Journal of Universal Computer Science, vol. 19, no. 13 (2013), 1892-1913
submitted: 5/11/12, accepted: 28/6/13, appeared: 1/7/13 © *J.UCS*

Business Process Management Applications based on Semantic Process Models: the ProcessGene Suite Case-Study

Avi Wasser
(University of Haifa, Israel
awasser@research.haifa.ac.il)

Maya Lincoln
(ProcessGene Ltd, Haifa, Israel
maya.lincoln@processgene.com)

Abstract In recent years, Business Process Management (BPM) applications have become central enablers for the generation, customization and utilization of business processes within and between organizations. One of the central techniques for extending the span of BPM applications is Natural Language Processing (NLP). This work aggregates and reviews previous works on NLP standardization for BPM. Based on these works, we present a set of BPM applications, aiming at extending the utilization of the knowledge embedded in business process repositories. To verify the industrial deployment, we then present an extended case study that examines the feasibility of the suggested applications in real life scenarios using the ProcessGene BPM suite.

Keywords: BPM applications, Business process model standardization, Business process repositories, Natural language processing.
Category: H.4

1 Introduction

Business process repositories are considered an important resource of organizational knowledge. These repositories facilitate visibility into the business of organizations, and therefore have a central role in enterprise analysis, strategy, and Information Technology (IT) efforts [see Krumbholz and Maiden 2001]. Therefore, in recent years, researchers have become increasingly interested in developing methods and tools for promoting the utilization of process repositories, and the topic has been discussed intensively both in academia and in industry [see Yan et al. 2012]. This work is aimed at: (1) presenting methods for enabling the utilization of such process repositories for business process management applications; and (2) validating the applicability of the presented methods in real-life scenarios.

Our work presents a content-based utilization framework that relies on the standardization of the content layer of business process repositories, as a basis for enabling several applications that leverage the usage of these knowledge reservoirs.

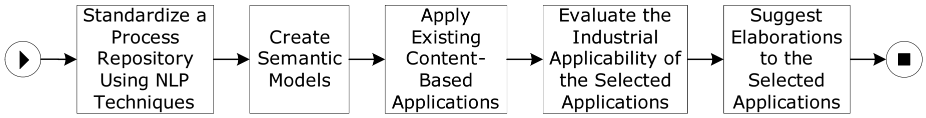

Figure 1: A high-level framework for utilizing standardized process repositories for content-based applications.

We propose a five-step meta-process to utilize process repositories, as illustrated in [Fig. 1] (using "Yet Another Workflow Language" (YAWL) [see van der Aalst and Ter Hofstede 2005]). First, we apply previously suggested methods for creating an operationally meaningful decomposition of a process repository. We use state-of-the-art Natural Language Processing (NLP) techniques to automatically decompose the content layer (text) of the repositories into its structured linguistic components (objects and actions and their qualifiers). As part of this decomposition, each business activity is encoded automatically as a *descriptor*, using the Process Descriptor Catalog ("PDC") notation [see Lincoln et al. 2007]. 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. Second, by analyzing the generated decomposition, we create seven action and object models, that represent operational aspects of the process repository, as suggested in [Lincoln et al. 2010a] and [Lincoln and Gal 2011b]. Third, we present applications from other works that use action and object models for solving problems related to the utilization of the process repository in the following domains: (1) design of new process models; (2) validation of changes in the repository; (3) search of process segments in the repository, (4) similarity measurement between process models; and (5) construction of process data ontologies, as a basis for further understanding the logic of process models. As a fourth step, we evaluate the industrial applicability of the selected applications by conducting a case-study and experiments based on a real-life process repository. Finally, we discuss how these applications can be extended and improved for better utilization of the process repositories by (1) deploying a larger set of semantic models; and (2) integrating complementing applications.

The suggested framework is demonstrated using a process repository that consists 31 real-life processes and 183 related activities from the software development industry. The case study and the experiments are conducted using an extension we developed to an industrial BPM software - the ProcessGene BPM Suite [see ProcessGene 2013].

The paper is innovative in the following ways: (1) it combines several process utilization methods under a common framework; (2) it suggests extending state-of-the art process utilization methods - based on other methods that operate in the same functional area; and (3) it examines the applicability of the different methods using the same process repository - and by that eliminating repository-specific biases.

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 activity decomposition model that is gathered from previous works, and is used in this work as the foundation for creating action and object taxonomies. We then present applications from previous works that currently rely on some parts of the standardized repository in [Section 4]. [Section 5] introduces the software tool and our empirical analysis, including discussion on how the applications can be extended. We conclude in [Section 6].

2 Related Work

Research on standardization and analysis of the content layer of business process models mainly focuses 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 [see Wahler and Kuster 2008, Pardo et al. 2012]. 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 [see Hull 2008, Cai et al. 2012]. Such object-centric approaches include artifact-centric modeling [see Nigam and Caswell 2003, Bhattacharya et al. 2007], adaptive business objects [see Nandi and Kumaran 2005], data-driven modeling [see Muller et al. 2007] and proclets [see Van der Aalst et al. 2001]. Further analysis of the object-centric model in terms of computing the expected coupling of object lifecycle components is presented in [Wahler and Kuster 2008].

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 [Kumaran et al. 2008] presents an algorithm that generates an information-centric process model from an activity-centric model. The works in [Lincoln et al. 2007, Lincoln et al. 2010a, Lincoln and Gal 2011b] present the concept of business process descriptor that decomposes process names into objects, actions and qualifiers, and suggest several taxonomies to express the operational knowledge encapsulated in business process repositories. In this work we take this model forward by:

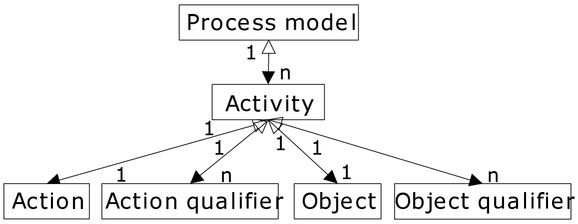

Figure 2: The activity decomposition model.

