Recommendation Based Process Modeling Support: Method and User Experience

Thomas Hornung[1], Agnes Koschmider[2], and Georg Lausen[1]

[1] Institute of Computer Science, Albert-Ludwigs University Freiburg, Germany
hornungt|lausen@informatik.uni-freiburg.de
[2] Institute of Applied Informatics and Formal Description Methods
Universität Karlsruhe (TH), Germany
koschmider@aifb.uni-karlsruhe.de

Abstract Although most workflow management systems nowadays offer graphical editors for process modeling, the learning curve is still too steep for users who are unexperienced in process modeling. The efficiency of users may decrease when starting process modeling with minimal expertise and no obvious modeling support. This paper describes the first contribution towards a theoretically sound and empirically validated analysis of a recommender-based modeling support who is geared towards both novices and expert users. The idea is to interpret process descriptions as tags which describe the intention of the process. This leads us to the notion of virtual documents or signatures. Based on these signatures we provide a search interface to process models stored in a repository. Additionally the user can invoke a recommendation function during modeling time and the system automatically identifies and suggests relevant process fragments. By adding two additional criteria, the frequency of process reuse and structural correctness, we arrive at a full-fledged modeling support system, which provides an easy to use interface to the user while retaining a high fidelity to the user's modeling intentions. We validated our support system with a user experiment based on real-life process models and our prototype implementation.

1 Introduction

Although most workflow management systems nowadays offer graphical editors for process modeling, the learning curve is still too steep for users who are unexperienced in process modeling. Pure awareness of the modeling language syntax is often insufficient. Profound working knowledge of the user is required to apply a modeling language in practice. [1] argues that user's modeling expertise is one main success factor of process modeling. Therefore, the user efficiency may decrease when starting process modeling with minimal expertise and no obvious modeling support.
To ensure a certain degree of modeling support, several authors proposed the reuse of process models [2], [3] but yet with little impact on the modeling context and user intention. Clearly, a full-fledged modeling support system is required,

which retains a high fidelity to the user's modeling intentions.
This paper describes the first contribution towards a theoretically sound and empirically validated analysis of a recommendation based modeling support, which assists the user twofold in modeling goal-oriented processes. Firstly, the user can search via a query interface for business processes or process parts (logically coherent groups of elements belonging together, e.g. approval, billing or assembly). The user can significantly save time in process modeling if a process matches the user request. Secondly, we use an automatic tagging mechanism in order to unveil the modeling objective of a user at process modeling time and to better fulfill the user's requirements. This feature of the modeling support system should be used if the user is not sure how to complete the process. In this case the results from the query can be unsatisfying due to the user's vague intention of the process model.
We validated our support system with an experiment using real-life process models and our prototype implementation. The evaluation confirmed that the modeling time and the number of operations of the reused processes can be reduced when using our process support tool. The evaluation results highlight which benefits users may have from our recommendation based modeling tool support:

- The system increases the efficiency of the user because users need less expertise to appropriately model processes,
- The tagging-based system increases the quality of the process models by highlighting the corresponding process parts that violate correctness criteria (e.g., structural deadlocks, which occur if an alternative flow initiated by an OR-split is synchronized by an AND-join),
- Our system overcomes the limitation of a controlled vocabulary for labeling process element names since the system considers process fragments with process vocabularies that are different from the one of the currently edited business process.

The remainder of the paper is structured as follows. Section 2 compares our approach with related work. Our tagging-based modeling support system will be explained in detail with a running example in Section 3. Section 4 presents our tagging algorithm and the creation of our process repository index. In Section 5 we will describe the business process search functionality and we will extend the search functionality in order to consider relevance. The cumulative ranking function and the complete recommendation algorithm is illustrated in Section 6. Initial evaluation results are presented in Section 7. Section 8 concludes the paper with an outlook on future research.

2 Related Work

Existing work in this area can be differentiated in four categories: (1) process reuse, (2) tagging, (3) process/service searching, and (4) research on ranking mechanisms.

