applied sciences

MDPI

Article

A Pattern Based Method for Simplifying a BPMN Process Model

Mateo Ramos-Merino *, Luis M. Álvarez-Sabucedo, Juan M. Santos-Gago and Francisco de Arriba-Pérez

Telematic Engineering Department, Escola de Enxeñaría de Telecomunicación, Universidade de Vigo, Campus Lagoas-Marcosende, 36.310 Vigo, Galicia, Spain; lsabucedo@gist.uvigo.es (L.M.Á.-S.); jsgago@gist.uvigo.es (J.M.S.-G.); farriba@gist.uvigo.es (F.d.A.-P.)

* Correspondence: mateo.ramos@gist.uvigo.es; Tel.: +34-986-814-073

Received: 26 April 2019; Accepted: 4 June 2019; Published: 5 June 2019

Featured Application: To work with a simplified version of a BPMN model that condenses the information of interest can be very interesting from a human point of view (the understanding of the model is facilitated). Moreover, an optimised version of a BPMN model can yield in more efficient results for mining process software and data analytic techniques.

Abstract: BPMN (Business Process Model and Notation) is currently the preferred standard for the representation and analysis of business processes. The elaboration of these BPMN diagrams is usually carried out in an entirely manual manner. As a result of this human-driven process, it is not uncommon to find diagrams that are not in their most simplified version possible (regarding the number of elements). This work presents a fully automatic method to simplify a BPMN process model document. A two-phase iterative algorithm to achieve this simplification is described in detail. This algorithm follows a heuristic approach that makes intensive use of a Pattern Repository. This software element is concerned with the description of feasible reductions and its enactment. The critical concept lies in the discovery of small reducible patterns in the whole model and their substitution with optimised versions. This approach has been verified through a double validation testing in total 8102 cases taken from real world BPMN process models. Details for its implementation and usage by practitioners are provided in this paper along with a comparison with other existing techniques concerned with similar goals.

Keywords: BPMN; simplification; process

1. Introduction

Business Process Model and Notation (BPMN) is an Object Management Group (OMG) standard for specifying business processes. The current version is 2.0.2 [1], and it allows the modelling of business processes both from a graphical perspective and from a machine-readable perspective (using an XML-based representation). It makes BPMN a suitable language for the description of processes in a visual way and their subsequent analysis using data analytic or process mining techniques [2]. Presently, BPMN is a leading industry standard for specifying workflows. It is broadly used, for example, in the domain of health-care protocols [3], software process tailoring [4], Data Quality management [5] or Internet of Things proceedings [6], among many other cases.

This paper describes an approach for the simplification of a BPMN process model aimed to optimise the number of elements included in the workflow. During the simplification task, the behaviour described initially must be preserved, i.e., the new simplified workflow model must be coherent with the original one, offering the same flow logic.

This simplification procedure aims to achieve two primary goals, namely:

- Provide a cleaner and easier to understand graphic representation. Therefore, a better understanding with less effort of these workflows will be provided to human readers [7].
- Reduce the complexity of the model for its processing by software agents. An optimised version of a BPMN model will yield more efficient results for process mining software and data analytic techniques.

It must be borne in mind that an efficient BPMN simplification can be a complicated task. In real scenarios, the number of possible paths in a workflow can grow very fast as the number of activities increases. Furthermore, due to loops and other structures in the model, the number of different combinations in the activities execution can cause an explosion in the number of states. This paper shows an iterative procedure to achieve the simplification objective. In the proposed approach, the deployment of a Pattern Repository will play a principal role in the simplification procedure. This pool of BPMN-based patterns is gathered by the authors, and it contains the core of the proposal.

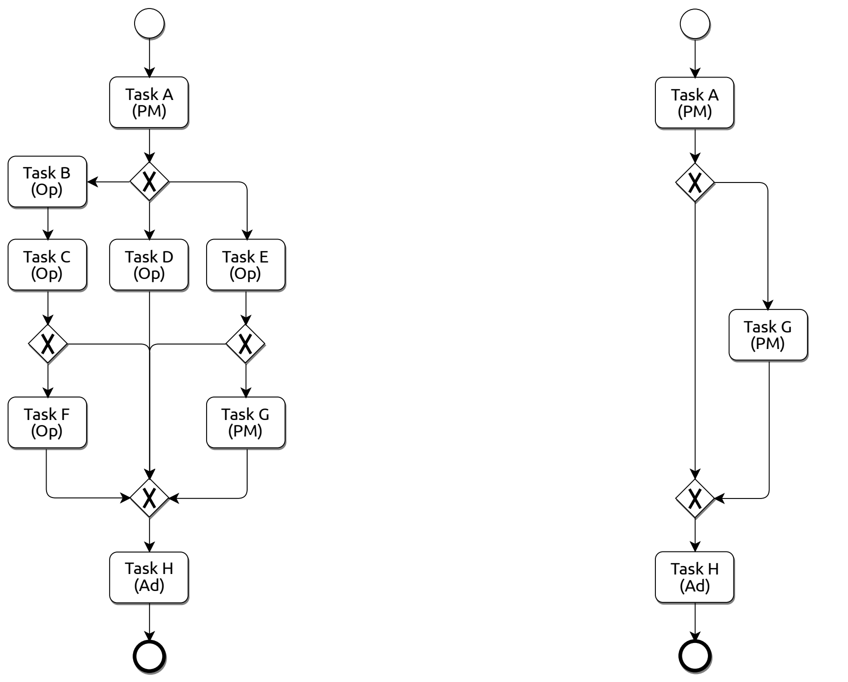
The main goal of this contribution is to provide a mechanism to simplify, if possible, a given BPMN Process type workflow (that describes a sequence or flow of activities with the objective of carrying out work). This type of workflows are commonly created in a non-automatic manner, i.e., a human operator creates them. First of all, this manual creation process is prone to introduce redundant or repetitive paths that do not provide new information to the behaviour of the model. In this situation, the proposed algorithm will identify the unnecessary elements and will generate a new simplified BPMN diagram offering an equivalent description of the workflow. Secondly, the application of the simplification algorithm is also intended for more complex issues. In particular, presently, it is common the generation of BPMN diagrams involving hundreds of activities from different perspectives and different levels of abstraction [8]. It provides a series of advantages [8] but also results in more complex flowcharts and more analysis time demanding models.

To illustrate that issue, the reader can consider Figure 1. It represents a process involving activities from three different roles: Project Manager (PM), Operator (Op), and Administrative (Ad). Let us suppose now that due to particular constraints, only a subset of these activities are relevant for concrete analysis. In a specific context, it could be interesting to focus on a particular role (or a set of them). Let's assume that we are concerned just with management operations, i.e., those performed by the Project Manager and Administrative roles. Therefore, it would be required to eliminate those activities in which the other roles (i.e., the Operator) participate. In this case, taking into account the entire model in Figure 1a is an unnecessary burden as only activities A, G and H are of interest. Therefore, before launching the analysis (either visually by a human or by any automatic techniques), it will be convenient to get a simplified model containing only the desired activities, like the model shown in Figure 1b.

In this example, not only the unneeded activities have been eliminated. The resulting model includes fewer Exclusive Gateways and fewer paths (sequence flows). In short, a new version of the entire workflow has been generated, but it was preserved its original behaviour (taking into account only those activities of interest). Despite the simplicity of this example, the complexity of this transformation is elevated for a real world BPMN model.

As the reader may note, if an activity or set of them do not provide any actual information, then removing it can be a real advantage. The simplification is clear in a more compact model: the fewer elements involved, the easier it is to understand the model. Of course, the resulting model can be tackled in a more efficient fashion in further analysis and operations. Analysis of the simplified workflow (including only the needed behaviour) can produce even more transparent and concise results due to the elimination of unnecessary elements. In this way, it is possible to have a general model with hundreds of activities and generate different perspectives depending on the particular interests. With this, we will have access to a complete family of processes.

This type of simplification attains greater significance in real world process models. In an organisation with hundreds of models containing hundreds of activities, it is possible to find superfluous or repetitive paths that do not provide new information to the behaviour of the model.



(**a**) Example BPMN model (**b**) Simplified model

Figure 1. Input and output of the simplification process.

This work presents an automatic heuristic-based approach to perform this type of simplification on any given BPMN model. Starting with a model (like in Figure 1a) and a list of the interesting activities, the objective is to achieve a simpler model, as presented in Figure 1b, containing just the relevant information for a specific goal. To get this result, several issues must be borne in mind as discussed in following sections.

The remainder of this paper is structured as follows. Next section highlights other approaches available in the literature related to this work. Then, Section 3 discusses about the pattern repository, a main piece of the proposed algorithm. Section 4 deeply describes the algorithm devised. Section 5 performs a discussion about the complexity of the algorithm. Afterwards, Section 6 presents the algorithm validation using 8102 cases taken from real world examples. Section 7 verifies the usability of the proposal in a real scenario. Finally, Section 8 points out the conclusions.

2. Related Work

The work presented in this paper is related to initiatives and projects in the Business Process Model Abstraction (BPMA) and Business Process Variability Modeling (BPMV) domains. Main similarities and differences regarding some of the most significant proposals in these areas are reviewed.

BPMA has been proposed in last years (see [8–11]) as a solution to a common problem appearing in large companies worldwide. Traditionally, companies maintain process model repositories including thousands of different workflows for each of the perspectives and goals to be tackled. As a consequence, the models within their repositories tend to include complex interrelationships, and they may even include overlaps or the same behaviour in different perspectives [8]. In this scenario, the primary purpose of BPMA is to design detailed models containing the whole set of information about the processes and to provide suitable abstraction mechanisms for generating the desired model view. This way, different kinds of users can access different views and perspectives on processes with an adapted visualisation. The approach proposed in our work can help to achieve these objectives, as it is intended to provide an algorithm to generate a simplification on a process model based on a set of relevant activities.

According to [8], and based on the cartographic generalisation by [12], BPMA deals with three main generalisation components: "why", "when" and "how". "Why" is concerned with the reasons and the goals of the abstraction. "When" includes the rules, thresholds and conditions to perform the abstraction. "How" describes the transformation and techniques for achieving the abstraction. This last one explains the method and the algorithms for performing the simplifications of the models. The simplification algorithm proposed is framed on the "how" generalisation component of BPMA.

The basic abstraction operations defined in BPMA are "elimination" (to delete elements in the model to reduce the coverage level) and "aggregation" (to create groups of elements to increase the granularity level). The former produces a model containing no information about the deleted elements while the latter preserves some part of the information. In the present work, the simplification algorithm is concerned with the elimination operation using a pattern matching method. In fact, some BPMA approaches, such as [13–15], perform eliminations using a sibling concept including Event-Driven Process Chains (EPCs) [16] fragments and transformation rules. There are also some other works that explore the pattern reduction method, using a closer approach to this proposal. In [17], five reduction rules that remove the undesired elements in the workflow model are presented. The authors aim to identify structural conflicts in the process model (like deadlocks and lack of synchronisation). They propose some correctness criteria to identify this structural conflicts by exploring all possible instance sub-graphs. However, starting with a brute force method to explore them may not be computationally effective in many contexts. To fight back this issue, they propose to perform firstly a simplification of the model based on these five rules previously identified. In our case, the objective is to delete from the model all the elements and structures that are correct but not significant for purposes of analysis, and, then, to explore the resulting model. However, even the simplification concept can be similar in some way to the previous works, the final goal is very different. The major similarities are related to the "adjacent reduction" and "closed reduction" rules. These rules can be applied in both works: [17] and the current one.

In any case, these works tackle the issue exclusively from a theoretical point of view, not a practical one. They do not provide a practical mechanism capable of carrying out and implementing these operations automatically. Additionally, these approaches are not based on BPMN notation, but on other languages and models (for example EPC). On the contrary, our proposal focuses, firstly, on providing a practical mechanism that, using a scalable repository of patterns, implements a solution in a fully automatic manner. Secondly, it is based on the BPMN language, the most widely used solution for the description of workflows in productive environments.