(a) testing it on real-life processes from the software development domain; (b) showing how the suggested taxonomies can assist in common usages of business process management (c) industrializing the approach using a real-life business process management (BPM) software suite.

3 The Activity Decomposition Model

This section describes a formal model of business process decomposition and analysis, gathered from previous works. We first introduce the descriptor model [see Section 3.1]. Then, based on the descriptor model, we introduce seven taxonomies of objects and actions [see Section3.2]. To illustrate the taxonomies we make use of the software development repository [see Section 1].

3.1 The Descriptor Model

In the Process Descriptor Catalog model ("PDC") [see Lincoln et al. 2007] 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]. Qualifiers provide an additional description to actions and objects. State-of the art Natural Language Processing (NLP) systems, *e.g.*, the "Stanford Parser" [see Stanford 2013] can be used to automatically decompose process and activity names into *process/activity descriptors*. For example, the activity "Develop requested functionality" generates an activity descriptor containing the action "develop," without an action qualifier, the object "functionality" and the object qualifier "requested."

3.2 Action and Object Based Taxonomies

The descriptor model has two basic elements, namely objects and actions, and it serves as a basis for several state of the art taxonomies, as follows: (1) in [Lincoln et al. 2010b] it was enhanced to create the *action hierarchy model*, the

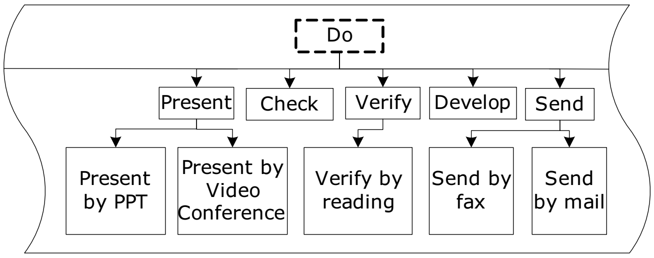

Figure 3: Segment of an action hierarchy model from the software development repository.

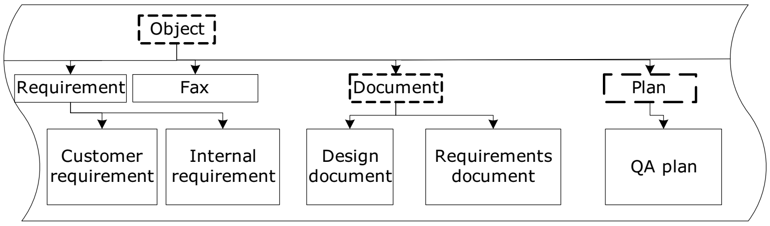

Figure 4: Segment of an object hierarchy model from the software development repository.

object hierarchy model, and the *action sequence model*, and the *object lifecycle model*; and (2) in [Lincoln and Gal 2011b] it was enhanced to create the *action scope model*, the *object grouping model*, and the *action influence model*.

3.2.1 The Action and Object Hierarchy Models

The action and object hierarchy models organize a set of activity descriptors according to the hierarchical relationships among business actions and objects, respectively. This hierarchical dimension of actions and objects is determined by their qualifiers [see illustrations in Fig. 3 and Fig. 4]. For example, consider the complete action "Send by fax." It is a subclass (a more specific form) of "Send" in the action hierarchy model, since the qualifier "By fax" limits the action of "Send" 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 object "Plan" was not represented without qualifiers in the software development process repository, and was completed from the more detailed object: "QA plan" by removing its object qualifier ("QA") [see Fig. 4].

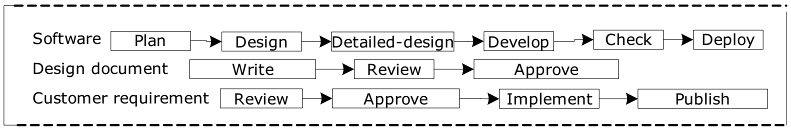

Figure 5: Segment of an action sequence model from the software development repository.

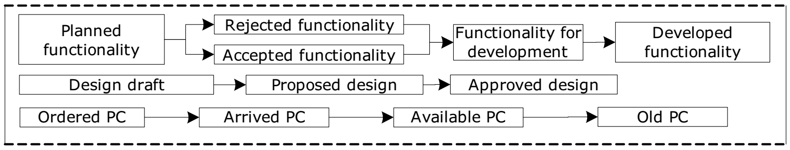

Figure 6: Segment of an object lifecycle model from the software development repository.

3.2.2 The Action Sequence Model

The action sequence taxonomy model organizes a set of activity descriptors according to the relationships among business actions and objects in terms of execution order. In this model, each object holds a graph of ordered actions that are applied to that object. A segment of the action sequence model of a software development repository is presented in [Fig. 5]. For example, the object "Student" is related to the following action sequence: "Interview" followed by "Accept," "Sign," and finally "Update."

3.2.3 The Object Lifecycle Model

The object lifecycle taxonomy model organizes a set of activity descriptors according to the relationships among business actions and objects in terms of execution order. For example, the object "Functionality" is part of the following object lifecycle: "Planned functionality"–> "Rejected/Accepted functionality"–>"functionality for development"–> "Developed functionality."

