Machine-Assisted Design of Business Process Models Using Descriptor Space Analysis

Maya Lincoln[1], Mati Golani[2], and Avigdor Gal[1]

[1] **Technion - Israel Institute of Technology**
mayal@technion.ac.il, avigal@ie.technion.ac.il
[2] **Ort Braude College, Israel**
matig@braude.ac.il

Abstract. In recent years, researchers have become increasingly interested in developing methods and tools for automating the design of business process models. This work suggests a method for machine-assisted design of new process models, based on business logic that is extracted from real-life process repositories using a linguistic analysis of the relationships between constructs of process descriptors. The analysis enables the construction of a descriptor space in which it is possible to define new process sequences. The suggested method can assist process analysts in designing new business processes while making use of knowledge that is encoded in the design of existing process repositories. To demonstrate the method we developed a software tool ("New Process Design Assistant" - NPDA) that automates the suggested design method. We tested our tool on the Oracle Applications ERP process repository, showing our approach to be effective in enabling the design of new activities within new business process models.
Keywords: New process model design, Business process repositories, Business process integration and management, Process choreographies.

1 Introduction

In recent years, researchers have become increasingly interested in developing methods and tools for automating the design of business process models. Process modeling is considered a manual, labor intensive task, whose outcome depends on personal domain expertise with errors or inconsistencies that lead to bad process performance and high process costs [10]. Hence, automating the reuse of constructs, gathered from predefined process models does not only save design time but also supports non-expert designers in creating new business process models. Research in this field encapsulates topics from the areas of software design and data mining [17,13,5,3], and is focused on structured reuse of existing building blocks and pre-defined patterns that provide context and sequences [4].

While most previous work focused on supporting the design of alternative process *steps* within *existing* process models, less work has been carried out on the design of *new* process models. Even the few works that addressed the design of new models were focused on a specific domain such as production

management [10,12]. This work aims at filling this gap by suggesting a generic method for designing new business process models related to any functional domain. The suggested method guides business analysts that opt to design a new business model, by suggesting process steps (activities) that are relevant to the newly created process model. The business logic for such suggestions is extracted from process repositories through the analysis of existing business process model activities. Each activity is encoded automatically as a *descriptor*, using the "PDC" notation, suggested first in [9] and further elaborated in this work for supporting the field of new process model design. The collection of all descriptors formulates a descriptor space, and distances between every two space coordinates are calculated in terms of business process conduct proximity. We show through an empirical evaluation that by utilizing the descriptor space it is possible to effectively support the design of new process models.

As a motivating example consider an airport process model that incorporates check-in related processes. Now, suppose that the airport management desires to extend the services provided to its customers by offering a new service: "check-in from home." In addition, it is also desired to outline the "check-out" process model as an extension of the current repository. Although these process models are new, the existing repository encapsulates know-how and business logic that are relevant and useful for their creation (*e.g.*, passenger check-in policies and procedures regarding security, luggage handling, passenger handling and document validation). In the above scenario, it would have been helpful for the process designer to design the new processes using a supporting system that relies on the reuse of previous know-how instead of doing this manually from scratch. To illustrate our methodology in this work we use a real-world case study for airport process design. Based on a "check-in" process that already exists in the repository, we demonstrate how it is possible to design the two, above mentioned, new business processes.

This work proposes an innovative method for assisting designers in designing brand new business process models while making use of knowledge that is encoded in the design of existing, related process models. Our work presents the following innovations: (a) it provides generic support to the design of new business process models; (b) it equally utilizes objects and actions for business content analysis: we make use of all activity linguistic components (object, actions and their qualifiers) concurrently, without special focus on objects (as object centric methods do) or on actions (as activity-centric methods do); (c) it takes the PDC model [9] several steps forwards and suggests an extended model that enables the extraction of business logic from business process repositories.

To implement the suggested method we developed a software tool, that was tested and demonstrated using the aviation industry case study and the Oracle Applications ERP process repository.

The rest of the paper is organized as follows: we present related work in Section 2, positioning our work with respect to previous research. In Section 3 we present an extended model for representing process activities based on the process descriptor notion, presented first in [9], and extended in this work

for the field of new process model design. In Section 4 we define and discuss the descriptor space and explain how to navigate in it. Then, we describe our method for designing new business process models in Section 5. Section 6 introduces the software tool and our empirical analysis. We conclude in Section 7.

2 Related Work

Most of the efforts invested in developing methods and tools for designing process models focus on supporting the design of alternative process steps within existing process models. Such a method is presented in [14] aiming to provide next-activity suggestions during execution based on historical executions and optimization goals. Recommendations are generated based on similar past process executions as documented in event logs. Similarly, [4] suggests an approach for helping business users in understanding the context and consequences of applying pre-defined patterns during a new process design.

Few works were devoted to the design of brand new process models within specific and predefined domains. The work presented in [10] utilizes the information about a product and its structure for modeling large process structures. [12] presents a method, named "the product-based workflow design," for designing new manufacturing related processes based on product specification and required design criteria. Our work offers a generic design method that is domain agnostic and does not rely on product design data.

A requirement for the support of business process design involves the performance of a structured reuse of existing building blocks and pre-defined patterns that provide context and sequences [4]. The identification and choice of relevant process components are widely based on the analysis of linguistic components - actions and objects that describe business activities. Most existing languages for business process modeling and implementation are activity-centric, representing processes as a set of activities connected by control-flow elements indicating the order of activity execution [18]. In recent years, an alternative approach has been proposed, which is based on objects (or artifacts/entities/documents) as a central component for business process modeling and implementation. This relatively new approach focuses on the central objects along with their life-cycles. Services (or tasks) are used to specify the automated and/or human steps that help move objects through their life-cycle, and services are associated with artifacts using procedural, graph-based, and/or declarative formalisms [6]. Such object-centric approaches include artifact-centric modeling [11,1], adaptive business objects [8], data-driven modeling [10] and proclets [15]. Further analysis of the object-centric model in terms of computing the expected coupling of object lifecycle components is presented in [18].

