Cased Based Reasoning in Business Process Management Design

Philipp Tueschen & Vítor Duarte dos Santos

Information Management Systems School, Universidade Nova de Lisboa, **Lisbon,** Portugal

This is the Author Peer Reviewed version of the following conference paper published by Springer:

Tueschen, P., & dos Santos, V. D. (2021). Cased Based Reasoning in Business Process Management Design. In R. Silhavy (Ed.), Artificial Intelligence in Intelligent Systems - Proceedings of 10th Computer Science On-line Conference, 2021 (Vol. 2, pp. 722-741). (Lecture Notes in Networks and Systems; Vol. 229). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-030-77445-5_65

Cased Based Reasoning in Business Process Management Design

Philipp Tueschen and Vítor Duarte dos Santos

Universidade Nova de Lisboa - Information Management Systems School, Lisbon, Portugal

Abstract. Case-based reasoning (CBR), another form of artificial intelligence, stores and retrieves past cases that can be adapted to find a solution to a current problem. The new solution can then be retained and made available to solve other future problems. Business Process Management analyzes and optimizes business processes to make them more effective and efficient for an organization's strategy to ultimately increasing shareholder value. CBR can help to support BPM, making better decisions with existing knowledge when solving process problems. This study investigates effectively store, retrieve, and adapt Business Process Management Notation (BPMN) solutions that best fit the underlying BPM problem using CBR as a tool. Therefore, a theoretical model was proposed, containing each CBR live cycle phase with different possible tools applied to BPMN diagrams, which was validated by expert interviews. This study concludes that a whole CBR life cycle can be applied to BPMN diagram problems with the need for human intervention. The objective was not to solve the whole problem but to contribute to a possible solution by using CBR through a theoretical model.

Keywords: Artificial Intelligence, Business Process Management, Case-Based Reasoning, Graph-Edit Distance, Learning by Analogy, XPDL

1 Introduction

Since the 90s artificial intelligence (AI) is becoming more and more useful; automating more and more tasks makes human life easy by copying the human way of learning, thinking, and reasoning. AI can conduct more and more complicated tasks, such as autonomous driving. This imitating of the human brain is accomplished based on the three basic concepts within the AI area: machine learning, deep learning, and neural networks [16].

Case-based reasoning (CBR) is part of the AI area as CBR learns by storing initial knowledge and retrieving it when needed. Further, CBR reuses this knowledge as a possible solution, which is then revised to solve a current problem. In the end, a successful solution to a problem is retained in the case database adding new knowledge to the CBR system, which is considered "learning"[15].

Business process management (BPM) problems rely on expert's knowledge to resolve them. As BPM focuses on finding improvement opportunities and

managing processes' consistency, experts rely on their past experiences to tackle the present problems.

CBR is imitating the expert's knowledge acquisition by storing knowledge from the past and using applicable knowledge to solve a present problem, so it is a good system to manage BPM problems more efficiently. In fact, Pichler [18] found that using CBR implemented in a BPM system allows them to execute processes that have clearly defined and standardized process models and react to events outside the limitations. Pichler [18] used Petri nets as a similarity function within the CBR system to find the most suitable cases to the underlying problem. However, this report will work with Carbonell's [6] learning by analogy, which compares semantics and considers the decisions' justifications of positive and negative outcomes. Further, Carbonell's "learning by analogyïs incorporated into the CBR system, supporting the retrieval and adaptation process.

Although Pichler [18] establishes that CBR can help to find solutions for BPM problems by using a similarity function establishing the distance between the attributes of the business process management notation (BPMN) diagrams, the article does not talk about the storing and retrieving of BPMN diagrams, which is an essential part of CBR in providing the best fit solution.

Existing literature established the possibility of translating BPMN models into XPDL language and that CBR can be an effective method for improving BPMN solutions [22][18]. However, there is a gap between how CBR could be used in comparing BPMN models and how CBR used with BPMN could be useful with different models then proposed by Pichler [18]. Hence, this thesis aims to find a solution to how BPMN can be stored effectively in XDPL and how the different stored XPDL files can be analyzed and retrieved effectively. This thesis also considers the BPMN adaptation, providing the best fit solution the user is looking for. This thesis's outcome is abstract knowledge; however, it could contribute to a more physical solution in the future [10].

2 Literature Review

Case-Based Reasoning CBR uses and adapts solutions that have been successful in the past to solve new problems. Case histories are collected for CBR, and their most important features are described. New solutions of cases are added as new knowledge to the CBR databases. Thus, as databases enable the management of larger volumes of cases, it makes the need for explicit models obsolete [15]. All in all, CBR uses successful past cases to suggest a solution to a current, new to the system, and similar problem. Kolodner[13] came up with four assumptions. After the current problem case could be described with its features, the most similar case stored can be allocated. Although this stored solution may not directly fit the current problem case's differences, it can be adjusted. The derived and authenticated new solution to the current problem can then be stored as a new case solution to solve a similar problem in the future [15]. This process builds the ground for the CBR working Cycle, that according to Aamodt and Plaza[2], is defined as the four Res:

1. Retrieve: After a current problem has been defined the most applicable solution is searched for and obtained from the case database
2. Reuse: The solutions obtained and used to solve a current problem
3. Revise: The obtained solution has to be adapted to the current problem to build a possible solution. If the solution is undesirable further adopting is required.
4. Retain: If the solution is desirable and verified as such it can be added as a new case to the database

Derivational Analogy Solving a problem with derivational analogy means to search for related problems in the database and adapt their solutions to be possibly relevant to the current problem, which is why Carbonell[6, p.3] defines it as the following: "Analogical problem solving consist of transferring knowledge from the past problem-solving episodes to new problems that share significant aspects with the corresponding past experience - and using the transferred knowledge to construct solutions to the new problems." Carbonell points out that it is not enough to find a solution similar to the current problem but rather about the knowledge, it contains, which can be retrieved and interpreted. It implies the necessity for a very detailed adaptation model to obtain a desired adapted solution from the past. As this way of problem-solving is a fundamental part of human cognition, the derivational analogy is undoubtedly part of AI [6].

