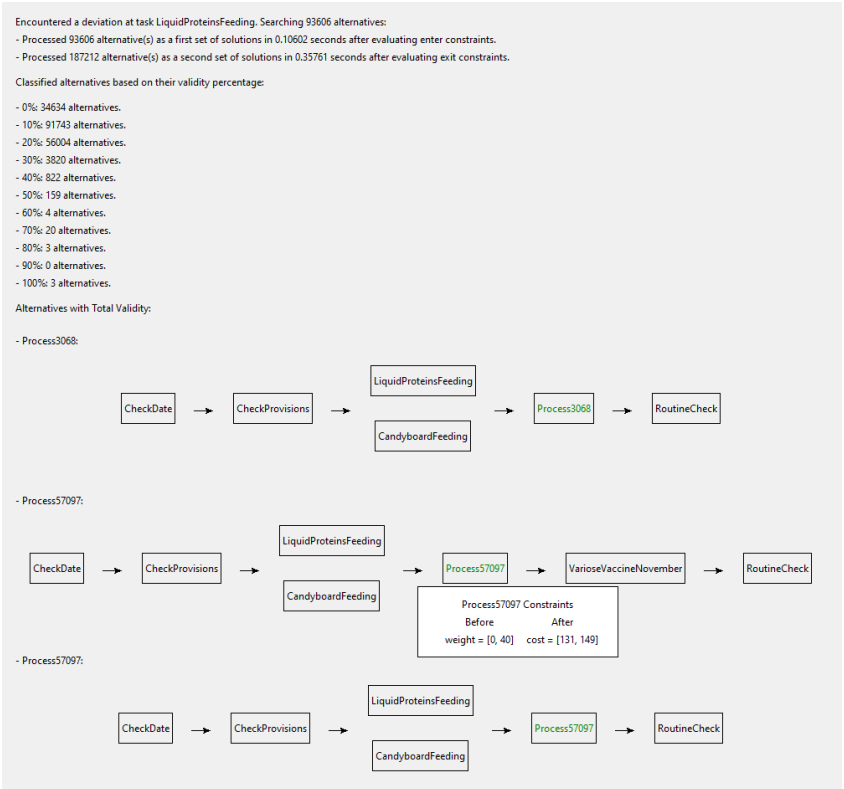

Figure 13. Classification of possible alternatives and display of the corrected process for valid alternatives.

Upon examining the elapsed time for each of the two filtering stages, for a small number of alternative tasks (less than 100,000), the displayed time was a small fraction of a second. This is due to the fact that the search process required very little time to complete given the simplicity of its requirements, and the Python "time" package inaccurately measures time when the intervals are very small. Increasing the number of generated alternatives increased the initial loading time and elapsed processing time, with greater precision.

The elapsed times were recorded in multiple cases with a growing pool of generated alternative tasks as shown in Table 3. The number of patterns written in the second column represents the actual number of generated tasks due to the elimination of duplicates. The values below are listed in seconds and are the averages of multiple executions of the script on the same device.

Table 3. Elapsed time during a search vs. the size of the library of patterns.

Generated Patterns	Patterns without Duplication	Total Elapsed Time (s)
1000	837	0.010
10,000	9930	0.153
100,000	93,606	1.589
1,000,000	905,217	26.291
2,000,000	1,809,169	55.908

As expected, there was a notable increase in the total elapsed time as the pool of alternative tasks grew. Yet, despite reaching an unreasonable amount at around 2,000,000, the script was able to process all alternatives in a matter of seconds and produced a meaningful report for the user, saving them hours of manual planning.

For each test case performed on the growing number of generated alternatives, the number of alternatives marked as a compatible solution and achieved partial validity are presented in Table 4.

Table 4. Number of patterns vs. the number of matching candidates.

Number of Patterns	100% Matching	70% Matching	50% Matching	30% Matching
1000 (837)	4	8	114	843
10,000 (9930)	6	26	280	5966
100,000 (93,606)	9	59	475	13,754
1,000,000 (905,217)	31	130	1174	25,149
2,000,000 (1,809,169)	64	274	2387	50,177

Alternative tasks with 100% validity were displayed on the user report at their correct location in the process as a suggested fix for the detected deviation. For the rest, it can be observed how the number of alternatives listed grew significantly larger as the percentage validity was less, or in other words the restrictions for them to be considered valid were looser. It is worth noting that, given the simplicity of the BPMN model used in these experiments, the process tasks in general have fewer constraints, resulting in the large number of alternatives with partial validity. For a realistic model where the tasks have many constraints, this number is expected to be lower in the range of high percentages of validity and higher as we approach 0% validity.

