Towards Auto-Suggested Process Modeling – Prototypical Development of an Auto-Suggest Component for Process Modeling Tools

Nico Clever, Justus Holler, Maria Shitkova, Jörg Becker

ERCIS – University of Muenster
Leonardo-Campus 3
48149 Muenster, Germany
{ nico.clever ¦ justus.holler ¦ maria.shitkova ¦ joerg.becker }@ercis.uni-muenster.de

Abstract: Companies have accepted process modeling as a powerful instrument for business reorganization, requirements specification in software development, knowledge management and other activities. Huge amounts of processes are being modeled in organizations nowadays. However, the re-use of existing process knowledge in order to simplify the modeling process has not yet been thoroughly studied and applied. We propose an auto-suggest component for process modeling tools, which, based on existing process knowledge, "auto-suggests" the process model elements step-by-step, thus saving the modeler time and effort.

1 Introduction

Business process management (BPM) has established itself as a mature field of reseach during the last decades. There are several commonly used process modeling notations [Ag04] and process modeling tools as a part of complex business process management suites supporting modeling projects [Ga10]. Companies have acknowledged process modeling as a powerful instrument for business reorganization, requirements specification in software development, knowledge management and other activities [In09]. As a result, large process model collections are created in organizations nowadays, such as APROMORE (2000 models with 20-100 tasks in each) [LR11], SAP reference model (600 models) [Me08] or the repository of Dutch local governments counsil (500 models) [Di11].

Given the size of process model collections and single process models, manual creation of all processes within the repository becomes a tedeous, time-consuming, and, thus, inefficient task. Therefore, we argue that the existing process knowledge from already created processes in the organization as well as from reference models should be re-used to facilitate modeling activities and enhance the efficiency in tems of model quality, modeling time and budget.

Driven by the research question, *how process modeling efficiency can be improved by using existing process knowledge*, we propose an auto-suggest component for process modeling tools. Similar to the T9 system for mobile phones or the auto-completion

function in a UNIX shell or web-search [In09], the active modeler is supported by automatic recommendations for the next process step. These recommendations are derived logically from a knowledge database filled with the process models created in the own or reference organizations.

The remaining paper is structured as follows: within the next section we describe the research design applied, followed by a literature review and the conceptual design of our auto-suggest prototype. The implementation and application section specifies the design decisions. In the last section we discuss the applicability of the conceptual design and prototypical implementation of the auto-suggest component and related future research and ongoing evaluation respectively.

2 Research design

In this article we follow the Design Science Research Methodology (DSRM) proposed by Peffers et al. [Pe07]. The DSRM is based on five consecutive phases depicted in Figure 1, namely problem identification and motivation, definition of the objective of the solution, design, demonstration and evaluation. Peffers et al. [Pe07] mention a sixth phase, communicating the research results, which is not depicted in the figure. This step is achieved by the ongoing research contributions.

In the first phase, we identify and motivate the problem by conducting a literature review [Vo09, WW02]. We used "auto-completion", "auto-suggest" and "recommendation-based" as keywords to find existing approaches in the area of process modeling facilitation and selected those articles which had an explicit focus on BPM. On the basis of the literature review, we found that this topic is not widely discussed and present in the research literature at the moment. Based on the research gap, we define the objective of our solution.

In the design phase, based on auto-suggest approaches in different IS fields such as software engineering, web design and mobile application design, we develop a conceptual model of the auto-suggest component for a BPM tool prototype [Be13]. This prototype was then implemented as a web-based process modeling tool to demonstrate its applicability and practicability.