Although most works in the above domain are either object or activity centric, only a few works combine the two approaches in order to exploit an extended knowledge scope of the business process. The work in [7] presents an algorithm that generates an information-centric process model from an activity-centric model. The work in [9] presents the concept of business process descriptor that decomposes process names into objects, actions and qualifiers. In this work we

take this model several steps forward by: (a) describing the relationships between the model components; (b) showing how the descriptor model can automatically be generated (using NLP methods); and (c) utilizing the qualifiers for identifying the relationships between descriptor components within a process repository.

3 The Activity Decomposition Model

This section describes a model of business process decomposition that supports process design. To illustrate the model components we make use of the aviation example from Section 1.

3.1 The Descriptor Model

The Workflow Management Coalition (WFMC) [2] defines business process as a "set of one or more linked procedures or activities which collectively realize a business objective or policy goal." An example of such business process model is the "Passenger check-in" process model, presented in Fig. 1. This figure is based on YAWL [16] with two slight visual representation modifications, convenient for our needs: (a) roles were added at the top of each activity; and (b) predecessor and successor processes are presented as nested activities at the beginning and at the end of the workflow.

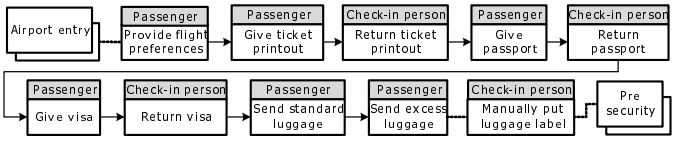


Fig. 1. An example: the "Passenger check-in" process model.

In the Process Descriptor Catalog model ("PDC") [9] each activity is composed of one action, one object that the action acts upon, and possibly one or more action and object qualifiers, as illustrated in Fig. 2, using UML relationship symbols. Qualifiers provide an additional description to actions and objects. In particular, a qualifier of an object is roughly related to an object state. State-of the art Natural Language Processing (NLP) systems, *e.g.*, the "Stanford Parser,"[3] can be used to automatically decompose process and activity names into *process/activity descriptors*.

For example, in Fig. 1, the activity "Manually put luggage label" generates an activity descriptor containing the action "put," the action qualifier "manually," the object "label" and the object qualifier "luggage."

3.2 A Descriptor Model for Process Design

We now enhance the PDC model of [9] to support process design. Our model has two basic elements, namely objects and actions, and we delineate four taxonomies from them, namely an *action hierarchy model*, an *object hierarchy model*,

[3] http://nlp.stanford.edu:8080/parser/index.jsp

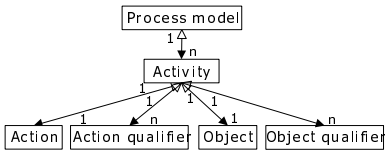


Fig. 2. The activity decomposition model.

an *action sequence model* and an *object lifecycle model*. The business action and object taxonomy models organize a set of activity descriptors according to the relationships among business actions and objects both longitudinally (hierarchically) and latitudinally (in terms of execution order), as detailed next.

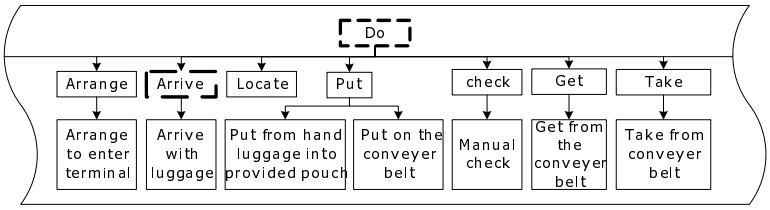


Fig. 3. A segment of the action hierarchy model extracted from the aviation processes.

The longitudinal dimension of actions and objects is determined by their qualifiers. To illustrate the longitudinal dimension of the aviation workflows, a segment of the action hierarchy model is presented in Fig. 3 and a segment of the object hierarchy model is presented in Fig. 4. Consider the complete action (the action and its qualifier) "Manual check." It is a subclass (a more specific form) of "Check" in the action hierarchy model, since the qualifier "Manual" limits the action of "Check" to reduced action range. It is worth noting that some higher-hierarchy objects and actions are generated automatically by removing qualifiers from lower-hierarchy objects and actions. For example, the action "Arrive" was not represented without qualifiers in the aviation processes repository, and was completed from the more detailed action: "Arrive with luggage" by removing its action qualifier ("with luggage") (see Fig. 3). In Section 5 we will show how such elements assist in designing new processes by enriching the underlying process repository range. This type of objects and actions are marked with a dashed border. In addition, a root node "Do" is added to any action hierarchy model and a root node "Object" is added to any object hierarchy model, effectively generating a single object and action tree from what would have been, in graph theoretic terminology, a forest.

To illustrate the latitudinal dimension of the aviation process repository, a segment of the action sequence model is presented in Fig. 5 and a segment of the object lifecycle model is presented in Fig. 6. Latitudinally, each object holds: (a) a graph of ordered actions (an "action sequence") that are applied to that object. For example, the object "Luggage" is related to the following action sequence: "Arrange" followed by "Send" (see Fig. 5); (b) a graph of ordered objects that expresses the object's lifecycle, meaning - the possible ordering of the object's

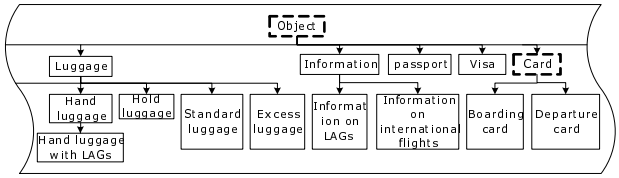


Fig. 4. A segment of the object hierarchy model extracted from the aviation processes.

states. This sequence is built by locating the same object with different qualifiers along the process diagram. For example, the object "Luggage" is part of the following object lifecycle: "Luggage" –> "Standard luggage"/"Excess luggage" –> "Labeled luggage" (see Fig. 6).