3.2.4 The Action Scope Model

The action scope model represents the relationship between an action within a process name (a "primary action") and the actions in its corresponding process model. The fact that a process repository consists of pre-defined process models is being used for learning about the scope of actions in the following way. Each primary action in the repository is related with a set of directional graphs of

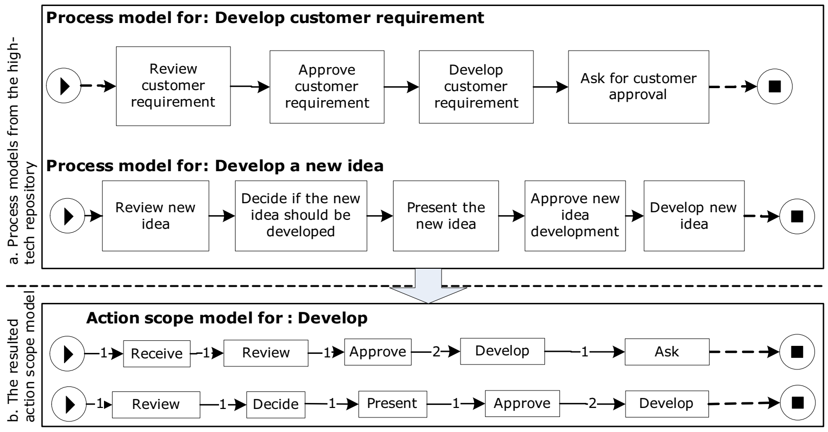

Figure 7: A segment of the action sequence model for the action "Develop."

actions that represent the order of actions within this primary action's segments. Since such a primary action can be part of more than one primary names, and since the same complete action may be represented more than once in the same process model segment - each edge in the action scope model is labeled with its weight, calculated by the number of its repetitions in the related process model segments. Graph splits are also represented in the action scope model.

Consider the following two processes from the software development repository: "Develop customer requirement" and "Develop a new idea." These processes are illustrated in [Fig. 7a]. Using these two process models, it is possible to generate an action scope model for the action "Develop" [see Fig. 7b]. According to this example, there are two optional action paths compatible to the "Develop" action starting by either "Receive" or "Review." Since "Develop" follows "Approve" twice in this model, the respective edge weight is set to 2.

3.2.5 The Object Grouping Model

The object grouping model represents the relationship between a primary object and the objects in its corresponding model segments. Since such a primary object can be part of more than one primary process segment, and since the same object may be represented more than once in the same process model segment - each object in the object grouping model is labeled with its weight calculated by the number of its repetitions in the related process model segments.

To illustrate, consider two processes from the software development repository: "Check software" and "Develop software." These processes are represented by corresponding graph segments as illustrated in [Fig. 8a]. Using these two pro-

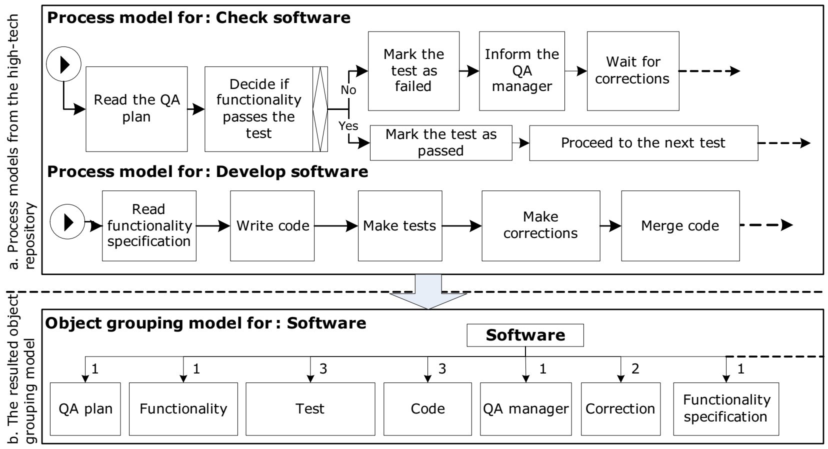

Figure 8: A segment of the object grouping model for "Software."

cess models, it is possible to generate an object grouping model for the object "Software," as illustrated in [Fig. 8b].

3.2.6 The Action Influence Model

An action influence model represents the relationship between a primary action and the flow of states (object qualifiers) of the primary object in model segments that correspond to the primary action. Each edge in the action influence model is labeled with its weight representing the number of its repetitions in the related process model segments.

To illustrate, consider the two process models named: "Plan development cycle" and "Plan project." They both deal with *plan,* but focus on different objects [see illustration in Fig. 9a]. By following changes to the qualifiers of the primary object in these process models we end up with the action influence model for "Plan" as illustrated in [Fig. 9b].

4 Applications

In [Section 2] we showed how the content layer of business process repositories can be standardized using the activity decomposition model. In this section we show how such standardized process repositories can be utilized for several applications in the domain of business process management. The presented applications are based on previous works that applied semantic analysis of standardized business process repositories, using natural language processing techniques.

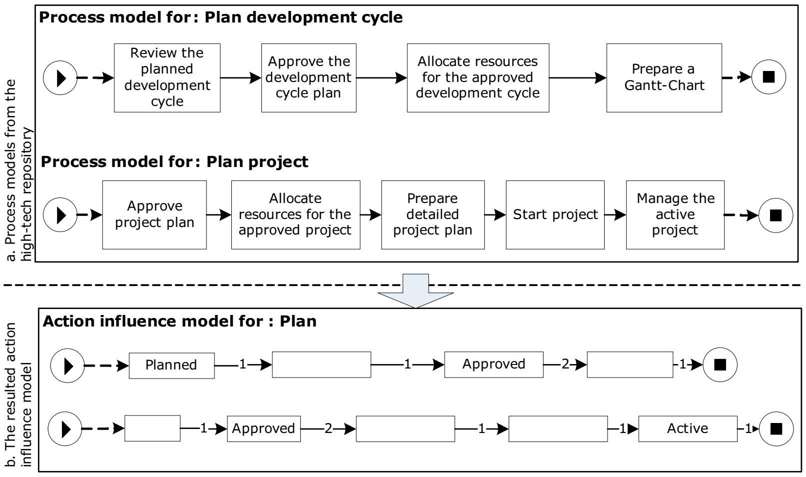

Figure 9: A segment of the action influence model for the action "Plan."