In [15] it is used some of the reduction rules previously identified in [17], but with a different purpose. In this case, the objective is closer to BPMA. The authors of this work are concerned with the "how" component and focus their efforts on the elimination operation. They identify Single Entry Single Exit (SESE) blocks and substitute them with edges or paths. In this procedure, the authors apply reduction rules for simplifying some elements in the process model. Besides of these similarities, the simplification rules are not the most important part of that contribution. The core contribution of these works is concerned mainly with the aggregation and elimination operations on the so-called SESE blocks. Also, they are not focused on BPMN, as they are using what they call "process schemes", a process graph with atomic activities and control elements between them. Moreover, our approach is not limited only to Single Entry Single Exit blocks. Multiple Entry Multiple Exit blocks are also analysed.

It is common in organisations to work with families of process variants. In this way, the conventional modelling languages have not specific support to represent these families. Over the past decade, several approaches have studied the Business Process Variability Modeling (BPVM) field in order to tackle this issue by extending the conventional business process modelling languages [18]. Presents a complete survey in the BPVM domain. It analyses 66 relevant publication from the year 2000 that cover 23 approaches to solving the problem. According to the variability mechanism, the approaches can be sorted into four classes: node configuration, element annotation, activity specialisation and fragment customisation. The approach proposed in the current paper can be categorised in the fourth class: fragment customisation. This class includes the works that propose the manipulation (insert, delete or modify) of entire fragments in order to create variation and customisation of the base process model. This class includes two main groups of approaches: Process variants by options (the most relevant works are [19–22]) and Template and Rules (the most relevant works are [23,24]). In the first group, the main idea is to mark the process model with the called adjustment points and to perform the change operations over these predefined points. This approach supports four types of operations to change the process flow: delete, insert, modify and move. For example, with the move operation, it is possible to relocate a fragment in the base model bounded by two adjustment points to another part bounded by two distinct adjustment points. The second group associates a set of business rules to a specific template model that will be used to perform the change operations. The different process model perspectives allowed in the family can be inferred by using the rules associated with the template model.

Meanwhile the main concepts behind all the BPVM approaches are very similar to our work, the philosophy and the application of the proposal are quite different. In particular, our approach uses a combination of the insert and delete basic manipulations to implement a complex operation: replace. In our case, a full Multiple Entry Multiple Exit fragment will be replaced by a simplified version offering the same behaviour but using fewer elements. One of the keys of our proposal lies in to obtain an efficient process model (i.e., with fewer elements). Moreover, our approach is generic and can be applied without modification to any BPMN process model. The most relevant works in the literature discussed in this section do not use this approach. They define rules and fragment for a specific family of models used by a specific organisation.

Other related works are those that deal with Business Process (BP) patterns [25]. The BP patterns are examples of process modelling that show how to connect activities within the workflow to solve a specific problem (i.e., synchronisation, parallel split, or exclusive choice, among others). To represent these patterns, a fully graphical notation oriented to human users is usually used. The Workflow Patterns Initiative has been working since 1999 in this field. The objective of this initiative is to provide a conceptual basis for business process design on different modelling perspectives (control flow, data, resource, and exception handling). Using the proposed patterns, the modelling stage of the business process is facilitated and a concise and straightforward workflow that adequately represents the desired behaviour is achieved. This goal has certain similarities with the one pursued by the present article. The BP patterns intend to approach the problem from the beginning, the modelling phase, while our proposal has another approach. A given model can change along its life cycle, so simplification in later phases, after the initial modelling stage, can be required. The following sections show actual situations (common in certain fields [26]) where it is required to remove a subset of the activities from a BPMN model. Under this premise, the simplification of the BPMN in the middle of its life cycle turns out to be of the utmost importance. In this context, the present proposal stands out, developing an alternative point of view and automating the task of simplification.

The gist of the above-mentioned works on BPMA and BP patterns has much in common with the fields of refactoring [27] and anti-patterns [28] which in turn are closely related to our proposal. Antipatterns share certain features with the simplifiable fragments of a process model and can be found for different modelling languages (EPC, BPMN, PetriNets, YAWL, among others). The task of refactoring resembles the action of searching and replacing these fragments with an optimised version. For a long time, there have been many works devoted to refactoring on varied contexts (for example centred on Unified Modeling Language (UML) diagrams [29], class diagrams [30] or use cases [31]) but it has been just in recent years when more proposals related to process models have appeared. In general, each work has different objectives and different refactoring premises (quality in terms of the number of elements, ease of understanding, consistency with changes, etc.). Ref. [32] focus on those process models that are shared by different users and analyse how to maintain a dynamic workflow in which several people interact along their life cycle. These works focus on evaluating the consistency of the patterns in this type of dynamic environments. There is a large number of works, for example [33,34], that aim to improve some aspect of the quality of the models through refactoring but do not present a fully automated mechanism. In these works, assistance to the user is particularly relevant as they suggest sets of model refactorings. Ref. [35] presents an interesting approach in which human interaction plays an important role in replacing fragments taking into account the semantics of the model and the real behaviour intended to be represented. Therefore, it is possible to propose substitutions that, even though not formally coherent, result in a final model more friendly and very simplified. However, in order to achieve this objective, the automation of the process is also sacrificed.

A set of works in the bibliography related to refactoring follow an approach that is known as Model transformations by demonstration (for example [36,37]). The general idea is first to monitor the substitution procedure performed by a human user. Thus, all atomic operations performed by the user are captured by comparing the initial and final models. In this way, they can propose a set of recommendations associated with the refactoring operations that are used to assist the simplification procedure. During the last few years, many works related to the application of refactoring and antipatterns techniques for the field of process models have emerged. However, and to the best of our knowledge, neither of them focuses on the type of patterns identified in this work nor shares the goal tackled by this work. This is due, in part, to the fact that the patterns considered by our work are not very common in the early stages of the life cycle of process models. This situation changes when analysing successive iterations in process remodelling. Notably, the patterns proposed here begin to appear, and their substitution becomes especially important when the process model has to undergo changes. In this scenario, it may be very useful to eliminate subsets of activities. These circumstances occur more often than is usually thought at first sight since the remodelling and enhancement of a process model is an essential task within the life cycle of BPMN models.

Another point that is not very usual in the field of refactoring and that occurs in the present work is based on using additional information about the process model itself to carry out the simplification. This aspect can be seen in the examples proposed in Section 7, which takes into account the roles of the people involved in the process to perform various types of simplifications. This approach is mentioned as a trend and research opportunity in [38]. This document also considers as a relevant trend the creation of fully automatic mechanisms for simplification. This feature is also a relevant aspect in the present work.

From a high level of abstraction, and contrary to other approaches already mentioned in the literature review, the main objective in the current work is to achieve a simpler and more efficient model to use in combination with additional analysis and process mining techniques to improve the understanding of the model by a human user (similar goals are discussed as future challenges for the Business Process Model domain in [39]). Furthermore, this work is performed over the preferred standard for the representation and analysis of business processes, BPMN, making it of a broader usage for the community.

3. Pattern Repository

To meet the proposed objective, authors decided to build a repository containing ordered pairs of patterns. Each of these pairs is made of the complex version of a pattern and its simplified version. This repository will be used later by the proposed algorithm to carry out the simplification of any given BPMN model. This section focuses on the identification of the minimal set of required patterns and its representation.

3.1. Repository Population

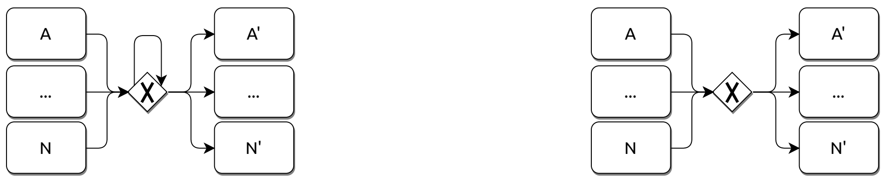
In the frame of the present work, a pattern is understood as a sub-part of a larger BPMN process model, i.e., a set of different elements (activities, gateways, connectors, etc.) with well-defined relationships. A pattern is considered as reducible (or simplifiable) if another pattern that represents the same behaviour than the first one but using fewer elements is possible. Figure 2a,b exemplify this idea with two simple patterns. As shown, the original pattern contains more sequence flows elements than the simplified one. However, replacing one block with the other one would not affect the behaviour of the global model. In the rest of the figures (Figures 3–6), the reader can observe the other pairs of patterns in the repository. For example, in Figure 5, a pattern including three different Exclusive Gateways is presented. The reader can note that one of the Exclusive Gateways (the superior one) is not needed. Thus, it can be removed without losing any relevant feature. Its inputs will be the new inputs for the lower right gateway. Applying this simplification reduces the total number of elements in the new model.

Taking into account the pattern repository, it can be followed that the pattern pool will contain two groups of patterns connected by a function. This function maps a set of reducible or simplifiable patterns (such as Figure 2a) to a set of simplified patterns (such as Figure 2b). Formally speaking, the function is surjective because there are different simplifiable patterns that can lead to the same simplified pattern.

To discover those pairs of patterns that constitute the repository, a systematic review of a broad set of BPMN models was carried out. A series of steps have been performed to finally identify the set of 5 simplifiable patterns considering the Exclusive Gateway elements of the BPMN notation. This type of elements is the most common one to define the behaviour at a decision point in real BPMNs. The process of discovery of the patterns has pursued two main objectives:

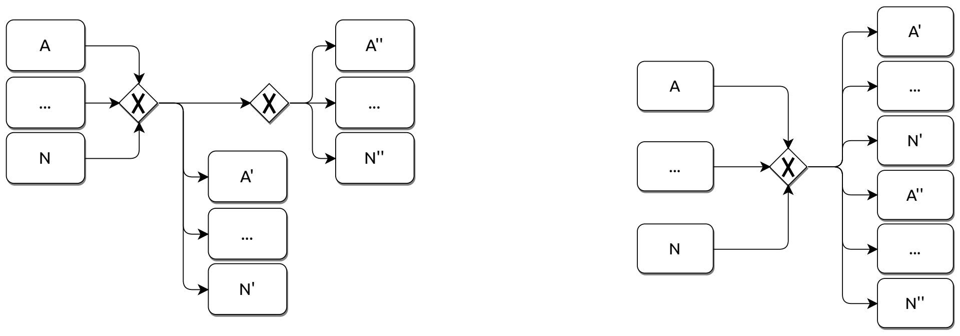
- Identify a set of pattern pairs that would substantially reduce a BPMN process model.
- Get the minimal set of them. Thus, greater efficiency is guaranteed when carrying out the search and replacement tasks performed by the simplification algorithm.

The identification of patterns was conducted using a limited set of BPMN models intended to discover simplifiable fragments visually. These BPMN models were generated from real models, inserting on them several redundant paths to get potentially highly simplifiable models. Starting with a BPMN model that refers to a real context, a random set of activities was eliminated (replacing them with a connection between the previous element and the next one). This process is similar to the procedure followed in stage 1 of the algorithm (see Section 4). By systematically repeating this procedure on different real BPMN models and eliminating different combinations of activities, a working set is obtained on which to identify the simplifiable patterns.



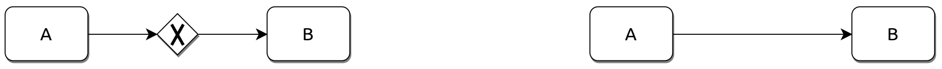
(**a**) Original pattern (**b**) Simplified pattern

Figure 2. Patterns pair A.



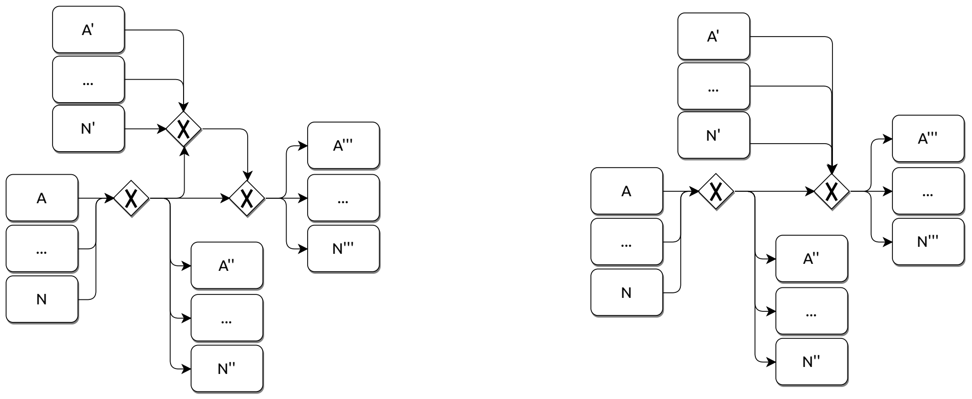
(**a**) Original pattern (**b**) Simplified pattern

Figure 3. Patterns pair B.