To ensure a certain degree of modeling support, several authors proposed the reuse of process models [2], [3] but yet with little impact on the modeling context and user intention. All this contributions lack an extensive query interface and a recommendation function for process parts during process modeling. Additionally, the proposed ranking functions are theoretic without empirical validation.
Concerning the annotation of resources with tags the approach of [4] is relevant for our approach. This approach describes a method, which automatically generates personalized tags for Web pages. The method of [4] relies on the idea, that the personalized tags are generated based on the user's Desktop documents. In the current implementation we unveil the user intention during process modeling with the edited process elements. We plan to evaluate the idea of [4] in order to tag project documents, which also may help to reveal user requirements during process modeling. We omitted this idea for the current implementation as project documents are difficult to obtain.
Regarding the process searching area, the set of proposals found in the literature [5], [6], [7] and [8] do not provide adequate techniques for searching processes concerning the user intention while reusing processes. For instance, the approach of [6] extends a rudimental Web Service search by supporting more complex service description search capabilities. [6] indexes business processes for efficient matching in Web Service infrastructures where the input query is a business process that is modeled as an annotated finite state automaton. This approach does not focus on searching and indexing business processes but rather on searching for complex service descriptions of services such as process aspects (by searching for optional and mandatory requirements within the business processes).
Ranking functions have been defined for Semantic Web Service Discovery [9] or for Information Retrieval [10] where the ranking functions are based on an ontology structure or human interaction. As [11] argues that the effectiveness of tags classifying blog entries are for manual tags less effective content descriptors than automated ones we decided to disregard human interactions for the process ranking at the moment.
To summarize, there are some approaches which partially use related methods to implement a modeling support system but on a rather theoretical level without existing empirical validation. The aim of our work is to present a modeling support system including a comprehensive query interface, recommender and ranking function, which are theoretically sound and initially empirically validated.

3 Running Example

Our current implementation of the support system is described in Figure 1. The user wants to model a process describing the handling of order requests. Her intention is to model this process from the perspective of customers. Via a query interface the user searches for process parts concerning *customer requests*. The results from the query were displayed according to a ranking function where the user selected the appropriate recommendation due to her modeling intention and

inserted the recommendation into her workspace via drag and drop.
Subsequently, the user continued to model the business process. At some point, she is not sure how to complete the process. Therefore she invokes the recommender system, which can be done in two ways. The user can either search via the query interface for fitting process parts (e.g., processes modeling *customer orders*) or invoke the recommender system by highlighting the corresponding elements for which the user wants to have a recommendation (in this figure the corresponding element group is highlighted with a gray rectangle). For the sec-

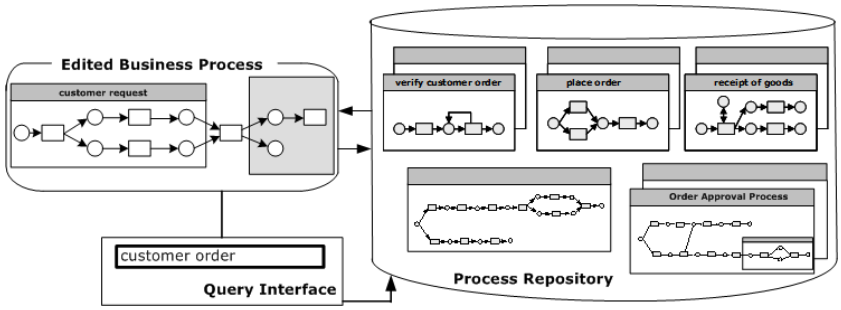

Figure 1. Possible user interaction scenarios for finding an appropriate process part.

ond alternative the recommender system automatically retrieves fitting process parts according to the user's modeling intention. The recommender component can only be invoked after modeling process elements (in contrast to the query interface, which is always accessible). Subsequently, the user can configure the process (part) suggestions in her workspace by inserting or deleting elements and save the modified process version in a process repository for further process reuse. In the initial development of our prototype we have populated our repository with 21 Petri net processes from real word projects and processes from the research literature concerning order and shipment procedures.
In the next two sections we present our process search algorithm.

