Data Model Development for Process Modeling Recommender Systems

Michael Fellmann, Dirk Metzger, Oliver Thomas

Data Model Development for Process Modeling Recommender Systems

Michael Fellmann[1], Dirk Metzger[2], Oliver Thomas[2]

[1] Rostock University, Institute of Computer Science, Germany
michael.fellmann@uni-rostock.de
[2] Osnabrück University, Information Management and Information Systems, Germany
{dirk.metzger, oliver.thomas}@uni-osnabrueck.de

Abstract. The manual construction of business process models is a time-consuming and error-prone task. To ease the construction of such models, several modeling support techniques have been suggested. However, while recommendation systems are widely used e.g. in e-commerce, such techniques are rarely implemented in process modeling tools. The creation of such systems is a complex task since a large number of requirements and parameters have to be addressed. In order to improve the situation, we develop a data model that can serve as a backbone for the development of process modeling recommender systems (PMRS). We systematically develop the model in a stepwise approach using established requirements and validate it against a data model that has been reverse-engineered from a real-world system. We expect that our contribution will provide a useful starting point for designing the data perspective of process modeling recommendation features.

Keywords: Enterprise Process Modeling, Recommender Systems, Requirements, Data Model.

1 Motivation and Relevance

Business process modeling and reorganization are still among the top-ten of relevant topics of today's CIOs [1]. However, the construction of semi-formal process models is even today, after two decades of research on business process modeling, a highly manual task involving substantial human effort. Regarding the modeling activity, effort is required to create models conforming to specified rules regarding the naming of model elements and the abstraction level of model elements. Concerning the naming of model elements, terminological problems are amongst the main problems when using conceptual (process) models [2]. Moreover, effort and difficulty arises due to the complexity of today's business processes. It might not be easy to figure out where to start modeling a process and where to stop and on which abstraction level to model [3, 4] since guidance in modeling is largely missing in current tools. These barriers call for process modeling support features, which assist users during process model-

ing and make suggestions how to complete a currently being edited process model. Such assistance functions are common features in programming environments (in terms of auto-completing e.g., Java code) or e-commerce systems (e.g., amazon.com). Although it has been demonstrated that assistance functions are beneficial in these domains [5] [6], assistance functions are not considered in commercial BPM tools. However, since recommender systems "generate meaningful recommendations to a collection of users" [7], development activities towards such systems should be given a priority in order to offer assistance functions in process modeling tools too.

Up to now, some proposals that lead to prototypical developments have been made in the area of recommendation-based process modeling [8] such as auto-completion approaches [9] [6] [10] [11] or auto-suggest features [12] [13] or recommendation methods for improving business process modeling [14]. However, these contributions rarely provide an explicit and detailed data model, so modelling the data perspective when building such tools has to start from scratch. It is not an easy task since a large number of requirements and parameters have to be addressed. To improve the situation and to fill this gap, we systematically develop requirements-based data model for process modelling recommender systems (PMRS) that can serve as a backbone for the development of modeling recommender systems. The development is based on a requirements catalog previously developed [15] from a literature analysis as well as from three different empirical studies that also involve business users.

The remainder is structured as follows. At first, we describe methodological aspects (Section 2). We then systematically construct the data model based on requirements before we present the integrated model (Section 3). We then critically review our model (Section 4) and finally end with a summary and conclusion (Section 5).

2 Methodological Considerations

We systematically develop the model in a stepwise approach using established requirements. These have been elicited from literature (for short: *R-Lit*), a survey among practitioners (for short: *R-Prac*), from a case study (for short: *R-Case*) and by demonstrating a prototypical system to real users (for short: *R-Prot*) that used it and commented on their experience. The detailed requirements elicitation is part of our previous work [15]. Since scientific progress is cumulative, we need to re-introduce these requirements throughout this paper in order to be able to construct our data model. The construction of the data model is done in a step-wise procedure. Thereby, requirements stemming from the before mentioned sources are used to derive partial data models. These models are then combined to a single integrated model. After the model has been constructed, we critically review our model by comparing it with a data model being reverse-engineered from a real-world system. This critical review leads to adaptations of the integrated model. Our research process is depicted in Fig. 1. Solid arrows mean "leads to", the dashed arrow means "go back and revise".