Building a model to adapt past case problems to the current situation requires specificity. The model needs to know what it means for both cases to have meaningful aspects in common. Further, what knowledge from these aspects is transmitted from the past solution to the current problem and how it occurs. Finally, it is important to know how these analogically associated aspects are selected from the case base [6].

Transformational Analogy Carbonell[1983, as cited in 14] depicts transformational analogy as taking part in the CBR adaptation where a solution and its sequence of actions from a previous problem is being modified into the solution of the new problem. The modification includes removing, adding, and/ or changing the sequence's actions. Hence, after the closest solution has been found, the derivational and transformational analogy can be used as a CBR strategy to form a reasoning process that helps adapt the old solution to the current problem [14].

Business Process Management

BPM analyses operational activities performed in organizations and identifies advantages of improving processes, guaranteeing their consistency. Thus, BPM helps operating processes faster, more accurately, at lower costs, and with reduced assets while increasing their flexibility. Focusing on End-to-End processes across organizational boundaries can reduce nonvalue-adding tasks, which exist due to departmental boundaries. Further, processes are continuously monitored

and improved if they no longer meet the customer's needs [12]. There are several methodologies with tools and techniques that help to identify redundant, inefficient, and ineffective processes or highlight specific processes for improvement [20]. Identifying business processes, their discovery , analysis, redesign, implementation, controlling and monitoring, is called the BPM Life Cycle [9].

The BPM as a management discipline can be divided into two approaches. Process improvement aims to analyze existing processes and consciously improve them while process reengineering questions existing processes from end-to-end and redesigns them from scratch. Therefore, BPM does not create business value by merely using IT or information systems to automate processes, but BPM uses IT and IS to enable process change, creating business value [9][12].

Business Process Management Notation 2.0 After processes are identified and discovered, they can be modeled and represented as a BPMN 2.0, which is a graphical notation in the form of a diagram. It provides an intuitive notation used by both technical and business users to represent complicated end-to-end process semantics, especially because many tools support this notation. It can also be identified as the most popular language for BPM [1]. Further, BPMN 2.0's Business Process Diagram (BPD) is the graphical notation and can be divided into groups, pools, and lanes. Pools can represent a business entity or a business role, while a lane is a subpartition of a pool representing specific business roles [17]. Within these divisions, BPMN uses four different essential elements. Flow objects consist of events, activities, and gateways. These are connected with Connecting Objects, divided by a Swimlane, and enriched with information through Artefacts. Thereby, actives are the work that is being performed, which can be further broken down into processes, subprocesses, and tasks. Events happen without any action and can be start events, intermediate events, or end events.

From BPMN to XPDL XPDL is an XML-Process Definition language being used by a variety of workflow applications. Since BPMN is only a diagram, in order to be executable, it has to be translated into XPDL [22]. Both BPMN and XPDL are flow-chart structures enabling a direct, simple, and complete translation from BPMN into the semantically identical and compatible XPDL. According to the WfMC[21], XPDL writes out the BPMN diagram providing a file format. It enables other software to read and recreate the same process.

As many elements serve as graphical representations, once BPMN has been translated into XPDL, the elements' order changes. Pool and Lane names graphically describe what each participant does in the form of containers. Thus, they hold information about activities, events, and gateways that are executed by the participant. However, these containers are not present in the XPDL format and cannot hold the participant information. Instead, the task level within the activity is the only entity that can hold the participant information. To associate the Pool or Lane with a task, XPDL utilizes the elements <participant>

and <performer>, which carry the actor executing the activity. Therefore, each <Task> holds a hidden <performer> [7].

Graph-Edit Distance A distance measure has to be defined to calculate the graph similarity of two labeled graphs. Graph-Edit Distance (GED), first reported by Sanfeliu et al. [19], is a widely accepted and used distance measure for labeled graphs due to its flexibility and sensitivity. The definition of GED is the minimum cost of an edit path between two graphs. The edit path is a sequence of edit operations that change a graph to an isomorphic graph of B by adding, deleting, and substituting a node or an edge. These three operations are linked to a non-negative cost, which as a sum defines the cost of an edit path. Further, the edit costs transfer the metric properties to GED [4]. However, computing GED is complicated since it is NP-hard also if edit costs are uniform. Just using GED in matching algorithms would lead to computational explosions. Consequently, there needs to be a tradeoff between computational complexity and precision [8].

Therefore, only a few algorithms can handle large graphs that would allow, in principle, to design a GED algorithm performing more accurately than currently possible [5]. Abu-Aisheh et al.[3] mentioned that there are currently no reliable algorithms to compute GED for big graphs having more than 16 nodes within an acceptable time frame. Thus, using pure GED is not be enough for a similarity search of process models, given that real-life process models often exceed 20 nodes. Therefore, heuristic algorithms have been proposed that are based on graph similarity. The difference between graph edit similarity and GED is that "[GES] of two graphs is the maximum possible similarity induced by a mapping between graphs [while the] GED of two graphs is the minimal possible distance induced by some mapping "[8, p. 53]. The best performing algorithm using GES was a greedy algorithm that maps between a pair of process graphs [8].