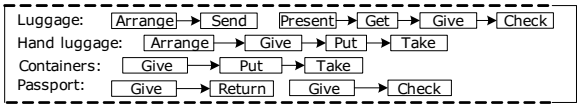


Fig. 5. A segment of the action sequence model extracted from the aviation processes.

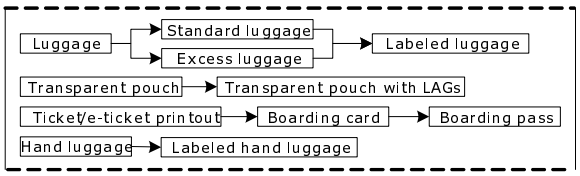


Fig. 6. A segment of the object lifecycle model extracted from the aviation processes.

4 The Quad-Dimensional Descriptor Space

Based on the activity decomposition model, it is possible to visualize the operational range of a business process model as a descriptor space comprised of objects and actions, related to each other and among each other in different relationship types. A navigation within this space can be a powerful tool for analyzing and utilizing the underlying business process knowledge encapsulated within a business process repository.

The descriptor space is a quad-dimensional space describing a range of activities that can be carried out within a process execution flow. The coordinates represent the object dimension, the action dimension, and their qualifiers. Therefore, each space coordinate represents an activity as a quadruple $AC = \langle O, OQ, A, AQ \rangle$, where O is an object, OQ is a set of object qualifiers, A is an action, and AQ is a set of action qualifiers.

The descriptor space is based on a given repository and contains coordinates that cover all existing activities. For example, the activity "Arrive at appropriate

terminal with luggage" can be represented by the following coordinate in the descriptor space: ⟨arrive, with luggage, terminal, appropriate⟩. This coordinate represents an actual activity in the business process: "Airport entry."

Once constructed, the descriptor space provides many coordinates that are not necessarily present in the current business process repository. For example, the coordinate ⟨put, in vehicle, luggage, null⟩ contains components (object, action and qualifiers) from the underlying business process repository, yet the specific combination does not represented an activity within this repository. We consider such combinations "virtual" combinations. These virtual combinations are explored, together with existing activities, in the process of the design of new business processes.

Between every two coordinates in the descriptor space we define a distance function that is tailored for our method. Our empirical analysis shows good results when using this proposed distance measure (see Section 6) and therefore it serves as our main tool for defining activity adjacency within the context of our method.

The proposed distance function in the descriptor space represents a linear combination of changes within each of its dimensions. Therefore, we define four specific distance measures by utilizing the structures that were gathered from existing business processes repositories (Section 3).

Definition 1. ***Object distance* (OD)***: Let O_i and O_j be two objects, OD_{ij} is the minimal number of steps connecting O_i and O_j in the object lifecycle model.*

For example, the object distance between "Luggage" and "Labeled luggage" is 2, since we need to move two steps forward from "Luggage" in order to reach "Labeled luggage" in the object lifecycle model (see Fig. 6).

In a similar way we define *Action distance*, AD, calculated based on the action sequence model. For example, the action distance between "Present" and "Check" when acted on "Luggage" is 3 (see Fig. 5).

Definition 2. ***Object hierarchy distance* (OHD)***: Let O_i and O_j be two objects, OHD_{ij} is the minimal number of steps connecting O_i with O_j in the object hierarchy model.*

For example, the object hierarchy distance between "Luggage" and "Hand luggage" is 1 (see Fig. 4).

In a similar way we define *Action hierarchy Distance*, AHD, calculated based on the action hierarchy model. For example, the action hierarchy distance between "Arrive with luggage" and "Put on the conveyer belt" is 4 (see Fig. 3).

OD, AD, OHD and AHD are combined to generate a specific distance function between any two activities AC_i and AC_j, as a weighted average, as follows:

$$Dist\,(AC_i, AC_j) = OD_{ij} + AD_{ij} + OHD_{ij} + AHD_{ij} \tag{1}$$

It is worth noting that the hierarchy distances (OHD and AHD) can always be calculated since the hierarchy models that they rely on are bi-directional trees. However, the distances OD and AD can be undefined in some cases (e.g. when

the two objects are not connected in the object hierarchy model, or when the two actions are not acted upon the same object and therefore do not take part in the same action sequence). In these cases the above distance components contribute a "no-connection" distance to the overall distance function. This distance is a tunable parameter.

As an example for the use of this distance function consider the two descriptors (luggage,hand,check,null) and (luggage,null,get,from the conveyer belt). To navigate from the first descriptor to the second, we first move one step up in the object hierarchy ($OHD = 1$) from the object "Hand luggage" to the object "Luggage" (see Fig. 4). Then, we recede two steps back from the action "Check" in the action sequence ($AD = 2$), resulting with the action "Get" (See Fig. 3). Finally, we drill down one step within the action hierarchy ($AHD = 1$), and retrieve the action "Get from the conveyer belt", and by that we reach the target descriptor. In total, and by assuming equal weights to all elements of the distance function, the distance between the two above coordinates is 1.

It is possible to navigate within the descriptor space (hence, moving from one descriptor to another) in a meaningful way. This navigation includes navigation along one or more of the four dimensions: the action hierarchal dimension, the object hierarchal dimension, the object lifecycle dimension and the action sequence dimension.

Navigating the Hierarchal Dimensions By navigating within the hierarchal dimensions of the descriptor space it is possible to move up to more general or drill down to more specific action and object scopes. For example, the activity "Put standard luggage" is represented by the descriptor (luggage,standard,put,null). A drill down in the action hierarchy retrieves more specific actions, e.g. "Put on the conveyer belt", a roll up in this hierarchy retrieves the more general action: "Do", and a navigation to sibling actions retrieves several actions such as "Locate" and "Check" (see Fig. 3). Similarly, a drill down in the object hierarchy yields no results, since "standard luggage" is a leaf node in the hierarchy graph. It is possible to roll up the object hierarchy in one or two steps, retrieving the actions: "Put luggage" and "Put object" respectively, and also navigate to sibling objects, *e.g.* "Excess luggage," resulting with three new activities, *e.g.* "Put excess luggage" (see Fig. 4).