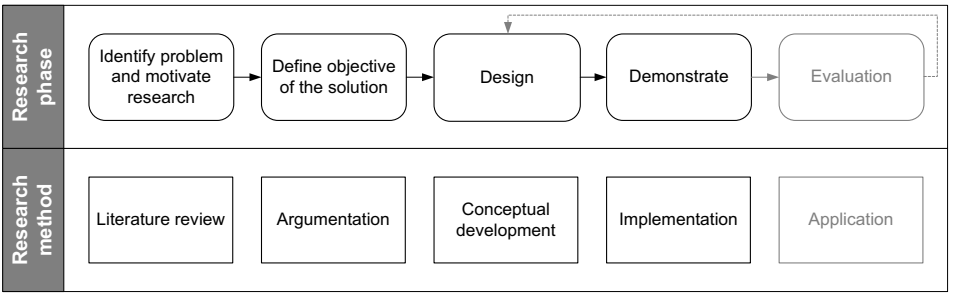

Figure 1: Research methodology

The last phase of the DSRM is greyed out in Figure 1: Research Figure , as the prototype evaluation is not addressed in this paper. The reason for it is the necessity of “learning" of the auto-suggest component, which will be discussed later in the paper. The evaluation is taking place at the time of this contribution and the results will be published after the evaluation is finished. Based on the evaluation results, we will address the improvement of the prototypical implementation.

3 Related work

In [HKO07], one approach for auto-completion based on business rules and structural constaints is presented. It uses Petri-nets described with the web ontology laguage OWL [MV04], which makes it possible to implement an efficient algorithm for (semi-) automatic similarity computation between process variants. Additionally, the semantic web rule language (SWRL) [Ho04] is used to describe structural and business-constraints. According to predefined rules, the system can automatically select process fragments that are then inserted at a particular step of the process model. The prototypical implementation of the conceptual idea from [HKO07] was presented in [HKO09]. The authors have extended the OWL description with two properties and created 18 SWRL rules which allow for automatic detection of suitable process fragments.

A recommendation-based modeling support system is proposed in [HKL08]. The system consists of a query interface, recommender and ranking functions and aims at re-using existing process fragments and, thus, at facilitating process modeling. Process fragments are described by tags or key words, which are derived from process descriptions according to the score function. [KO11] extends this approach by additionaly integrating factors which influence process modeling (modeling purpose, view, role, model properties and complexity). The solution also includes a collaboration component – the recommender system tracks the users who have already utilized the recommended process fragment and shows this information in the query results screen which supports unexperienced modelers in their recommendation choice.

[Be06] presents a conceptual description of an approach for auto-completion of business process models based on the OWL which allows for an automatic analysis of process models. During the modeling process, a recommendation mechanism determines possible subsequent fragments of all templates by computing similarities. The system also ensures structural correctness of the models when suggesting fragments (check for deadlock freeness and soundness of the resulting model). The prototypical implementation of the approach is not yet finished or presented by the authors.

In [MP08], an alternative approach for reusing business process fragments is proposed. The authors construct a formal model for the description of process models using a π-calculus for dynamic aspects and an ontology stack for static aspects of process models. The authors have not yet finished the querying framework, so the approach is not completely implemented.

Table 11. Auto-suggest approaches in literature

Source	Modeling language	Degree of automatisation	Technology used	Impl.
[HKO07]	Petri nets	Semi-automatic	Ontology, structural correctness rules	no
[HKO09]	Petri nets, (possibly applicable to WS-BPEL, EPC)	Semi-automatic	Ontology, structural correctness rules	(yes)
[HKL08] [KO11]	Petri nets	Manual query	Key words	yes
[HKL08] [KO11]	Petri nets	Semi-automatic	Key words, structural correctness rules	(yes)
[Be06]	Petri nets	Semi-automatic	Ontology, structural correctness rules	no
[MP08]	On the example of BPMN	Manual query	Ontology, π-calculus	no

In Table 11, the approaches to auto-suggest process modeling described above are summarized. All of the approaches focus on the suggestion of process fragments, and not single elements. Most of the solutions, except [HKO09] and [MP08], are targeted to Petri nets, as this language has a formal representation which allows for checking the suitability of suggested fragments according to structural correctness properties such as deadlock-freeness or soundness. Most of the approaches are based on representing a process model with the web ontology language OWL, but two of them, [HKL08] and [KO11], use key words for describing the models. It has to be stated that, except for the query interface by [HKL08] and [KO11], the implementation of the approaches is either not performed or not clearly presented in the articles.

Thus, based on the literature review, we argue that there is still a gap in the area of auto-suggest functionality for process modeling. There is no universal language-independent approach to facilitate manual process modeling, which re-uses existing process knowledge and makes real-time suggestions during model creation. In the next section, we present a conceptual design for such an auto-suggest component for process modeling.

