

Design by Selection: A Query-based Approach for Business Process Modeling

Ahmed Awad¹ Matthias Kunze¹ Sherif Sakr² Mathias Weske¹

¹ Hasso-Plattner-Institute
University of Potsdam, Germany

{ahmed.awad, matthias.kunze, mathias.weske}@hpi.uni-potsdam.de

² School of Computer Science and Engineering
University of New South Wales, Australia
ssakr@cse.unsw.edu.au

Abstract. During business process design, working procedures in organizations are represented by process models. It is an important task in any process improvement project, but also both time consuming and error prone. While many organizations maintain large process model repositories, we observe that the information these repositories carry is not fully exploited during process modeling. In this paper, we present a novel approach to business process design called Design by Selection, which takes advantage of process repositories during design and facilitates reuse of process model components. These components can be static or flexible. Static ones represent the specific aspects of the process model, while flexible components realize re-use: They are defined by visual queries, which return matching process model components to be embedded in the overall process. As a result, process models can be designed in a more efficient and less error-prone way.

Key words: Business Process Design, Reuse, Querying business processes, Business process repositories

1 Introduction

Business process models are created by business process designers with the aim to capture business requirements, enable better understanding of business processes, facilitate communication between business analysts and IT experts, and serve as a basis for the derivation of executable business processes. Model design requires determining the activities that need to be performed, ordering of their execution, handling exceptional cases that can occur, etc. In any organization, business process models represent a main source of business knowledge which is scattered among several IT systems, business documents and the minds of involved people. This knowledge is reused each time a process model is created or updated, however, in an ad-hoc and generally uncontrolled fashion. Moreover, process design usually involves different perspectives, e.g. from business experts, top managers, compliance officers, etc. This makes the integration of these different views a time consuming and a tedious task. Nevertheless, it can be the case that a similar situation has been addressed before in the design of business

processes. In this case, it would be of great value to have systematic way to access and reuse existing process models in order to cut down the design cost of a new process model.

Imagine a financial institution that aims to extend its services through the offering of checking accounts in combination with a credit card. The institution is likely to already have several processes in place, which 1) validate the trust in the customer to balance its account after withdrawal through the credit card, e.g. the approval of loans or overdraft limits, and 2) obtain all required information to open a bank account, e.g. setting up a savings account. Reusing existing processes reduces the risk of neglecting steps required by the institution's policies and leverages existing resources and skills. Instead of designing a process from scratch and validating it against existing guidelines, a process designer would express new process fragments in a detailed way and express those parts of the process that shall stem from reuse in a declarative way. This intentional process description will then be used to query existing process definitions, e.g. from a process repository, for a set of adequate process fragments and to propose them to the designer who completes the process design by selection from this set.

“A repository is a shared database of information about engineered artifacts” [5], i.e., business process repositories store the primary artifacts—business process models—as well as information about them—*metadata*—such as: business domain, author, dates of manipulation and instance logs. The complexity of such repositories and their structure can be arbitrarily large, eventually fostering complete business ontologies [13,15]. Business process repositories have been developed along with techniques to access models, and associate them with metadata [6,15,24]. While search and retrieval of process models are largely based on keyword and full text search, certain approaches to effectively *query* process models according to their semantics have been proposed recently [2,4,16]. Amongst the use cases motivating the development of query languages was the support for process reuse. However, corresponding work mainly discussed the languages and the technical details on matching queries to processes without providing a vision on how reuse could be supported.

In this paper, we propose an approach that encourages reuse of process models, built on top of existing components, namely the open modeling platform Oryx [6], an open process modeling platform and repository, and the BPMN-Q query language [2,21], as a means to access and retrieve process components from the repository. We call this approach *Design by Selection*, by which users design so called *partial process models* that comprise both, a process modeling language and a process model query language, in an embedded fashion. The process model query language is similar to the process modeling language with regards to its notation, and thus, although being declarative, eases expressing a certain intention in the partial process model. To come up with suggestions, embedded queries will be matched against models stored in a repository to extract matching components. These components will be composed along with concrete parts in the original partial model to provide complete process models and to present them to the process designer as a ranked list, of which the designer can accept one as is or modify it arbitrarily. This approach promises several benefits:

- Time and effort of modeling can be reduced.
- Reuse is enabled on a process fragment level and precomposes a potential completion.

- A partial process model may contain several embedded queries; each could represent a view on the developed process [17], which relaxes the learning curve, particularly for novices in a business domain.
- The quality and maturity of the newly developed process models can be improved.
- Redundancy between several process models can be reduced, especially in case of process variants [14].

The remainder of this paper is structured as follows: In Section 2, we introduce the underlying concepts of this work, process models, process model queries, and similarity matching, followed by the proposed concept of partial process models, cf., Section 3 and a discussion of composing and ranking results of completions found in process model repositories, cf., Section 4. An architectural proposal for implementing the framework based on existing components is presented in Section 5. Finally, we discuss related work in Section 6, before concluding the paper with a prospect on future work.