All the numerical results above were obtained by running the scripts on an ASUS ROG Strix G15 laptop with an Intel Core i7-10750H CPU, which affected the performance especially when the number of alternatives tested was in the millions. If ran on a server with dedicated hardware and optimization, such as multithreading, the loading time is expected to be significantly less for both the script that generates alternatives and the main script that loads and processes the said alternatives.

5. Discussion and Limitations

This paper presents a novel approach to address unexpected situations in a workflow by introducing three basic types of repairs with the ability to verify and validate each injected solution.

Unlike conventional approaches, where repairs are either predicted deviations or exceptions that are solved locally, this study introduces a dynamic approach that modifies the model by inserting or deleting parts of the workflow. In other words, the novelty lies in the fact that the library does not inherently contain solutions mapped to each problem, but rather it is populated with patterns stored by different collaborators. These patterns serve as potential solutions for unexpected problems, which distinguishes our approach from the existing literature. Furthermore, a unique aspect of this work is the replacement of not only one block on the main model but multiple blocks, accomplished through the utilization of a bypass-type repair mechanism.

In this paper, candidate solutions were evaluated and sorted based on performance indicators, and the decision maker can define an objective function to select the best solution automatically or manually. The proposed method utilizes a collaborative library of random patterns that are stored by domain experts. This library serves to identify compatible solutions based on the current state of the system, thus leveraging the expertise of both internal and external sources to address unexpected problems. Unlike conventional approaches found in the literature, this library does not store solutions for each problem. Instead, it stores patterns that may or may not serve as candidates for an unexpected deviation in the execution of a model.

One of the notable strengths of the proposed approach is its searching mechanism, which involves the system searching through a large number of patterns to identify, rank, and select the most suitable solution set. This was demonstrated in the previous section. The underlying concept behind this mechanism is to swiftly eliminate patterns that are not compatible by utilizing basic computing tools. For example, the system can compare if a range of values falls within a given interval or compute the likelihood of an event occurring. By employing these simple computational techniques, the potential set of solutions can be rapidly narrowed down to a smaller subset of patterns. Subsequently, more sophisticated tools can be applied to refine and accurately rank the results. In order to comprehensively evaluate the performance of our proposed mechanism, additional tests were conducted as part of the experiment. Specifically, the mechanism was tested on various library sizes, encompassing different scenarios including 1000; 10,000; 100,000; 1,000,000; and 2,000,000 patterns. The results of these tests are summarized in Table 3. This table provides valuable insights into the performance of our approach and demonstrates its effectiveness in terms of time efficiency.

This approach is in stark contrast to using intricate methods, such as simulations, from the outset, which are time-consuming to reduce the set of patterns. The proposed approach is designed to be efficient and effective by leveraging simple tools in the initial stages of the search process, and then using more advanced techniques as the set of candidates is narrowed down. By adopting this approach, the proposed method can provide a rapid and accurate identification of solutions.

In this paper new characterization and notations were provided, enriching the BPMN elements to better describe a deviation, a pattern and the system state. Basic BPMN gates were put in focus in terms of what is happening to the state token when it propagates throughout these gates. These characterizations were necessary to be able to compute where the output of a pattern can be injected later and how many blocks on the main model can be bypassed.

These characterizations provided a valuable framework for accurately calculating the impact of candidates on the system and identifying the most effective course of action to repair the workflow. It would have been challenging to reach the degree of accuracy and efficiency needed to appropriately resolve unforeseen issues in the workflow without these characterizations. The ease of implementing the methodology described in this

study through coding, as shown in the prior proof-of-concept case study, is one of its key benefits. By characterizing and formulating the approach, it was possible to easily add new characterizations to the existing BPMN elements and implement the entire approach using Python. This ease of implementation is a crucial factor in making the approach accessible and feasible for real-world applications, as it enables practitioners to quickly and efficiently incorporate it into their existing systems and workflows.

In addition, the proposed iterative solution injection approach involves the system continuously analysing historical data to suggest optimizations at the main model level. By learning from past experiences and adapting to new circumstances, the system can continually improve its performance. This approach opens up new possibilities for artificial intelligence studies in the future, where the system could use the historical database to further enhance the model's design. With this approach, the system can identify recurring deviations and propose improvements to the main model to prevent or minimize their impact.