4.1 Machine-Assisted Design of Business Process Models

The work in [Lincoln et al. 2010a] suggests a generic method for designing new business process models related to any functional domain. Modeling is considered a manual, labor intensive task, whose outcome depends on personal domain expertise with errors or inconsistencies that may lead to bad process performance and high process costs [see Muller et al. 2007]. Hence, automating the reuse of constructs, gathered from predefined repositories does not only save design time but also supports non-expert designers in creating new business process models.

The *process delineator* is a stepwise method aimed at supporting the task of new process model design. The 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. To do that, it relies on an underlying process descriptor space and at any phase it either refines an existing process activity or suggests a next process activity.

The business logic for such suggestions is extracted from the following four models: the action and object hierarchy model, the action sequence model and the object lifecycle model. Each activity is encoded automatically as a *descriptor*, using the "PDC" notation. 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.

The *process delineator* was further elaborated in [Wasser and Lincoln 2012b] by adding semantic learning capabilities that opt to improve the quality of generated business process models. The learning mechanism analyzes, in real-time,

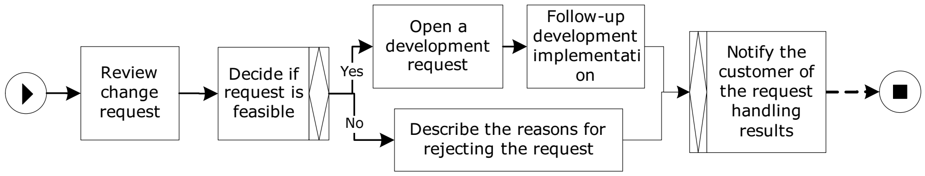

Figure 10: A search result for "how to handle a customer change request."

the linguistic relationships between process descriptors and adjusts them according to human inputs that are accumulated during the modeling process.

4.2 Content-Based Validation of Business Process Modifications

The work in [Lincoln and Gal 2011a] presents a content-based validation framework that uses organizational standards to evaluate the correctness of both newly designed and modified processes. A unique feature of the proposed framework, in the context of process validation, is the creation of a repository that captures organizational standards by using natural language processing analysis to capture simultaneously action and object patterns. The paper's contribution to the compliance domain is in the *dynamic* construction and adjustment of patterns - avoiding the need to design and maintain external, static rules.

The authors propose to *automatically* extract business logic from process repositories using the PDC model, and the taxonomy models of action sequence, object lifecycle, and object and action hierarchies that support the validation process. The proposed method includes three steps for content-based validation: (1) deficiency identification (using existing descriptors as a benchmark reference), (2) validation score calculation, and (3) generation of a ranked list of possible corrections.

4.3 Searching Business Process Repositories Using Operational Similarity

The search framework proposed in [Lincoln and Gal 2011b] receives natural language queries and returns a ranked list of related *process models*. The business logic is extracted from process repositories through the analysis of the following three taxonomies: the action scope model, the object grouping model, and the action influence model [see Section 3]. The proposed method *dynamically* segments a process repository according to the ad-hoc request as expressed in the user's search phrase.

As an example for clarifying the application's usability, consider an employee interested in finding out "how to handle a customer change request." An expected outcome of this retrieval request would be a *segment* from the process repository that represents the order of activities that one should follow in order to achieve the required process goal, as illustrated in [Fig. 10]. The benefit of such a retrieval framework is that the result is ready for execution.

The retrieval output is related to the search phrase in *operational* terms. For example, [Fig. 10] provides a segment that is only marginally similar to the search phrase text. Specifically, one of the search phrase terms ("Handle") is not represented by any of its activities while the terms "Change" and "Customer" are represented by only one activity.

The above search framework was also found useful in supporting BPM related decisions [see Wasser and Lincoln 2012a].

4.4 Measuring Similarity Between Process Models

The work in [van Dongen et al. 2008] proposes a method for measuring the similarity between process models using semantic analysis. Similarity in this paper is based on textual proximity of process components. The framework is aimed at eliminating redundancies in process repositories and enabling an efficient comparison of process models. The framework is highly useful for large process repositories, in which manual detection of redundancies is a highly complicated task.

The framework includes calculation of the degree of behavioral similarity between process models. This metric is measured using causal footprints as an abstract representation of the behavior captured by a process model.

4.5 An Automatic Construction of Process Data Ontologies

Some works focus on automatic construction of process data ontologies, as a basis for further understanding the logic of process models. Several works have been suggested, each focuses on extracting different aspects of the business conduct as encoded in the process repository. The work in [Belhajjame and Brambilla 2009] proposes a query-by-example approach that relies on ontological description of business processes, activities, and their relationships, which can be automatically built from the workflow models themselves. The work in [Bozzon et al. 2010] automatically extracts the semantics from searched conceptual models, without requiring manual meta-data annotation, while basing its method on a model-independent framework.