2 Preliminaries

This section formally introduces process modeling and querying, which forms the groundwork for *partial process models*, which are described later.

2.1 Business Process Modeling

Currently, there is a number of graph-based business process modeling languages, e.g. BPMN [1], EPC [11], YAWL [22], and UML Activity Diagram [18]. Despite their variance in expressiveness and modeling notation, they all share the common concepts of tasks, events, gateways (or routing nodes), artifacts, and resources, as well as relations between them, such as control flow. [20] Without loss of generality, we can abstract from particular node types as their execution semantics are not vital to structural query matching, which is rather based on the concept of a process graph.

Definition 1 (Process Model). A process model P is a connected graph $(N, E, role)$, where $N = N_C \cup N_A \cup N_R$ is a set of nodes with disjoint sets N_C (a nonempty set of control flow nodes), N_A (a set of artifacts), and N_R (a set of resources), and a set of edges $E = E_C \cup E_D$, with disjoint sets $E_D \subseteq (N_D \times N_A) \cup (N_A \times N_D)$ (a set of directed data access edges connecting control flow nodes with artifacts) and $E_C \subseteq N_C \times N_C$ (a nonempty set of directed control flow edges) where $\bullet n$ ($n\bullet$) stands for the set of immediate predecessor (successor) nodes of $n \in N_C$.

A process model has exactly one start event $n_{start} \in N_C$ with no incoming and at least one outgoing control flow edge, i.e., $|\bullet n_{start}| = 0 \wedge |n_{start} \bullet| \geq 1$, and exactly one end event $n_{end} \in N_C$ with at least one incoming and no outgoing control flow edge, i.e., $|\bullet n_{end}| \geq 1 \wedge |n_{end} \bullet| = 0$. Each other control flow node $n \in N_C \setminus \{n_{start}, n_{end}\}$ is on a path from n_{start} to n_{end} .

$role : N_C \rightarrow (\mathbb{N} \cup \perp)$ assigns a control flow node, especially tasks, to a resource³.

³ In compliance with [23] we understand resources as entities that are capable of conducting work.

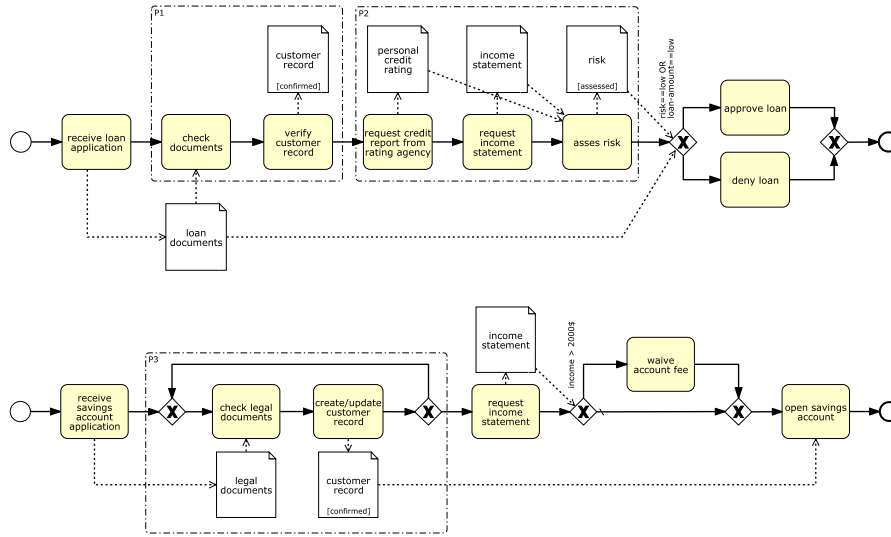


Fig. 1. Process Models and Potential Process Model Components

A connected sub-graph of a process model is a *process model fragment*. With regards to partial process models, we refer to a specific type of process model fragments that have a *single entry node* and a *single exit node* [25] as *process model components*.

Definition 2 (Process Model Component). A connected subgraph $(N', E', role)$ of a process model $(N, E, role)$, where $N' \in N, E' \in E$, is a process model component PC iff it has exactly one incoming boundary node $n_{in} \in N'$, i.e., $\bullet n_{in} \subseteq N \setminus N'$ and one outgoing boundary node $n_{out} \in N'$, i.e., $n_{out} \bullet \subseteq N \setminus N'$.

Figure 1 illustrates two process models found in a business process model repository. The notation used is BPMN [1]: Rounded rectangles describe an activity, or a unit of work, and arcs between them define sequence flow and thus causal relationships among the activities. Diamond shapes express gateways, where $+$ denotes that all outgoing branches need to be activated, X denotes one path must exclusively be chosen, and $\$$ denotes a subset of the outgoing branches needs to be followed according to the specified condition. Empty circles represent start and end event of a process, respectively. Artifacts are illustrated with dog-eared paper sheets.