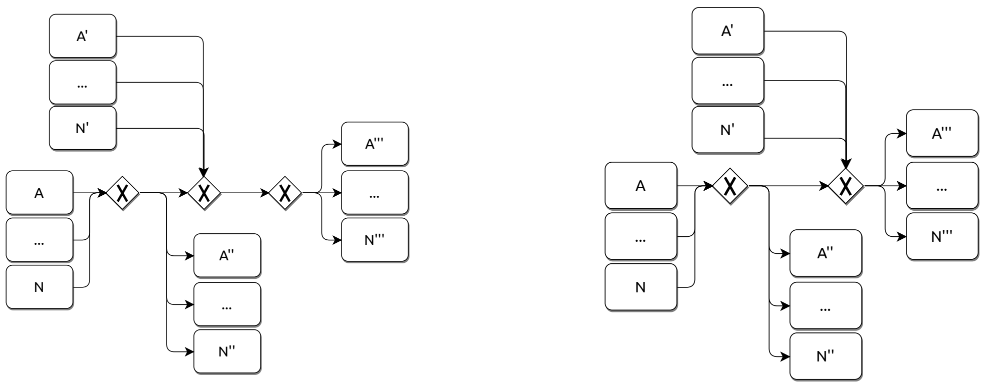
(**a**) Original pattern (**b**) Simplified pattern

Figure 4. Patterns pair C.



(**a**) Original pattern (**b**) Simplified pattern

Figure 5. Patterns pair D.



(**a**) Original pattern (**b**) Simplified pattern

Figure 6. Patterns pair E.

On a first step, the process of identification of patterns was carried out individually by each of the authors. Each one of the BPMN models of this working set was analysed following a series of stages: (i) identify a simplifiable fragment; (ii) carry out a repeated simplification of this fragment until all the occurrences of it have been eliminated; (iii) analyse the resulting model and check if there is still redundancy; (iv) if so, identify a new type of simplifiable fragment and (v) repeat steps (ii), (iii) and

(iv) until a model that does not show redundancy is obtained. With this first step, each of the authors obtained a set of simplifiable BPMN fragments.

On a second stage, the pairs of patterns identified by each author was put together, and the duplicated ones were discarded. Patterns that could be broken down into several smaller patterns were also eliminated. This feature is especially significant to guarantee the minimum possible number of patterns in the repository. Finally, a set of 5 pairs of patterns shown in Figures 2–6 were obtained.

3.2. Pattern Representation

The patterns in the repository are software components. In particular, the repository patterns are represented as Java classes implementing the SimplifiablePattern interface shown on the Lisitng 1. This interface includes the methods necessary to search and replace a simplifiable pattern in the original model with a simplified one:

- **searchPattern** This method has to look for the reducible pattern in the model to simplify. If the pattern is found, it returns the point in the model in which the pattern has been identified.
- **reducePattern** This method has to simplify a pattern in a model (found previously by the previous method) by replacing the reducible pattern with the simplified pattern associated (that contains less elements).

```java
public interface SimplifiablePattern{
  public BpmnId searchPattern(Bpmn model);
  public Bpmn reducePattern(Bpmn model, BpmnId startElmnt);
}
```

Listing 1: Interface SimplifiablePattern.

Thus, the Java classes intended to contain the information about the pair of patterns will play a central role as they will include not just the graph models but also the artifacts to support the operations required in the simplification process (described in the next section).

The population of the repository begins with the identification of the pair of patterns: the simplifiable pattern and its simplified version (cf. Figures 2–6). From the analysis of both versions, the code must be developed in order to transform the BPMN fragments information into a Java class that contains the behaviour of the simplifiable pattern and the actions that must be performed on the BPMN to apply the simplification. As an example, the patterns pair shown in Figure 2 is implemented on the Listing 2.

```java
public class PA implements SimplifiablePattern{
@Override
public BpmnId searchPattern(Bpmn model){
 ArrayList<BpmnExGateway> exGateways = model.getExclusiveGateways();
 for(BpmnExGateway gateway : exGateways)
 {
  ArrayList<BpmnId> outSequencesIds = gateway.getOutSequencesIds();
  for(BpmnId seqId : outSequencesIds)
  { //check if some sequence returns to gateway
   BpmnId nextElmnt = model.getSequenceById(seqId).target();
   if (nextElmnt.equals(gateway.getId()))
   {//Element found
    return new BpmnId(gateway.getId());
   }
  }
 }
 return null;
}

@Override
public Bpmn reducePattern(Bpmn model, BpmnId start){
 BpmnId gId = startElement;
 BpmnExGateway gateway = model.getGatewayById(gId);
 ArrayList<BpmnId> outSequencesIds = gateway.getOutSequencesIds();
 for(BpmnId seqId : outSequencesIds)
 {//search all sequence returning to gateway
  BpmnId nextElmnt = model.getSequenceById(seqId).getTarget();
  if (nextElmnt.equals(gateway.getId()))
  {//Element found
   model.deleteSequence(seqId);
   gateway.deleteOutgoing(seqId);
   gateway.deleteIncoming(seqId);
  }
 }
 return model;
}
}
```

Listing 2: First pair of patterns implemented as a Java class.

The first method of the interface, *searchPattern*, consists mainly in a loop along all the exclusive gateways in the model to find out if one of them matches with Figure 2a. For each gateway, it looks for a sequence that exits and returns to it (i.e., doing nothing). If a gateway with these conditions is found, the method returns it. Conversely, the second method, *reducePattern*, receives as input parameter the starting element of the pattern found by the previously described method (if the case). The objective now is to convert the original pattern into its simplified version (cf. Figure 2a,b) inside the whole BPMN process model. In this case, it looks for all the sequences that are not necessary and deletes them.

The reader may note the use of the Java class *Bpmn*. This class is intended to parse an XML file describing a BPMN compliant document into a set of data structures representing the BPMN process model. Also, it offers a collection of high level operations to operate on the model facilitating the implementation of the patterns as previously described (cf. Figure 7). It includes, for example, methods to get the input and output sequences of an element or to delete and modify the connections between the activities.

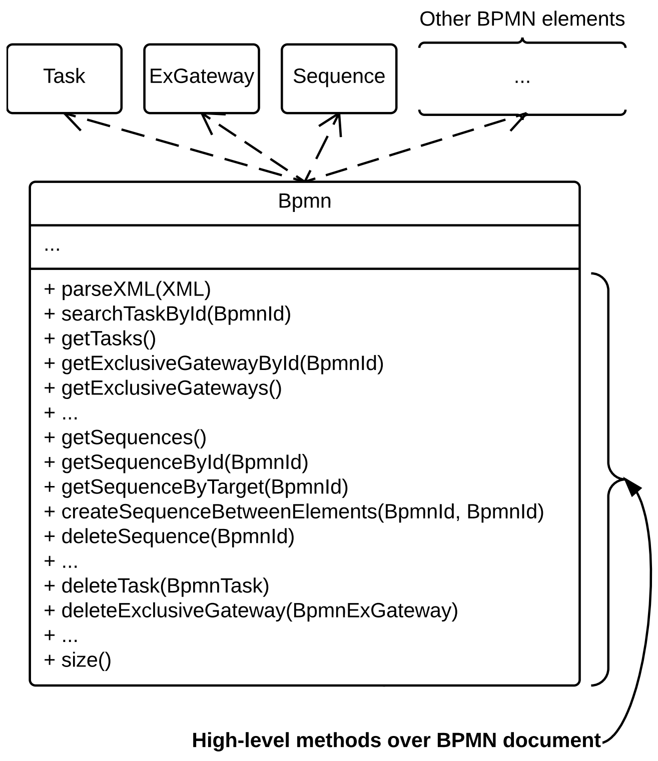

Other BPMN elements
Task
ExGateway
Sequence
...
Bpmn
...
+ parseXML(XML)
+ searchTaskById(BpmnId)
+ getTasks()
+ getExclusiveGatewayById(BpmnId)
+ getExclusiveGateways()
+ ...
+ getSequences()
+ getSequenceById(BpmnId)
+ getSequenceByTarget(BpmnId)
+ createSequenceBetweenElements(BpmnId, BpmnId)
+ deleteSequence(BpmnId)
+ ...
+ deleteTask(BpmnTask)
+ deleteExclusiveGateway(BpmnExGateway)
+ ...
+ size()
High-level methods over BPMN document

Figure 7. The BPMN class

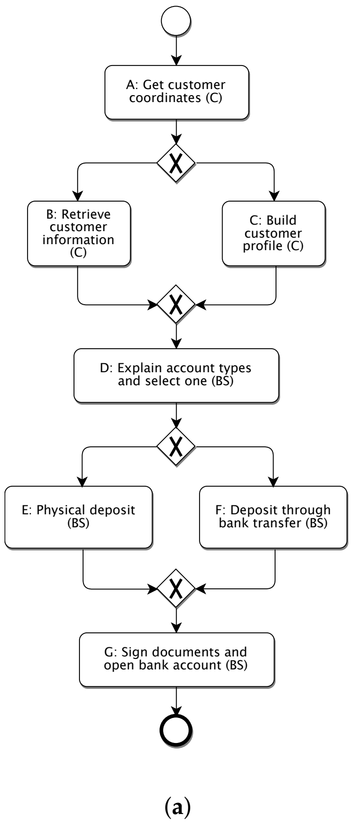
To illustrate the implementation of the pattern repository it has been presented a straightforward example involving a pair of patterns. The separation of the execution into the two methods can sound to the reader unnecessary. However, real usage has brought into light that it is very useful for the sake of code simplicity and scalability in the more sophisticated patterns. The complete repository of all the patterns identified by the authors was implemented under this scheme. This repository is easily scalable by adding new patterns as new Java classes.

4. Algorithm Description

The model simplification algorithm is tackled in a two-stage approach. The first stage is concerned with the generation of the new version of the model by removing the undesired/unneeded tasks. Afterwards, using this new model as input, the optimised version of the model is obtained as the result of an iterative procedure that uses the repository of patterns described previously. Eventually, a redesigned and simpler version of the model, if possible, is obtained.

In the following subsections a real world example will be presented to illustrate the phases of the algorithm. An excerpt of a workflow that describes the process of opening a bank account is considered. In this workflow (cf. Figure 8), activities can be classified into two groups:

- Operations performed by the client using a computer terminal (C).
- Operations involving interaction with the bank staff (BS).


A: Get customer coordinates (C)
B: Retrieve customer information (C)
C: Build customer profile (C)
D: Explain account types and select one (BS)
E: Physical deposit (BS)
F: Deposit through bank transfer (BS)
G: Sign documents and open bank account (BS)
(a)

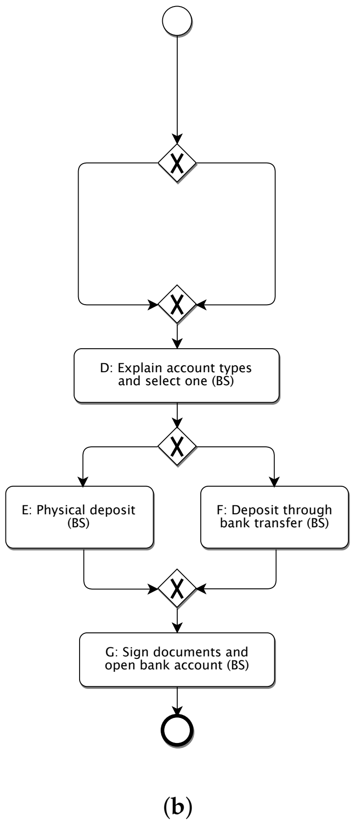

D: Explain account types and select one (BS)
E: Physical deposit (BS)
F: Deposit through bank transfer (BS)
G: Sign documents and open bank account (BS)
(b)

Figure 8. Opening a bank account. (**a**) All the operations; (**b**) Operations involving interaction with the bank staff.