5 Implementation and Case Study

5.1 Implementation

We have developed an extension to the ProcessGene BPM Suite that: (1) automatically decomposes process and activity names into process descriptors [see

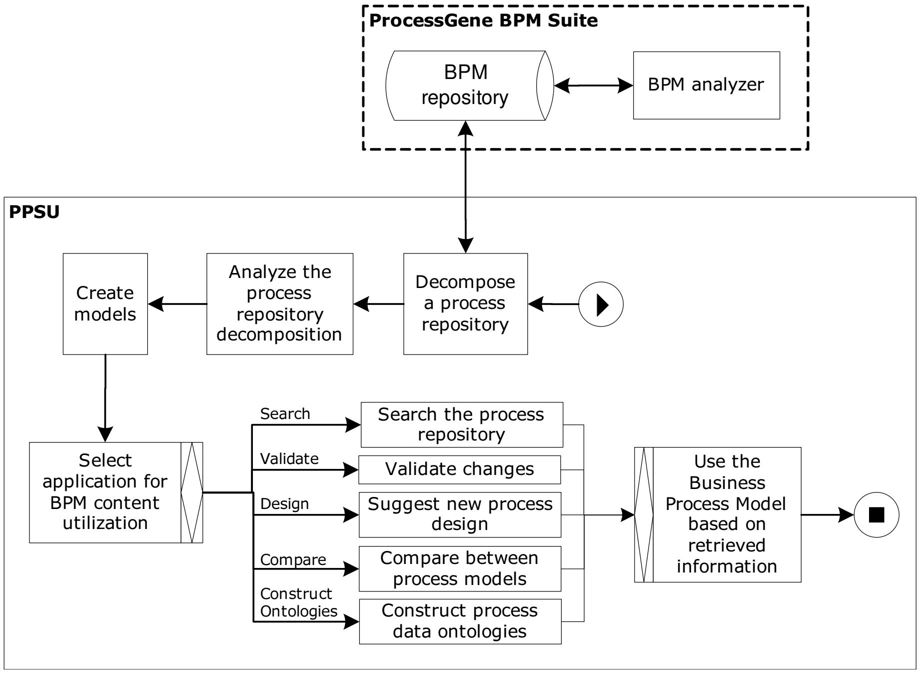

Figure 11: An illustration of the ProcessGene Process Semantics Utilizer.

Section 3.1]; (2) automatically builds the seven action and object Taxonomies [see Section 3.2]; and (3) implements the five BPM applications for improving the utilization of business process repositories [see Section 4]. We named this extension: "ProcessGene Process Semantics Utilizer" (PPSU), as illustrated in [Fig. 11] (using "Yet Another Workflow Language" (YAWL) [see van der Aalst and Ter Hofstede 2005]). Given the ProcessGene BPM analyzer, and based on an existing process repository, the PPSU guides users in utilizing the knowledge embedded in the process repository.

The PPSU utilization process is composed of six steps, as follows. First, we create an operationally meaningful decomposition of a process repository. We use state-of-the-art Natural Language Processing (NLP) techniques to automatically decompose the content layer (text) of the repository into its structured linguistic components (objects and actions and their qualifiers). As part of this decomposition, each business activity is encoded automatically as a *descriptor*, using the Process Descriptor Catalog ("PDC") notation [see Section 3.1].

Second, we analyze the generated decomposition and third, we create seven action and object models, that represent operational aspects of the process repository [see Section 3.2].

As a fourth step, the user decides which application is required for his current BPM utilization needs. The BPM tools include the following applications: (1) design of a new process model; (2) validation of changes made to an existing process repository; (3) search for process models using natural language queries; (4) comparison between two process models; and (5) construction of process data ontologies - for further analysis and understanding of the repository content. In the fifth step we use the above action and object models for providing the required assistance. Although each application is different they all share the following steps. Utilization starts with a user request phrased in natural language. Then, this input is decomposed into linguistic components. As a result of this phase both the user intention (input) and the methodology's underlying knowledge (the process repository) are represented in a common language. The next phase includes an analysis of the process repository and the user input aimed at fulfilling the specific goal. As a result, a set of solution suggestions is generated and then ranked according to the user's input, so that higher ranked suggestions are believed to be closer to the user needs. The specific methods for new process model generation, validation, search, comparison, and ontologies construction are detailed in [Sections 4.1, 4.2, 4.3, 4.4, and 4.5] correspondingly. Finally, at the sixth step the user receives results and supporting information and conducts his usage and decisions accordingly.

The PPSU system implements a client-server architecture. Server side logic is implemented in PHP using a MySql database. PPSU uses a Natural Language (NL) parser - the "Stanford Parser" - as a web service for decomposing sentences into linguistic components. The client runs within an Internet browser and is implemented in HTML and JavaScript, with AJAX calls to the server.

5.2 Case Study

To verify industrial deployment, we present in this section an extended case study that examines the feasibility of the above presented applications in real life scenarios. The case-study was carried out using the ProcessGene BPM suite professional edition [see Section 11], and based on the software development industry business process repository (see Chapter 1).

5.2.1 An Example for Designing a New Process Model

To examine the applicability of the method for new process model design [see Section 4.1] in real-life scenarios, we present two examples, as follows. The software development process repository covers activities starting from the customer requirement processing, through the implementation and customization of new software modules and terminating as the software is approved by the customer and brought into production. The newly designed processes are related to the

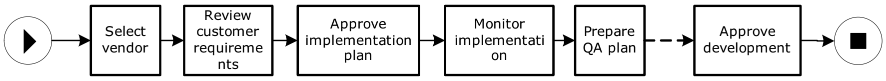

Figure 12: The new designed process diagram for "Outsource software development."

software development domain, but are not covered by the actual process repository. The first new process, "Outsource software development," extends the process repository by outsourcing the software development activities, instead of executing in-house development. The second new process, "Create automatic tests" extends the process repository by adding a new process for automatic software-testing.