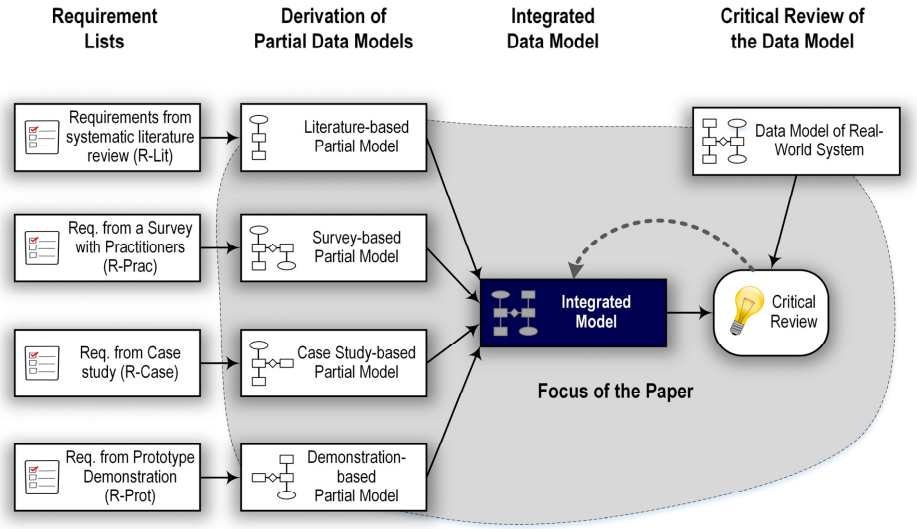

Fig. 1. Overview on Research Process Taken

3 Requirement-based Design of the Reference Model

In this section, we systematically derive our data model based on requirements. We use the term *implementation* to denote that a requirement is embodied in the data model, meaning the data model reflects the requirement.

3.1 Implementation of Requirements from Literature (R-Lit)

Requirements from literature have been elicited by reviewing related works found in the databases SCIENCE DIRECT, ISI WEB OF KNOWLEDGE, SPRINGER and EBSCO that in sum cover approx. 950.000 journals, books and conferences (see [15] for more details). Requirements fall into three broad categories: Requirements in regard to the *content* that is recommended, the *recommendation capabilities* and the *recommendation system*. We introduce each requirement briefly and comment on how these requirements are reflected in the data model. The data model in this section (cf. Figure 2) and subsequent sections are constructed using the well-established notation of Entity Relationship Diagrams that supports the specification of data structures on a conceptual level.

Content-related Requirements

R-Lit-1. *Recommendation of basic process model constructs.* The system should be able to recommend constructs such as elements, their structure and labels.

R-Lit-2. *Recommendation of additional process model constructs.* The system should provide recommendations for other constructs such as resources.

R-Lit-3. *Provide descriptive meta-information about the recommendations.* The system should provide relevant meta-information about the suggested elements.

R-Lit-4. *Provide provenance information about the recommendation.* The system should provide information to judge the quality of the recommendation.

We implement R-Lit-1 by introducing the entity *Basic Recommendation Element*, R-Lit-2 by introducing the entity *Additional Recommendation Elements* (cf. Figure 2). We generalize both by introducing a more abstract *Recommendation Element*-entity. In order to capture descriptive meta-information about elements that may also comprise provenance information (i.e. source of origin) as demanded by R-Lit-3 and R-Lit-4 respectively, we introduce the attribute *Metadata* and associate it to the entity *Recommendation Element*.

Capability-related Requirements

R-Lit-5. *Ensure recommendations with a high semantic quality.* The system should provide recommendations that are adequate and lead to a high semantic model quality.

R-Lit-6. *Flexible and easy application of recommendations.* The system should make it easy to work with recommendations and may guide the user in the selection of suggestions.

R-Lit-7. *Use personalization mechanisms.* The system should provide personalized recommendations tailored to the needs of the specific user in her specific modeling situation.

R-Lit-8. *Adjustable filtering options for recommendations.* The user should be able to adjust the filtering criteria for recommendations.

R-Lit-9. *Adjustable amount of recommendations.* The user should be able to adjust the amount of recommendations.

R-Lit-10. *Multiple recommendation strategies.* Recommendations should be determined using different calculation strategies in order to fit the user requirements.

The semantic quality of recommendations is mainly dependent on the concrete algorithm used to calculate recommendations as well as on the concrete instance data, so R-Lit-5 is not directly relevant for the data model. Likewise, R-Lit-6 cannot easily be reflected in the data model on account of its non-functional nature (e.g. due to terms such as “flexible” and “easy”).