The proposed approach has several limitations that should be acknowledged. Firstly, the types of repairs introduced, namely entry/exit corrections and bypass, are considered basic repairs. There may be room to explore additional types of repairs and alternative methods of injection. Further research could investigate more sophisticated repair techniques to expand the repertoire of available repair options.

Secondly, the focus of this research was primarily on forward repairs, addressing deviations and enabling the system to proceed with the remaining steps. However, it is essential to recognize that backward repairs, which involve revisiting previous steps to rectify errors or inconsistencies, may also play a crucial role in real-world scenarios.

More, it is important to note that resource management and scheduling aspects were intentionally excluded from the scope of this paper. While the proposed approach addresses deviations within the BP, the management of resources and scheduling considerations remain separate concerns. Future research could explore how resource management and scheduling techniques can complement this approach, providing a more comprehensive solution for process optimization.

The paper aimed to showcase the applicability of the proposed approach in real-world scenarios, with a particular focus on the apiculture domain due to time limitations. However, it is recognized that conducting broader tests in various other domains would provide further validation of the approach in diverse contexts.

6. Conclusions

In conclusion, this paper presented a novel approach to address unexpected deviations in a workflow. The proposed method leverages a collaborative library of random patterns curated by domain experts, and employs a rapid searching mechanism that eliminates non-compatible patterns using simple computing tools. The method introduces new characterizations and notations to enhance BPMN elements and adopts an iterative solution injection approach.

The effectiveness of the proposed method was demonstrated through its implementation in a proof-of-concept case study in the apicultural domain, aiding decision-making processes. Furthermore, the approach offers the potential for future integration of artificial intelligence and simulation studies, expanding its capabilities and scope.

During the evaluation of the proposed mechanism, we conducted comprehensive tests on various library sizes, covering a range of scenarios. The results of these tests showcase the performance of our approach and highlight its time efficiency.

While the proposed method represents a significant improvement over previous approaches, it is important to recognize its dynamic and efficient nature in addressing unexpected deviations. Notably, the solution injection capability allows bypassing one or several blocks within the model, enhancing its versatility and adaptability.

Finally, the ease of implementation of the proposed method ensures accessibility and feasibility for real-world applications. This combination of enhanced functionality, practical

implementation, and proven performance makes the proposed method a compelling solution for effectively managing workflows and aiding decision-making processes.