2.2 Business Process Model Querying

Based on the definition of process models and process model components, we introduce the concept of process model queries, as a means to obtain business process components from a collection of business processes by structurally matching a query to each of them. BPMN-Q is a visual process model query language designed to help business process designers access repositories of business process models [2]. The language supports querying all the control and artifact concepts of business process models. Moreover,

it introduces a set of new *abstraction* concepts that are useful for different querying scenarios.

Definition 3 (BPMN-Q Query). A BPMN-Q query is a tuple

$Q = (QC, QA, QCF, QP, QDF, QUF, isAnonymous, role)$ where:

- QC is a finite set of control flow nodes in a query,
- QA is a finite set of data artifacts in a query,
- $QCF \subseteq QC \times QC$ is the control flow relation between control nodes in a query,
- $QP \subseteq QC \times QC$ is the path relation between control nodes in a query,
- $QDF \subseteq (QD \times QC \times QS) \cup (QC \times QD \times QS)$ is the data access relation between control objects and data objects.
- $QUF \subseteq QA \dot{\times} QP$ is the set of undirected data flow between data objects and paths⁴
- $isAnonymous : QC \cup QD \rightarrow \{true, false\}$ is a functions that determines whether control flow nodes and data objects in a query are anonymous.
- $role : N_C \rightarrow (\mathbb{N} \cup \perp)$ assigns a control flow node to a resource.

BPMN-Q Constructs. A BPMN-Q model is called a query. A query declaratively describes a structural connectivity that must be satisfied by a matching process model. In addition to the core business process modeling concepts, BPMN-Q introduces new concepts which serve its querying purpose.

- **Path edges** connecting two control flow nodes represent an abstraction over an arbitrary set of control flow nodes that could be in between the matching process model.
- **Undirected data flow edges** are used to connect data objects to a path edge in a query. This is also an abstraction mechanism where one looks for paths on which there are nodes that access the specified data object.
- **Anonymous activities and data objects** can be used if no restriction should be put on the labels of matched process model elements, e.g. to query any activity accessing a particular data object. In such cases elements of a query would be labeled with the character @ to declare it as an anonymous activity or data object, respectively.

Matching Queries to Processes. A BPMN-Q query is matched to a candidate process model via a set of refinements to the query. With each refinement nodes (edges) in a query are replaced with the corresponding nodes (edges) of the matching process model. If one node can have more than one possible replacement within the process model, a new refined copy of the query is created for each possible replacement. We call the replacement a resolution of an element of the query.

Since BPMN-Q is designed to match queries to process definitions in a repository, it is necessary to identify a candidate set of process models that can have the chance to provide a match to the query, rather than scanning the whole repository. To evaluate path edges, we rely on the transitive closure of nodes regarding the control flow relation.

Definition 4 (Reachability). Let $P = (N, E, role)$ be a process model. Let E_C^* be the transitive closure over the control flow relation E_C . For each node $n \in N_C$, we define the set of reachable nodes from n as $\llbracket n \rrbracket_{E_C^*} = \{m : (n, m) \in E_C^*\}$.

⁴ $\dot{\times}$ is the non Cartesian product, unordered pairs, defined as $X \dot{\times} Y = \{\{x, y\} : x \in X \wedge y \in Y\}$

Relying on Definition 4, we can explain how path edges are evaluated. A path edge ($source, target$) evaluates to a sub-graph in which all nodes are 1) reachable from $source$ node, 2) $target$ node is reachable from every node, and 3) edges between nodes in the sub-graph are those between the nodes in the process model.

A process model component $(N', E', role)$ of a process model $(N, E, role)$ is a resolution to the path edge ($source, target$), iff

- $N' = \{n : n \in \llbracket source \rrbracket_{F^*} \wedge target \in \llbracket n \rrbracket_{F^*}\}$,
- $E' = \{(x, y) : x, y \in N' \wedge (x, y) \in E_C\}$,

A process model is said to match the query if it satisfies all control flow, data flow and path edges as in Definition 5.

Definition 5 (Matching). A process model $P = (N, E, role)$ matches a query $Q = (QC, QA, QS, QCF, QP, QDF, QUF, isAnonymous, role)$ if:

- $QC \subseteq N_C$ control flow nodes in the query find resolvants in the process, for which the roles of resolvants also match,
- $QA \subseteq N_A$ data artifacts in the query find resolvants in the process,
- $QCF \subseteq E_C$ Control flow relations in the query are included in the process,
- $\forall p \in QP \exists (N', E') : sub-graph(N', E') \text{ is not empty,}$
- $QDF \subseteq E_A$ Data flow in the query is included in the process,
- $\forall \{a, p\} \in QUF \exists a' \in N_A, t' \in N_C : t' \in N' \wedge ((a', t') \in E_A \vee (t', a') \in E_A)$
where N' is the set of control nodes resulting from evaluating path edge p .