4 Semantic Annotation of Business Process Models

Usually keyword extraction algorithms use as input documents and return a list of significant keywords, which outline the content of a document. The number of occurrences of each keyword implies a ranking in the sense that the keyword that appears most often is more relevant to the document than keywords that appear less often. We adopt the intention of keyword respectively tagging techniques to improve searching for fitting business processes. The tag extraction and scoring for business processes is inspired by the *Term and Document Frequency* measure, which is very fast to compute (cf. [12]). Each place and transition in a Petri net representation of a business process model is labeled with a description, which

specifies the purpose of each activity or state respectively. Therefore, we can regard these words as tag candidates and thereby the whole Petri net as a virtual document. This allows us to use standard Information Retrieval (IR) techniques (cf. [13]) to build up an index over business process models. Here, we first remove common English words from the set of tag candidates because they appear so often in a typical natural language corpus that they do not convey any meaning specific to the business process. This phenomenon is often referred to as Zipf's law, which states that the frequency of any word is inversely proportional to its rank in the frequency table [14]. After stop word removal each keyword is assigned a tag score for this business process based on a modified version of the *tf∗ idf metric*[3]:

$$\text{TagScore}(t_i) := \frac{\text{TF}(t_i)}{\Sigma_{j=1}^{N} t_j} * log(\frac{|P|}{|(p_j : t_i \in p_j)|})$$

Here, $\text{TF}(t_i)$ is the frequency of the tag t_i in transition or place labels, N is the total number of distinct tag candidates (after stop word removal), $|P|$ denotes the total number of indexed business processes and $|(p_j : t_i \in p_j)|$ is the number of business processes, where the tag t_i appears. The purpose of the *idf* part ($log(\frac{|P|}{|(p_j:t_i \in p_j)|})$) is to decrease the impact of words that are common over all business processes. In order to bridge the gap between different modeling vocabularies we determine for each keyword the set of synonyms via WordNet[4] and assign the same tag score to each word in the synonym set.
As mentioned above, the user can identify different distinct process fragments and assign a title to them (e.g., order approval, complaints handling, order receipt). To make these fragments searchable as well, we index them in the same way as if they were regular business processes and additionally store a pointer to the business process with which they are associated, e.g. for a business process which consists of three distinct process fragments, we would include four virtual documents in our index: the whole process, and each fragment as well.
In the next section we illustrate the supported retrieval possibilities for our annotated business processes and process fragments, whereas in Section 6 we describe the overall ranking algorithm used for recommending appropriate process fragments.

5 Searching for Process Fragments

The user has the possibility to use a tag based search functionality at each stage during the process modeling phase and can choose whether she wants to search for process parts, whole business processes or both. Since we used the open source Java search engine Lucene[5] as the underlying index and search framework, the scoring of the results is based on a mixture of the Vector Space Model (VSM) and the Boolean model. The key idea of the VSM is to represent each

[3] *term frequency ∗ inverse document frequency*
[4] `http://wordnet.princeton.edu/`
[5] `http://lucene.apache.org`

document, i.e. business process in our case, as a multi-dimensional vector, where the dimension is the total number of unique keywords that occur over the whole corpus, i.e. all indexed business processes. This vector constitutes the signature of the process which is later used for retrieval. A query is interpreted as a vector in this space and the similarity of the query to documents is computed based on the cosine similarity ($cos(\theta) = \frac{v_{query} * v_{process}}{||v_{query}|| * ||v_{process}||}$) between the two vectors. More specifically, a business process b_j would be represented by the vector $b_j = [\text{TagScore}(t_1), \text{TagScore}(t_2), \ldots, \text{TagScore}(t_K)]$. Because we have enriched each tag with the additional synonym set, the dimension of the vector is not N, the total number of unique tags (except stop words), but $K = \Sigma_{i=1}^{N} |\text{SynSet}(t_i)|$, where $\text{SynSet}(t_i)$ denotes the set of all synonyms of t_i.
Continuing our example from Section 3 the user is searching for both process parts and entire business processes modeling *customer orders* (see Figure 2). The user activated WordNet in order to suggest processes, where process ob-