4 Conceptual design of the auto-suggest component for process modeling

4.1 Architectural design scenarios

Concerning the design of the auto-suggest functionality for process modeling, three different scenarios are applicable. The scenarios differ in their degree of specialization in favor of a concrete modeling environment and, thus, directly affect the flexibility of the

proposed solution. The first scenario seeks for a direct integration of the auto-suggest component with an existing process modeling environment. The second and third scenarios favor generic, environment-independent solutions, either based on a specific modeling language or not.

The first implementation scenario relies on a complete integration of the auto-suggest component into the modeling environment (Figure 2). This scenario does not require or make use of an external data source. It solely uses the existing models in the modeling environment or a predefined exemplary set of models. Due to the close integration with the modeling language and tool, semantic characteristics can be considered for providing the suggestions. For example, models in the notation of the Event-driven Process Chain (EPC) in its original form consist of alternating events and functions [Sc00]. When providing a suggestion for an element of type „event“, all the „events“ will be excluded from the suggestion set. Since no interface to an external database, e. g. other modeling environment has to be designed, no non-trivial generalizations of heterogenous process information is necessary on the one hand. On the other hand, this is a clear disadvantage of this solution, since only process elements can be transferred between two instances of the same modeling environment via export and import and, therefore, the reutilization of existing knowledge is hindered.

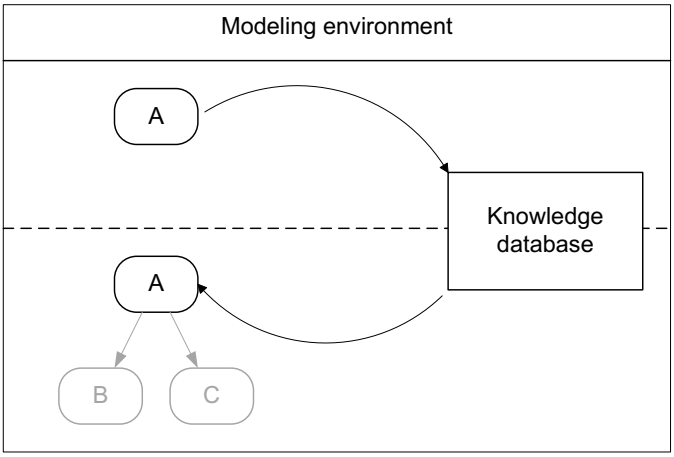

Figure 2: Completely integrated auto-suggest architecture

In the second scenario, the business logic and database of the auto-suggest component are outsourced from the actual modeling environment to a self-contained application. Both the modeling environment and the auto-suggest component communicate via a specifically designed interface. However, since the suggestions are – as they are in the integrated scenario – provided for one and the same modeling language, specific structural and semantic characteristics can still be utilized. As shown in Figure Figure 3, the architecture of this solution allows for the outsourcing of the auto-suggest component to an external server which, in turn, simplifies the exchange of knowledge between different process repositories.