4.1. First Stage: Deleting Undesired Activities

On the first stage of the proposed schema, undesired activities selected by the user must be removed. The resulting model must be still coherent. Therefore, the algorithm must delete the appropriate sequence flows in the BPMN models too (i.e., sequence flows connecting the deleted activity). Also, new sequences connecting the surrounding elements of a deleted activity (previous with next ones) must be created. In Figure 8 an example is shown of the input and the output of this stage, in case of aiming at getting a simplified version of the workflow for operations requiring interaction only with the bank staff, Figure 8a (representing the input) is considered as the starting point. In this line, Figure 8b represents the output model where sequence flows replace the operations without interaction with bank staff.

As the reader can note, in this new model, not all elements in Figure 8b are required to represent the desired behaviour. This issue will be tackled later on, on the second stage of the algorithm.

A fragment of the Java implementation for the first stage is presented in the Listing 3. The first stage function loops over the list of activities to be deleted and removes the activities defined by the user. To delete one single activity it is mandatory to establish a direct connection between the previous element and the upcoming one (another activity, a gateway, etc.). Upon the new connection is created, it is possible to delete the old activity and the old sequences as they are no longer of any use in the model.

```java
public Bpmn firstStage(Bpmn model, ArrayList<BpmnId> tasksToDeleteIds){
  for(Id taskToDeleteId : tasksToDeleteIds)
  { //Search each task to delete...
    BpmnTask task2Del = model.searchTaskById(taskToDeleteId);

    //... and delete it safetely
    model= this.deleteTask(model, task2Del);
  }
  return model;
}

//Deletes a task in a safetely way
private Bpmn deleteTask(Bpmn model, BpmnTask task2Del){

  //Search sequences Ids
  BpmnId inSeq = task2Del.getInSequenceId();
  BpmnId outSeq = task2Del.getOutSequenceId();

  //Search elements before/after these sequences
  BpmnId elmntBeforeTaskId = model.getSequenceById(inSeq).getSource();
  BpmnId elmntAfterTaskId = model.getSequenceById(outSeq).getTarget();

  //New connection skipping the activity
  model.createSequenceBetweenElements(elmntBeforeTaskId,elmntAfterTaskId);

  //Delete activity and old sequences
  model.deleteTask(task2Del);
  model.deleteSequence(inSequenceId);
  model.deleteSequence(outSequenceId);

  return model;
}
```

Listing 3: First Stage Java code.

In this first stage, a simple method to remove the undesired activities is used: to delete one, a direct connection between the previous element and the upcoming one (another activity, a gateway, etc.) is established. Upon the new connection is created, it is possible to delete the old activity and the old sequences as they are no longer of any use in the model.

4.2. Second Stage: Simplifying the Model

In this stage, the goal is to simplify the outcome from the previous stage to obtain a simpler version. As the reader can note from Figure 8b, the workflow produced in the first stage can include unneeded BPMN elements and, therefore, it can be simplified to conduct more efficiently further operations on it. This is because the first stage eliminates the activities creating connections between previous and upcoming elements.

Figure 9 shows the general architecture for the second stage. As the reader can see, different elements must work together to conduct the simplification of the model. The pattern repository uses a *Bpmn* class containing general utility functions to perform operations over the BPMN model. The Patterns Engine, in the centre of the figure, can use the patterns included in the repository to offer a high level API that is used to perform the simplification.

An excerpt of the Java implementation for this engine and the API deployed is shown in the Listing 4. The piece of code includes some essential methods that will be used for the second stage algorithm. The engine maintains a list of the patterns available in the repository (check line 3). This list is dynamically populated in the constructor method of the engine. It allows looping over all the patterns simplifying each one in the model. The engine also offers methods to simplify of the next pattern in the list (lines 20–31), to check if the list has been entirely explored (lines 33–38), and to rewind the pattern iterator (lines 40–42).

```
 1 public class PatternsEngine
 2 {
 3   ArrayList<SimplifiablePattern> patterns;
 4   int current;
 5 
 6   //Engine constructor
 7   PatternsEngine(){
 8     //Load dinamically all patterns in repository
 9     //Exception handling omited
10     Reflections r = new Reflections("repository.package");
11     Set<Class<? extends SimplifiablePattern>> repo =
12                                              r.getSubTypesOf(SimplifiablePattern.class);
13 
14     patterns = new ArrayList<>();
15     for (Class<? extends SimplifiablePattern> pat : repo) {
16       patterns.add(pat.newInstance());
17     }
18   }
19 
20  //Simplify next pattern in the BPMN model
21  public Bpmn simplifyNext(Bpmn model){
22     SimplifiablePattern aPattern = patterns.get(current);
23     if(this.hasNext()){
24        currentPattern++;
25     }
26     BpmnId startElmnt = aPattern.searchPattern(model);
27     if (startElmnt != null){
28        model=aPattern.reducePattern(model, startElmnt);
29     }
30     return model;
31   }
32 
33   public boolean hasNext(){
34     if(currentPattern < patterns.size())
35       return true;
36     else
37       return false;
38   }
39 
40   public void rewindPatterns(){
41     currentPattern = 0;
42   }
43 }
```

Listing 4: Patterns Engine Java code.

In this stage, the algorithm, using the API offered by the *PatternsEngine*, looks for existing sub-models in the input model that matches any of the existing ones in the repository. Upon each matching, the corresponding transformation is conducted. This process keeps on going until no more changes on the model are possible. The process of searching and reducing a pattern is performed until no more reductions can be evaluated over the model.

Finally, as an example, in Figure 10 the reader can see the outcome of the second stage on the model from Figure 8.

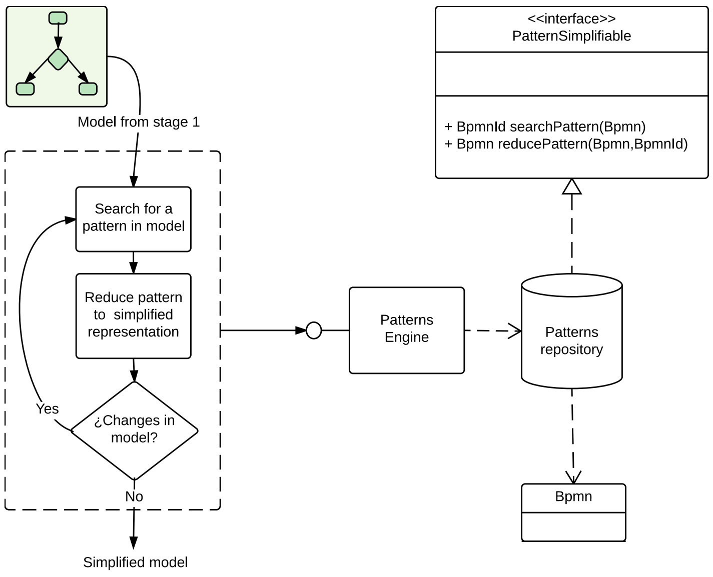

Model from stage 1
Search for a pattern in model
Reduce pattern to simplified representation
¿Changes in model?
Yes
No
Simplified model
Patterns Engine
Patterns repository
<<interface>> PatternSimplifiable
+ BpmnId searchPattern(Bpmn)
+ Bpmn reducePattern(Bpmn,BpmnId)
Bpmn

Figure 9. Architecture from second stage.

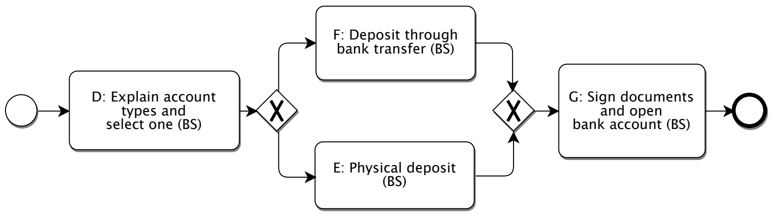

D: Explain account types and select one (BS)
F: Deposit through bank transfer (BS)
E: Physical deposit (BS)
G: Sign documents and open bank account (BS)

Figure 10. Result after stage two.

5. Complexity

In this section, the complexity of the algorithm is analysed from two different points of view: (i) regarding the size of the input BPMN model and (ii) regarding the number of pattern pairs in the repository.

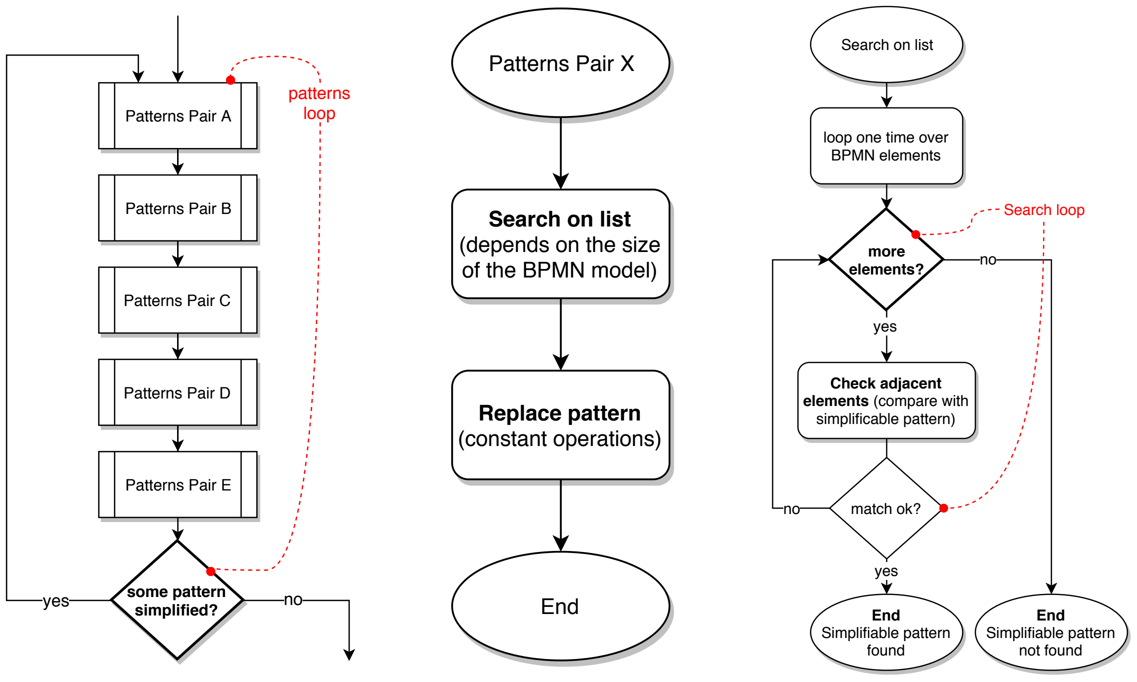
To carry out this analysis, the flow diagrams shown in Figure 11 will be used. These diagrams contain the basic behaviour of the algorithm with the most relevant operations from the point of view of time complexity. Firstly (see Figure 11a), the algorithm consists of a patterns loop that runs through the set of pairs of patterns within the repository. As explained in the Section 4, this loop is executed until no more patterns to simplify can be found. In this way, the number of executions of the loop depends on the complexity of BPMN model under consideration.

In each execution of the patterns loop, each of the pairs of patterns is probed (with the search operation) for a possible replacement (see Figure 11a).

The replacement of a pattern (Figure 11b) has a complexity of $O(1)$ because its execution time does not depend on the size of the BPMN model or the number of patterns in the repository. The replacement operation involves simple modifications on the BPMN model, actually, just adjusting or deleting some

BPMN elements. The reader can see on the Listing 2 the Java code responsible for the replacement of pattern A (lines 21–36).

Regarding the search operation (Figure 11b,c), its complexity cannot be disregarded. The search for a possible simplifiable pattern needs to carry out a loop that checks each element included in the BPMN model. However, it is sufficient to go through the entire list of elements only once during a search operation. In each iteration the focus is on one BPMN element and its immediately adjacent elements (those connected directly with a sequence flow stereotype). The idea is to check if the relationship between the considered element and its adjacent elements is identical to the one expressed in the simplifiable pattern. These check operations have a complexity of $O(1)$ (typically it will be a comparison between pairs of values) and their execution time does not depend on the size of the BPMN, nor the number of patterns in the repository. As the reader can note, the time consumption in this model is due to the search loop (Figure 11c), which only depends on the size of the BPMN model linearly.