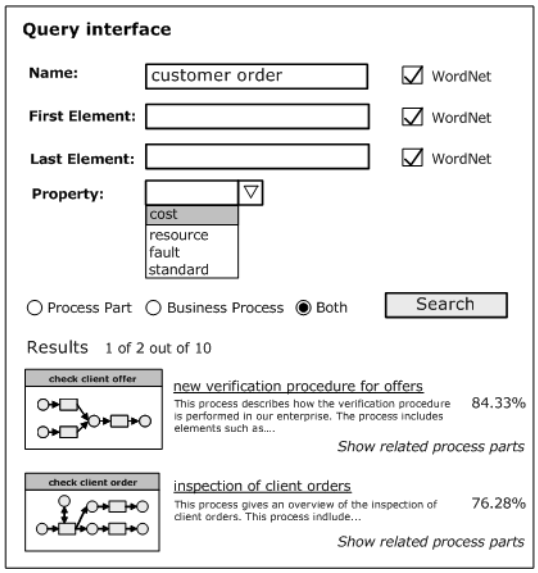

Figure 2. Query for all process fragments that are related to *customer orders*.

jects have been labeled with respect to a different vocabulary. The user can narrow down the number of recommendations by the criteria *First Element* and *Last Element* searching for a specific first or last element(s) in the process. An additional search criterion is the process property, where *cost* signifies a low cost process, *resource* indicates a process with full exploitation of resources, *fault* is a process with minimal fault rate and *standard* signifies a standard process. This four properties result from our practical process modeling experiences. If required, the user can introduce more annotation properties.

The recommender system found 10 results, which match the user's modeling intention and displays them ranked by their Lucene score. If the user is interested in a recommendation she can open a larger view of the process fragment by double clicking on the picture.
Besides the standard Boolean operators, such as `AND`, `OR`, and `NOT` the user can pose wildcard queries and perform fuzzy searches based on the Levenstein distance, or Edit distance algorithm [15].
Additionally, in Figure 2 the user can preview related process parts for each recommendation (see *Show related process parts*). The idea is that process parts that succeed or precede the part in question and were used in the same modeling domain the user is in at the moment (e.g. Manufacturing) can help to estimate the degree of fitness of a recommended part.
Therefore each business process model and thus each process part that occurs in this model is classified into a modeling domain before it is added to the process repository. We assume that the number of possible domains is usually fixed within a company and hence we can provide the user with an interface where she can choose to which domain the process belongs. Additionally, the process property can be provided in this stage. After a sufficient amount of process models are in the repository we could use automatic classification techniques such as a Naïve Bayes classifier [16] to automatically highlight a domain the process model is most likely to belong to, or the most likely process property respectively.
If the user now clicks on the *Show related process parts* button, the system shows two ranked lists of related processes, i.e. the preceding and the succeeding process parts.

6 Ranking of Recommendations

The process recommendations depicted in Figure 2 show results of a user query, which are ranked according to the Lucene score. If the user invoked the query search before starting modeling any node (and the suggested recommendations have never been selected by someone else), then the recommendations will exclusively be ranked as explained in Section 5. But the ranking mechanism changes if the user invokes the recommender function or the query interface once she already modeled process elements in her workspace. Then the ranking also depends on the modeling context (e.g., activities which were modeled and the control flow).
To completely rank fitting recommendations for the second scenario we extend the Lucene score, which was introduced in Section 5, with two additional criteria. Firstly, the frequency a user selected a specific process fragment in the past and secondly the number of structural errors. In our scenario a structural error can only occur in the interconnected process (to be composed of the edited business process and the recommended process).
An interconnected business process is considered *structurally correct* if it complies with the well-structuredness property [17]. This structural property for

business processes is violated if for example an alternative flow initiated by an OR-split is synchronized by an AND-join. The benefit of this property is a good process modeling style, which makes understanding of the processes models easy and supports the detection of undesirable deadlocks[6]. The verification of structural properties is performed once for all process fragments that match the automatically generated Lucene query mentioned above. For instance, the interconnection of the edited business process (excluding the highlighted elements) with the first recommendation in Figure 2 would include a structural problem (an AND-split is synchronized by an OR-join, which is specified in the literature as a TP handle). Nevertheless, the user can insert this recommendation into her workspace. But, she needs to decide how to improve this business process. We assume, that the processes in the repository are already analyzed and thus we exclude (structural) deadlocks for them.