Please note that the current BPMN-Q implementation does not recognize roles, due to the fact that they are rarely used, yet does its extension only yield moderate effort.

2.3 Similarity Matching BPMN-Q Queries.

Assume a query with a path from activity “request credit report” to activity “assess risk” be created by a business process designer to search for processes that validates trust in a customer. A basic process model query processor would look for process models having activity labels strictly matching those in the query. Thus, process models having activities of the form “request credit rating”, “evaluate risk”, etc., will not be recognized as matches by the query processor, even though they are *semantically* relevant to the query. The search with strict matching of activity labels would be sufficient in organizations with a high maturity level in designing processes, or where process modelers are bound to a controlled vocabulary. In such situations, process designers are strictly aligned with a rather fixed set of labels to describe their process steps. Unfortunately, this is not the general case.

To tackle this problem, we employed information retrieval techniques to automate the discovery of *semantically* similar activities while avoiding the cost of manual labeling. Thus, we enhanced the query processor with a semantic query expander component [3], which searches for similar activities to those employed in the original query in the process model repository. Then the BPMN-Q query gets modified by substituting each of its activities with similar ones. With such a substitution step, new BPMN-Q query graphs are generated to constitute the expanded BPMN-Q query set.

Process components matching a query model will have a similarity score assigned ranging from 0 to 1. A similarity score of 1 indicates an exact match between the query and the process. Lower similarity scores indicate that a match was found between a

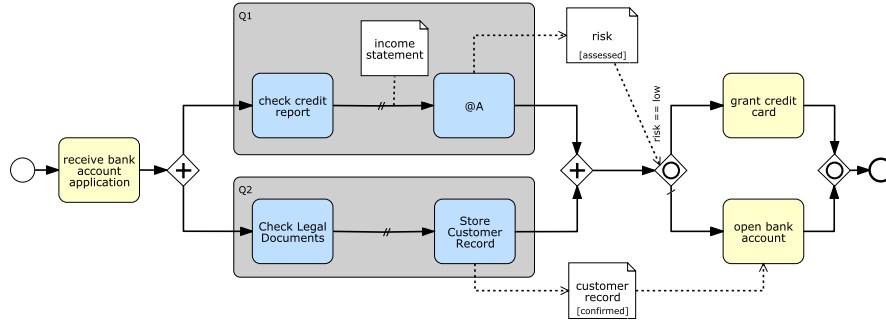


Fig. 2. Partial Process Example, consisting of process model fragments and process model queries.

semantically similar query and the process model. Users are given the opportunity to attribute their search with a threshold, ranging from 0 to 1, that controls the minimum similarity accepted in queries. For more details about the BPMN-Q query language and its similarity matching mechanism, we refer the reader to [2,3].

3 Partial Process Models

The approach presented in this paper—*Design By Selection*—is based on the notion of *partial process models* that describe a desired process model through a combination of process model fragments and process model queries.

Definition 6 (Partial Process Model). A partial process model $\mathfrak{P} = (\mathfrak{F}, \mathfrak{Q}, \mathfrak{A}, \mathfrak{E}, role)$ is a connected graph that consists of disjoint sets of process model fragments \mathfrak{F} and process model queries \mathfrak{Q} connected through directed edges $\mathfrak{E} \subseteq \mathfrak{F} \times \mathfrak{Q} \cup \mathfrak{Q} \times \mathfrak{F}$, where each outgoing boundary control flow node $n_{out} \in N$ of a process model fragment $F \in \mathfrak{F}$ is connected to exactly one incoming boundary control flow node $n_{in} \in QC$ of a process model query $Q \in \mathfrak{Q}$ and vice versa.

Artifacts $a \in \mathfrak{A}$ are connected to both, process model fragments and process model queries, via a data access edge $(a, n)\{n \in N_C\}$ connected to a control flow node in a process model fragment and a data access edge $(a, n)\{n \in QC\}$ connected to a control flow node in a process model query or a path $(a, p)\{p \in QP\}$ in a query, respectively.

$role : (N_C \cup QC) \rightarrow \mathbb{N} \cup \perp$ uniformly assigns control flow nodes of process model fragments and process model queries to a resource.

\mathfrak{A} and $role$ are separately included in the definition of partial process models, because they define the context of a query specified by the surrounding process model fragments. At query evaluation time, they will be compiled into the respective process model queries to reflect the respective query's context within potential matches.