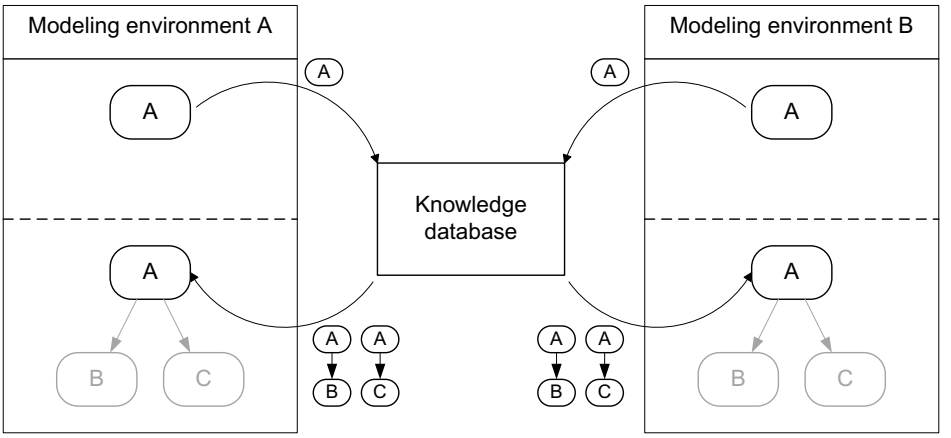

Figure 3: Self-contained, language-dependent auto-suggest architecture

In the third scenario, the auto-suggest component is not only self-contained but language-independent, as well. Here, only the denotations of the process elements – also called labels – are transferred to the knowledge base and used for suggestion. Unlike in the other scenarios, utilization of the language-specific semantic or syntactic characteristics for suggestion provision is not possible. A precondition for the conjoint utlization of such a central language-independent knowledge base for several different modeling environments is a consistent use of denomination conventions/patterns for process elements. In

Figure 4**Fehler! Verweisquelle konnte nicht gefunden werden.**, the general architecture of such a central process modeling knowledge base scenario is shown. As outlined in the figure, the semantics incorporated by the previous solutions – in terms of concrete model elements represented by rounded rectangles – are left out. This represents storing process modeling knowledge in the external database purely based on the labels of the process elements.

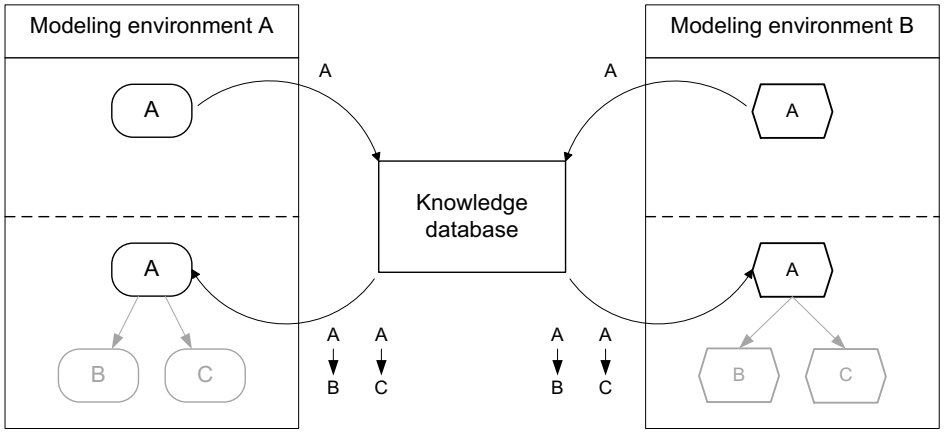

Figure 4: Self-contained, language-independent auto-suggest architecture

4.2 Additional requirements

Next to selecting the architechtural design scenario for the prototype implementation, some additional requirements have to be considered. These requirements will be discussed in this section.

First of all, the selection process of a potential suggestions has to be determined. Several criteria are relevant in this case:

- Frequency of the sequence – an element would be selected if it is the successor of the predecessor in a large set of models in the knowledge base.
- Date of the insertion – an element would be selected depending on whether it was lately used as the successor of the previous element and would not be selected if it has not been used in a specific timespan.
- Previous users of the suggestion – following [KO11], social structures of process modeling is an important aspect to consider. Thus, entries from experienced users or own entries could be selected preferably.
- Frequency of suggestion-adoptions – a ratio between how often a successor was suggested and how often it was actually accepted as a valid process model element. The higher the ratio, the more likely an element will be suggested. After a certain threshold the element would not be suggested anymore.
- The similiarity of element labels – calculated according to the Levenshtein distance[3] between two labels without taking semantic peculiarities of certain modeling languages into account [HM11].

On the basis of these criteria, learning of the knowledge database is realized. It can be divided into two phases. In the first phase, process models and their elements are loaded into the database by transferring predecessor-successor-pairs of elements. In the second phase, the frequency for suggestion (score) of these pairs is continuously adjusted as soon as new pairs of connected elements are created, or the existing ones are deleted in the modeling environment. For our prototype, a rather basic selection methodology is chosen. It is a combination of two aspects mentioned above, the frequency of usage and the date of the last usage counted in days. This is done to prove the applicability of such a selection method, but at the same time to be able to evaluate the learning functionality of the prototype in a comprehensible way. Thus, the selection formula is:

$$S(h,a) = h * e^{-\frac{\log 2}{365} * a}$$

S represents the score assigned to a pair while h represents the frequency of usage and a represents the date of the last usage counted in days. In simple words, this exponential function can be interpreted in the way that the score of a certain connection is cut in half if it has not been used in a year.