Navigating the Sequential Dimensions By navigating within the sequential dimensions of the descriptor space it is possible to navigate to: (a) preceding and succeeding actions that act on the descriptor's object and (b) advancing to a more advanced state of the object processing or receding to a less advanced state. Following the above example, by advancing to a successor action, the activity "Take standard luggage" is retrieved (see Fig. 5) and by receding to a former state of the object, the object: "luggage" is retrieved, resulting in the activity "Put luggage" (see Fig. 6).

As demonstrated above, by navigating the hierarchal and sequential dimensions, it is possible to move to new coordinates within the descriptor space.

Note that some of these new coordinates, e.g. (luggage,standard,check, null), were not originally represented in the aviation process repository, but rather combined from different process descriptor components. Such new activities can enrich the range of process content on one hand, but on the other hand can also add non-relevant navigation results. In Section 5.4 we refer to the issue of how to sort and asses the relevance of such "virtual" navigation results.

5 The *Process Delineator* Method for Assisting the Design of Process Models

Having described the descriptor model and the descriptor space, we are now ready to introduce a method for assisting a process designer in designing new process models based on data gathered from an existing process repository. This is achieved by using the *process navigator*, that assists users in creating new processes in a stepwise procedure. It relies on an underlying process descriptor space and at any phase it either refines an existing process activity or suggests the next process activity. Since the descriptor space has a large number of elements, a general search within this space may be very expensive. Therefore, we will hereby suggest a more efficient navigation method that is tailored for our specific target.

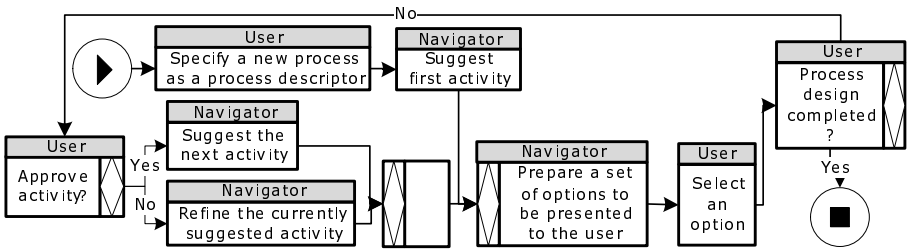


Fig. 7. The process navigator mechanism.

The process navigator is illustrated in Fig. 7. The design process starts when a designer (e.g. a process architect or a business analyst) defines a need for designing a new process. This need is translated into a process descriptor format. For example, if a process analyst wants to design a new process named: "Send luggage from home", it will be transformed into a process descriptor in the following format: object="luggage", action= "send", object qualifier="null", action qualifier="from home".

Based on the process descriptor input, the process navigator produces options for the first process activity (see Section 5.1). The designer reviews the output option list and either selects the most suitable first activity for the newly designed process, or suggests her own first activity for the process. At any next phase the designer either requests to refine the current activity (see Section 5.2) or advance to design the next activity (see Section 5.3). Each time the process navigator is requested to suggest activities as part of the design process (suggest the first

activity, a refinement or a next step activity) it outputs a list of options, sorted and flagged according to the option's relevance to the current design phase (see Section 5.4).

Having received the option list at each refinement or next activity phase, the designer is required to point at the most suitable option either by: (a) selecting one of the suggested options; (b) combining a new process descriptor using process descriptor components from different options in the list; (c) choosing any other process descriptor from the PDC; (d) inserting a new process descriptor.

After selecting the most suitable process activity, the designer examines the newly designed process and decides whether the generated process flow achieves the process goals. If goals are achieved, then the designer terminates the design procedure; else - the design procedure continues and the designer is required once again to decide how to proceed (whether to further refine the currently suggested activity or to continue to the next activity design). This design procedure continues until the process goal is achieved.

It is worth noting that the process navigator enables the production activities that were not represented in the original business process repository, but rather were combined from process descriptor components. These activities are important for the design of new processes, since they can enrich the design process by expanding the optional range of business conduct.

5.1 Suggesting the First Process Activity

To suggest the first process activity, the process navigator focuses on the new process descriptor's object (defined as input by the designer) as its main knowledge anchor, since this object is the subject of the new process. It then attempts to match it with an initial action that can be acted on this object. To do that, the process navigator searches the target object and its more specific objects within the object hierarchy model. It then creates first activity suggestions in the format of activity descriptors comprised of the retrieved objects and the first action that acts upon them. Results are then sorted and flagged according to the method presented in Section 5.4. Continuing the example above, the following first activity options will be suggested (see Fig. 5): "Arrange luggage" and "Give hand luggage".

5.2 Refining the Currently Suggested Process Activity

A refinement of the currently suggested process activity (the "reference activity") means slightly modifying the reference activity's content in a way that the modified activity will still be related to the reference activity but at the same time will express a slightly different operational conduct. This modification can be conducted in five orthogonal methods, and at any refinement phase the process navigator applies each method and collects their results to a unified list of activity refinement options. The five refinement directives are listed below. To illustrate each of these methods we will show how the action "Get luggage" can be refined.

Action and Object Refinement To refine the reference action, the process navigator navigates the descriptor space by drilling down the action hierarchal

dimension. It then combines the retrieved, more specific, actions with the object presented in the reference activity. The refinement of objects is done in a similar manner. By applying an action refinement on our example's reference activity the refinement option: "Get luggage from the conveyer belt" is retrieved (see Fig. 3). Symmetrically, by applying the object refinement method, the following refinement options are produced: "Get hand luggage," "Get hold luggage," "Get standard luggage" and "Get excess luggage" (see Fig. 4).