Figure 2 shows a partial process model corresponding to the use case illustrated in Section 1. The first intention, to validate trust in the customer, is represented by BPMN-Q query $Q1$, i.e., a set of actions that take the *income statement* of a client

into account and an anonymous activity that produces a context artifact *risk* in the state *assessed*. The query in the lower part addresses the second intention, i.e., the creation of a customer record, and matches process components that contain activities labeled similar to *check legal documents* and *store customer record*, yielding a *complete customer record*, cf., *Q2*. The right part of the process model illustrates how artifacts from the queries are incorporated into the remaining process model, e.g. the assessed risk will have an influence on the decision, whether the customer will be granted a credit card. Such relationships make up the context of queries in a partial process model. Figure 3 shows an example of a composed process model based on the defined partial process model in Figure 2. In this composed model, the process model components *P2* and *P3* are extracted from the process models of Figure 1 where *P2* represents the matching element of *Q1* and *P3* represents the matching element of *Q2*.

Evaluation of a process model query $Q \in \Omega$ returns zero or more process model components. Obviously, the process model query in the partial process model *PM* can be replaced by one of the returned process components, since both have exactly one incoming and one outgoing boundary node, as defined above, which results in a complete business process model. However, each business process query can return a potentially large number of matched business process components from which one needs to be chosen. This is addressed by the term “selection”, i.e., the users who developed the partial process model are supposed to choose from a ranked list of alternatives that comply with the given business process queries. To achieve this, the results of the single business process queries need to be combined into a complete process model, as discussed previously, and a meaningful ranking must be established to support the user in selecting a process model proposal.

4 Ranking of Model Compositions

The main task of a query processor is to evaluate the BPMN-Q queries of the defined partial process model against the process model repository. For each BPMN-Q query, a result set is returned comprising *matched* process model components. These matched components could represent exact or similar matches for the query models, cf., Section 2.3. In case of similar matches, each matched process model component is then attached with its similarity score (*SS*) which is computed during the query evaluation process. In case of an exact match, the value of this similarity score is equal to 1 for each matched component of the result set.

In principle, having different matched components for each query model, which usually belong to different process models, yields that we can have several possible compositions that originate from distinct process models. Each possible composition needs to include exactly one component from the answer set of each query in the partial process model. Hence, the number of the possible compositions is equal to the number of queries in a partial process model multiplied by the number of returned results for each query. Clearly, it is inconvenient for process designers to go through this potentially very large list of compositions to select among them. Therefore, the set of possible compositions are *ranked* according to various criteria, applying a ranking process that starts by initially ranking the matched process model components inside the answer set of each query based on their similarity scores. Then, it computes a ranking score for each

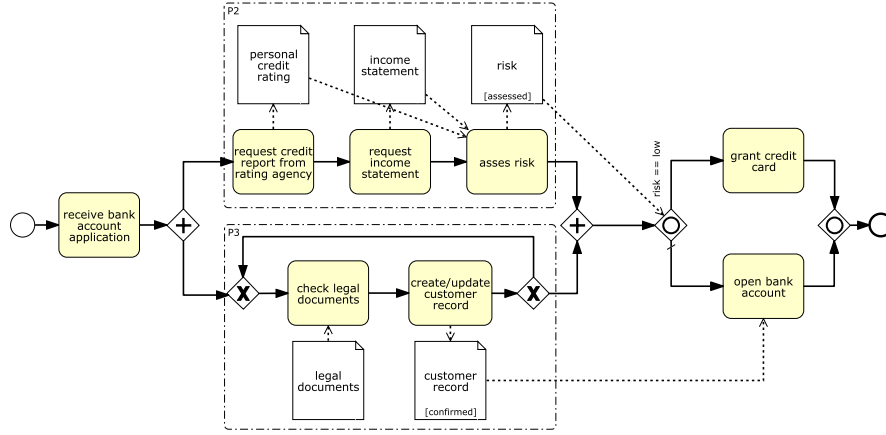


Fig. 3. Complete Process Example, composed of proposals P2 and P3, cf., Figure 1.

possible composition model. This ranking score is obtained by *fusing* the intermediately computed scores of different scoring elements, which are described as follows.

- **Combined Similarity Matching Score (CS):** Computing this scoring element for a candidate composition model ccp is achieved by multiplying the similarity scores of its individual components.

$$CS(ccp) = \prod_{i=1}^n SS(mc_i)$$

where n represents the number of the model components and $SS(mc_i)$ refers to the similarity score of the individual model components, cf., Section 2.3.