In Figure 1 the user highlighted three elements for which she wants to have a recommendation for both process fragments as well as whole business processes (see Figure 2). To determine relevant process parts, we extract the labels of each highlighted process object (place or transition) and remove common stop words, which yields the set t_{raw}. The remaining query tag candidates t_{raw} are then expanded with their related synonym sets, similarily as described in Section 5, resulting in the set t_{query}[7]. The initial process fragments are then determined by querying the Lucene index, where the query term is the concatenation of all tags in the set t_{query}. In the remainder of this chapter we present two additional criteria which are used to tweak the rank of the thus found process fragments.

As already mentioned previously we assume that users independently declare logically coherent process parts, which are stored with a title and optionally a description and a process property in the repository. This runs the risk that users store useless process parts in the repository because no consistency check is applied in order to evaluate the usefulness (a process part with one element may be regarded as useless). To remedy this we integrated the frequency a process fragment has been selected in the past into our ranking. If process fragments have been refreshed, respectively updated, the user will be informed about this with a remark. The updated processes are assigned the same frequency score as the old process version. If users decide against the updated process (and favor more often another recommendation) then the frequency score will automatically decrease over time.

To calculate the frequency a user selected a process we adapted the user count algorithm presented in [18]. Let U and P be the set of all users and processes, and p_{ij} is the number of selections of process j by the user i. The rating r_{uk_1} for the number of users u who have selected the process k is:

$$r_{uk_1} := \frac{\sum_{i \in U} t_{ik}}{|U|}$$

[6] An undesirable deadlock is a situation where a process instance is waiting for a progress, which cannot be performed because some task cannot be finished (a desirable deadlock is a situation where the process instance has finished its progress and the instance can not be reinvoked again).

[7] Note that $|t_{query}| = \Sigma_{i=1}^{N} |\text{SynSet}(t_{raw_i})|$, where $|t_{raw}| = N$.

where t_{ik} is calculated by the following equation:

$$t_{ik} := \left\{ \begin{array}{cc} 0 & (p_{ik} = 0) \\ 1 & (p_{ik} >= 1) \end{array} \right\}$$

t_{ik} is 0 if the user i has never selected the process k; otherwise it is 1.
The ranking r_{uk_2} for the number of selections of all users is calculated by:

$$r_{uk_2} := \frac{\sum_{i \in U} p_{ik}}{1 + \sum_{i \in U} \sum_{j \in P} p_{ij}}$$

The range of this value is $[0, 1)$. The score freqScr for a user u selecting a process p can then by determined as:

$$\text{freqScr} := \frac{r_{uk_1} + r_{uk_2}}{2}$$

Imagine the second process in Figure 2 has been selected more often than the first one (e.g., 5 vs. 3 times). After reranking (due to the frequency) the recommender system would list the process *check client order* higher than the process part *check client offer*.
The covered fitting recommendations are further reranked by the criterion of structural correctness [17]. Syntactically correct processes are ranked higher than process recommendations that will cause undesirable deadlocks in case of interconnection.
Due to capacity and resource restrictions we decided not to integrate a complete deadlock verification, which would be performed when searching for fitting processes and also whenever a user inserts a process fragment into her workspace. Instead, we favor only the verification of structural errors, which can easily be detected whenever the user inserts new nodes into her workspace. Structural errors are generally easy to find and correct. Generally, the score of the correctness degree depends on the relative number of structural errors where TP handles decrease the score less than PT handles (an OR-split is synchronized by an AND-join). A score of 1.0 for the correctness degree indicates a completely structurally correct recommendation. The penalty for structural conflicts is determined based on the frequency with which these conflicts occur for the considered process fragments. More formally:

$$\text{corrScr} := \left\{ \begin{array}{l} 1 - 0.1 * \frac{N}{|PT| + |TP|} \text{ for PT handles} \\ 1 - 0.2 * \frac{N}{|PT| + |TP|} \text{ for TP handles} \end{array} \right\},$$

where N is the number of all recommended process fragments, $|PT|$ the number of recommended process fragments which would result in PT handles if the user would insert them into her workspace and $|TP|$ is defined similarly for TP handles. If the equation would result in a negative value for the structural correctness metric, i.e. corrScr < 0, we define corrScr to be 0. The intuition is that possible PT and TP handles are punished more severely if they occur rarely and less severely if almost every recommended process fragment would result in a TP or PT handle if inserted into the workspace.
The overall ranking for recommendations results from the following weighted equation:

$$R := w_1 * \text{searchScr} + w_2 * \text{corrScr} + w_3 * \text{freqScr}$$

where $\Sigma_{i=1}^{n} w_i = 1$, and the Lucene score is assigned the greatest weight, i.e. the Lucene score has the most significant influence on the ranking.
The reranking of search results of Figure 2 gives the following final descending order as shown in Figure 3. Process elements, which cause (in case of interconnection) structural problems are highlighted with a gray rectangle. The choice

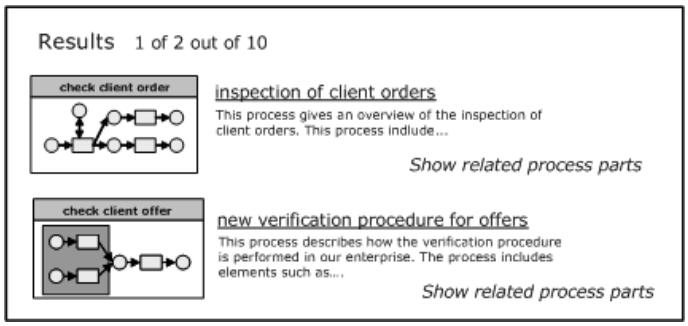

Figure 3. New order of recommendations after reranking.

for these ranking criteria is supported by our evaluation. The interviewed persons stated that element labels are the most important reason for choosing a recommendation followed by the process result and process structure. The user relevance feedback was unimportant. Instead the user appreciated the frequency score and the configuration window.
Furthermore, the standard process parts are element groups consisting of up to ten elements. From this point of view it makes no sense to consider the model size, the density of process elements or the average connector degree in the ranking, which have been identified by [19] as main factors for business process understandability.

7 Evaluation

To validate our modeling support system we conducted an evaluation. We completed our evaluation after the tenth person because the last four persons did not significantly change the recommendations. Instead they selected the processes edited by preceding interviewed persons. Among the ten interviewed persons were four beginners, two approved modelers and four advanced modelers. In the initial development of our prototype we have populated our repository with process models from real word projects concerning order and shipment processing. Additionally, we collected a set of process models from the research literature regarding the same application area. Before starting the evaluation the repository contained 21 process models including 15 process parts (which we manually

declared from the 21 process models).
For these processes we build a questionnaire that should answer the following questions:

- Can the modeling time be reduced using the modeling support system?
- Can the number of operations (deletion, insertion) be reduced when reusing processes from the repository?

Mainly, the interviewed persons had experiences for improvement and documentation purposes and stated that the most influences on their process modeling are (modeling/enterprise) goals and requirements. All interviewees asked that they are modeling from left to right[8]. Finally, most persons declared that they are searching in the WWW or ask the corresponding persons for relevant information in order to model the business process. This statements confirmed our presumption that users spent some time to find relevant information. We therefore conclude, that a search capability is beneficial for process modeling.
Next, in the questionnaire we asked the users to model three business processes. For the first and the third process we provided detailed information about the process solution. The second process instruction was short: *model a business process for order approval*.
Except one person all interviewees started their process modeling tasks with the search interface. Subsequently, they inserted either all or only some elements of a recommendation or even several elements from different recommendations into their workspace. Then they finished their process modeling or continued their searching. Table 1 shows the average search results for the three business processes to be modeled. The average of searches performed for the first process

Average Number of...	1st Model	2nd Model	3rd Model
... searches performed	1.8	1.0	1.5
... recommendations proposed	38	26.77	37.5
... recommendations viewed to find fitting process	5.3	3.55	3.6
... recommendations selected	1.7	1.22	1.6

Table 1. Overview of performed searches and requested recommendations.

model is almost two, for the second is one and for the third is the middle number. Next, the average of selected recommendations for the first and the third model converges to two and for the second model lies in the middle between 1 and 2.