The first example supports the design of a new business process for: "Outsource software development." The generated output (new process model) of this example is illustrated in [Fig. 12] as a YAWL (Yet Another Workflow Language) diagram. The design process starts when the (human) process designer inserts the following process descriptor: (action="outsource", action qualifier=null, object="development", object qualifier="software") to the process assistant and determines that the first activity is: "Select vendor." Respectively, the process assistant searches the descriptor space, looking for next activity possibilities. Note that the object "Vendor" already appears in the repository, in processes related to service and equipment suppliers. The result set includes the following activities: "[1] Send order form," "[2] Review requirements" and "[2,~] Notify vendor." The designer selects the option "Review requirements" and decides to refine it. After one refinement the activities "Review customer requirements" and "Approve requirements" are proposed by the process designer application, and the user selects the first option. In the next design phase after three refinements the activity "Approve implementation plan" is selected. As a result, the process designer suggests an option list for the next activity that starts with the option: "[1] Monitor actual implementation." This first option is selected for refinement and as a result an option list is suggested, containing the option: "[1] Monitor implementation." This option is selected as a valid option in the process model. After eleven additional design phases, the process design application reaches the phase of approving the outsourced development (last activity). In this case the activity "Sign development" was selected and after one refinement the activity "Approve development" is reached.

The designer is now interested to design the new business process: "Create automatic tests." The design process is conducted in a similar manner to the one presented above and results in the process diagram presented in [Fig. 13]. An interesting observation related to this design process is the low number of

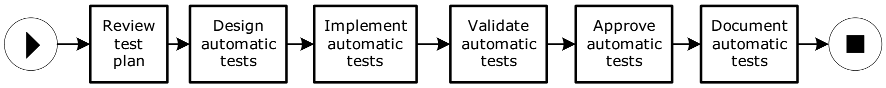

Figure 13: The new designed process diagram for "Create automatic tests."

refinements (three) during the entire design process. The first suggested activity required one refinement and the second designed activity required two refinements. Afterwards, once the design process reached the object "Automatic tests," the next correct suggested activities were proposed as part of the first suggested option list at each phase. The reason for such relatively high accuracy steams from the fact that the rest of the newly designed process is similar to the software development process (that already exists in the repository), having the same actions that act upon different objects ("Automatic tests" instead of "Software").

In summary, the case study demonstrated that it is possible to apply the process design application in a real-life scenario. On average, 2.1 iterations are required for reconstructing the proper activity at each design phase. The design of the second process model required even less steps than the design of Inventory processes (1.7 vs. 2.6 steps on average, respectively). It should be noted that the location of the correct activity was very high in the ranked list of suggested activities (average location: 2.3). This location was even higher at phases that did not involve refinement (average location: 1.4); and was a little lower in steps in which a refinement was required (3.1 on average). This may be due to the fact that refinement steps include a much larger amount of alternatives.

The results indicate the usefulness of using a descriptor repository in identifying activities for a new business process. We also demonstrated that the method is effective in the given real-life scenario, both in terms of the number of design steps and in the number of refinements that are needed.

Finally, it is possible to elaborate this work, by calculating descriptor space distances also using the three additional models: the action scope model, the object grouping model, and the action influence model [Sections 3.2.4,3.2.5, and 3.2.6, correspondingly]. In addition, it is may be possible to integrate the process model search method proposed in [Lincoln and Gal 2011b] in order to generate model segment suggestions at each design phase, instead of single activity suggestions.

5.2.2 An Example for Validating Business Process Modifications

To verify applicability in real-life scenarios, we chose a set of three real-life processes from the software development repository, comprising 34 activities altogether. From these three processes we created a *database* of process descriptors and generated their object and action taxonomies [see Section 3].

To evaluate the suggested method we used the evaluation framework suggested in [Lincoln and Gal 2011a]. Accordingly, we carried out three experiments, each one 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) find the last activity in the predecessor process of the removed process (*anchor activity*) - as was defined in the original database (before the selected process was removed); (c) run the *validator application (VA)* in a stepwise manner. At each phase we: (i) add an activity (*new activity*) from the removed process to the existing process repository - so that it follows the anchor activity. Then, the *VA* validates this *add* change and outputs its validation results for each error type it identifies; (ii) delete that same activity from the process repository and validate this *delete* change using the *VA*. In case no suggestion is generated for the examined change, a penalty score of 30 is given.

For example, consider an experiment where the process "Outsource software development" [see Fig. 12] is removed from the database. At first, we try to add the first activity: "Select vendor" to the process repository, without the process "Outsource software development". Then, we call the *VA* and evaluate the validity of this *add* change. We then delete "Select vendor" from the process repository (after recording it in the repository) and again, invoke the *VA* for validating the *delete* change. In a similar manner, in the following seven sub-experiments (one sub-experiment per activity), we add a different activity from the examined process as a *new* activity and then delete it from this model, creating a *delete* change. Note that after each sub-experiment the process model returns to its original form (a database without the removed process) so that the sub-experiments are independent and their execution order is immaterial.

[Tab. 1] presents a summary of the experiment results. As the changed activity (the suggested new activity or the deleted activity) is further away from the anchor activity, the lowest (best) score (LS) of the proposed change in the suggested corrections list increases, meaning that the error requires a larger number of correction steps. In addition, the position of the correct activity (PCA) in the correction suggestion list for add changes was higher as the new activity was closer to the anchor activity in the original repository. In other words, we found that as a newly suggested (added/deleted) activity is less adequate in the existing process model (in terms of distance from the anchor activity), the suggested corrections require more changes in the original descriptor space and the suggestion for the correct activity is ranked lower in the suggestion list.

The sharp increase in *LS-Add* and *LS-Delete* that occurred when the distance from the anchor activity grew from 5 to 6 (an increase from 5.5 to 9.8) and from 6 to 7 (an increase from 6.7 to 11.3), respectively [see Tab. 1], can be explained

Table 1: Experiment results.