- **Homogeneity Score (HS):** The process model components of each candidate composition model can belong to the same or to distinct original process models. Having different model components of the candidate composition belonging to the same original process model increases its homogeneity, which leads to an increase in its chances of being more meaningful and highly appealing so as to be accepted by the process designer. For example, the combination between the process model components $P1$ and $P2$ (Figure 1) will be more homogeneous than the combination between the process model components $P2$ and $P3$. Thus, we start by computing the homogeneity between each *unique different pair* (PHS) of the model components as follows

$$PHS(mc_i, mc_j) = \begin{cases} 0 & \text{if the original models of the pair are different} \\ 1 & \text{if the original models of the pair are the same} \end{cases}$$

where mc_i and mc_j represent a unique different pair of model components. In general, the number of unique different pairs n is equal to $\frac{c(c-1)}{2}$ where c is the number of process model components. The homogeneity score of a candidate model composition ccp is then computed as follows

$$HS(ccp) = \frac{\sum_{i=1}^n PHS(udp_i)}{n}$$

where udp represent a unique pair of model components (mc_i, mc_j) and n represent the total number of unique different pairs.

- **Metadata Matching Score (MS)**: As mentioned above, process model repositories store metadata (descriptive attributes about process models). Clearly, the homogeneity between model components which originate from *different* process models increases if there is an *overlap* in their metadata. For example, model components that have been developed by the same user or in the same department have a higher chance of being more homogeneous or meaningful. Therefore, the metadata matching score (MS) is considered to be complementary to the general homogeneity score (HS). Here, we also start by computing the metadata homogeneity score (MHS) for each *unique different* pair of model components as follows.

$$MHS(mc_i, mc_j) = \frac{T(mc_i) \cap T(mc_j)}{c}$$

where mc_i and mc_j represent a unique different pair of model components, T represents the set of values for the metadata attributes of a model component and c represents the total number of the metadata attributes. The metadata matching score of a candidate model composition ccp is then computed as follows.

$$MS(ccp) = \frac{\sum_{i=1}^n MHS(udp_i)}{n}$$

where udp represents a unique pair of model components (mc_i, mc_j) and n represents the total number of unique different pairs. It should be noted that for this scoring element we consider only those pairs of model components which originate from *different* process models. The key reason behind this is that those pairs of components, which originate from the *same* process model, have already had their score increased on the homogeneity scoring element (HS).

- **Reusing Popularity Score (RS)**: For different reasons, e.g. clarity, simplicity, completeness, reputation of the designer, one process model can be more reusable than another. To make use of this fact, for each process model in a repository, we keep track of its *reuse ranking* (RR). This ranking gets automatically increased when any of its components is reused in a newly composed model. Thus, this scoring element can make effective use of the “wisdom of the crowd” in the context of social systems. Based on the reuse ranking of the stored process models, we then compute the reusing popularity score for each candidate composed model ccp as follows.

$$RS(ccp) = \sum_{i=1}^n RR(mc_i.originalModel)$$

where mc refers to a model component, $originalModel$ refers to the original process model from which the process model component originated and n represents the total number of model components.

The *final ranking score* of a candidate composed model (ccp) is computed by fusing the values of its scoring elements using the following weighted function.

$$FinalScore(ccp) = w_1 * CS(ccp) + w_2 * HS(ccp) + w_3 * MS(ccp) + w_4 * RS(ccp)$$

where w_i represents a weighting factor for a scoring element which can be configured and adjusted by the end-user, while $\sum_{i=1}^n w_i = 1$. Initially, process designers can rely

on a uniform regression parameter where all weighting factors have the same value, i.e. $w_i = 1/n$. With the continuous usage of the system, workload data can be gathered to generate significant training datasets that can be used as an input for a regression analysis process to deduce optimized weighting factors [10].

In practice, the set of the possible compositions generated by the combination of the matched process model components for the BPMN-Q queries can be large. Therefore, the framework needs to apply *filtering* mechanisms in order to avoid overwhelming the process designer with less valuable possible compositions such as: 1) The model composer needs to ignore the matched process model components for the BPMN-Q queries with a similarity score (SS) that is less than a user-defined threshold t_1 . Similarly, the possible compositions with combined similarity matching scores (CS) that are less than another a user-defined threshold t_2 can be ignored. Moreover, the composer may return only the top K possible compositions based on their final ranking score ($FinalScore$) where the value of K is user-defined. 2) The composition environment needs to allow the process designer to specify some *constraints* on the composition scheme. For example, the process designer can describe that matched process model component of Q_1 and Q_2 must belong to the same original process model or must have an overlap in a particular subset of their metadata.

5 Framework Architecture

In this section, we envision an architecture for the *Design By Selection* framework for business process modeling, illustrated in Figure 4, which consists of the following main components.