Two additional aspects have to be considered prior to the development of the prototype. First, handling of sensitive data plays an enormous role in the development of an auto-suggest component for process modeling. Since business process modeling is often a

[3] The Levenshtein distance is a metric, measuring the distance between two string sequences.

matter of highly sensitive organizational information, the possibility of restricting the access to intra-organizational knowledge has to be accounted for in the concept.

Second, the extent of suggested elements is of high importance. Since the aim of auto-suggested process modeling is to save the modeler time, it would be suboptimal if too many suggestions would be provided by the tool. Hence, the challenge is to provide a number of choices which is as big as necessary and as small as possible. Based on the calculation of the weighted score $S(h, a)$, two approaches are feasible. First, a threshold could be defined which has to be exceeded by an element in order to be suggested. Second, a fixed number of suggestions provided by the tool could be defined ordered by the decreasing score $S(h, a)$. Advantage of the first method is a well-formed result set matched to the modeler's needs if the threshold is well-chosen and the quality of the knowledge in the database is good. Pitfall of the first method is a possibly too broad result set if the initialing element is an often used one. This, in turn, is not the case if the second method is applied. If an adequate number of elements to be suggested is defined[4], the modeler can get an overview quickly and choose the fitting suggestion. Pitfall of this method could be that the modeler could be provided with suggestions which are of no interest in the explicit modeling context. For our prototype, we chose the latter method, because its pitfall of possibly providing unfitting suggestions is more acceptable than an excess supply of suggestions. However, extensive evaluation of the prototype will be carried out in order to assess and possibly correct the decisions made in the conceptual design phase of the auto-suggest component.

4.3 Architecture of the auto-suggest component

Given the multiplicity of process modeling tools and notations, as it was stated in the introduction of the article, the self-contained, language-independent approach is chosen for the auto-suggest component. Since it is detached from a specific modeling language, the application of the auto-suggest functionality is possible in a much broader context and is not limited by language-specific characteristics. Thus, a more meaningful application with respect to the potential of auto-suggest based process modeling itself is enabled.

Due to the choice of a self-contained solution, the auto-suggest component has two main tasks. On the one hand, it has to provide an interface to the process modeling tools which are delivering the learning input. As a matter of course, the tools which are to benefit from the functionality and to be included in the evaluation, have to match this interface when providing the information. On the other hand, it has to provide the business logic for the selection of the recommended successors of process elements. As an additional third, trivial task, the component serves as a process knowledge storage, containing all the information about existing internal and reference processes. The architecture of the resulting auto-suggest component and its interfaces is depicted in Figure 5.

.

[4] Adequate in this case always depends on the modeling purpose and the modeling context. Furthermore, a comprehensive evaluation of the prototype has to be carried out in order to provide a general statement in this context. Thus, no generally appicable rule can be provided at the current point.