Author Contributions: Conceptualization, C.K., F.T., C.Y., A.A., N.D. and G.Z.; methodology, C.K., F.T., C.Y., A.A., N.D. and G.Z.; software, C.K., and K.J.; validation, C.K., F.T., C.Y., A.A., N.D. and G.Z.; formal analysis, C.K., F.T., C.Y., A.A., N.D. and G.Z.; investigation, C.K., F.T., C.Y., A.A., N.D. and G.Z.; resources, C.K., F.T., C.Y., A.A., N.D. and G.Z.; data curation, C.K., F.T., C.Y., A.A., N.D. and G.Z.; writing—original draft preparation, C.K., K.J., F.T.; writing—review and editing, C.K., F.T., C.Y., A.A., N.D. and G.Z.; visualization, C.K., F.T., C.Y., A.A., N.D. and G.Z.; supervision, C.K., F.T., C.Y., A.A., N.D. and G.Z.; project administration, F.T., C.Y., A.A., N.D. and G.Z.; funding acquisition, C.K., F.T., C.Y., A.A., N.D. and G.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Russell, N.; van der Aalst, W.M.; ter Hofstede, A.; Edmond, D. *Workflow Patterns*; MIT Press Ltd.: Cambridge, MA, USA, 2016; Volume 8, p. 384.
2. Nqampoyi, V.; Seymour, L.F.; Laar, D.S. Effective Business Process Management Centres of Excellence. In *Research and Practical Issues of Enterprise Information Systems*; Tjoa, A.M., Xu, L.D., Raffai, M., Novak, N.M., Eds.; Springer: Cham, Switzerland, 2016; pp. 207–222.
3. *IEEE Std 1320.1-1998*; IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0. IEEE: Piscataway, NJ, USA, 1998; pp. 1–116. [CrossRef]
4. van der Aalst, W.M.P. Everything You Always Wanted to Know About Petri Nets, but Were Afraid to Ask. In *Business Process Management*; Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J., Eds., Springer: Cham, Switzerland, 2019; pp. 3–9.
5. Davis, W.S. *Tools and Techniques for Structured Systems Analysis and Design*; Addison-Wesley Longman Publishing Co., Inc: Reading, MA, USA, 1983.
6. Grigorova, K.; Mironov, K. Comparison of business process modeling standards. *Int. J. Eng. Sci. Manag. Res.* **2014**, *1*, 1–8.
7. Specification, O.A. OMG Systems Modeling Language (OMG SysML™), V1. 0, 2007. Available online: https://www.omg.org/spec/SysML/1.0/PDF (accessed on 10 June 2023).
8. Cimino, M.G.; Vaglini, G. An Interval-Valued Approach to Business Process Simulation Based on Genetic Algorithms and the BPMN. *Information* **2014**, *5*, 319–356. [CrossRef]
9. White, S.A.; Miers, D. *BPMN Modeling and Reference Guide: Understanding and Using BPMN*; Future Strategies Inc.: Toronto, ON, Canada , 2008.
10. El Kassis, M.; Trousset, F.; Daclin, N.; Zacharewicz, G. Business Process Web-based Platform for Multi modeling and Distributed Simulation. In Proceedings of the I3M 2022-International Multidisciplinary Modeling & Simulation Multiconference, Rome, Italy, 19–21 September 2022.
11. OMG. Business Process Model and Notation (BPMN), Version 2.0. 2011. Available online: https://www.omg.org/spec/BPMN/2.0/PDF (accessed on 10 June 2023).
12. Freund, J.; Rücker, B. Real-Life BPMN: Includes an Introduction to DMN. 2019 . Available online: https://www.infomath-bib.de/tmp/data2/Real-Life%20BPMN%20-%20edition%204.pdf (accessed on 10 June 2023).
13. Bendraou, R.; da Silva, M.A.A.; Gervais, M.-P.; Blanc, X. Support for Deviation Detections in the Context of Multi-Viewpoint-Based Development Processes. In Proceedings of the CAiSE Forum, Luxembourg, 5–9 June 2012 ; pp. 23–31.
14. Rabah-Chaniour, S.; Zacharewicz, G.; Chapurlat, V. Process Centered Digital Twin. GDR MACS (2022). Online Oral Presentation. Available online: https://action-jn.sciencesconf.org/resource/page/id/5 (accessed on 31 May 2022).
15. Speakman, S.; Tadesse, G.A.; Cintas, C.; Ogallo, W.; Akumu, T.; Oshingbesan, A. Detecting Systematic Deviations in Data and Models. *Computer* **2023**, *56*, 82–92. [CrossRef]
16. Kady, C.; Chedid, A.M.; Kortbawi, I.; Yaacoub, C.; Akl, A.; Daclin, N.; Trousset, F.; Pfister, F.; Zacharewicz, G. IoT-Driven Workflows for Risk Management and Control of Beehives. *Diversity* **2021**, *13* , 13070296. [CrossRef]
17. Hodge, V.; Austin, J. A survey of outlier detection methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
18. Tariq, Z.; Charles, D.; McClean, S.; McChesney, I.; Taylor, P. Anomaly Detection for Service-Oriented Business Processes Using Conformance Analysis. *Algorithms* **2022**, *15*, 257. [CrossRef]
19. Ni, W.; Zhao, G.; Liu, T.; Zeng, Q.; Xu, X. Predictive Business Process Monitoring Approach Based on Hierarchical Transformer. *Electronics* **2023**, *12* , 1273. [CrossRef]