Previous Related Work Pichler [18] builds a business process management system using supply chain event management to identify and manage events that occur during process execution. It monitors processes, tracks, and traces the whole process down to its activities' progress and sends out an alert if there is a disturbance in the process, such as activities that do not finish on time. CBR is used to find similar past cases from the CBR database, which could be applied to the current event. The process and its model where the event occurred function as a basis for CBR systems search. The current and the old case have to be compared to find a suitable case. In other words, the similarity between the cases has to be calculated. The higher the similarity, the more compatible are the two cases. Therefore, Pichler uses Petri Nets and a process description. The current case is characterized through the process instance, its execution status, and its description, which is then compared to the case base. Although Pichler [18] states the critical part needed to compare the old with new cases, it lags an explanation of how each case's descriptions are being translated to be comparable, since different people use different semantics in describing a case or a process. Aagesen and Krogstie [1]. found that people working on process design tend to

use text annotation extensively to work around design issues. Hence, much text would have to be analyzed instead of the process itself. Having a diagram in the first step to visualize and identify the process' problem would help to search for a solution with specific keywords in the case base.

3 Storing BPMN Diagrams

Using the CBR life cycle, the first cases have to be stored to build a case database that can be later used to optimize BPMN diagrams' problems (see Fig. 1). Before any BPMN diagram can be stored first, the key information must be retrieved for proper labeling to index each diagram correctly. To be able to retrieve information, the BPMN diagram has to be first converted to an XPDL file, which keeps the diagram's structure. Thus, the indexing system is only working with the BPMN's respective XPDL files, which can later be translated back into a BPMN diagram. Therefore, only the XPDL file of each BPMN diagram has to be stored. There are specific sections within XPDL that contain information needed to build the labels for the BPMN diagram's index. Creating the BPMN diagram index at the storing stage helps reduce the computing time later in the retrieving phase. Further, the BPMN information used is limited to the diagram itself. No additional information, such as activity time, is used to analyze this thesis for simplicity reasons.

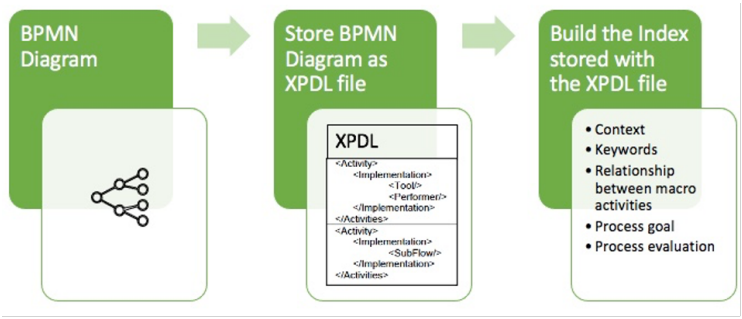

Fig. 1. From a BPMN Diagram to its Index

Building the Index

The index's labels are only effective if they adequately capture and describe the process's content. Hence, each XPDL file's labels can be broken down into the following five different categories, namely context, keywords, the relationship between macro activities, process goal, and process evaluation. Together they capture the actors involved in the process, the tasks they fulfill, and how they relate to each other. Further, they contain the goal the process is supposed

to achieve and its success or failure. This information will help later in the retrieving process to faster allocate processes with similar goals and use positive and negative outcomes to understand better how the right solution should look. Therefore, it strengthens the understanding of how processes are structured in detail and what professions are involved in getting to the desired goal.

Information in the original Diagram The context label consists of the process and participants' names for retrieving, which a few steps have to be taken. From where and what information is obtained from the XPDL file can be seen in Figure 2. First, the workflow process tag provides the name of the process. Next, as previously mentioned, BPMN's pool- and lane -names contain the information about the process participants, which are not directly represented within XPDL. Since only a task or activity can hold the participants' information, the participants in XPDL associated with the pool and lane names of the BPMN diagram have to be retrieved from the activity itself. The context can show if a process needs different actors or if the current actor setup is a good fit for the current process problem. In the rest of this paper, the current problem refers to the problem case that is tried to be solved with the herein developed artifact.

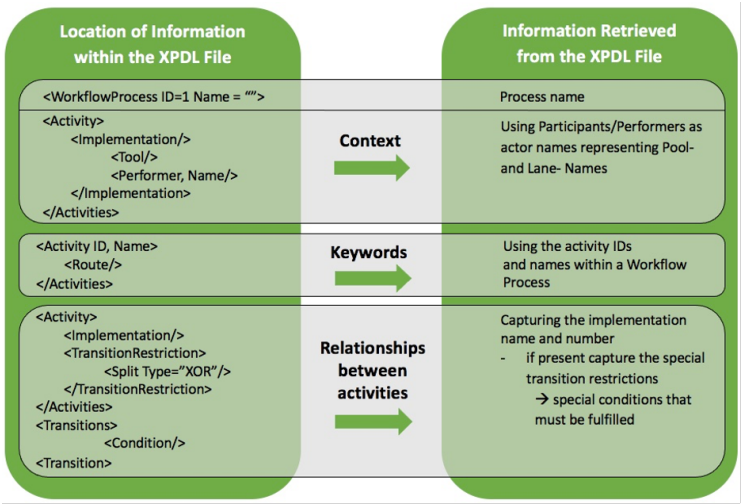

Fig. 2. XPDL Information Extraction

Further, the macro activities are used as keywords. Therefore, the keyword label summarizes the XPDL activity tags within a workflow process, which provides information about what happens during each activity. One process's activities can then be compared to other process's activities on similarity and if the current problem process might have to change, add or delete activities.

```
(<PackageHeader>
    <XPDLDescription> Goal = {}>
 </PackageHeader>
 <WorkflowProcess ID>)
```

List. 1. XPDL Header adding the Goal Label

```
(<PackageHeader>
    <XPDLDescription> Goal = {}, Success = {True or False}>
 </PackageHeader>
 <WorkflowProcess ID>)
```

List. 2. XPDL Header adding the Success Label

Next, the relations between macro activities have to be captured. Here the implementations, transitions, and conditions explain what happens with the token after each activity. In other words, as they are like the BPMN's gateway, they describe how two activities are connected, for example, through an exclusive or parallel gateway. This gives additional information on how complex the flow of the diagram is structured. When compared to other processes, it can show the differences in structure and flow to reach the goal of the process. Although having the same goal, they can be vastly different and thus can show opportunities for improvement of the current problem process.As a process can involve multiple participants, activities, or relationships, it is favorable to store them all together in a list. These lists can then be compared to each other position by position.