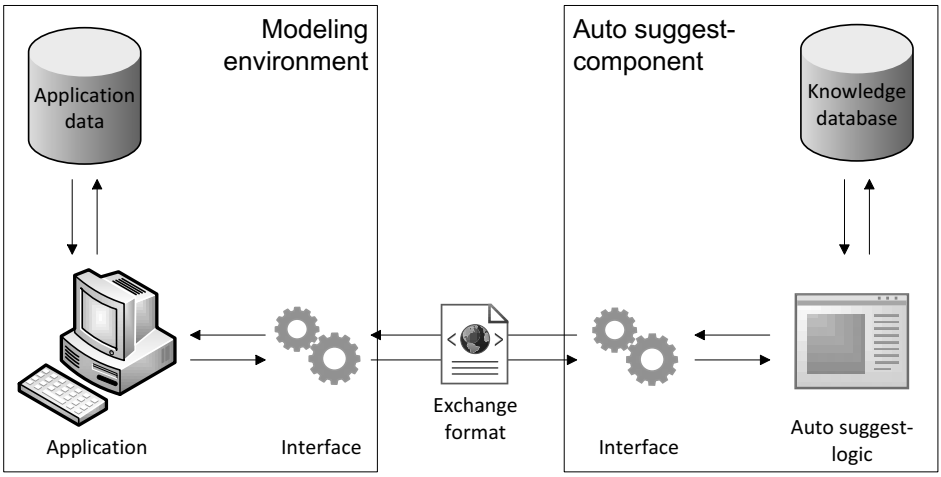

Figure 5: Architecture of the auto-suggest component

The data exchange format between the modeling environment(s) and the auto-suggest component, should account for three requirements. First, it has to be simple to be integrated into the interfaces of various modeling environments allowing for a wider application of the approach. Second, it has to be powerful enough to transfer all the necessary information. Third, in consideration of the excecution time requirements of auto-suggested process modeling it has to allow for a short processing time of a suggestion-request and the respective response from the component. As the de-facto standard for data exchange on the web fulfilling all of these requirements, the JSON (JavaScript Object Notation) format is chosen for our prototype [Nu09].

5 Implementation and application

As pointed out in the conceptual design, our auto-suggest component for process modeling tools prototype is a self-contained, modeling language-independent web application. It can be viewed as a client-server application, where the auto-suggest component acts as a server and the associated modeling tools are clients. The application provides the following functionality:

- Web-interface: To respond to requests from the modeling tools, the application has a web-interface which provides the suggestions via the JSON-based exchange format.
- Knowledge database: To store the information about the process elements which is delivered by the modeling environments, the prototype possesses a high-performance database. The interface to this database allows for an efficient processing of the incoming requests, e. g. either storing information or accessing existing knowledge.

- Auto-suggest logic: To assess the quality of the suggestions and provide an adequate and fitting amount of recommendations, the prototype features the auto-suggest logic with the characteristics described in the previous section. Thus, it calculates the score of a certain suggestion based on the formula $S(h,a) = h * e^{-\frac{\log 2}{365} * a}$, which is dependent on the frequency of recommendation usage and the date when the recommendation was used for the last time. To put it simply, the score is halved if a connection has not been used in the last 365 days. This approach provides only a predefined number of suggestions to keep the time-saving effect for the modeler as high as possible. Furthermore, it accounts for sensitive (organizational) information as it allows the connections to be flagged for usage in a specific modeling environment only.

To enable the usage of the prototype, modeling environments have to be connected to the prototype in order to facilitate the learning of the knowledge base and to evaluate the effect of the auto-suggest functionality on the modeling time. Hence, a web-based modeling tool should be extended by an interface to communicate with the auto-suggest component. Likewise, it provides the following functionality:

- Obtain suggestions: For obtaining suggestions, the interface recognizes the label of the process element and sends a request for suggested elements to be placed after this element.
- Maintain connections: The interface is able to provide the auto-suggest component with the data set from the modeling environment to account for an initial knowledge database setup. Furthermore, it provides the component with changes in the modeling environment, either by inserting new connections between process elements or by changing the existing ones.
- Display suggestions: The modeling environment displays the suggestion provided by the auto-suggest component by outlining them in their potential position.
- Apply suggestions: The interface of the modeling environment allows for inserting the suggested element if the user accepts a certain suggestion.

Figure 6 shows the information flows between modeling environment and auto-suggest component. First, the user selects a process element in the modeling environment (1). The modeling tool then transfers this information to the web-interface (2) which, in turn, converts the information to the JSON-based exchange format and sends it to the auto-suggest component (3). The interface of the component converts the data to the native format and passes it to the auto-suggest logic (4). A query to the knowledge database is constructed to find possible successors of the element (5). The database responds with a set of possible recommendations (6). The auto-suggest logic calculates the scores for each returned element and passes the results via the web-interface (7) to the web-interface of the modeling tool (8). The data is converted back to the native format of the modeling tool and proceeds it to the application (9). The suggestions are then shown to the user (10). If the user decides to accept a suggestion (11), the further steps are carried out. As this is optional, i. e. that the user does not have to accept the suggestion, the further steps are denoted in brackets in the figure. The accepted suggestion is processed

by the modeling tool and stored in the process repository (12). The resulting process model is then shown to the user (13). In parallel, the information that the user has accepted the suggestion, is passed to the auto-suggest component via the web-interfaces (14, 15 and 16). The auto-suggest logic then increases the frequency of this particular connection in the underlying knowledge base (17).