Action and Object Generalization The generalization method is similar to the action and object refinement method, only this time the process navigator navigates the descriptor space by moving up the action and the object hierarchal dimension, respectively.

Advance an Object's State or an Action To advance the object's state within an activity, the process navigator navigates the descriptor space by moving forward in the object lifecycle sub-dimension. In a symmetrical manner, to advance an activity's action, the process navigator moves forward in the action sequence sub-dimension of the descriptor space. It then combines the retrieved, more advanced (successor), actions with the reference object. In our example the objects "Standard luggage" and "Excess luggage" represent more advanced states of the object "Luggage" (see Fig. 6) and the action "Give" follows the action "Get" in the action sequence applied on "Luggage" (See Fig. 5). Therefore, the following three refinement suggestions are constructed: "Get standard luggage", "Get excess luggage" and "Give luggage".

Recede to a Less Processed State of the Object or to a Former Action The receding method is similar to the above advancing method, only this time the process navigator navigates the descriptor space by moving backwards in the object lifecycle and action sequence sub-dimensions.

For example, the action "Present" is acted on "Luggage" before this object is taken (before the action "Get" is applied) (see Fig. 5), hence creating the refinement option: "Present luggage."

Move to a Sibling Action or Object In order to move to a sibling action, the process navigator moves horizontally within the action hierarchal sub-dimension. By fixing the reference action's level, it retrieves sibling actions for this action. Moving to a sibling object is conducted in a similar manner. Continuing our example, a navigation to sibling actions to "Get" retrieves a list of activities that includes: "Check luggage" and "Take luggage" (see Fig. 3). In the same manner, a search for sibling objects, retrieves a list of activities, that includes: "Get passport" and "Get visa" (see Fig. 4).

5.3 Suggesting the Next Process Activity

In order to suggest the next process activity for the newly designed process, the process navigator takes the process execution flow one step forward. This advancement can be achieved in two alternative ways: either by advancing to a later action that acts on the currently accepted (the reference) object, or advancing to a sibling object combined with the reference activity's action. The rationale behind the last directive is that in some process flows the same action

is operated on sibling objects in order to fulfill a certain process goal (e.g. in the aviation processes, the "Check-in" process includes the two consecutive activities: "Send standard luggage" and "Send excess luggage"). Note that in the same manner the process navigator can suggest previous process activities, in case the user points at any activity in the newly designed process and requests predecessor suggestions. The only difference is that in this case predecessors instead of successors will be retrieved and matched to the reference object.

To illustrate how the process navigator generates next activity suggestions, lets assume that the activity: "Give passport" has been currently selected by the designer and she now wishes to create the next process activity. To do that, the process navigator searches the descriptor space for next activity options. It finds in the action sequence model two options: "Check passport" and "Return passport" (see Fig. 5). In addition, sibling objects to "Passport" are also retrieved from the object hierarchy model, creating additional options such as "Give visa", "Give luggage," and "Give information" (see Fig. 4).

5.4 Preparing a Set of Output Options

The process navigator assesses the output options in each navigation phase and combines an ordered option list to assist the user in selecting the most suitable option. The process navigator sorts the options according to their relevance to the current design phase based on two considerations. First, on proximity to the design phase reference coordinate - which represents the last selected activity when suggesting a refined or next activity, or to the targeted process descriptor when suggesting the first process activity. Second, the process navigator consider to what extent was it changed comparing to actual activities that were part of the underlying process repository. Therefore, the construction of the ordered option list is conducted according to the following four stages: (a) sort by proximity to the reference activity; (b) internally sort by similarity to processes in the repository; (c) add a random option to avoid getting stuck in a local optimum; and (d) flag each option, as further detailed below.

Sort by Proximity to the Reference Activity The process navigator calculates the distance between the reference coordinate and each of the list options (see definition 1), and sorts the list in an ascending order - from the closest to the most distant option.

Internally Sort by Similarity to Processes in the Repository The process navigator also takes into account the extent to which it changed compared to actual activities that were part of the underlying process repository. For this purpose the process navigator distinguishes between three levels: (a) *No change*- the suggested activity is represented "as is" within the underlying business process repository. These options are not marked by any flag; (b) *Slight modification* - there is an actual activity in the underlying business process repository containing the same object and action with different qualifiers. These options are marked with "~"; (c) *Major change* - the object and action within the suggested activity were not coupled in any of the activities within the underlying business process repository. These options are marked with "M".

Therefore, after sorting the options by their proximity to the reference activity, each group of options with equal distances is internally sorted in an ascending order - presenting the no change options at the beginning of the list, since these options posses more credibility using the knowledge gathered by the process navigator. According to the example presented in Section 5.3, several options were generated as candidates for next activities to be conducted after the activity "Give passport." Most of these options were produced by combining the action "Give" with siblings of the object "Passport", hence having the same distance from the reference activity. Nevertheless, these options can further be differentiated. For example, "Give visa" is an actual activity in the aviation process repository, and therefore is flagged as such. Nevertheless, "Give luggage" has no representation in this repository, but since "Give hand luggage" does, this option is flagged using the "~" sign. Since there is no descriptor that combines the action "Give" and the object "Information" in this repository, the option "Give information" is flagged by "M".

Add a Random Option To avoid getting stuck in a local optimum, the process navigator adds at any stage a random activity from the descriptor space, that shifts the reference activity to a new random coordinate, in a similar manner as in simulated annealing (or mutation in genetic algorithms). Thus, the process navigator can provide new suggestions that are based upon a proximity sort to this new reference activity.

Flag Each Option After assessing each option's relevance to the current navigation phase and sorting the option list accordingly, the process navigator tags each option according to the following coding rules. First, since more than one option can achieve the same distance value, the numerical distance value is presented at the beginning of each option. Second, the change level is added to the distance indication, separated by a comma. For example, the option "Give luggage" from the example above will be flagged "[2,~]".