We determined for all three process tasks the Pearson correlation coefficients. None correlation coefficient is significant at a 95% confidence level (see Table 2). The only *demonstrative* correlation can be stated for the third process instruction. If we differentiate the number of searches performed according to the user's

[8] As mentioned previously, the current implementation supports both modeling techniques (starting at the modeling trigger or at the modeling output).

modeling experiences, then modeling beginners posed more queries than advanced users. One reason could be unsatisfying recommendation results or their uncertainty which process to choose. Advanced modelers mostly decided for one recommendation and customized it. But, the number of valid cases (interviewed persons) is too small in order to make any generalizable claims.

task	**corr. parameter**	**corr. coefficient**	**p-value**
1	user exp. vs. # selected rec.	-0,0834	0,4093
	user exp. vs. # searches performed	0,0891	0,4033
2	user exp. vs. # selected recommendations	-0,2978	0,2017
	user exp. vs. # searches performed	0,0891	0,4033
3	user exp. vs. # selected recommendations	0,408	0,1208
	user exp. vs. # searches performed	0,5215	0,061

Table 2. Correlation parameters for the three modeling tasks in the questionnaire.

If the interviewees decided to use the query interface then the user could decide which recommendation to open. To realize this we prepended a table-based result list as depicted in Table 3 including the Lucene Score, the Frequency Score and the average number of operations (insertions and deletions) users have performed after adding the recommended process part into their workspace. To

#	Process Name	Score	Frequency	Operations
1	CheckOrder	95.02	5	15
...	...	...	...	...
10	Handle Customer Order	48.85	3	20

Table 3. Ranked process recommendations.

control the number of configurations we adopted the methods presented in [20] for version control of workflow process definition. This prepended representation of query results has two advantages. (1) Several users posed only *meaningless* queries such as searching for elements labeled *received order*, which is modeled in a variety of processes. Thus, this representation helps the user to find fitting processes (due to the process name) even when the query was non-declarative. (2) This representation is highly efficient compared to the view of recommendations (like in Figure 3), which requires a lot more time to load all models. With this prepended representation only the selected recommendations will be loaded in the graphical view.
Figure 4 shows our evaluation results concerning our initial questions to be answered by the questionnaire. The modeling time and the number of operations

for the three modeling examples in the questionnaire decreases by the number of interviewee. The first person spent the most time to find a suitable process for reuse and customized the recommendations. The following interviewees reused the processes of the first person and edited them slightly. Subsequently, the last six interviewees adopted the processes of their predecessors with minimal modifications.

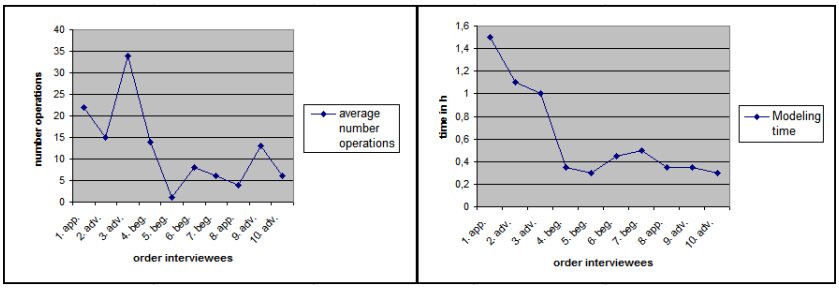

Figure 4. Reduction of modeling time relative to interviewed persons.

We determined the Pearson correlation coefficients for the correlation parameters time vs. number of processes and process parts in the repository and number of operations vs. number of processes and process parts in the repository. Both correlation parameters are significant at a 95 % confidence level (with a correlation coefficient of -0,82 and -0,6). Consequently, we conclude that our support system reduces modeling time and the number of operations, if suitable business processes are available in the repository.