- **Process Modeling, Querying, and Composition Environment** provides the process designer with a user-friendly *modeling interface* [6]. Users express their intention by means of a partial process model, cf., Section 3. The *query interface* extracts the set of process model queries from the partial process model, and passes them on to the query processor. The returned set of queries will then be *composed* with the model fragments from the partial model and *ranked* through the *model composer*, cf., Section 4.
- **Process Model Repository.** Instead of building the Design by Selection framework on top of a proprietary repository, it shall be connected to several, potentially disparate repositories, obtain and maintain process models stored remotely. Repositories do not only store models [5], but also a set of metadata, which is used to control the composition and ranking process.
- **Uniform Language Interface** translates process models of specific languages, cf., Section 2.1, to a common representation suitable to process model querying, i.e. complying with Definition 1. This allows to query a larger set of process models and can further be used to unify the query interface and processor toward different process definition languages [20].
- **Query Processor & Process Model Indexes.** The *query processor* evaluates the queries received from the query interface [21]. It provides support for relaxation and refinement of user queries. In case the queries do not return sufficient results, the query processor is able to relax the query according to some similarity notions [3,7]. Similarly, if a query returns too many results, the user needs to be provided with a

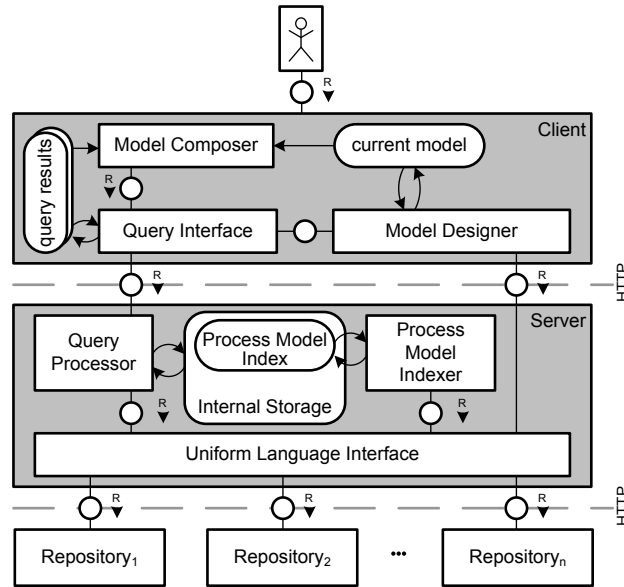


Fig. 4. Framework Architecture

possibility of refining and improving their request. In order to further improve the searching, process models could be *indexed* upfront to expedite query evaluations.

As mentioned earlier, some of the components of this framework do already exist. The client, particularly the *model designer*, is the Oryx editor, an extensible process modeling platform for research, designed to model and manage process models online [6]. *Query interface* and *query processor* for BPMN-Q [2] have been implemented as plugin to the Oryx editor and are able to run process model queries against the Oryx process model repository. We have not implemented a *uniform language interface* yet, but similar work has been done by La Rosa et al. [20], who propose a generic process model representation for an advanced process model repository. Our architecture acknowledges the existence of a multitude of different process model repositories, which is the rationale behind the decoupling of a process modeling, querying, and composition system from a particular process model repository.

6 Related Work

The issue of accelerating business process design has attracted many researchers. In [17] Mendling and Simon propose to use schema integration approaches from database research to come up with a unified process model that satisfies the different perspectives of the stakeholders, obtained by interviews. Later on, these perspectives are merged and some restructuring rules are applied to refine the final model. Compared to our approach, a view could even be declaratively represented as a query. Moreover, we can contribute to the creation of the view by letting the designer query the repository for similar situations and then discuss the result with the respective stakeholders. Unlike the

approach by Mendling and Simon, it is not necessary to state semantic relations between the different views. This can be achieved by the composition and ranking components.

Several approaches have been proposed to measure the similarity between business process models. Ehrig et al. [8] have presented an approach for measuring the similarity between business process models semantically modeled with the Web Ontology Language (OWL). The approach relies on detecting *synonyms* and *homonyms* of process element names where the degree of similarity between process models correlates positively with the number of used synonyms and negatively with the number of used homonyms. Dijkman et al. [7,19] have described another lexical technique called *N-to-M* label matching which matches any node in one model with any node in the other model as long as the *string edit distance* between their labels is above a given threshold value. The authors have also described two *structural* matching techniques named as *greedy graph* matching and *A-star* graph matching. These algorithms are considered to be variants of the *graph edit distance* similarity measure. The *greedy* algorithm computes the optimal mapping by starting with an empty mapping, then adds the pair of nodes to the mapping that most increases the graph matching score. The algorithm continues adding pairs of nodes until there are no more pairs that increase the graph matching score. The *A-star* algorithm tries to deal with this problem by selecting a partial mapping with the lowest graph edit distance in each step. However, the number of partial mappings that need to be maintained depends on the number of pairs of nodes that can be matched. Therefore, they limit the search space by considering only highly similar pairs instead of trying to match all pairs of nodes. In principle, our framework is fully agnostic with respect to integrating and reusing any similarity matching technique for process models into the query processor component.