In contrast to that, R-Lit-7 can be embedded in the data model by introducing the relation *Selection* between the entities *Recommendation Element* and *User*. In this way, decisions to include a recommended element in the model that a user has been made are recorded as “selections” and can be leveraged for computing future recommendations.

R-Lit-8 and R-Lit-9 are requirements that permit a user to adapt the system to his or her individual preferences. Hence, an entity *User Settings* has been introduced as a specialized form of a more general *Settings*-entity. Since selecting the best recommendation calculation technique as required by R-Lit-10 may be a matter that requires experience, this is probably best set by an expert user engaged in setting up the system. So R-Lit-10 has been attributed to this more general *Settings*-entity.

System-related Requirements

R-Lit-11. *Support knowledge base evolution.* The system should provide capabilities such as versioning, change management, importing new content or learning.

R-Lit-12. *Compatibility to existing tools and languages.* The system should work with existing modelling languages and in conjunction with existing tools.

We reflect R-Lit-11 by introducing a separate entity *Element-Set* that represents an arbitrary number of collections of elements (e.g. process activities to be suggested for several business domains). In addition, to provide for versioning and change management capabilities, the attribute *Metadata* is associated to *Element-Set*. R-Lit-12 demanding compatibility to existing tools finally is not eligible for representation in the data model. Figure 2 shows the derived partial model from the requirements along with the requirements that are depicted as graphical annotations.

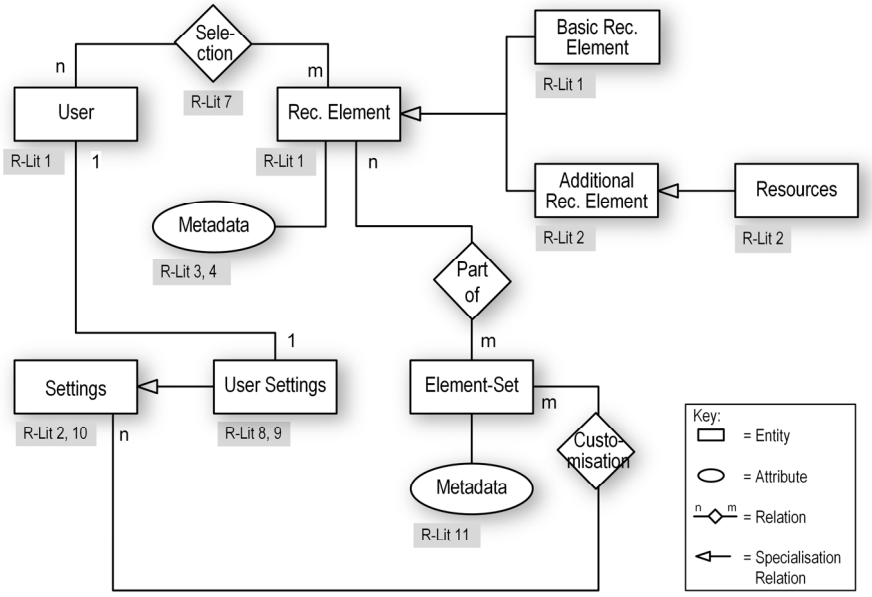

Fig. 2. Literature-based Partial Model

3.2 Implementation of Requirements from Practitioners (R-Prac)

The following requirements have been derived based on studies that involved 48 participants as described in more detail in [15].

R-Prac-1. *Various sources for recommendations.* The recommendation system should be able to generate recommendations from various sources.

R-Prac-2. *Provenance information.* The recommendation system should provide background information regarding the source and quality of a recommendation.

R-Prac-3. *Display of recommendations on request.* Recommendations should be provided when the user requests the system to do so.

R-Prac-4. *Multiple ways of displaying recommendations.* The recommendation system should provide multiple ways of displaying the recommendations varying in their degree of non-obtrusiveness.

We implement R-Prac-1 and R-Prac-2 by introducing an additional attribute *Metadata* and associating it with the entity *Element-Set*. In this way, the source and the origin of a recommendation can be recorded and exploited by the algorithm computing the recommendations. Adjustments to the provision of suggestions that a user or system administrator might want to set according to R-Prac-3 and R-Prac-4 can be stored in the *User Settings* and *Global Settings* respectively. Figure 3 shows the derived partial model from the requirements.