Information not in the original Diagram The following information is not directly translated from a BPMN diagram into XPDL; however, it has beneficial effects when added to XPDL as it makes comparisons easier. Knowing the goal of a process at the beginning will help later to compare processes better. Processes with the same or similar goal can be grouped and thus limit the search space for similar solutions.

These similar solutions can the compared based on the previously explained labels context, keywords, and relationship between macro activities. Hence, if similar goals are available in the database, they can be an additional supportive label and can be added as a name into XPDL's package header (see List. 1). Further, it is essential to have a process evaluation knowing whether the process was a success or failure (see List. 2).

Negative examples support building the new models by showing how the process's goal could not have been reached. However, a more effective method is to separate the negative examples since this binary label would strongly polarize a later similarity search. Another possible method is leaving successful and unsuccessful cases in the similarity search without having the current problem's success labeled, which would provide more freedom of search.

Since the other cases in the case database would still possess this label, the other labels' similarity would determine the current problem's success or failure. Thus, there would be negative and positive examples retrieved, providing broader knowledge that would support a more accurate solution in the end. On the other hand, successful models possibly provide an instant alternative solution to the current problem. Having established all five labels, they can be used to describe and classify each XPDL file process. Combined, they form the index for each BPMN process consisting of the process ID and the five labels. This construct can then be stored together with each XPDL file in the case database and be queried against the before-mentioned index's labels. However, as the previous arguments show, there are many possibilities to combine and restrict labels during the similarity search to obtain one or multiple solutions to a current problem. These combinations and restrictions have to be chosen by the user's preferences and tailored to the knowledge required to find a solution to the current problem.

4 Retrieving BPMN Diagrams

In the second phase of CBR matching cases to a current BPMN problem are being retrieved. Each XPDL file belonging to a BPMN diagram has been stored with its own index consisting of five different labels in the form of lists. The retrieving phase utilizes Carbonells suggested difference function in the form of a semantic search engine and a similarity metric to compare the lists with each other. The closest matches are then being returned for the adaptation step.

Difficulties of retrieving BPMN Diagram

Retrieving analogical BPMN diagrams means searching for some degree of similarity related to content or structure. The words and language each BPM specialist used can be different within the process schema, for example, naming the activities. Hence, two different BPM specialist can solve the same topic or problem using different words which are semantically similar. However, a normal search would only be able to find similar words with tokens or an edit-distance-based system. Therefore, semantics play a significant role when searching for similar BPMN diagrams.

Simultaneously, solutions in the case database may solve the same or a similar problem while having a completely different structure. In another scenario, the solutions may solve different problems with very similar structures. Thus, the structure is also important and can be different for each situation.

Consequently, two underlying problems have to be solved. The different diagram structures that could be beneficial and the used words that could be synonymous. While the different usage of words could mean the problem is covering a different problem, a different structure could just implicate another creative way of solving the current problem.

Semantic Search Model

A semantic search model queries the database's index for its keywords. There are many search engines that can retrieve information based on the similarity of words calculated by different edit-distances, but that is not the goal of this engine. Retrieving the same words limits the search to a strong match of similarly spelled words leaving just the possibility of a different diagram structure. This, however, would disregard the other powerful impact of semantically similar words. Therefore, the semantic search model uses a semantic engine that also finds semantically similar keywords or a list of keywords.

As previously established, the index's information is obtained through the original XPDL file of the BPMN diagram (context, keywords, and the relationships of the activities) and some additional information (goals and success), which has to be cleaned first. A preprocessing stage has to take place, normalizing and tokenizing the data.

The semantic search model then uses the preprocessed index and calculates a semantic similarity score for each label for each diagram in the database. In order to calculate the similarity score, the semantic search model has to build a vector space, grouping together semantically similar words using constructs like unsupervised learning models. Afterward, the semantic search model calculates an overall similarity score based on the weight the user assigned to each label (different levels of importance for different labels) and returns the most similar solution. Hence, priorities have to be set on which labels are most important to the user. As a result, this paper proposes that the semantic model is trained by the whole database (see Fig.3).

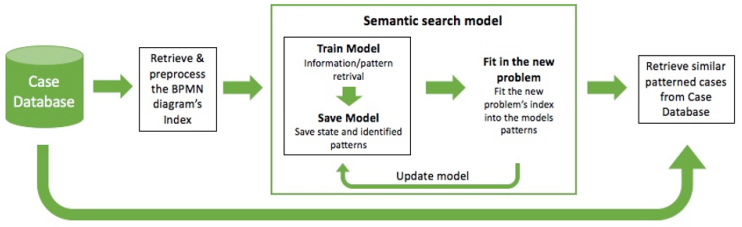

Fig. 3. Sematic Search Model used for retrieving similar cases

It breaks down the index and learns what phrases and word patterns, which are not necessarily linked to a specific label, exist in the database. Once a problem diagram is introduced to the model, it will compare the problem diagram's words to the database's pre-identified patterns. The space of patterns the new problem fits most can then be assumed to be semantically similar and retrieved. Hence, the model categorizes the new problem according to pre-identified patterns. For example, there could be a pattern of how activities are being named around a particular process goal. Thus, the model remembers that these names are used

when a particular process goal has to be achieved. If the words of the current problem are using words associated with this pattern, the model will retrieve the diagrams with the same word patterns from the database. As only word patterns are being analyzed, the index's context or goal labels can vary from the current problem giving a possible solution more freedom and more knowledge. Here the current problem's index must not contain the label of the process' success because the semantic engine will bring into a context space around other processes sorted around the label success or failure, which is not desirable when looking for knowledge that helps to find a successful solution to the current problem. The success label will help sort the most similar cases at a later stage. It is also important to notice that this analysis does not consider the order of activities in the diagram, as it just looks at the similarity of word patterns in the index, giving the search engine further freedom.