6 Implementation and Case Study

6.1 Implementation

We developed an IT system that implements the suggested method for designing new process models. Given a process name, and based on an existing process repository, the system guides users in creating new process workflows. The system implements a client-server architecture, in which the client is responsible for presenting and collecting data from the user and the server is responsible for processing the user's input data and suggest directions for advancing the design process. Server side logic is implemented in PHP using a MySql database. The client runs within an Internet browser and is implemented in HTML and JavaScript, with AJAX calls to the server.

The server side high-level architecture includes five main components (see Fig. 8): (a) the new process design assistant, responsible for managing and orchestrating the process design mechanism; (b) the process repository database that contains the existing business process repository, in terms of activity descriptors and object and action taxonomies - as described in Section 3; (c) the

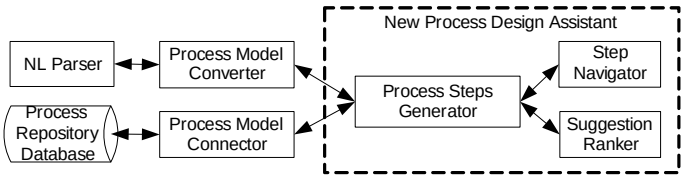


Fig. 8. The new process design assistant high-level architecture.

process model connector, which provides an interface for communicating with the process repository database; (d) the process model converter, responsible for converting inputted business process repositories into a normalized data structure as saved in the process repository database. Currently, our system supports the conversion of repositories expressed in BPEL or BPMN; (e) the Natural Language (NL) parser, an existing web service for decomposing sentences into linguistic components. We use the "Stanford Parser" (see Section 3.1). This web service decomposes activity names into descriptor components.

The new process design assistant is further decomposed into four main components: (a) the process steps generator, responsible for producing suggested activities for each design phase. It communicates with all other components and presents the user at each stage with options for advancing the design process; (b) the step navigator, which is responsible for navigating in the process model database, and for retrieving a list of relevant activity options; (c) the suggestion ranker, responsible for ranking the suggested activity options at each stage.

6.2 Case Study: An Example for Designing a New Process Model

To illustrate the proposed framework we present two examples from the field of aviation. The aviation process repository covers airport activities starting from the passenger's entry to an airport, through document handling and security checks and terminating as the passenger boards the airplane. The newly designed processes are related to the aviation field, but are not covered by the process repository. The first new process, "Passenger checkout," extends the process repository by handling passenger related activities conducted *after* an airplane arrives at its destination. The second new process, "Send luggage from home," extends the process repository by offering an additional service to passengers *before* their arrival at the airport. Using these examples we will show how business knowledge, gathered from a process repository, can be utilized to guide the design of new processes. Note that the design process is an iterative process; in some cases after examining the newly designed process, the designer decides to modify it by adding activities in the middle of the flow.

The first example, illustrated in Fig. 9 as a YAWL diagram, supports the design of a new business process for: "Passenger checkout." The (human) process designer inserts the following process descriptor: (action="checkout", action qualifier=null, object="passenger", object qualifier=null) to the process navigator and determines that the first activity is: "Give passport." Respectively, the

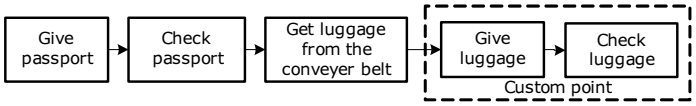
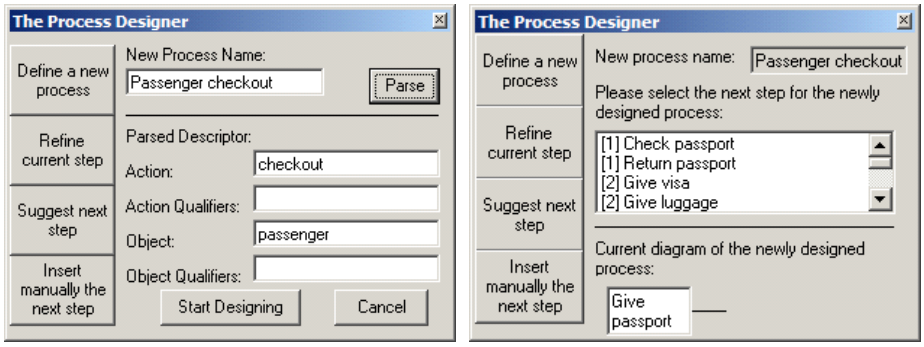


Fig. 9. The new designed process diagram for "Passenger checkout".

process navigator searches the descriptor space, looking for next activity candidates for the newly designed process. As a result, several activity options are retrieved, sorted and flagged, creating an option list that starts with the following activities (see sub-sections 5.3 and 5.4): "[1] Check passport," "[1] Return passport," "[2] Give visa," "[2] Give luggage" and "[2,M] Give information." The designer selects the option "Check passport" and decides that this activity is suitable. She then notices the option "Give luggage" further along the same list (located three options ahead), and decides to combine this activity after "Check passport," since she knows that this sort of activity will be required at the customs point. The designer then asks the process navigator to provide next step options. In the same previous manner, options are extracted and presented as a sorted list starting with: "[1] Check luggage," "[2] Give visa" and "[2,M] Give information". The designer selects the first option, "Check luggage." By reviewing the newly designed process, the designer realizes that an activity may be missing before "Give luggage", since the passenger may not have carried his luggage with him to the airplane. She then points at "Give luggage" and asks for previous activity suggestions. The process navigator searches for next step activities, but this time it navigates backwards in the action sequence. This search results in an option list containing the option "Get luggage" (see Fig. 3). The designer selects this option, and asks the process navigator to refine it, since it seems to lack sufficient details to express the activity required in this context. As a result the process navigator retrieves an ordered list of several options with different flags, among them: "[1] Get hold luggage," "[1,~] Get luggage from the conveyer belt," "[1,~] Get hand luggage," "[2] Get passport" and "[2,M] Get visa" (see sub-section 5.2). The designer reviews the proposed list and decides that the activity "Get luggage from the conveyer belt" describes the action required there. Note that this activity was selected although it was not represented "as is" in the business process repository. Re-examining the newly designed process, the designer decides that her goal is achieved and hence terminates the design procedure.