Distance from the anchor activity	0	1	2	3	4	5	6	7	8
LS-Add (avg.)	3.1	3.5	4	4.2	4.6	5.5	9.8	12.3	14.7
LS-Delete (avg.)	3.2	3.7	4.0	4.5	5	5.1	6.7	11.3	11.9
PCA-Add (avg.)	1.6	1.8	2.2	3.0	3.9	4.3	5.9	10.6	21.8

by the usage of the penalty score. Similarly, this penalty score also explains the sharp increase in *PCA-Add*, that occurred when the distance from the anchor activity grew from 6 to 7 (an increase from 5.9 to 10.6).

To summarize, the experiments show the usefulness of using the process validation application presented in [Lincoln and Gal 2011a] for validating changes in an existing business process repository. The application was found to be effective for the given software development repository, both in terms of the correction suggestions it produces and in its ranking mechanism.

Similarly to the former method, we suggest elaborating the validation application, by referring also to the three additional models: the action scope model, the object grouping model, and the action influence model [Sections 3.2.4,3.2.5, and 3.2.6, correspondingly]. In addition, we suggest to integrate the method proposed in [van Dongen et al. 2008] for measuring the similarity between process models using semantic analysis. This method may assist in ranking the generated correction suggestions, by comparing the suggestions to the original user intention (the change he made to the process repository).

5.2.3 An Example for Searching Business Process Repositories Using Operational Similarity

To evaluate the usability of the process model search method in real-life scenarios, we followed the evaluation framework suggested in [Lincoln and Gal 2011b]. Accordingly, we chose a set of 11 real-life processes from the software development repository, comprising 69 activities altogether. Using these processes we created a "process repository database," that includes the 11 process models and their derived taxonomies.

To evaluate the suggested method we conducted 11 experiments. At each experiment, a single process was removed from the database and then was searched according to its name. More specifically, each experiment was conducted according to the following steps: (a) preparation: removal of one of the process names from the database and reconstruction of the taxonomies so that the database will contain the process model (its graph segment) but the three taxonomies will not contain any of its descriptor components; (b) search for the removed process in the database, using its name as the search phrase.

In order to calculate each search result ranking score, we used the similarity method described in [van Dongen et al. 2008] [see Section 4.4]. In addition, the threshold parameters for sifting irrelevant result candidates (defined in [Lincoln and Gal 2011b]) were set to: $th^{OGM} = 0.2$, $th^{AIM} = 0.3$.

The search results have then been objectively evaluated with respect to the original, removed, process. For each of the 11 experiments we also chose the "best result," the one most similar to the goal process model, calculated using the similarity method presented in [van Dongen et al. 2008]. The metrics for measuring the effectiveness of the method, as detailed below, assess both the quality of an average result, as well as the average quality of the best result in each experiment - showing the best performance of the search method.

We used the five metrics proposed in [Lincoln and Gal 2011b] to measure the effectiveness of the method. For all results and for the best result we computed: (1) the average percentage of correct activities, showing how similar the results are with respect to the goal, correct result; (2) the average percentage of redundant activities, showing the amount of redundant, unnecessary activities in the retrieved results. Finally, we identified the average location of the best result in the list of search results - showing the effectiveness of the grading mechanism, that aims at locating the best result at the beginning of the result list.

The experiment results are as follows. On average, each result contained 82.5% of the goal process model activities. On average, the best result has a higher overlap of 87.9% with the goal process model and was ranked in a high location in the list of results (1.6), usually in the first place. In addition to the correct activities, the results also contained, on average, 11% of redundant activities out of the goal process model, while this percentage was lower (6.4%) in the best result.

These experiments have shown the usefulness of using the process model search application in searching for business process models based on their operational meaning. We also showed the method to be effective in the given software development repository, both in terms of the similarity between the results and the goal result and with respect to the ranking of the best result.

We propose elaborating the method's segmentation process by referring also to execution flows as expressed by the action sequence model and the object lifecycle model [Sections 3.2.2 and 3.2.3, correspondingly]. In addition, an integration of the query-by-example approach proposed in [Belhajjame and Brambilla 2009] may extend the returned result range and may produce better search results.

5.2.4 An Example for Measuring Similarity Between Process Models

We used the method for measuring similarity between process models [see Section 4.4] as part of the search method experiments [see Section 5.2.3], in order to determine the ranking of search result options. Since the search method experiments have shown the usefulness of the search method, and since the similarity method had a significant contribution to the effectiveness measures, we conclude that the similarity method is applicable and be effective in the real-life scenario, both in terms of determining similarity between process models and with respect to the ranking of the best result.

We suggest elaborating the similarity method proposed in [van Dongen et al. 2008] by referring also to *operational* similarity, as expressed in the activity decomposition model [see Section 3]. Distances in this case can be calculated in terms of proximity in the seven action and object taxonomies.

5.2.5 An Example for Automatic Construction of Process Data Ontologies

Measuring the usability of ontology construction for leveraging the usage and the understanding of business process repositories cannot be measured by machine, since it involves the opinion and input of (life) users. Such experiments are beyond the scope of this paper and we consider them as future work.

We suggest extending the above methods by targeting the automatic extraction and usage of the operational layer (the "how-to") and the business rules encapsulated in the process repository. These can be explored using the taxonomies presented in [Section 3].

6 Conclusions

In this paper we presented a framework for utilizing process repositories for business process management applications. The framework included an NLP analysis and standardization of the content layer of business process models as a basis for several such applications. We then showed how the suggested applications can be extended and improved, based on notions and techniques from other related applications. To verify the industrial deployment, we presented an extended case study that examines the feasibility of the suggested applications in real life scenarios using the ProcessGene BPM suite.