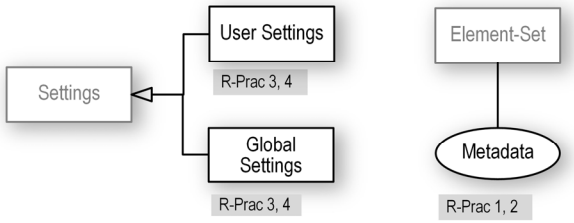

Fig. 3. Practitioner Survey-based Partial Model

3.3 Implementation of the Requirements from the Case Study (R-Case)

Requirements that have been collected involving 100 participants in a case study as described in more detail in [15].

R-Case-1. *Understandable recommendations.* Since one main positive aspect of using standardized activities has been that their interpretation is less ambiguous, the PMRSs should use such standardized activities.

R-Case-2. *Recommendation of "uncommon", innovative contents.* For example, the system may suggest activities that are executed typically in another industry and in that way inspire the process design.

R-Case-3. *Extension capability of the pre-defined contents.* To provide a remedy for missing activities, the recommendation system should include a feature to extend the internal knowledge base.

R-Case-4. *Benchmarking feature.* The system should facilitate benchmarking e.g. by suggesting Key Performance Indicators (KPI) or by enabling a comparison of KPI values.

R-Case-5. *Advanced model processing features.* The system should offer advanced features for the translation of models in multiple languages (e.g. process taxonomies such as PCF exist in different languages), to compare models or to show which area of enterprise activities they cover based on their semantics.

R-Case-6. *On/off switch and decent presentation of recommendations*. The system should be switched off easily and the recommendations should be presented decently.

We reflect R-Case-1 by the attribute *Metadata* associated to the entity *Recommendation Element* since additional information stored as metadata might help to provide understandable recommendations. Their "uncommonness" or innovativeness as requested by R-Case-2 however is an attribute of the data being stored in the *ElementSet* and as such cannot be reflected in terms of structures in the data model. Further, import features for new content as demanded from R-Case-3 as well as advanced model processing features required by R-Case-4 have to be implemented as part of the functionality of a PMRs and thus are not reflected in the data model. In order to provide a benchmarking feature as implied by R-Case-4, Key Performance Indicators are required since they form the basis of any comparisons. These factors can be suggested similar to other *Additional Recommendation Elements* and thus are introduced as a specialization of the latter. An on/off-switch required by R-Case-6 can be realized as part of the *User Settings* data. Figure 4 shows the derived partial model from the requirements.

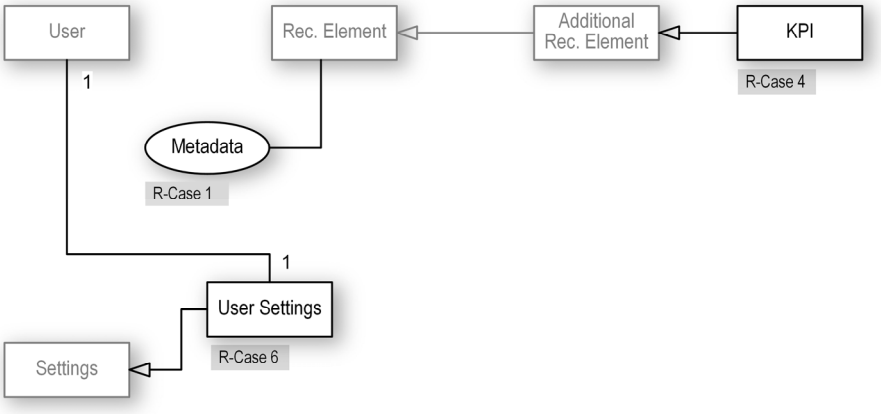

Fig. 4. Case Study-based Partial Model

3.4 Implementation of the Requirements from Assessing a Prototype (R-Prot)

This section describes the implementation of the following requirements that have been collected by assessing a prototype with 66 business users as described in more detail in [15].

R-Norm-1. *Recommendation of organizational units.* The system should recommend additional elements such as organizational units executing the activities.

R-Norm-2. *Recommendation of resources.* The system should recommend resources such as documents, tools or information systems.

R-Norm-3. *Customized specific taxonomies.* To make sure a plethora of potential use cases is covered, the predefined contents in the system should be customizable.