Patterns Pair A
Patterns Pair B
Patterns Pair C
Patterns Pair D
Patterns Pair E
patterns loop
yes
some pattern simplified?
no
Patterns Pair X
Search on list (depends on the size of the BPMN model)
Replace pattern (constant operations)
End
Search on list
loop one time over BPMN elements
Search loop
more elements?
no
yes
Check adjacent elements (compare with simplificable pattern)
no
match ok?
yes
End Simplifiable pattern found
End Simplifiable pattern not found

(**a**) (**b**) (**c**)

Figure 11. Flow diagrams to analyse the complexity. (**a**) Algorithm flow diagram; (**b**) Patterns pair flow diagram; (**c**) Search simplifiable pattern.

It is also worth noting that it is not uncommon for a BPMN model to show a tree-like design, i.e., to include many branches on its representation. On first thought, this could pose an issue for searching a particular fragment. However, to perform the search for the simplifiable pattern, and due to the model representation that the BPMN notation allows, it is not necessary to dive for the various bifurcations that a model can present. In fact, it is just required to check elements one by one, regardless of its position on the model, to evaluate the adjacent elements in order to verify if a simplifiable pattern can be spotted.

Summing up, it can be stated that the search operation has a complexity $O(n)$. The reader can check an example of a search operation on the Listing 2, which shows the Java code responsible for the search of pattern A (lines 3–18). The creation of the search loop (considering each element of Exclusive Gateway type) is shown on line 5. The number of iterations of this loop depends directly on the size of the BPMN model.

5.1. Complexity Regarding the Size of the BPMN Model

As the reader may anticipate, it is quite common to encounter with large, in terms of elements, BPMN models. Therefore, it is convenient to get an idea beforehand of the scalability of the proposal in terms of number of BPMN elements. This is the reason to estimate the complexity of the given algorithm from the point of view of elements within the BPMN model under consideration.

For the patterns loop (Figure 11a, the key is to interpret how the number of iterations varies. This value is very dependent on the BPMN model under study but it is not easy to estimate beforehand the number of possible simplification within the model, i.e., the number iterations. The worst case would be to assume that for a given BPMN a simplification should be made for each element of the model. Thus, the relationship between the number of simplifications and the size of the BPMN is direct and linear. As a consequence, it can be set that the complexity can no longer be worse than $O(n)$. For example, the reader can imagine a BPMN model of size x containing 10 simplifiable patterns, 2 of each of the types contained in the repository. This means that the patterns loop is executed 5 times. If we now imagine a BPMN model of size 10x, it is reasonable to think that it could potentially present 10 times more simplifiable patterns. In this way, the patterns loop would be executed this time 50 times. In this way, we will consider the execution of the patterns loop with a complexity of $O(n)$ with respect to the increase in size of the BPMN model.

In the process of searching and, if the case, replacement of the pairs of patterns, the key lies in the search operations. As mentioned, the replacement of a pattern has a complexity $O(1)$. However, the time spent searching for a pattern depends directly and linearly on the size of the BPMN model. Therefore, this operation has a complexity $O(n)$. Since the repository contains 5 pairs of patterns, the search operation is performed in sequence 5 times. Therefore, according to the rule of the concatenation of actions, the time complexity of the set of searches will be the maximum order of the actions present in the set, that is, $O(n)$.

For the complete algorithm, taking into account the complexity of the loop and the complexity of the search and applying the iteration rule, it follows that the total complexity of the algorithm is $O(n^2)$. Therefore, the execution time grows quadratically with the size of the BPMN model.

5.2. Complexity Regarding the Size of the Repository

Another factor that could affect the complexity of the algorithm is the number of simplifiable patterns in the repository. As it could impact on the feasibility of the model, it was decided to study the complexity of the algorithm considering this variable.

When the number of patterns in the repository increases, the calls to the search and replacement methods will increase for the same patterns loop execution. However, the time taken to execute the individual search method of each of the patterns does not change (since this only depends on the size of the BPMN model). Moreover, increasing the number of patterns does not have to increase the number of iterations of the patterns loop, as it is just linked to the size of the BPMN model itself. Therefore, the actual increase of the execution time by increasing the number of patterns is directly linked with the new pairs of patterns, adding a search operation of $O(n)$ for each pair added. The number of iterations of the loop is not linked with the numbers of pairs of patterns in the repository. For example, the reader can imagine an example in which the patterns loop is executed 5 times. If initially, the repository has 5 patterns, this leads in an execution of 25 search operations. Assuming now that the repository has for example 50 patterns, and that the patterns loop is still running 5 times, this results in 250 search operations. In this way, it follows that the complexity of the algorithm concerning the number of patterns is $O(n)$. Time grows linearly with the number of patterns in the repository.

6. Validation. Testing the Simplification

The authors were prone to the experimental exploration of results to assess the validity of the proposed solution. Therefore, it was decided to run it against a repository of BPMN models. After each execution, a consistent model representing more simply (when possible) the behaviour of the process must be generated. The behaviour described by the new models must be coherent with the original ones, and new behaviours cannot be generated. The objective is to check the correct simplification of each tested model.

6.1. Repository of BPMN Models and Inputs for the Algorithm

The input for the algorithm is composed by a BPMN model and a list of interesting activities. To generate the input sets our starting point is a repository including 63 different models taken from real world examples and collected from different sources:

- 38 models have been translated from the Event-Driven Process Chains (EPC) model collection provided by Laue et al. that was used for the study [40].
- 10 models have been translated to BPMN from different Petri Nets and others 15 models have been discovered from trace logs. The original Petri Nets and the trace logs have been taken from [41] (available online in [42]).

For each model on the original repository, a set of about 60–150 different combination of interesting activities were created. In this way, 8102 different inputs for the second stage of the algorithm were produced.

Executing the algorithm against this combination of up to 8102 different inputs parameters will produce 8102 simplified models in which the reducible patterns identified should have been simplified. To check the correct execution of the algorithm, the simplifications were checked according to the corresponding to the inputs. Two main aspects were analysed in each case:

- If at least one pattern has been identified and simplified, the number of elements in the simplified model must be less than the number of elements in the input model (regarding gateways and sequences).
- No wrong behaviour is introduced in the simplified model. The preserved activities must keep the causal dependency between them. If, for example, activity A is followed by activity B, but A never follows B, then it is assumed that there is a causal dependency between A and B (that must be preserved in the simplified model).

6.2. How to Check the Causal Dependency?

To assure this feature the footprint matrix and fitness concepts from process mining techniques were applied. The footprint matrix represents the causal dependency between the activities in a log or in a model (more about this concept can be learned in [43,44]). This concept is used in some process discovery algorithms, like the α-algorithm [45], helping it to scan the event log for particular patterns. In this case, the footprint matrix will be calculated for a process model, containing the relation between each pair of activities in the workflow. Let M be a process model and a, b two activities that belong to the process model M. It is possible to establish the following relationships:

- $a >_M b$ if only if there is a possible path $p = \langle t_1, t_2, ..., t_n \rangle$ and $i \in 1, ..., n-1$ such that $p \in M$ and $t_i = a$ and $t_{i+1} = b$. This relation is also known as "direct follows" relation.
- $a \rightarrow_M b$ if only if $a >_M b$ and $b_M a$
- $a \#_M b$ if only if $a_M b$ and $b_M a$
- $a ||_M b$ if only if $a >_M b$ and $b >_M a$

The footprint matrix is commonly used in basic conformance checking techniques providing a log-to-model comparison. However, this same approach can be used in log-to-log and model-to-model comparisons [46]. In our case, we will use the model-to-model comparison approach to quantify their similarity. The correct way is to compare the unsimplified model's footprint matrix (end of stage one) with the simplified model's footprint matrix (end of stage two). These two footprints matrix are calculated and compared using the fitness measure for each one of the 8102 inputs. The objective is to check that no wrong behaviour is introduced in the simplified model. The measurement that will be used for the comparison is the fitness. It represents the similarity between two footprints (belonging to two different process models M_1 and M_2) using a 0.0 to 1.0 scale. In our case, a value of 1.0 in all comparisons will be the goal. In Equation (1) the fitness measurement for comparing two footprints (f_{M1} and f_{M2}) is defined [46].

$$fitness = 1 - \frac{\text{different cells between } f_{M1} \text{ and } f_{M2}}{\text{total cells in } f_{M1} \text{ and } f_{M2}} \tag{1}$$

For clarity, it is possible to visualise this procedure with an example. Take the process model from the end of stage one represented in Figure 8 as model M_1. It must be compared with the process model from the end of stage two represented in Figure 10 (M_2). The reader can check the footprint matrix for both models. Table 1 contains the footprint for model M_1 and Table 2 for model M_2.

Table 1. Footprint matrix for model M_1.

	d	**e**	**f**	**g**
d	$\#_{M1}$	$\rightarrow_{M1}$	$\rightarrow_{M1}$	$\#_{M1}$
e	$\leftarrow_{M1}$	$\#_{M1}$	$\#_{M1}$	$\rightarrow_{M1}$
f	$\leftarrow_{M1}$	$\#_{M1}$	$\#_{M1}$	$\rightarrow_{M1}$
g	$\#_{M1}$	$\leftarrow_{M1}$	$\leftarrow_{M1}$	$\#_{M1}$

Table 2. Footprint matrix for model M_2.

	d	**e**	**f**	**g**
d	$\#_{M2}$	$\rightarrow_{M2}$	$\rightarrow_{M2}$	$\#_{M2}$
e	$\leftarrow_{M2}$	$\#_{M2}$	$\#_{M2}$	$\rightarrow_{M2}$
f	$\leftarrow_{M2}$	$\#_{M2}$	$\#_{M2}$	$\rightarrow_{M2}$
g	$\#_{M2}$	$\leftarrow_{M2}$	$\leftarrow_{M2}$	$\#_{M2}$

In this example, both matrices contain the same values, and as a result, the fitness value is 1.0.

6.3. Results

The procedure described is repeated for the 8102 cases in order to carry out the validation. Upon the review of the result, the value of 1.0 in the fitness measurement, the maximum value possible, has been obtained for each input set.

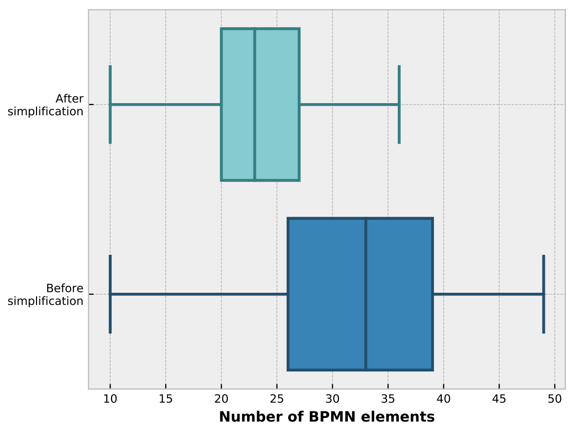
By analysing the simplified models a table with the most significant measures is elaborated. An elements contage is essential to see the effect of the simplification process. The reader can see a fragment of the results on Table 3.

Table 3. Results table excerpt.

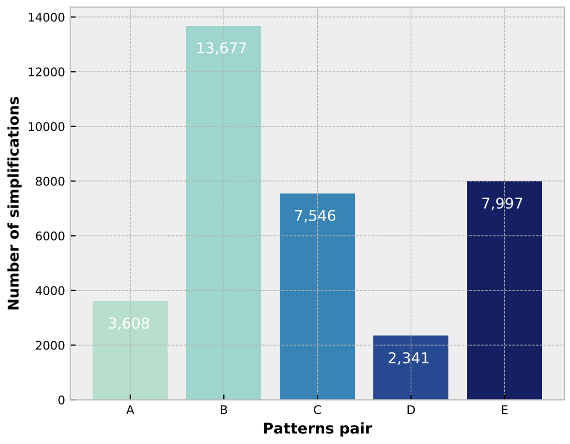
	Original Model				Simplified Model									
ID	**Tasks**	**Gats**	**sFlows**	**Total**	**Tasks**	**Gats**	**sFlows**	**Total**	**A**	**B**	**C**	**D**	**E**	**Fitness**
...	.	.	.	.	.	.	.	.	.	.	.	.	.	.
pm15_073	6	14	28	48	6	6	17	29	0	3	3	0	2	1.0
pm15_074	6	14	28	48	6	4	14	24	0	4	3	0	3	1.0
pm15_075	6	14	28	48	6	5	16	27	0	4	3	0	2	1.0
pm15_076	6	14	28	48	6	5	16	27	0	4	3	0	2	1.0
pm15_077	6	14	28	48	6	5	16	27	0	4	2	0	3	1.0
...	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Each row in the table corresponds with one of the 8102 cases tested. The first column contains the model ID. Columns 2, 3 and 4 contain a contage element for the original model representing the number of gateways, activities and sequence flows. Column 5 represents the total of elements in the original model. Columns 6, 7 and 8 contain a contage element for the simplified model representing the number of gateways, activities and sequence flows. Column 9 contain the total of elements in the simplified model. Columns 10 to 14 represent the number of patterns of each type that has been identified and simplified in each model. Finally, column 15, represents the fitness measure (comparing both footprints of the original and the simplified one).