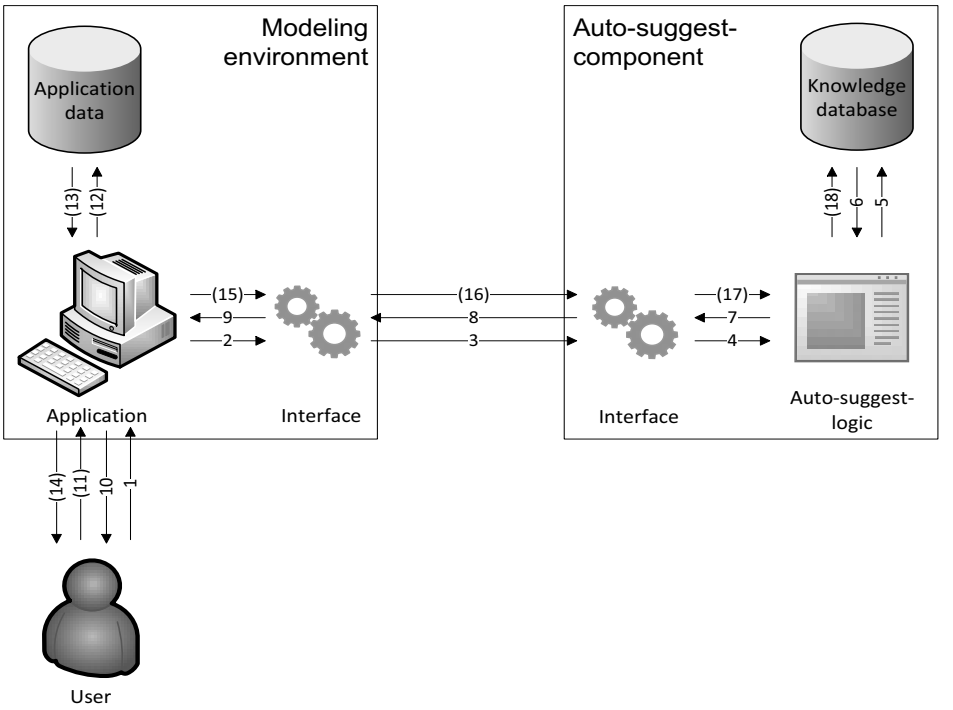

Figure 6: Prototypical information flow of the suggestion process

6 Discussion and further research

In this article, we have proposed an auto-suggest component for process modeling tools. An introduction of such a component should facilitate process modeling activities by supporting the user in manual creation of the process models.

We have designed our prototype as a self-contained, modeling language-independent web application with a rather simple recommendation mechanism to prove its applicability. We are going to extend and improve the recommendation mechanism after the evaluation of our solution has finished. The evaluation of the recommendation mechanism and the overall protoype is taking place at the time of this contribution and first results will be available in the near future. At the current state we can say that the protoype successfully demonstrates the concept's feasibility. As the perceived usefullness has to be evaluated by users in concrete modeling settings, we cannot yet tell if the research goal is fully reached. To prove it, we need to access the change in

modeling speed, quality and overall usability of the solution. Therefore, we are planning to conduct a number of experiments to measure time spent with and without the auto-suggest component. Furthermore, we want to elaborate on possiblities of increasing comparability of process lables and, therefore, process elements themselves. Possible approaches to solving this problem can be found in [HKO09, HKL08].

Bibliography

[Ag04] Aguilar-Savén, R. S.: Business process modelling: Review and framework. In International Journal of Production Economics. 90, 2004; pp. 129–149.