R-Norm-4. *Mobile version of the recommender.* Due to the fact that an increasing amount of work is done on the go, a mobile version should be offered.

R-Norm-5. *Interface to other systems.* Data inside the PMRSs used for recommendations such as taxonomies of pre-defined activities or organizational units should be updated frequently via interfaces to systems containing that data.

R-Norm-6. *Support multiple platforms.* As there are different platforms and architectures used in companies the support of the most important of them is needed to make sure the system gains acceptance.

R-Norm-7. *„Intelligent recommendations".* This requirement is more an overall characteristic of the whole system and demands that recommendations should be made on the right time in the right manner with adequate content.

R-Norm-8. *Show recommendation context.* The user of the system should be informed about the semantic context of a recommendation that is offered.

The requirement of suggesting additional elements as stated in R-Prot-1 and R-Prot-2 can be implemented by introducing the requested elements as specialized *Resource* entities *Documents*, *Tools* and *Information Systems*. The requirement of customizing the element collection used to calculate recommendations stated in R-Prot-3 can be reflected by the *Part-of*-relation between *Recommendation Elements* and *Element-Set*. In this way, the same recommendation elements may be part of different element sets (e.g. an element set for each industry the recommendation tool is used in). The usage of the PMRs in a mobile version (R-Prot-4) and its interfaces to other systems (R-Prot-5) as well as support for multiple platforms (R-Prot-6) are outside the scope of the data model. Also, "intelligent" recommendations (R-Prot-7) are an obligation of the algorithm that operates on top of the data. However, showing the recommendation context as requested by R-Prot-8 may be supported using the information stored in the attribute *Metadata* associated to the *Recommendation Element*-entity. Figure 5 shows the derived partial model from the requirements.

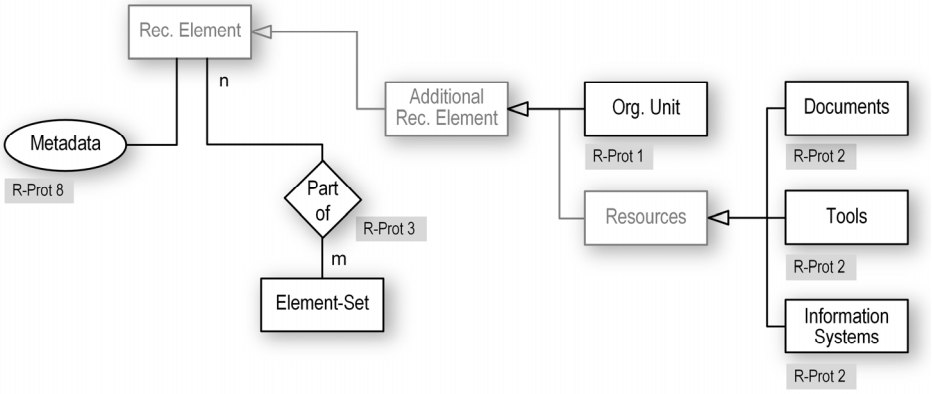

Fig. 5. Demonstration-based Partial Model

3.5 Integrated Data Model

The partial models that have been developed in the previous sections are integrated into a single model that is depicted by Figure 6.