8 Conclusion and Future Work

We presented a system for supporting users at modeling time which is focused on reducing both the modeling time and increasing the structural correctness at an early stage by providing a search functionality for process fragments stored in a repository. Additionally we proposed a novel recommendation algorithm which ranks process fragments based on three different criteria. First, a modified version of the Term and Document Frequency measure, which has been adapted for business processes. Second on the reuse of process fragments and third on the structural correctness of process parts. Our user evaluation suggests, that the recommender system reduces the number of required editing operations and of the modeling time.
For future work we plan to investigate a more elaborate multi-stage matching procedure, e.g. combining the Term and Document Frequency with the results of a Naïve Bayes classifier on process instances by using a voting prediction combiner (cf. [21]). Furthermore, as already mentioned in the related work section we

plan to tag user guides or project documentations to additionally unveil user's requirements.

References

1. Bandara, W., Gable, G.G., Rosemann, M.: Critical Success Factors of Business Process Modeling. Technical report, Preprint series of Queensland University of Technology (2007)
2. Madhusudan, T., Zhao, J.L., Marshall, B.: A Case-based Reasoning Framework for Workflow Model Management. Data Knowl. Eng. **50** (2004) 87–115
3. Kim, J.H., Suh, W., Lee, H.: Document-based Workflow Modeling: a Case-based Reasoning Approach. Expert Syst. Appl. **23** (2002) 77–93
4. Paul, Costache, S., Nejdl, W., Handschuh, S.: P-TAG: Large Scale Automatic Generation of Personalized Annotation Tags for the Web. In: WWW, New York, NY, USA, ACM Press (2007) 845–854
5. Ghose, Aditya; Koliadis, G.C.A.: Process Discovery from Model and Text Artefacts. IEEE Congress on Services (9-13 July 2007) 167–174
6. Mahleko, B., Wombacher, A.: Indexing Business Processes Based on Annotated Finite State Automata. International Conference on Web Services (2006) 303–311
7. Shen, Z., Su, J.: Web service discovery based on behavior signatures. In: IEEE International Conference on Services Computing, IEEE Computer Society (2005) 279–286
8. Weijters, T., van der Aalst, W.: Process Mining: Discovering Workflow Models from Event Based Data. In: BNAIC. (2001) 283–290
9. Skoutas, Dimitrios; Simitsis, A.S.T.: A Ranking Mechanism for Semantic Web Service Discovery. IEEE Congress on Services (9-13 July 2007) 41–48
10. Xu, J., Li, H.: AdaRank: a Boosting Algorithm for Information Retrieval. In: ACM SIGIR, ACM (2007) 391–398
11. Brooks, C.H., Montanez, N.: Improved Annotation of the Blogopshere via Autotagging. In: WWW, Edinburgh, UK (2006)
12. Efthimiadis, E.N.: A User-centred Evaluation of Ranking Algorithms for Interactive Query Expansion. In: ACM SIGIR, ACM (1993) 146–159
13. Salton, G., Mcgill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA (1986)
14. Zipf, G.K.: Human Behaviour and the Principle of Least-Effort. Addison-Wesley, Cambridge MA (1949)
15. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In: Proceedings of IJCAI-03 Workshop on Information Integration on the Web. (2003) 73–78
16. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
17. van der Aalst, W.M.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers (1998) 21–66
18. Ohsugi, N.; Monden, A.M.K.: A Recommendation System for Software Function Discovery. APSEC (2002) 248–257
19. Mendling, J., Reijers, H., Cardoso, J.: What Makes Process Models Understandable? In: BPM. LNCS, Springer (2007) 48–63
20. Zhao, X., Liu, C.: Version Management in the Business Process Change Context. In: BPM. LNCS, Springer (2007) 198–213
21. Bozovic, N., Vassalos, V.: Two-Phase Schema Matching in Real World Relational Databases. In: ICDE Workshops. (2008) 290–296