Throughout the 8102 cases considered, models needing a deeper simplification than others were found. Through the boxplot of Figure 12a, the reader can see the effect and the number of simplifications that have been made on the models of the dataset. In this chart, it is shown the effect of the second stage on the algorithm. As the reader can note, there is a clear trend towards obtaining models with fewer elements after the patterns simplification phase. The median with respect to the number of elements in a model has decreased from 33 (before the simplification process) to 23.



(**a**) BPMN elements simplified



(**b**) Patterns pairs simplified

Figure 12. Simplification results from the 8102 cases.

It is also interesting to analyse the times that each one of the patterns present in the repository has been found and simplified. In total, for the 8102 cases tested, 35,169 simplifications were conducted. On Figure 12b, the reader can verify the distribution of these simplifications according to each pattern. The repetition of the pattern B is evident, doubling the occurrences to the next most popular pattern, the E.

The complete table with the raw information for the 8102 tested cases is available for the reader in an online data repository [47] at the Open Science Framework platform. This repository contains also the 62 original BPMN models, the 8102 BPMN models obtained after stage one and the 8012 BPMN simplified models.

7. A Proof of Concept on a Real Scenario

An additional evaluation has been carried out to verify the usability of the proposal in a real scenario. The authors have been working during recent years in several projects dedicated to issues related to traceability in a broad sense. In the due course of this researching activities, the authors have tackled the problems derived from hazardous medicines in the health field [48]. This leads us to propose extensions to the BPMN notation for its application in Hazard Analysis and Critical Control Points (HACCP) [26] scenarios or to the identification of gaps and needs for standardization of hospital protocols of a dangerous nature [49], among others. Given this trajectory, the proof of concept for this proposal was focused on the field of the management of hazardous drugs (HDs). HD are those drugs whose handling poses a danger to people (carcinogenicity, teratogenicity, reproductive toxicity, low dose organ toxicity, genotoxicity, among others) [50]. In [49], the authors focus their efforts on the identification of the stages that make up the life cycle of the HDs and, for each of them, certain action protocols are defined. Subsequently, in [51], an assessment is made of the advantages of applying the BPMN extension proposed in [26] for this domain of interest. In this work, 47 experts in the domain evaluated the benefits of the proposal. An adaptation of a BPMN model presented in this last work was selected to check the usability of the simplification proposal. The chosen BPMN model can be seen in Figure 13.

This model summarizes the tasks carried out for the administration of an HD. During this stage, two roles are involved: Nurse (N) and Auxiliary Nurse (AN). There are also four possible ways to administer the HD and they are shown in Table 4.

By collecting the opinions of sanitary experts at different levels about the processes related to the management of dangerous medicines, some existing limitations arose. The models currently in use tend to include all the actions carried out by different health profiles. In this way, it is possible, in a single diagram, to represent several perspectives. However, this introduces a drawback: each of the individual roles involved in the process may have too much information (some of the data can be considered as noise). In this way, the interpretation of the model is sometimes more complicated than necessary and could lead to errors. For example, a person with the AN role may not be interested in knowing the sequence of tasks that another person with the N role must perform. Therefore, it is preferable for the person with the AN role to have a model simplified that represents only those tasks required for his/her role. Following, two particular scenarios that show this idea are presented.

Table 4. Types of HD administration.

	Administration Method	Key on the Diagram
Administration 1	intravenous	TI
Administration 2	subcutaneous	TII
Administration 3	ophthalmic	TIII
Administration 4	urinary catheter	TIV

As a first example, let's assume that a person with the AN role wishes to visualise only the activities involving his/her participation. Besides, he/she is only interested in those tasks involving ophthalmic administration, as this person only is capable of performing this procedure. In this line, the goal of simplification would be to produce a new simpler model that contains only the relevant information for this context. Applying the proposed two-phase algorithm, it would be possible to obtain the BPMN models shown in Figure 14a,b (output of stages 1 and 2, respectively). The model in Figure 14b represents in a much more synthetic way the relevant information for the AN role.

For the second example, let's consider the case of a person also involved in the administration of the HD under the role N that wants to check the tasks that must perform. In this case, the list of activities removed in the first stage is different from the first example. The new resulting models generated for this new context are shown in Figure 15a,b.

As shown, in both cases, the simplification procedure strives to offer a simpler model. This new model allows analysing the desired information with a representation that contains fewer elements resulting in a clearer model for the user with a particular rol. The difference between the models represented in Figure 14a,b (the same for Figure 15a,b) is evident. Even both represent the same behaviour but, in the cases of Figures 14b and 15b, the model is simpler and faster to interpret by the human user.

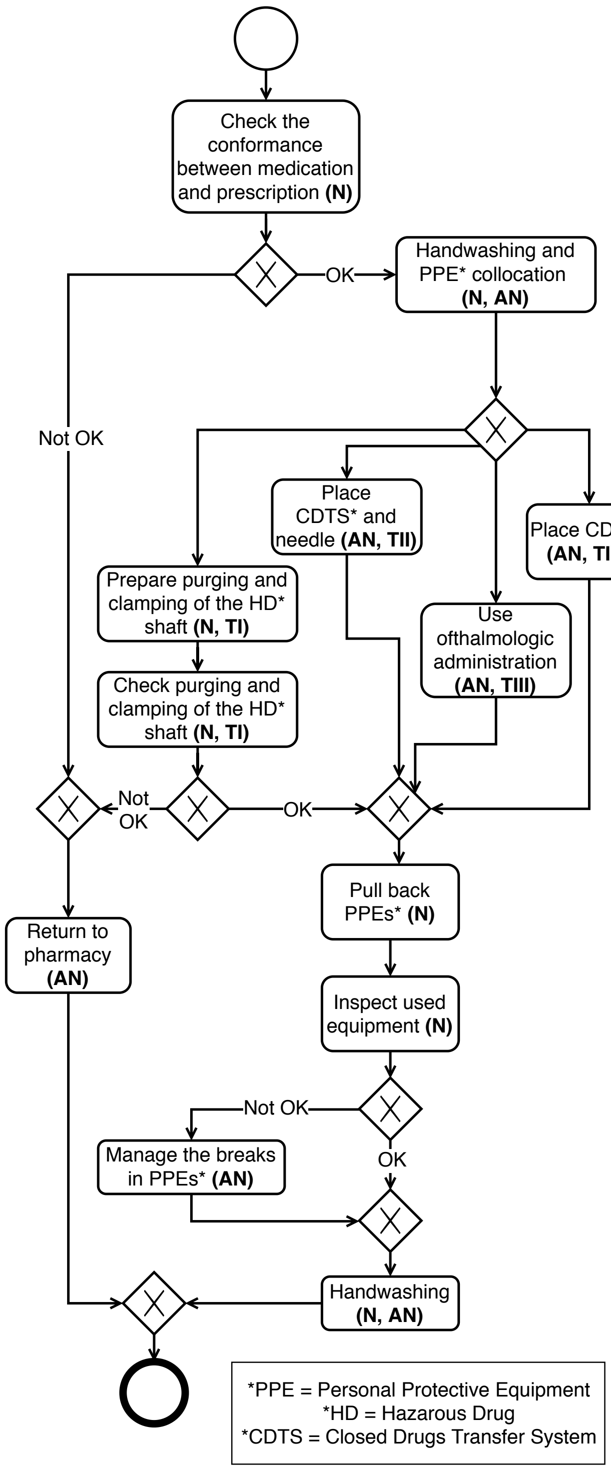
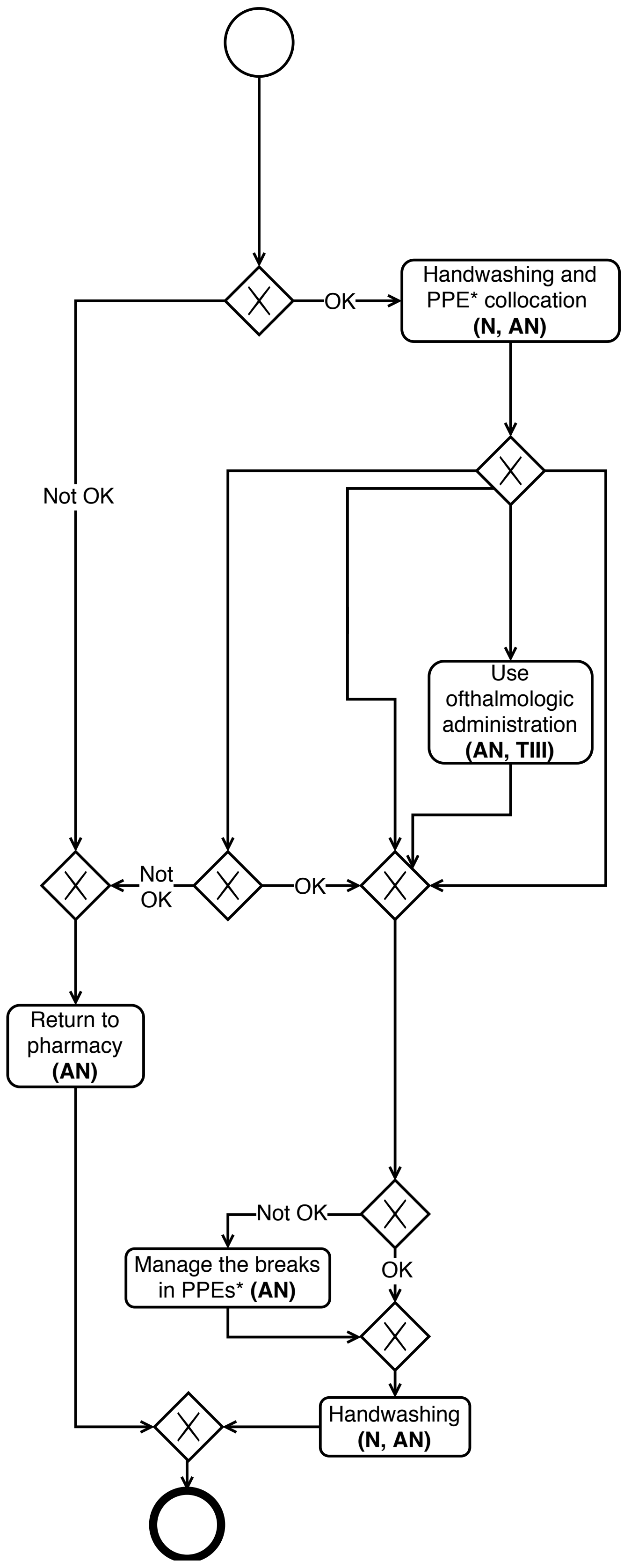
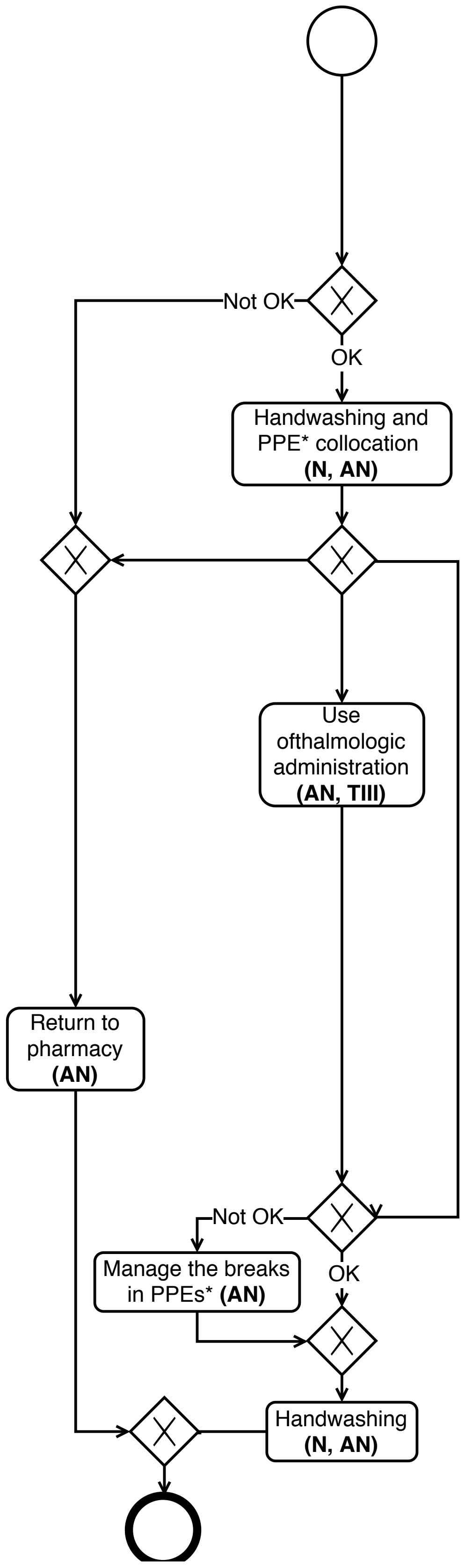