[Be06] Betz, S.; Klink, S.; Koschmider, A.; Oberweis, A.: Automatic user support for business process modeling. In Proceedings of the Workshop on Semantics for Business Process Management. 2006; pp. 1–12.

[Be13] Becker, J.; Clever, N.; Holler, J.; Püster, J.; Shitkova, M.: Semantically Standardized and Transparent Process Model Collections via Process Building Blocks. In Proceedings of the The Fifth International Conference on Information, Process, and Knowledge Management - eKNOW 2013. Nice, 2013; pp. 172–177.

[BKR11] Becker, J.; Kugeler, M.; Rosemann, M.: Process Management: A Guide for the Design of Business Processes. Springer, Berlin, 2011.

[Di11] Dijkman, R.; Dumas, M.; Van Dongen, B.; Käärik, R.; Mendling, J.: Similarity of business process models: Metrics and evaluation. In Information Systems. 36, 2011; pp. 498–516.

[Ga10] Gartner: Magic Quadrant for Business Process Management Suites. 2010.

[HM11] Haldar, R.; Mukhopadhyay, D.: Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach. In arXiv preprint arXiv:1101.1232. 2011.

[Ho04] Horrocks, I.; Patel-schneider, P. F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M.: SWRL : A Semantic Web Rule Language Combining OWL and RuleML. In W3C Member submission, 21, 2004; 79.

[HKO07] Hornung, T.; Koschmider, A.; Oberweis, A.: Rule-based autocompletion of business process models. In Proceedings at the 19th Conference on Advanced Information Systems Engineering (CAiSE). 247, 2007.

[HKO09] Hornung, T.; Koschmider, A.; Oberweis, A.: A recommender system for business process models. In 17th Annual Workshop on Information Technologies & Systems (WITS). 2009.

[HKL08] Hornung, T.; Koschmider, A.; Lausen, G.: Recommendation based process modeling support: Method and user experience. In Conceptual Modeling-ER 2008. 2008; pp. 265–278.

[In09] Indulska, M.; Green, P.; Recker, J.; Rosemann, M.: Business process modeling: Perceived benefits. In Conceptual Modeling-ER 2009. 2009; pp. 458–471.

[KO11] Koschmider, A.; Oberweis, A.: Designing Business Process with a Recommendation-Based Editor. In Data & Knowledge Engineering. 70, 2011; pp. 483–503.

[LR11] La Rosa, M.; Reijers, H.: APROMORE: An advanced process model repository. In Expert Systems with Applications. 38, 2011; pp. 7029–7040.

[Me08] Mendling, J.; Verbeek, H. M. W.; Van Dongen, B. F.; Van der Aalst, W. M. P.; Neumann, G.: Detection and prediction of errors in EPCs of the SAP reference model. In Data & Knowledge Engineering. 64, 2008; pp. 312–329.

[MV04] McGuinness, D. L.; Van Harmelen, F.: OWL web ontology language overview. In W3C recommendation. 10, 2004; 2004-03.

[MP08] Markovic, I.; Pereira, A.: Towards a formal framework for reuse in business process modeling. In Business Process Management Workshops. 2008; pp. 484–495.

[Nu09] Nurseitov, N.; Paulson, M.; Reynolds, R.; Izurieta, C.: Comparison of JSON and XML data interchange formats: A case study. In Computer Applications in Industry and Engineering (CAINE). 2009.

[Pe07] Peffers, K.; Tuunanen, T.; Rothenberger, M. A.; Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. In Journal of Management Information Systems. 24, 2007; pp. 45–77.

[Sc00] Scheer, A.-W.: Aris - Business Process Modeling. Springer, Berlin, 2000.

[Vo09] Vom Brocke, J.; Simons, A.; Niehaves, B.; Riemer, K.; Plattfaut, R.; Cleven, A.: Reconstructing the Giant: On the Importance of Rigour in Documenting the Literature Search Process. In Proceedings of the ECIS 2009. Verona, 2009; pp. 2206–2217.

[WW02] Webster, J.; Watson, R. T.: Analyzing the Past to Prepare for the Future: Writing a Literature Review. In MIS Quarterly. 26, 2002; pp. xiii–xxiii.