Further, the model's training time could be very time-consuming depending on the database's size; however, the training does not have to be repeated for every search process because, without new cases, the patterns and phrases will not change. Instead, after a threshold of new diagrams has been stored on the database, the training process can be conducted during system downtime.

Nevertheless, the built index, which the model uses for the analysis, can be categorized as short text. Short text analysis has shortness and sparsity, which is a critical challenge for traditional text mining tools [11]. Traditional text mining tools require cohesive text to train the model to learn patterns and other information, which can later be found in other texts/documents. In the model's case, the patterns can be seen as semantic similarity clusters. Although much research studied the application of text mining tools for whole documents and more extended text parts, other papers suggest much current research about short text analysis, for example, using discovering topic representative terms or self-teaching convolutional neural networks [23] [24]. Hence, the semantic search model is still a theoretical construct, which practical development is out of scope for this thesis and is left to future research. Therefore, future tests are necessary to prove the viability of the above suggest construct and if the indexes are long enough and with adequate information to find and retrieve semantically similar indexes.

Graph-Edit Distance

Before, the semantic search model looked at the words used in different diagrams to find some similarities. This section looks at the BPMN diagrams' structure that visualizes the process flow from beginning to end. As previously established, the diagram structure holds much additional, important information. If a process is solved similarly to another process, it means that the structures coincide. The more the processes differ in the solution, so do their structures. Hence, when comparing two diagram structures for similarity, there should always be some threshold that allows for some structure differences providing some alternative structure solutions.

The need to compare two diagrams requires some metrics measuring their similarity, for which usually some distance has to be calculated. Therefore, this thesis proposes GED as a measure since the BPMN diagram's process structure can be translated into a graph [8]. An example of the mapping from an XPDL to a graph can be seen in Figure 4.

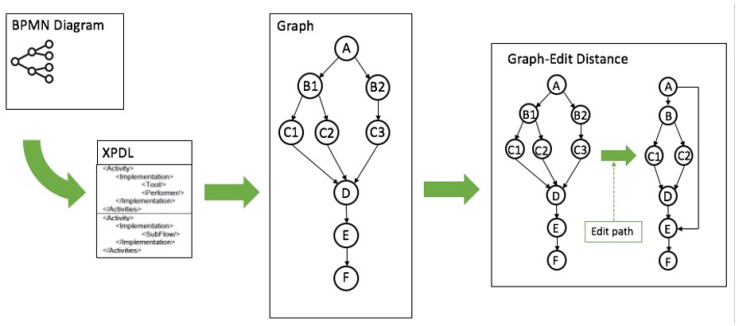

Fig. 4. From XPDL to Graph-Edit Distance

Once the BPMN diagram is translated into a graph, its activities are now called vertexes, and its connecting objects are called edges. All other parts of the diagram are not considered in this step. GED can be applied to any graph where it calculates the minimum operations needed to adjust one graph to the other. Therefore, each vertex/edge from the first graph has to be mapped to one particular vertex/edge of the second graph or a dummy vertex/edge, which can absorb structural transformations.

Changing one graph to the other is achieved through the edit operations insert, delete, and substitute, which sequence is called edit-path. This edit-path is applied to the graph's vertexes and edges. Next, the extent of the transformations applied to a graph by the edit-path is represented by a cost function. The cost function can measure the magnitude of distortions since each edit operation has a predefined cost. Hence, the edit-distance between two graphs is defined by the edit path with the lowest cost function indicating the highest structural similarity. Since there is a lot of research on calculating graph similarity, there are already a few algorithms using the GES, a GED form. According to the research, GES can be better handled by a greedy algorithm, which is currently the leading algorithm, especially when graphs get as big as a BPMN diagram with more than 18 nodes. However, the test and possible implementation of this GED form and algorithm is out of this thesis's scope. That is why this paper will use GED, which can later be translated into GES.

Retrieving using a Semantic Search Model and Graph-Edit Distance

The previous sections covered the retrieving phase analyzing the written and structural parts of a BPMN diagram. Now, as a final step, the retrieving phase

has to coordinate these two analysis parts. First, both analyses need a threshold of the degree of similarity to retrieve similar cases, which has to be done by the user. The model needs the thresholds to find some similarity that narrows down the possible solutions because, without a degree of similarity, the system does not know what cases to retrieve. Once the thresholds have been set, the order of the two parts has to be set up. Since semantics can better describe a solution's content and whether it has dealt with a similar or the same problem, the most logical order is the semantical analysis followed by the structural analysis. Accordingly, the system must first learn the semantical context of possible solutions to the problem, which it will then retrieve. After, it can look at the structures of the same context, possible solutions. According to the threshold, based on the graph-edit distance or the GES and a greedy algorithm, it can then retrieve the solutions with the highest similarity. These solutions can then be used in the next step to adapt the current problem to the successful retrieved solution.