Our approach further covers the topic of assisted completion of process models, in that it takes one or several fragments of a model and proposes a set of completions that satisfy the modeler's intention, which they describe via queries. A similar problem has been addressed by Koschmider et al. [9,12], who propose a semi-automatic completion framework for a special type of petri nets. A collection of petri nets, a query petri net and a set of textually defined rules that characterize the desired completion are transposed into an ontology. Automatic inference techniques are used to match the query model and rules structurally against the existing model collection. The result is ordered solely by the degree of similarity of a matched model to the query. Our approach is significantly different, in that 1) BPMN-Q proposes a graphical query language that is very similar to the process definition language in notation and semantics, 2) allows several points of inserting queries in a partial process model, and 3) offers a comprehensive ranking model that does not only take the syntax of a process model into account, but also relations between matches found and metadata stored along with models in a repository.

7 Conclusion and Outlook

In this paper, we introduced a new approach to accelerate business process design. The approach is based on the idea of investigating a business process repository by means of queries. In a partial process model, the user can define new functionality imperatively and specify functionality to be looked up declaratively with process model queries. The results of each embedded query are composed and ranked in order to provide the user

with the closest result to their queries. This approach promises several benefits by reusing existing business knowledge materialized in existing process models. The reuse is not only on the level of a whole process model, but rather on a fine grain level which is in the form of process model components. Moreover, the approach collects components from different process models. Another benefit is that each query could represent a view on the process design. Thus, we also address the issue of collaborative process design in a simple way.

Depending on the embedded queries, the time taken to match, compose, and rank varies. With large business process model repositories, the processing time of retrieving partial process model components can be very long. As future work, we intend to define upfront analyses of a process model query that inform the user about embedded queries, which might be meaningless or yield a potentially very large result sets. Moreover, we plan to extend our framework to support the design of *federated* business process models that reflect virtual global views on existing process model components while keeping the underlying process model autonomous. In other words, each federated model will serve as a template for the integration between fragments from the underlying process models.

References

1. Business Process Modeling Notation 1.2 (BPMN 1.2) Specification, Final Adopted Specification. Technical report, OMG, 2009.
2. A. Awad. BPMN-Q: A Language to Query Business Processes. In *EMISA*, 2007.
3. A. Awad, A. Polyvyanyy, and M. Weske. Semantic Querying of Business Process Models. In *EDOC*, pages 85–94, 2008.
4. C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *VLDB*, 2006.
5. P. Bernstein and U. Dayal. An overview of repository technology. In *VLDB*, 1994.
6. G. Decker, H. Overdick, and M. Weske. Oryx - sharing conceptual models on the web. In *ER*, 2008.
7. R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *BPM*, 2009.
8. M. Ehrig, A. Koschmider, and A. Oberweis. Measuring Similarity between Semantic Business Process Models. In *APCCM*, 2007.
9. T. Hornung, A. Koschmider, and A. Oberweis. Rule-based autocompletion of business process models. In *CAiSE Forum*, 2007.
10. C. Hwang, D. Hun Hong, and K. Seok. Support vector interval regression machine for crisp input and output data. *Fuzzy Sets and Systems*, 157(8), 2006.
11. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”. Technical Report 89, Institut für Wirtschaftsinformatik, Saarbrücken, 1992.
12. A. Koschmider and E. Blanchard. Automatic user assistance for business process modeling. In *RCIS*, 2007.
13. A. Koschmider and A. Oberweis. Ontology based business process description. In *EMOI-INTEROP*, 2005.
14. R. Lu and S. Sadiq. Managing Process Variants as an Information Resource. In *Business Process Management*, 2006.
15. Z. Ma, B. Wetzstein, D. Anicic, and S. Heymans. Semantic business process repository. In *SBPM*, 2007.
16. I. Markovic. Advanced querying and reasoning on business process models. In *BIS*, 2008.

17. J. Mendling and C. Simon. Business process design by view integration. In *BPM Workshops*, 2006.
18. OMG. UML 2.0 superstructure specification. Technical report, 2004.
19. L. García-Bañuelos R. Dijkman, M. Dumas and R. Krik. Aligning Business Process Models. In *EDOC*, 2009.
20. M. La Rosa, H. Reijers, W. Aalst, R. Dijkman, J. Mendling, M. Dumas, and L. Garcia-Banuelos. Apromore : An advanced process model repository. <http://eprints.qut.edu.au/27448/>, 2009.
21. S. Sakr and A. Awad. A Framework for Querying Graph-Based Business Process Models. In *WWW*, 2010.
22. W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Inf. Syst.*, 30(4), 2005.
23. W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1), 2003.
24. J. Vanhatalo, J. Koehler, and F. Leymann. Repository for business processes and arbitrary associated metadata. In *BPM Demo*, 2006.
25. J. Vanhatalo, H. Völzer, and J. Koehler. The refined process structure tree. In *BPM*, 2008.