20. Ducq*, Y.; Vallespir, B. Definition and aggregation of a performance measurement system in three aeronautical workshops using the ECOGRAI method. *Prod. Plan. Control* **2005**, *16*, 163–177. [CrossRef]
21. Heguy, X.; Zacharewicz, G.; Ducq, Y.; Tazi, S.; Vallespir, B. A performance measurement extension for BPMN: One step further quantifying interoperability in process model. In *Enterprise Interoperability VIII: Smart Services and Business Impact of Enterprise Interoperability*; Springer: Cham, Switzerland, 2019; pp. 333–345.
22. Ougaabal, K.; Zacharewicz, G.; Ducq, Y.; Tazi, S. Visual Workflow Process Modeling and Simulation Approach Based on Non-Functional Properties of Resources. *Appl. Sci.* **2020**, *10*, 4664. [CrossRef]
23. Kherbouche, O.M.; Ahmad, A.; Basson, H. Using model checking to control the structural errors in BPMN models. In Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science (RCIS), Paris, France, 29–31 May 2013; pp. 1–12. [CrossRef]
24. Mallek, S.; Daclin, N.; Chapurlat, V. The application of interoperability requirement specification and verification to collaborative processes in industry. *Comput. Ind.* **2012**, *63*, 643–658. [CrossRef]
25. Da Silva, A.S.; Ma, H.; Zhang, M. A GP approach to QoS-aware web service composition including conditional constraints. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 2113–2120. [CrossRef]
26. Zhang, S.; Paik, I. An efficient algorithm for web service selection based on local selection in large scale. In Proceedings of the 2017 IEEE 8th International Conference on Awareness Science and Technology (iCAST), Taichung, Taiwan, 8–10 November 2017; pp. 188–193. [CrossRef]
27. Borgida, A.; Murata, T. Tolerating exceptions in workflows: A unified framework for data and processes. In Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration, San Francisco, CA, USA, 22–25 February 1999; pp. 59–68.
28. Svendsen, A. *Application Reconfiguration Based on Variability Transformations*; Technical Report 2009-566 School of Computing; Queen's University Kingston: Kingston, ON, Canada, 2009; p. 4.
29. Gao, H.; Huang, W.; Yang, X.; Duan, Y.; Yin, Y. Toward service selection for workflow reconfiguration: An interface-based computing solution. *Future Gener. Comput. Syst.* **2018**, *87*, 298–311. [CrossRef]
30. Rosa, M.L.; Aalst, W.M.V.D.; Dumas, M.; Milani, F.P. Business process variability modeling: A survey. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–45. [CrossRef]
31. Gottschalk, F.; Van der Aalst, W.M.; Jansen-Vullers, M.H. Configurable process models–a foundational approach. In *Reference Modeling*; Springer: Cham, Switzerland, 2007 ; pp. 59–78.
32. Becker, J.; Delfmann, P.; Knackstedt, R. Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models. In *Reference Modeling: Efficient Information Systems Design through Reuse of Information Models*; Springer: Cham, Switzerland, 2007 ; pp. 27–58.
33. Bayer, J.; Kettemann, S.; Muthig, D. *Principles of Software Product Lines and Process Variants*; Technical Report; 2004. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=13e31abc3bbbf13af8a71157b4761839d7587bab (accessed on 10 June 2023).
34. Hallerbach, A.; Bauer, T.; Reichert, M. Issues in modeling process variants with provop. In Proceedings of the Business Process Management Workshops: BPM 2008 International Workshops, Milano, Italy, 1–4 September 2008; Revised Papers 6; Springer: Cham, Switzerland, 2009; pp. 56–67.
35. Haidar, H.; Daclin, N.; Zacharewicz, G.; Doumeingts, G. Enterprise Modeling and Simulation. In *Body of Knowledge for Modeling and Simulation: A Handbook by the Society for Modeling and Simulation International*; Springer: Cham, Switzerland, 2023; pp. 221–247.
36. Hauder, M.; Pigat, S.; Matthes, F. Research Challenges in Adaptive Case Management: A Literature Review. In Proceedings of the 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, Ulm, Germany, 1–2 September 2014; pp. 98–107. [CrossRef]
37. Herranz, C.; Martín-Moreno Banegas, L.; Dana Muzzio, F.; Siso-Almirall, A.; Roca, J.; Cano, I. A Practice-Proven Adaptive Case Management Approach for Innovative Health Care Services (Health Circuit): Cluster Randomized Clinical Pilot and Descriptive Observational Study. *J. Med. Internet Res.* **2023**, *25*, e47672. [CrossRef] [PubMed]
38. Badakhshan, P.; Conboy, K.; Grisold, T.; vom Brocke, J. Agile business process management: A systematic literature review and an integrated framework. *Bus. Process Manag. J.* **2020**, *26*, 1505–1523. [CrossRef]
39. Abrahamsson, P.; Conboy, K.; Wang, X. 'Lots done, more to do': The current state of agile systems development research. *Eur. J. Inf. Syst.* **2009**, *18*, 281–284.
40. Conboy, K. Agility from first principles: Reconstructing the concept of agility in information systems development. *Inf. Syst. Res.* **2009**, *20*, 329–354. [CrossRef]
41. Russell, N.; van der Aalst, W.M.; Ter Hofstede, A.H. *Exception Handling Patterns in Process-Aware Information Systems*; BPM Cent. Rep. BPM-06 BPMcenter. Org; 2006; Volume 208. Available online: https://www.researchgate.net/profile/Nick_Russell2/publication/228618501_Exception_handling_patterns_in_process-aware_information_systems/links/0fcfd50850ad0af074000000/Exception-handling-patterns-in-process-aware-information-systems.pdf (accessed on 10 June 2023).
42. Adams, M.; Ter Hofstede, A.H.; Van Der Aalst, W.M.; Edmond, D. Dynamic, extensible and context-aware exception handling for workflows. In Proceedings of the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS: OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, 25–30 November 2007; pp. 95–112.