Retrieving possible solutions based on similarity always poses a knowledge barrier problem. As the system looks for some similarity, it is impossible to retrieve all knowledge present in the database. There might be a solution with no semantic or structural similarity to the current problem but still offers a viable solution. This solution might even be from a completely different domain. However, since there is no similarity, the system will not be able to detect this kind of solution. Thus, these solutions also cannot be considered by the system.

5 Adapting and Retaining BPMN Diagrams

The CBR live cycle's adaptation phase is about the knowledge transfer from the past solutions, retrieved from the case base to the current problem, which is not just about the similarity between the past solution and the current problem. According to Carbonell (1983), it is important to look at what kind of knowledge is transmitted for adapting the old solution to the current problem. Ideally, the adaptation model consists of positive and negative problem solutions, which share the same issues. Additionally, past solutions and the current problem should also have similar reasons of decisions taken available for a thorough understanding of the two problems at hand. If the reasons for decisions taken are congruent, the past solution could solve the current structure's problem.

In the current construct, the adoption phase can have positive and negative solutions available, the amount of which is based on the similarity threshold set in the retrieving phase. Additionally, the ratio of positive and negative examples can vary randomly for each CBR life cycle depending on the similarity score.

However, not all information Carbonell suggests is available in the BPMN diagrams. For example, the reasons for decisions taken are often not written down in the diagram and thus are difficult to obtain. If information about the decisions taken were present in the diagram, it would be in the form of process descriptions used to explain some parts of the diagram. Since not all parts of the diagram are always explained, the reasons for decisions taken are often incomplete and would make the text analysis even more complicated and time-consuming. Hence, due

to incompleteness and increased complexity of the retrieving phase, the reasons for decisions taken are not considered in this adapting approach.

In the end, there are only two points of knowledge available for the system to derive a valuable solution for the current problem, namely semantical context and a degree of similarity between the process models' segments. Consequently, there is not enough information to implement derivational analogy since the system cannot reuse decisions taken in the retrieved solutions and build a new solution from null. Therefore, this report suggests a transformational analogy for the system to build the solutions to the current problem based on past solutions, which can deal with the context and structure of retrieved solutions and the current problem. Additionally, next to the transformational analogy, the adaptation system takes into account positive and negative solutions. Now that the basic concepts of the adaptation system are set, detailed procedures can be developed.

The Adaptation System

The adaptation system (see Fig. 5) consists of multiple steps, from applying old knowledge to validating the new solution and its functionality. Since the semantic engine already retrieves the contextually most similar cases in the previous retrieving step and the GED distance further narrows down the search to the structurally closest solutions, all these solutions can be temporarily stored in a separated pool (Step 1 of Fig. 5). Next, the adaptation process (2) applies the changes (Fig. 6). The GED has already calculated all changes that have to be made between the vertices and edges of the old solution and the current problem in the retrieving phase. In this case, the vertices are the process's activities, and the edges are their connections. Thus, the adaptation process implements the most straightforward change calculated by the GED, making the current problem's process successful.

In order to validate the successful change from the current problem to the new viable solution, the changes within the BPMN process's XPDL file have to be temporarily stored (3). Since the transformational analogy applies the best successful solution to the current problem, the negative examples have not been considered yet. Therefore, the negative retrieved solutions can be separated into an extra pool (4) so that in the next step, the temporary new solution can be compared to the negative examples (5).

Since the system is now dealing with negative examples, the desired outcome is a low similarity between the new temporary solution and the case base's negative solutions. However, they will have some congruities, given all cases have been retrieved beforehand based on a high similarity score. Hence, the comparison looks at whether there is less similarity to the negative examples than before in both semantics and structure. Therefore, the comparison uses the previous similarity score that has been established during the retrieving phase and calculates a new overall similarity score. The assumption is that if the old similarity score of the current problem and the negative solutions are similar in the beginning, the newly calculated score should be lower after the adaptation of a successful

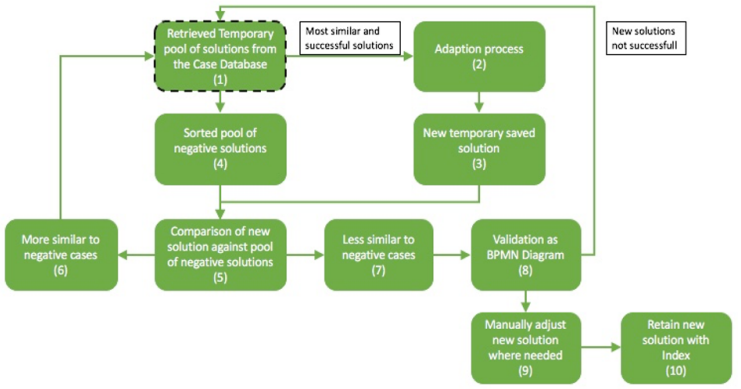

Fig. 5. Adaptation Process Overview

solution. Thus, according to the previous assumption, if the new solution is not less similar to the negative examples or if the new solution is even more similar than before, it would indicate a negative, unsuccessful new solution (6). As a result, the next best retrieved successful solution would have to be retrieved from the temporary pool (1) to go through the adaptation and comparison process (2-5) again. Once the similarity is less than before (7), the assumption is that the new solution to the current problem must be more successful than before.