Check the conformance between medication and prescription (N)
OK
Handwashing and PPE* collocation (N, AN)
Not OK
Place CDTS* and needle (AN, TII)
Place CDTS* (AN, TIV)
Prepare purging and clamping of the HD* shaft (N, TI)
Use ofthalmologic administration (AN, TIII)
Check purging and clamping of the HD* shaft (N, TI)
Not OK
OK
Pull back PPEs* (N)
Return to pharmacy (AN)
Inspect used equipment (N)
Not OK
Manage the breaks in PPEs* (AN)
OK
Handwashing (N, AN)
*PPE = Personal Protective Equipment
*HD = Hazarous Drug
*CDTS = Closed Drugs Transfer System

Figure 13. BPMN model for the HD adminsitration scenario.

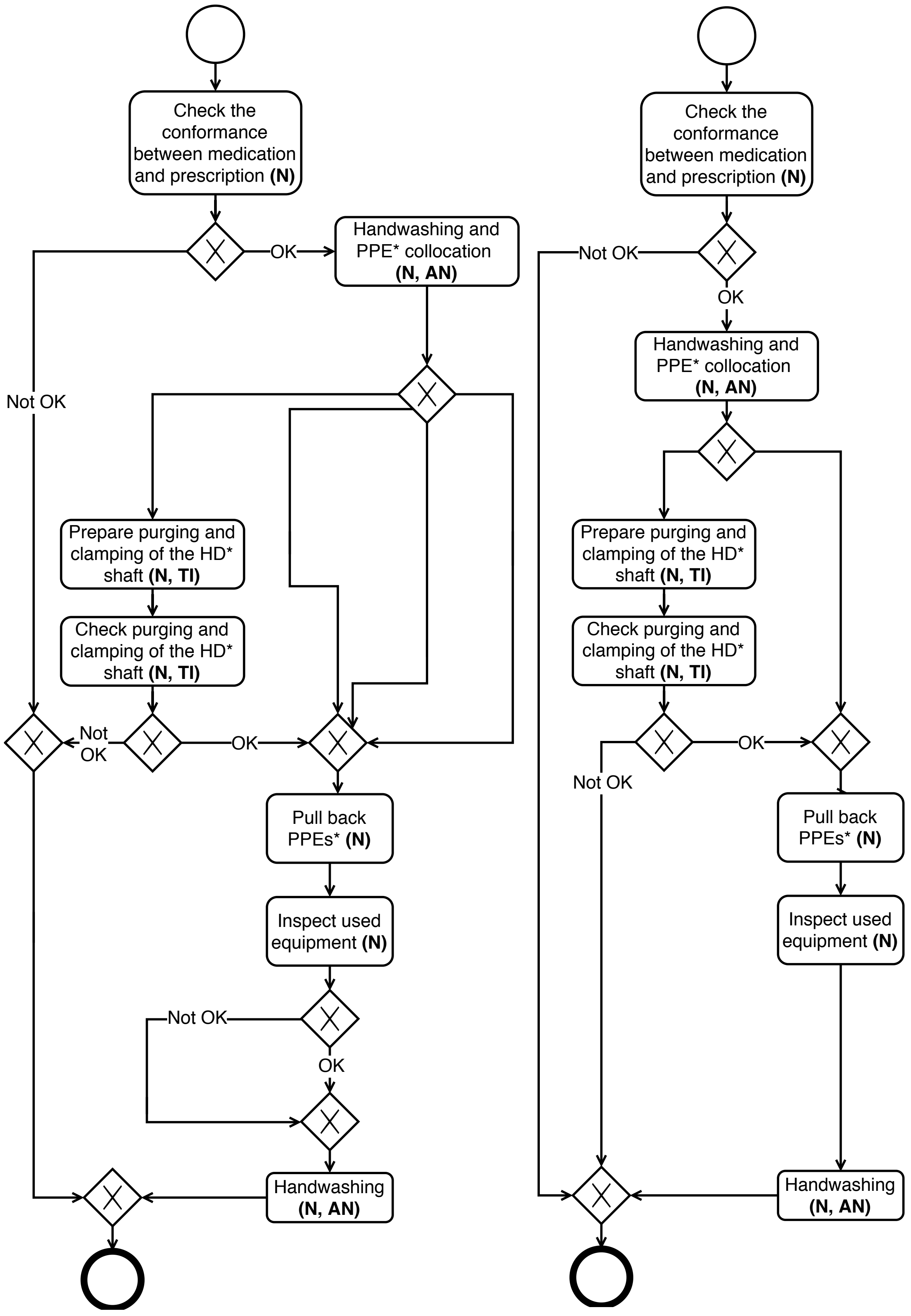

OK
Handwashing and PPE* collocation **(N, AN)**
Not OK
Use ofthalmologic administration **(AN, TIII)**
Not OK
OK
Return to pharmacy **(AN)**
Not OK
OK
Manage the breaks in PPEs* **(AN)**
Handwashing **(N, AN)**

(**a**) BPMN after first stage


Not OK
OK
Handwashing and PPE* collocation **(N, AN)**
Use ofthalmologic administration **(AN, TIII)**
Return to pharmacy **(AN)**
Not OK
OK
Manage the breaks in PPEs* **(AN)**
Handwashing **(N, AN)**

(**b**) BPMN after second stage

Figure 14. BPMN models for the AN role example.


Check the conformance between medication and prescription (N)
OK
Not OK
Handwashing and PPE* collocation (N, AN)
Prepare purging and clamping of the HD* shaft (N, TI)
Check purging and clamping of the HD* shaft (N, TI)
Not OK
OK
Pull back PPEs* (N)
Inspect used equipment (N)
Not OK
OK
Handwashing (N, AN)
Check the conformance between medication and prescription (N)
Not OK
OK
Handwashing and PPE* collocation (N, AN)
Prepare purging and clamping of the HD* shaft (N, TI)
Check purging and clamping of the HD* shaft (N, TI)
OK
Not OK
Pull back PPEs* (N)
Inspect used equipment (N)
Handwashing (N, AN)

(**a**) BPMN after first stage (**b**) BPMN after second stage

Figure 15. BPMN models for the N role example.

In addition to facilitating the visual reading, the subsequent application of business process techniques is improved (e.g., analysis of most used routes, failure analysis or conformance checking analysis, among others). Many times, the application of these techniques produces results with a difficult interpretation since it is necessary to be familiar with this type of techniques to make the most of the results obtained. In this line, the cleaner the BPMN model, the easier the interpretation and the less error will arise by humans.

8. Conclusions

Presently, the usage of BPMN models for the characterisation and the analysis of processes is common in the academic and industrial environment. From its usage, many advantages arise such as better analysis of costs, more in-depth tracking of raw material and final products, more complete documentation for auditing processes, etc. On this daily usage, we may face the need to perform transformations (aggregation or removal) on steps, processes, or tasks.

To be able to apply in an optimum manner the already existing techniques in the domain of Data Analytics and Process Mining to conduct analysis and studies, it is desirable to work on a minimum but equivalent BPMN model. It is at this point where this proposal comes into the scene. It is possible to perform transformations to achieve a simplified model by taking advantage of the proposed scheme. Even if it is not possible to guarantee the attainment of the optimal solution; we can get, in an entirely automatic fashion, an optimised version of the original model ensuring non-wrong solutions.

This proposal was successfully used in support of actual projects such as [48]. In due course of the mentioned project, it turned out to be necessary to be able to generate a simplified model with a subset of the activities. In this project, protocols are defined following the Hazard Analysis and Critical Control Points (HACCP) [52] system. In HACCP environments it is common to use a subset of the BPMN language (this has been taken into account in the approach of the proposal). These protocols are documented graphically using an extension of BPMN, BPMNE2 [26] that is oriented to identify certain critical aspects related to the security of the process instances. The monitoring of them is carried out using the ToCP platform [53,54]. To carry out the audit, Process Mining techniques are used, mainly conformance checking, on the event logs generated in the monitoring points. Before carrying out these analyses, it is convenient to eliminate activities that do not generate event logs from the BMPNE2 models, obtaining a simplified graph. Therefore, it is possible to use tools such as ProM [55] immediately and to reduce the analysis times while improving the graphic understanding for humans.

As already discussed in the review of related works, there exist some similarities between the work presented in this paper and other approaches. Nevertheless, to the best knowledge of the authors, no previous works describe an approach oriented to these contexts with enough detail level to guide further attempts on the field.

With the heuristic algorithm designed, it is achieved a generic, open and flexible approach to perform the simplification of BPMN models. A relevant feature is the generation of a repository for patterns. This element, a key element of the implementation, was designed in an extensible way. Therefore, it is easy to populate the repository with new patterns identified for improving the simplification process.

Despite the utility of the proposal, certain aspects could be improved in the future. A pending issue for future enhancement is linked to the fact that patterns in the repository have to be defined manually by experts in the domain. This same fact causes that it cannot be assured formally that all the possible patterns have been found in such a way that achieves the simplest possible final diagram. Having to implement manually (currently using Java) the patterns themselves and their simplified versions is a difficulty for its broad adoption. As a future line, the creation of an automated mechanism that performs this task could be tackled. Nevertheless, it seems to the authors quite unlikely that to total amount of patterns should be large increased.

To fulfil this goal, the use of a formal and machine-understandable language to define the pairs of patterns would be useful. Thus, in a first stage, this language would be used to represent the pairs of patterns found. Subsequently, an automated software mechanism would translate these patterns into a representation that could be used for the simplification algorithm. In this line it is interesting to analyse the research themes of BP patterns [25], refactoring [27], anti-patterns [28], BPMV [18] or BPMA [11].

It should be noted that only Process-type BPMN diagrams have been taken into account, and, in particular, the focus has been set on Exclusive Gateway type elements. An immediate future line would be to expand this study to include more elements available in the BPMN specification of Process type as well as collaboration diagrams and choreographies. However, the analysis carried out in this work and using the repository model for patterns and the algorithm described can be considered a good starting point to fulfil this broader goal taking into account the popularity of the Exclusive Gateways in the real world examples (according to the experience of the authors about the 90% of the used gateways are exclusive). This line of future development would need to straightforwardly define new pairs of simplifiable patterns and add them to the repository.

The authors are confident that they have developed a solution that may support further research in other domain taking advantage of this work. Also, efforts are currently being devoted to the provision of an open and interoperable repository to store and analyse patterns eligible for these optimisation processes that may benefit the entire community.

Author Contributions: M.R.-M., L.M.Á.-S. and J.M.S.-G. have performed conceptualization, methodology, investigation and funding acquisition. M.R.-M. and F.d.A.-P. have written the original draft and the software. All the authors have contributed to data curation, validation, formal analysis and writing—review the final draft.