The designer is now interested to design the new business process: "Send luggage from home." The design process is conducted in a similar manner to the one presented before and results in the process diagram presented in Fig. 11. An interesting observation in this design process is that the designer selects more often next step activities that share the same action applied on sibling objects. The business logic behind this phenomenon is that this process expresses a more interactive business conduct in which one party (the passenger) exchanges items with the other party (the airport representative). For example, the search for



(a) The designer's request for designing the new process: "Passenger Checkout"

(b) The third phase in designing the new process: "Passenger Checkout"

Fig. 10. The designer's request for designing the new process: "Passenger Checkout".

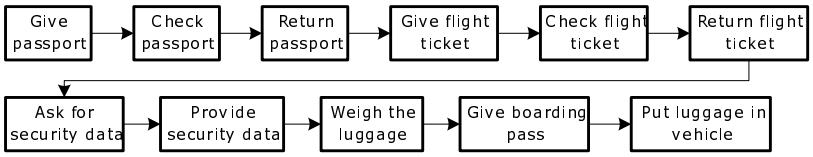


Fig. 11. The new designed process diagram for "Send luggage from home".

next activity after "Give passport" resulted in two activities that were added to the design flow at the same design phase: "Give flight ticket" and "Give boarding pass." In a similar manner, "Check flight ticket" was also added to the process design together with "Return passport" as a result of searching the next activity to follow "Check passport." Note that in both cases after adding two activities at the same design phase, the user added complementary activities between them, and therefore these activities are not adjacent to each other in the final activity flow.

Another interesting observation is the usage of the activity "Put luggage in vehicle." While the original business process repository contained the action "Put in vehicle" applied only to the object "Baby carriage," the terminating activity combines this action with the object: "Luggage." This was achieved by the following design phases: after accepting the activity: "Give boarding pass," the designer asks for next activity suggestions and receives an option list. Knowing that in order to fulfill the process goal, the last activity should be related to luggage, she selects the option: "Give luggage" and asks to refine it since it does not provide enough information. When examining the refinement suggestions, the designer encounters the option "Put luggage" and decides that the action "Put" is even more suitable for the purpose. Nevertheless, further activity details are still missing, and therefore an additional refinement phase is conducted, this time outputting the activity: "Put luggage in vehicle" that fits exactly the designer needs. The designer reexamines the newly designed process, and although the

last designed activity is flagged with "M", hence represents a major change to the underlying process data, the designer approves the new process design as a complete design that fulfills the process goal.

6.3 Experiments

We now present an empirical evaluation of the proposed method effectiveness. We first present our experimental setup and describe the data that was used. Based on this setup we present the implemented methodology. Finally, we present the experiment results and provide an empirical analysis of these results.

Experiment Setup The "New Process Design Assistant" software (NPDA, see Section 6.1) was installed on a workstation running Windows XP, IIS6, PHP 4.8 and MySQL 5.0. This workstation served both as the server and the client, running Internet Explorer 7 as the application container and presentation layer. The "no-connection" distance (defined in Section 4) was set to 500.

Data We chose a set of 14 "real-life" processes from the Oracle Business Model (OBM),[4] comprising: (a) nine business processes from the "Procurement" category, with 96 activities altogether; and (b) five business processes from the "Inventory" category, with 31 activities altogether. The "Procurement" data set contains related, sequential activities and therefore represents a focused operational area. The "Inventory" data set represents an extended business area, featuring loosely coupled business logic. Using the selected 14 processes we created a "process repository database" (see Section 6.1).

Evaluation Methodology In order to evaluate the suggested method we conducted 14 experiments. At each experiment, a single process was removed from the database and then reconstructed using the NPDA software. This "machine assisted reconstruction" enables us to objectively measure the method's effectiveness.

Each experiment was conducted according to the following steps: (a) preparation: remove one of the processes from the database so that the database will not contain any of its descriptor components; (b) run the NPDA in a stepwise manner. At each phase we try to identify an activity ("goal activity") that is compatible with the removed process, according to the following steps: (1) if the goal activity's linguistic components are represented in the Process Repository Database, run the "find next activity" algorithm (see Section 5.3). If the output list contains the goal activity - continue to reconstruct the next goal activity. Else, run the "activity refinement" algorithm (see Section 5.2). If the option list produced by the refinement step does not include the goal activity, choose the activity that shares the largest amount of common descriptor components with the goal activity as a basis for an additional refinement. If, after 10 successive refinements, the required activity is still not represented by one of the output options, it is inserted manually as the next process activity and the design process is continued by locating the next activity; (2) else (the goal activity's linguistic components are not represented in the Process Repository Database), the next goal activity is inputted manually by the experimenter.

[4] http://www.oracle.com/applications/tutor/index.html.

Results and Analysis Table 1 presents a summary of the experiment results. Each experiment of creating a new process model was based on a database with the set of all activity descriptors in all process models, excluding the set of activity descriptors of one goal process. This means that we aim at recreating the goal activities from a partial set of activity descriptors. On average, for 89% of the goal activities, all descriptor components were contained both in the goal process and in another process (see column #3). This was the case despite the relatively small experiment size (13 processes, whereas the entire OBM includes around 1,500 processes), highlighting the amount of similarity one would expect when designing new processes based on an existing repository. For the remaining 11%, at least one descriptor component was missing. In such a case, the activity was inserted manually during the design process. It is worth noting that for the 89% of activities that had the potential of reconstruction from the database, 100% were reconstructed successfully using our method (see Table 2).

Table 1. Experiment results.