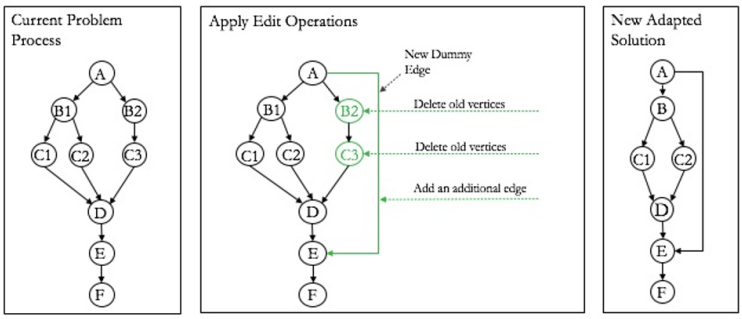

Fig. 6. Adaptation Process using Transformational Analogy

For validation purposes, the new solution, transformed into a BPMN Diagram, is then tested for a correctly running process (8). If it is also successful at this stage, the current problem can be solved with the new solution. However, this new solution was built based on the structure disregarding the context as it is assumed to be similar. Thus, there is the need to check for misinterpreted

activities by the system, which contextually and structurally do not fit precisely the place the system put it (9). For example, there is always the possibility of two times the same activity at different places in the process diagram. It can result from the two processes having the same activity before the adaptation but at different places. In addition, due to the double copy of one activity, there exists the possibility of a missing activity. In this case, it is required to manually adjust the new activities' order and language according to the current problem's context and compare the two diagrams' structure. Further, the user must also verify the process according to the needs and, if needed, adjust the process. Once the new problem solution is found viable, it can be retained in the case database as a new solution (10). Therefore, the new solution to the current problem is indexed as described in the section storing BPMN diagrams.

6 Discussion

This section analyzes the developed artifacts' usefulness, critique, and possible improvements based on the validation section's answers. The outcome will be a general evaluation of the proposed model.

All interviewees agreed to the usefulness of the theoretical model and its learning from past cases. Although they agreed to the usefulness, they gave feedback to different parts of the model. It was seen positively that the model would work faster in finding old, similar cases than humans improving the knowledge management's performance.

Especially, the pool of retrieved, similar solutions could lead to a more oriented outcome. Although some concerns were expressed that there can still be no fitting solution, the model retrieves the most similar solutions based on a similarity count system, so a solution is always returned. Hence, there should also be a person checking the new process's viability, which is also part of the model. Further, the interviewees mentioned a certain similarity between the model and its use in consultancy companies on a basic level, comparing it to their knowledge management learning from old cases since they do not always start from scratch for every project.

These statements show that learning from past cases has positive aspects and is already used in some professional environments, confirming that CBR used in the developed model is useful.

However, there has also been criticism of the model of different aspects. Some interviewees would have preferred more practical examples and data tests, such as company data, which goes beyond the Thesis's scope. As mentioned in previous sections, the purpose of this Thesis is to develop a theoretical model. If the model is deemed valid, and useful future research involves its practical implications.

Further, it was argued that the BPMN diagram alone is too general to draw similarities and does not provide enough data. Many other parts of the model influence the process's simulation, which is the most crucial part of validating the new process. With this additional information, it is tough for CBR to analyze because every case is very individual. So, if CBR was used, it has to be focused

on qualitative analysis. As the beginning of the Thesis mentioned, it has only been focused on the BPMN Diagram for simplicity of the theoretical model.

The recommendations focused mainly on the BPMN details used to find similar cases and to adapt them afterward. If one deletes information and parts of the processes 100, they are not available anymore if any other problems arise with the process where the deleted parts could be useful. It is never practical to delete information 100. Instead, there should be a backup database that stores the deleted parts. As a result, a backup database and practical tests in the adaptation stage should be considered. While the backup database can be incorporated into the adaptation stage, the technical implications are future research.

Further, there should be some kind of validation in every step that verifies why one moves forward. The model's validation steps have been considered, primarily through the final human validation step of the new process's correctness. As mentioned under criticism, just searching for the similarity between the BPMN diagrams is not enough since more detailed information is important to make the process work and verify it through simulations. Thus, it was recommended to continue to find similarities between the BPMN diagrams as proposed. However, instead of just storing and retrieving the diagram, once a similar old diagram has been found, all its additional information should be retrieved with it in order to properly understand the old solution to adapt it correctly to the current problem. Therefore, even the old AS-IS and TO-BE models would be essential to understand the justification. This could be added to the current theoretical model increasing the human validation step's importance. All in all, the theoretical model itself has been evaluated to be useful, although it requires adjustments for additional BPMN information in the adaptation part. It became evident that the human factor in checking and analyzing the new and final process through simulation is significant for its validation. Hence, these points will have to be considered for the theoretical model's practical implications.

7 Conclusion

Based on extensive research in BPM and CBR, together with side topics, such as GED, it can be concluded that a whole CBR life cycle can be applied to BPMN diagram problems with the need for human intervention.

As BPM at its core controls and optimizes the business processes to make them more effective and efficient, it ultimately increases shareholder value. CBR, as part of AI, tries to mimic human learning by analogy. Hence, CBR is adopted to support BPM, making better decisions with existing knowledge when solving process problems. Therefore, BPMN diagrams must be translated first into XPDL files before CBR concepts and transformational analogy can be applied. Using the CBR life cycle helped to structure the solution's steps to build a closed learning process.

During the storing phase, each XPDL file obtains an index with five labels that contain the needed information for efficient comparisons between old solutions and the current problem later on. After the old solutions have been stored in the

case database, the retrieving phase starts with semantically analyzing the current problems' index. Further, it will place the current problem in the semantic space with the old solutions that use similar wording, which is expected when dealing with similar solutions. The old solutions the semantic analysis finds closest to the current problem are then extracted for structural analysis. Here the XPDL structure is translated into a graph to apply GED distance to calculate the smallest edit distance to adjust the old solution to the current problem. Thus, the smallest GED distance indicates the final most similar solution. In the final adaptation phase, the transformational analogy is used, fundamentally applying the smallest Graph-Edit Distance to the current problem. Afterward, the new structure of the current problem has to be tested for its viability. If the user finds the current problem's solution successful, its wording must be adjusted before it is stored in the case database starting a new CBR cycle. While the proposed artifact's usefulness is generally approved, it needs improvement in some areas.