Funding: This research was partially funded by XUNTA DE GALICIA under the grant number ED481A-2017/323 (Axudas predoutorais). The APC was funded by the University of Vigo.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Object Management Group (OMG). *Business Process Model And Notation (BPMN)*; Version 2.0.2; OMG Document Number formal/13-12-09 2014. Available online: https://www.omg.org/spec/BPMN/2.0.2/PDF (accessed on 4 June 2019).
2. Van Der Aalst, W.; Adriansyah, A.; De Medeiros, A.K.A.; Arcieri, F.; Baier, T.; Blickle, T.; Bose, J.C.; van den Brand, P.; Brandtjen, R.; Buijs, J.; et al. Process mining manifesto. In *International Conference on Business Process Management*; Springer: New York, NY, USA, 2011; pp. 169–194.
3. Telang, P.R.; Kalia, A.K.; Singh, M.P. Modeling healthcare processes using commitments: An empirical evaluation. *PLoS ONE* **2015**, *10*, e0141202. [CrossRef] [PubMed]
4. Pillat, R.M.; Oliveira, T.C.; Alencar, P.S.; Cowan, D.D. BPMNt: A BPMN extension for specifying software process tailoring. *Inf. Softw. Technol.* **2015**, *57*, 95–115. [CrossRef]
5. Vaziri, R.; Mohsenzadeh, M.; Habibi, J. TBDQ: A Pragmatic Task-Based Method to Data Quality Assessment and Improvement. *PLoS ONE* **2016**, *11*, e0154508. [CrossRef] [PubMed]
6. Meyer, S.; Ruppen, A.; Hilty, L. The Things of the Internet of Things in BPMN. In *International Conference on Advanced Information Systems Engineering*; Springer: New York, NY, USA, 2015; pp. 285–297.
7. Mendling, J.; Reijers, H.A.; van der Aalst, W.M. Seven process modeling guidelines (7PMG). *Inf. Softw. Technol.* **2010**, *52*, 127–136. [CrossRef]
8. Smirnov, S.; Reijers, H.A.; Weske, M.; Nugteren, T. Business process model abstraction: A definition, catalog, and survey. *Distrib. Parallel Databases* **2012**, *30*, 63–99. [CrossRef]
9. Bobrik, R.; Reichert, M.; Bauer, T. View-based process visualization. In *Business Process Management*; Springer: New York, NY, USA, 2007; pp. 88–95.
10. Eshuis, R.; Grefen, P. Constructing customized process views. *Data Knowl. Eng.* **2008**, *64*, 419–438. [CrossRef]
11. Tsagkani, C.; Tsalgatidou, A. Abstracting BPMN models. In Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece, 1–3 October 2015; pp. 243–244.

12. McMaster, R.B.; Shea, K.S. *Generalization in Digital Cartography*; Association of American Geographers: Washington, DC, USA, 1992.
13. Polyvyanyy, A.; Smirnov, S.; Weske, M. Reducing Complexity of Large EPCs. In Proceedings of the Lecture Notes in Informatics: Modellierung betrieblicher Informationssysteme, Gesellschaft für Informatik, Bonn, Saarbrücken, Germany, 27–28 November 2008; pp. 195–207.
14. Polyvyanyy, A.; Smirnov, S.; Weske, M. On Application of Structural Decomposition for Process Model Abstraction. In Proceedings of the Business Process, Services Computing and Intelligent Service Management, Leipzig, Germany, 23–25 March 2009; pp. 110–122.
15. Bobrik, R.; Reichert, M.; Bauer, T. *Parameterizable Views for Process Visualization*; University of Twente: Ernscott, The Netherlands, 2007.
16. Van der Aalst, W.M. Formalization and verification of event-driven process chains. *Inf. Softw. Technol.* **1999**, *41*, 639–650. [CrossRef]
17. Sadiq, W.; Orlowska, M.E. Analyzing process models using graph reduction techniques. *Inf. Syst.* **2000**, *25*, 117–134. [CrossRef]
18. Rosa, M.L.; Van Der Aalst, W.M.; Dumas, M.; Milani, F.P. Business process variability modeling: A survey. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 2. [CrossRef]
19. Hallerbach, A.; Bauer, T.; Reichert, M. Managing Process Variants in the Process Lifecycle. In Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS'08), Barcelona, Spain, 12–16 June 2008 .
20. Hallerbach, A.; Bauer, T.; Reichert, M. Guaranteeing soundness of configurable process variants in Provop. In Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing, Vienna, Austria, 20–23 July 2009; pp. 98–105.
21. Hallerbach, A.; Bauer, T.; Reichert, M. Issues in modeling process variants with provop. In *International Conference on Business Process Management*; Springer: New York, NY, USA, 2008; pp. 56–67.
22. Hallerbach, A.; Bauer, T.; Reichert, M. Capturing variability in business process models: The Provop approach. *J. Softw. Maint. Evolut. Res. Pract.* **2010**, *22*, 519–546. [CrossRef]
23. Kumar, A.; Yao, W. Process materialization using templates and rules to design flexible process models. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*; Springer: New York, NY, USA, 2009; pp. 122–136.
24. Kumar, A.; Yao, W. Design and management of flexible process variants using templates and rules. *Comput. Ind.* **2012**, *63*, 112–130. [CrossRef]
25. Russell, N.; van der Aalst, W.M.; ter Hofstede, A.H. *Workflow Patterns: The Definitive Guide*; MIT Press: Cambridge, MA, USA , 2016.
26. Ramos-Merino, M.; Santos-Gago, J.M.; Álvarez-Sabucedo, L.M.; Alonso-Roris, V.M.; Sanz-Valero, J. BPMN-E2: A BPMN extension for an enhanced workflow description. *Softw. Syst. Model.* **2018**. [CrossRef]
27. Mkaouer, M.W.; Kessentini, M.; Bechikh, S.; Cinnéide, M.Ó.; Deb, K. On the use of many quality attributes for software refactoring: A many-objective search-based software engineering approach. *Empir. Softw. Eng.* **2016**, *21*, 2503–2545. [CrossRef]
28. Morales, R.; Soh, Z.; Khomh, F.; Antoniol, G.; Chicano, F. On the use of developers' context for automatic refactoring of software anti-patterns. *J. Syst. Softw.* **2017**, *128*, 236–251. [CrossRef]
29. Mansoor, U.; Kessentini, M.; Wimmer, M.; Deb, K. Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm. *Softw. Qual. J.* **2017**, *25*, 473–501. [CrossRef]
30. Jensen, A.C.; Cheng, B.H. On the use of genetic programming for automated refactoring and the introduction of design patterns. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, OR, USA, 7–11 July 2010; pp. 1341–1348.
31. Khan, Y.A.; El-Attar, M. Using model transformation to refactor use case models based on antipatterns. *Inf. Syst. Front.* **2016**, *18*, 171–204. [CrossRef]
32. Küster, J.; Völzer, H.; Favre, C.; Branco, M.C.; Czarnecki, K. Supporting different process views through a shared process model. *Softw. Syst. Model.* **2016**, *15*, 1207–1233. [CrossRef]

33. Sahraoui, H.; Syriani, E. Recommending Model Refactoring Rules from Refactoring Examples. In Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Copenhagen, Denmark, 14–19 October 2018; pp. 257–266.
34. Bodhuin, T.; Canfora, G.; Troiano, L. SORMASA: A tool for Suggesting Model Refactoring Actions by Metrics-led Genetic Algorithm. In Proceedings of the 1st Workshop on Refactoring Tools, WRT, Berlin, Germany, 30 July–3 August 2007; pp. 23–24.
35. Camunda. BPMN Examples–Best Practices for Creating BPMN 2.0 Process Diagrams. 2018. Available online: https://camunda.com/bpmn/examples/ (accessed on 5 May 2019).
36. Brosch, P.; Langer, P.; Seidl, M.; Wimmer, M. Towards end-user adaptable model versioning: The by-example operation recorder. In Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, Vancouver, BC, USA, 17 May 2009; pp. 55–60.
37. Sun, Y.; Gray, J.; White, J. MT-Scribe: An end-user approach to automate software model evolution. In Proceedings of the 2011 33rd International Conference on Software Engineering (ICSE), Honolulu, HI, USA, 21–28 May 2011; pp. 980–982.
38. Mariani, T.; Vergilio, S.R. A systematic review on search-based refactoring. *Inf. Softw. Technol.* **2017**, *83*, 14–34. [CrossRef]
39. Alotaibi, Y. Business process modelling challenges and solutions: A literature review. *J. Intell. Manuf.* **2016**, *27*, 701–723. [CrossRef]
40. Gruhn, V.; Laue, R. A heuristic method for detecting problems in business process models. *Bus. Process Manag. J.* **2010**, *16*, 806–821. [CrossRef]
41. Vázquez-Barreiros, B.; Mucientes, M.; Lama, M. ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm. *Inf. Sci.* **2015**, *294*, 315–333. [CrossRef]
42. Vázquez-Barreiros, B.; Mucientes, M.; Lam, M. ProDiGen Software. From Log to Petri Net. 2015. Available online: http://tec.citius.usc.es/processmining/prodigen (accessed on 2 February 2019).
43. Van Der Aalst, W. Process mining: Overview and opportunities. *ACM Trans. Manag. Inf. Syst.* **2012**, *3*, 7. [CrossRef]
44. Bose, R.; Van Der Aalst, W.M.; Zliobaite, I.; Pechenizkiy, M. Dealing with concept drifts in process mining. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 154–171. [CrossRef] [PubMed]
45. Van der Aalst, W.; Weijters, T.; Maruster, L. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1128–1142. [CrossRef]
46. Van Der Aalst, W.M. *Process Mining: Data Science in Action*; Springer: New York, NY, USA, 2016.
47. Ramos-Merino, M.; Santos-Gago, J.; Álvarez-Sabucedo, L.; de Arriba-Pérez, F. Simplification of BPMN Process Models. Data Repository. 2019. Available online: https://osf.io/94scm/?view_only=4cebebc2e80c43fe94bb18bf16a3b95b (accessed on 17 April 2019).
48. Sanz-Valero, J.; Sabucedo, L.Á.; Wanden-Berghe, C.; Roris, V.A.; Gago, J.S. Sun-pp236: Deployment of a tag-based system to ensure traceability management of parenteral nutrient mixtures. *Clin. Nutr.* **2015**, *34*, S111. [CrossRef]
49. Bernabeu-Martínez, M.A.; Merino, M.R.; Gago, J.M.S.; Sabucedo, L.M.Á.; Wanden-Berghe, C.; Sanz-Valero, J. Guidelines for safe handling of hazardous drugs: A systematic review. *PLoS ONE* **2018**, *13*, e0197172. [CrossRef] [PubMed]
50. Connor, T.H.; McDiarmid, M.A. Preventing occupational exposures to antineoplastic drugs in health care settings. *CA Cancer J. Clin.* **2006**, *56*, 354–365. [CrossRef] [PubMed]
51. Ramos-Merino, M.; Álvarez-Sabucedo, L.M.; Santos-Gago, J.M.; Sanz-Valero, J. A BPMN Based Notation for the Representation of Workflows in Hospital Protocols. *J. Med. Syst.* **2018**, *42*, 181. [CrossRef] [PubMed]
52. Khandke, S.; Mayes, T. HACCP implementation: A practical guide to the implementation of the HACCP plan. *Food Control* **1998**, *9*, 103–109. [CrossRef]
53. Rorís, V.M.A.; Sabucedo, L.M.Á.; Wanden-Berghe, C.; Gago, J.M.S.; Sanz-Valero, J. Towards a mobile-based platform for traceability control and hazard analysis in the context of parenteral nutrition: Description of a framework and a prototype app. *JMIR Res. Protoc.* **2016**, *5*, e57. [CrossRef]

54. Alonso-Rorís, V.M.; Álvarez-Sabucedo, L.; Santos-Gago, J.M.; Ramos-Merino, M. Towards a cost-effective and reusable traceability system. A semantic approach. *Comput. Ind.* **2016**, *83*, 1–11. [CrossRef]
55. Verbeek, H.; Buijs, J.; Van Dongen, B.; van der Aalst, W.M. Prom 6: The process mining toolkit. *Proc. BPM Demonstr. Track* **2010**, *615*, 34–39.