43. Jasinski, A.; Qiao, Y.; Fallon, E.; Flynn, R. A Workflow Management Framework for the Dynamic Generation of Workflows that is Independent of the Application Environment. In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Virtual Conference, 17–21 May 2021; pp. 152–160.
44. Andree, K.; Ihde, S.; Pufahl, L. Exception handling in the context of fragment-based case management. In Proceedings of the Enterprise, Business-Process and Information Systems Modeling: 21st International Conference, BPMDS 2020, 25th International Conference, EMMSAD 2020, CAiSE 2020, Grenoble, France, 8–9 June 2020; pp. 20–35.
45. Górski, T.; WOźniak, A.P. Optimization of Business Process Execution in Services Architecture: A Systematic Literature Review. *IEEE Access* **2021**, *9*, 111833–111852. [CrossRef]
46. Xie, L.; Luo, J.; Qiu, J.; Pershing, J.A.; Li, Y.; Chen, Y. Availability "weak point" analysis over an SOA deployment framework. In Proceedings of the NOMS 2008—2008 IEEE Network Operations and Management Symposium, Salvador, Brazil, 7–11 April 2008; pp. 473–480. [CrossRef]
47. Mennes, R.; Spinnewyn, B.; Latré, S.; Botero, J.F. GRECO: A Distributed Genetic Algorithm for Reliable Application Placement in Hybrid Clouds. In Proceedings of the 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, 3–5 October 2016; pp. 14–20. [CrossRef]
48. Dyachuk, D.; Deters, R. Service Level Agreement Aware Workflow Scheduling. In Proceedings of the IEEE International Conference on Services Computing (SCC 2007), Salt Lake City, UT, USA, 9–13 July 2007; pp. 715–716. [CrossRef]
49. Dyachuk, D.; Deters, R. Ensuring Service Level Agreements for Service Workflows. In Proceedings of the 2008 IEEE International Conference on Services Computing, Honolulu, HI, USA, 8–11 July 2008; Volume. 2, pp. 333–340. [CrossRef]
50. Zhong, Y.; Li, W.; Wang, X.; Fan, J. An approach to agent-coalition-based Automatic Web Service Composition. In Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Wuhan, China, 23–25 May 2012; pp. 37–42. [CrossRef]
51. Qiqing, F.; Xiaoming, P.; Qinghua, L.; Yahui, H. A Global QoS Optimizing Web Services Selection Algorithm Based on MOACO for Dynamic Web Service Composition. In Proceedings of the 2009 International Forum on Information Technology and Applications, Chengdu, China, 15–17 May 2009; Volume 1, pp. 37–42. [CrossRef]
52. Hashmi, K.; AlJafar, H.; Malik, Z.; Alhosban, A. A bat algorithm based approach of QoS optimization for long term business pattern. In Proceedings of the 2016 7th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 5–7 April 2016; pp. 27–32. [CrossRef]
53. Kalasapur, S.; Kumar, M.; Shirazi, B.A. Dynamic Service Composition in Pervasive Computing. *IEEE Trans. Parallel Distrib. Syst.* **2007**, *18*, 907–918. [CrossRef]
54. Ukor, R.; Carpenter, A. Flexible Service Selection Optimization Using Meta-Metrics. In Proceedings of the 2009 Congress on Services - I, Los Angeles, CA, USA, 6–10 July 2009; pp. 593–598. [CrossRef]
55. Russell, N.; Van Der Aalst, W.M.; Ter Hofstede, A.H.; Edmond, D. Workflow resource patterns: Identification, representation and tool support. In Proceedings of the CAiSE, Porto, Portugal, 13–17 June 2005; Springer: Cham, Switzerland, 2005; Volume 5, pp. 216–232.
56. Bocciarelli, P.; D'Ambrogio, A. A BPMN extension for modeling non functional properties of business processes. In Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, Boston, MA, USA, 4–9 April 2011; pp. 160–168.