This dissertation was developed with the purpose to contribute to the development of learning from past solutions by building an overall system that stores, retrieves, and adapts old BPMN solutions to current BPMN processes. Therefore, this work did not have the objective to solve the whole problem but to contribute to a possible solution by using CBR. Combined with BPMN diagrams each CBR phase needs a different and unique approach to bring the whole CBR life cycle together as one working solution. The solution is constructed on a theoretical level and thus does not provide a practical implementation.

Limitations

This research focused on building a theoretical model; it did not develop the technical side because it is outside the thesis's scope due to time constraints. Further, the developed semantic search engine is a theoretical construct with not yet reached technological requirements. Until now, most text mining tools for natural language processing need excessive amounts of concise text to be trained and to conduct an analysis. Most search engines look for the exact word or phrases of words on other documents limiting the retrieval of other matching knowledge.

Moreover, big BPMN diagrams can still be too big for similarities to be calculated by GED or GES in an A* algorithm. Lastly, adaptation is a challenging part as it is limited to blindly applying the structure without checking if the content of each activity also fits in the same space. There are still many problems in the adaption phase, so the user must intervene to check the outcome, maybe even for the temporary outcomes.

Future Work

Future research has to consider all the additional metadata that describes the diagram's process in detail. However, the model theoretically has the ability to store the additional information and retrieve it together with the BPMN diagram.

Additionally, deleted sequences in the adaptation phase should be stored in a backup database for possible later use.

From the technical side, further research needs to determine the semantical model's and GED's capabilities using practical examples to confirm their viability and usefulness.

Hence, future research should extend the model's theoretical possibilities and its technical implications, which will minimize the need for user intervention.

References

1. Aagesen, G., Krogstie, J.: Bpmn 2.0 for modeling business processes. In: Handbook on Business Process Management 1: Introduction, Methods, and Information Systems, pp. 219–250. Springer Berlin Heidelberg (jan 2015)
2. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7(1), 39–59 (1994)
3. Abu-Aisheh, Z., Raveaux, R., Ramel, J.Y., Martineau, P.: A parallel graph edit distance algorithm. Expert Systems with Applications 94, 41–57 (mar 2018)
4. Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L.: Comparing heuristics for graph edit distance computation. In: VLDB Journal. vol. 29, pp. 419–458. Springer (jan 2020)
5. Blumenthal, D.B., Gamper, J.: On the exact computation of the graph edit distance. Pattern Recognition Letters 134, 46–57 (jun 2020)
6. Carbonell, J.G.: Derivational analogy : a theory of reconstructive problem solving and expertise acquisition. Tech. rep., Carnegie Mellon University, Pittsburgh (1985)
7. Cheng, R., Sadiq, S., Indulska, M.: Framework for business process and rule integration: A case of BPMN and SBVR. In: Lecture Notes in Business Information Processing. vol. 87 LNBIP, pp. 13–24. Springer Verlag (jan 2011)
8. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 5701 LNCS, pp. 48–63. Springer, Berlin, Heidelberg (2009)
9. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
10. Gregor, S., Hevner, A.R.: Positioning and presenting design science research for maximum impact. MIS Quarterly: Management Information Systems 37(2), 337–355 (2013)
11. Grida, M., Soliman, H., Hassan, M.: Short text mining: State of the art and research opportunities. Journal of Computer Science 15(10), 1450–1460 (oct 2019)
12. Hammer, M.: What is business process management? In: Handbook on Business Process Management 1: Introduction, Methods, and Information Systems, pp. 3–16. Springer Berlin Heidelberg (jan 2015)
13. Kolodner, J.: Promoting Transfer through Case-Based Reasoning: Rituals and Practices in the Learning by Design Classroom and Evidence of Transfer. San Mateo, CA:Morgan Kaufmann (1995)
14. Kuchibatla, V., Muñoz-Avila, H.: An analysis on transformational analogy: General framework and complexity. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 4106 LNAI, pp. 458–473. Springer Verlag (2006), `https://link.springer.com/chapter/10.1007/11805816{_}34`

15. Pantic, M.: Introduction to Machine Learning & Case-Based Reasoning. Tech. rep., Imperial College London (2006)
16. Paul Sciglar: What Is Artificial Intelligence? Understanding 3 Basic AI Concepts - Robotics Business Review. Robotics Business Review (2018)
17. Pérez-Castillo, R., Piattini, M.G.: Uncovering essential software artifacts through business process archeology. IGI Global (oct 2013)
18. Pichler, A.: Flexibilität in Business Process Management Systemen durch Case-based Reasoning. In: Wirtschaftsinformatik Proceedings. pp. 589–597 (2011)
19. Sanfeliu, A., Sanfeliu, A., Fu, K.S.: A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. IEEE Transactions on Systems, Man and Cybernetics SMC-13(3), 353–362 (1983)
20. Van Der Aalst, W.M., La Rosa, M., Santoro, F.M.: Business process management: Don't forget to improve the process! Business and Information Systems Engineering 58(1), 1–6 (jan 2016)
21. WfMC: XPDL - Workflow Management Coalition (2019), `https://www.wfmc.org/standards/xpdl`
22. White, S.A.: XPDL and BPMN. Tech. rep., Future Strategies Inc., WfMC (2003)
23. Xu, J., Xu, B., Wang, P., Zheng, S., Tian, G., Zhao, J., Xu, B.: Self-Taught convolutional neural networks for short text clustering. Neural Networks 88, 22–31 (apr 2017)
24. Yang, S., Huang, G., Cai, B.: Discovering Topic Representative Terms for Short Text Clustering. IEEE Access 7, 92037–92047 (2019)

All links were last followed on January, 2021.