Column #	1	2	3	4	5	6	7
Column name	# of total processes in DB	# of total activities in DB	% of goal activities represented in the DB	Avg. # of steps per design phase	Avg. location of correct option in 'next activity'	Avg. location of correct option in 'refine activity'	Avg. location of the correct option per design phase
Avg.-all	14	127	89.0%	2.0	1.2	2.8	2.6
Avg.-Procurement	9	96	90.6%	1.9	0.8	3.0	2.8
Avg.-Inventory	5	31	83.9%	2.1	1.9	2.4	2.3

In addition, Table 1 shows that on average, two iterations are required for reconstructing a goal activity (see column #4). The design of Procurement processes required slightly less steps than the design of Inventory processes (1.9 vs. 2.1 steps on average, respectively). It should be noted that the location of the goal activity was very high in the ranked list of suggested activities (average location: 2.6, see column #7). This location was even higher at phases that did not involve refinement (average location: 1.2, see column #5); and was a little lower in steps in which a refinement was required (2.8 on average, see column #6). This may be due to the fact that refinement steps include a much larger amount of alternatives. Again it should be noted that results within the Procurement category were better than results within the Inventory category - probably due to the larger database representing Procurement processes. Another reason may be the consecutive nature of procurement processes vs. the loosely coupled business logic of the Inventory processes.

Table 2 analyzes the number of refinements that are needed to design the correct goal activity. For each number of refinements, we record the percentage of cases where this number of refinements was needed. We also record, for each

Table 2. Distribution of successful predictions vs. the number of required refinements.

# of refinements	0	1	2	3	4	5	6	7	8	9
% of successful predictions	12%	35%	27%	12%	4%	2%	2%	1%	1%	3%
Cumulative	12%	48%	75%	88%	92%	94%	96%	96%	97%	100%

number of refinement i, the cumulative percentage of cases where up to i refinements were needed. We observe, for example, that in 88% of the cases the system can reconstruct the goal activity after a maximum of three refinements. These results clearly demonstrate the speed and efficiency of the suggested method. Moreover, in all experiments the refinement process converged into a maximal number of nine refinements in the worst case. As hypothesized earlier- a larger database would probably yield even better results.

To summarize, we have shown the usefulness of using a descriptor repository in identifying activities for a new business process. We also showed the method to be effective in the given experimental setup, both in terms of the number of design steps and in the number of refinements that are needed.

7 Conclusions

We proposed a mechanism to automate the reuse of constructs gathered from predefined process models. Such a mechanism saves design time and supports non-expert designers in creating new business process models. The proposed method, software tool, and experiments provide a starting point that can already be applied in real-life scenarios, yet several research issues remain open, including: (1) an extended empirical study to further examine the quality of newly generated processes; (2) an extended activity decomposition model to include an elaborated set of business data and logic (*e.g.*, roles and resources); and (3) defining a learning mechanism that will take into account previous designer preferences and adjusting (in real time) the process delineator mechanism.

As a future work we intend to investigate further language semantics by using more advanced natural language processing techniques, as well as semantic distances between words. Finally, we intend to apply the techniques we have developed to create new methods for workflow validation.

Acknowledgments

Many thanks to Samia Mazhar and the BPM Group at QUT for providing access to the aviation business process data. We also thank Roman Kushnarenko for his assistance with the experiments.

References

1. K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. *Lecture Notes in Computer Science*, 4714:288, 2007.

2. WM Coalition. The workflow management coalition specification - terminology & glossary. Technical report, Technical Report WFMC-TC-1011, Workflow Management Coalition, 1999.
3. M. Golani and S.S. Pinter. Generating a process model from a process audit log. *Lecture notes in computer science*, pages 136–151, 2003.
4. T. Gschwind, J. Koehler, and J. Wong. Applying patterns during business process modeling. In *BPM*, volume 5240, pages 4–19. Springer, 2008.
5. C. Hofmeister, P. Kruchten, R.L. Nord, H. Obbink, A. Ran, and P. America. A general model of software architecture design derived from five industrial approaches. *The Journal of Systems & Software*, 80(1):106–126, 2007.
6. R. Hull. Artifact-centric business process models: Brief survey of research results and challenges. *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008*.
7. S. Kumaran, R. Liu, and F.Y. Wu. On the duality of information-centric and activity-centric models of business processes. *Lecture Notes in Computer Science*, 5074:32–47, 2008.
8. S. Kumaran and P. Nandi. Adaptive business object: a new component model for business applications. *white paper, IBM TJ Watson Research Center, http://www. research. ibm. com/people/p/prabir/ABO. pdf*.
9. M. Lincoln, R. Karni, and A. Wasser. A Framework for Ontological Standardization of Business Process Content. *International Conference on Enterprise Information Systems*, pages 257–263, 2007.
10. D. Muller, M. Reichert, and J. Herbst. Data-driven modeling and coordination of large process structures. *Lecture Notes in Computer Science*, 4803:131, 2007.
11. A. Nigam and N.S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.
12. H.A. Reijers, S. Limam, and W.M.P. Van Der Aalst. Product-based workflow design. *Journal of Management Information Systems*, 20(1):229–262, 2003.
13. G. Schimm. Process miner - a tool for mining process schemes from event-based data. In *JELIA '02: Proceedings of the European Conference on Logics in Artificial Intelligence*, pages 525–528, London, UK, 2002. Springer-Verlag.
14. H. Schonenberg, B. Weber, B.F. van Dongen, and W.M.P. van der Aalst. Supporting flexible processes through recommendations based on history. In *International Conference on Business Process Management (BPM 2008)*, volume 5240, pages 51–66. Springer, 2008.
15. WMP Van der Aalst, P. Barthelmess, CA Eliis, and J. Wainer. Proclets: A framework for lightweight interacting workflow processes. *International Journal of Cooperative Information Systems*, 10(4):443–482, 2001.
16. W.M.P. van der Aalst and AHM Ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.
17. W.M.P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.
18. K. Wahler and J.M. Kuster. Predicting Coupling of Object-Centric Business Process Implementations. In *Proceedings of the 6th International Conference on Business Process Management*, page 163. Springer, 2008.