The paper contributes to the present literature in several aspects: (1) it combines several process utilization methods under a common framework; (2) it suggests improvements to state-of-the art process utilization methods; and (3) it provides a consistent case-study that examines the applicability of the different methods using the same process repository - and by that eliminates repository-specific biases.

The proposed framework provides a starting point that can already be applied in real-life scenarios, yet several research issues remain open. We mention three such extensions here. First, extending the list of relevant applications that can utilize the standardized model - beyond the scope that was presented in this paper. Second, adding a case study and experiments to measure the efficiency of the proposed improvements to the listed applications. Thirdly, conducting experiments to measure the usability of state-of-the-art ontology construction methods for leveraging the usage and the understanding of business process repositories- as well as to the structured extension of such repositories.

Acknowledgments

Many thanks to ProcessGene Ltd for providing access to ProcessGene BPM Suite, and to the ProcessGene best-practice process repository.

References

[Belhajjame and Brambilla 2009] Belhajjame, K., Brambilla, M.: Ontology-based description and discovery of business processes; Enterprise, Business-Process and Information Systems Modeling; (2009), 85-98.

[Bhattacharya et al. 2007] Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models; Lecture Notes in Computer Science; 4714 (2007), 288.

[Bozzon et al. 2010] Bozzon, A., Brambilla, M., Fraternali, P.: Searching repositories of web application models; Web Engineering; volume 6189 of Lecture Notes in Computer Science; 115; Springer Berlin / Heidelberg (2010).

[Cai et al. 2012] Cai, H., Xu, B., Bu, F.: A conceptual ontology-based resource meta-model towards business-driven information system implementation; Journal of Universal Computer Science; 18 (2012), 17, 2493-2513.

[Hull 2008] Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges; On the Move to Meaningful Internet Systems: OTM 2008; Springer (2008), 1152-1163.

[Krumbholz and Maiden 2001] Krumbholz, M., Maiden, N.: The implementation of enterprise resource planning packages in different organizational and national cultures; Information systems; 26 (2001), 3, 185-204.

[Kumaran et al. 2008] Kumaran, S., Liu, R., Wu, F.: On the duality of information-centric and activity-centric models of business processes; Lecture Notes in Computer Science; 5074 (2008), 32-47.

[Lincoln and Gal 2011a] Lincoln, M., Gal, A.: Content-based validation of business process modifications; ER 2011; (2011a), 495-503.

[Lincoln and Gal 2011b] Lincoln, M., Gal, A.: Searching business process repositories using operational similarity; On the Move to Meaningful Internet Systems: OTM 2011; (2011b), 2-19.

[Lincoln et al. 2010a] Lincoln, M., Golani, M., Gal, A.: Machine-assisted design of business process models using descriptor space analysis; Business Process Management; (2010a), 128-144.

[Lincoln et al. 2010b] Lincoln, M., Golani, M., Gal, A.: Machine-assisted design of business process models using descriptor space analysis; Technical Report IE/IS-2010-01; Technion (2010b); http://ie.technion.ac.il/tech_reports/1267736757_Machine- Assisted_Design_of_Business_Processes.pdf.

[Lincoln et al. 2007] Lincoln, M., Karni, R., Wasser, A.: A Framework for Ontological Standardization of Business Process Content; International Conference on Enterprise Information Systems; (2007), 257-263.

[Muller et al. 2007] Muller, D., Reichert, M., Herbst, J.: Data-driven modeling and coordination of large process structures; Lecture Notes in Computer Science; 4803 (2007), 131.

[Nandi and Kumaran 2005] Nandi, P., Kumaran, S.: Adaptive business objects-a new component model for business integration; Proc. Intl. Conf. on Enterprise Information Systems; (2005), 179-188.

[Nigam and Caswell 2003] Nigam, A., Caswell, N.: Business artifacts: An approach to operational specification; IBM Systems Journal; 42 (2003), 3, 428-445.

[Pardo et al. 2012] Pardo, C., Pino, F. J., García, F., Piattini, M., Baldassarre, M. T.: An ontology for the harmonization of multiple standards and models; Computer Standards & Interfaces; 34 (2012), 1, 48-59.

[ProcessGene 2013] ProcessGene: Processgene website; http://www. processgene.com (2013).

[Stanford 2013] Stanford Group, T. S. N. L. P.: The stanford parser website; http://nlp.stanford.edu:8080/parser/index.jsp (2013).

[van der Aalst et al. 2001] Van der Aalst, W., Barthelmess, P., Eliis, C., Wainer, J.: Proclets: A framework for lightweight interacting workflow processes; International Journal of Cooperative Information Systems; 10 (2001), 4, 443-482.

[van der Aalst and Ter Hofstede 2005] van der Aalst, W., Ter Hofstede, A.: YAWL: yet another workflow language; Information Systems; 30 (2005), 4, 245-275.

[van Dongen et al. 2008] van Dongen, B. F., Dijkman, R. M., Mendling, J.: Measuring similarity between business process models; Advanced Information Systems Engineering, 20th Int. Conference, CAiSE 2008, Montpellier, France; Springer; (2008), 450-464.

[Wahler and Kuster 2008] Wahler, K., Kuster, J.: Predicting Coupling of Object-Centric Business Process Implementations; Proceedings of the 6th International Conference on Business Process Management; Springer; (2008), 163.

[Wasser and Lincoln 2012a] Wasser, A., Lincoln, M.: Ontology based method for supporting business process modeling decisions; Springer; (2012a).

[Wasser and Lincoln 2012b] Wasser, A., Lincoln, M.: Semantic Machine Learning for Business Process Content Generation; 20th International Conference on Cooperative Information Systems (CoopIS); (2012b).

[Yan et al. 2012] Yan, Z., Dijkman, R., Grefen, P.: Business process model repositories-framework and survey; Information and Software Technology; 54 (2012), 4, 380-395.