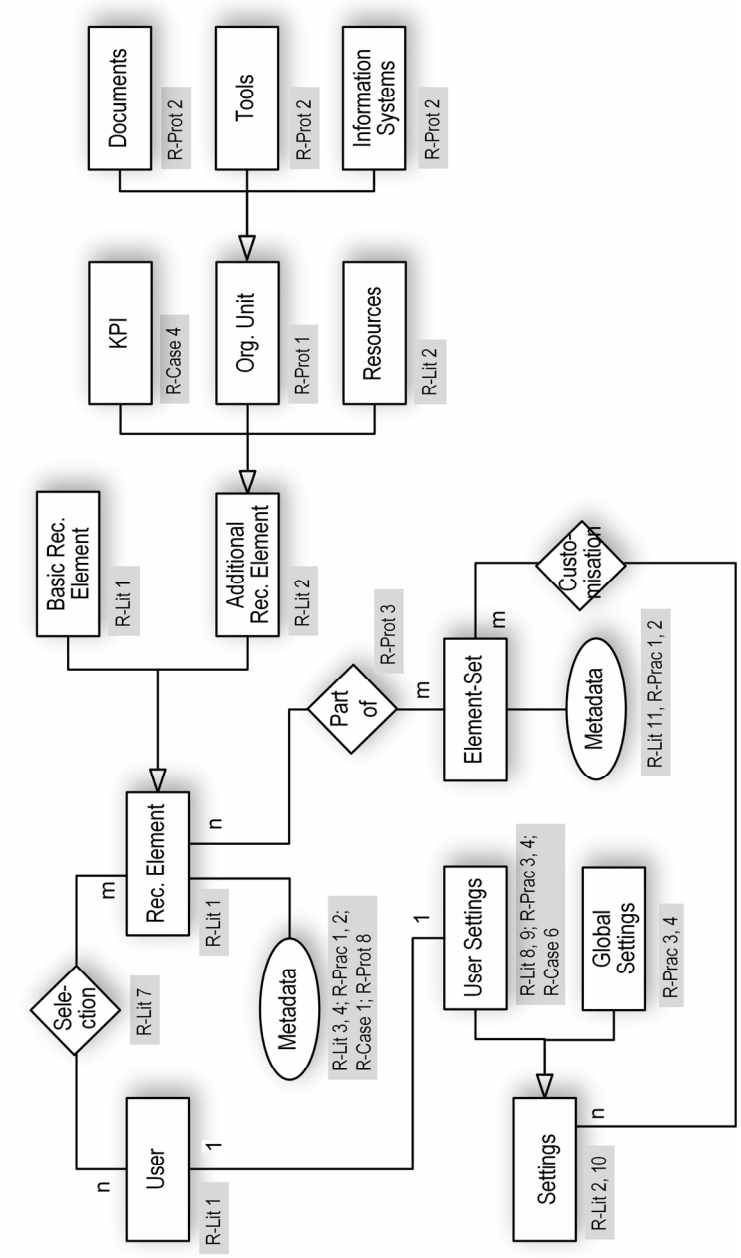

Fig. 6. Integrated Model

4 Critical Review and Refinement of the Information Model

In this section, we first introduce a real-world implementation of a PMRS. After this, we present the data model of the implementation in the next section.

4.1 Introduction of a Real-World PRMs Implementation

The prototype has been built with the purpose to design an artifact that is useable, testable and evaluable. We used web technologies on the frontend in conjunction with a server-based backend (PHP, MySQL). With this tool, a modeler is able to create simple linear process models (cf. Figure 7a) and use the recommending and autocomplete-feature of the system. The auto-completion feature (cf. Figure 7b) allows completing the labels of model elements based on keywords typed in an input field. The recommendation part of the system is capable of suggesting the next element (cf. Figure 7c). The system had been populated with process information (the labels and hierarchy of approx. 1000 enterprise functions) from the Process Classification Framework (PCF) in version 6 (see www.apqc.org), although other ontologies in semantic business process modelling [16] might also provide relevant contents.

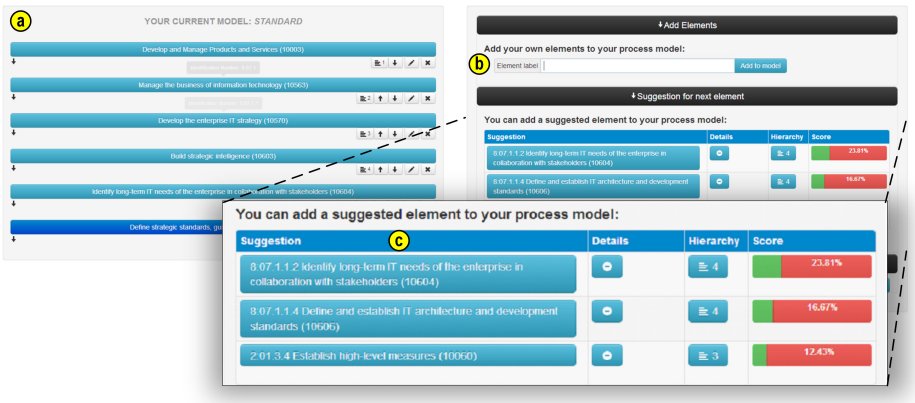

Fig. 7. User Interface of the PMRs Implementation

4.2 Data Model of the Implemented Tool

The tool was implemented initially to serve as a testbed and evolved over time. Hence the data model that has been derived systematically was not present in the beginning of the tool development. In this way, the data model of the real-world tool provides a good opportunity to review and refine the requirements-based normative data model leveraging the experiences from a real-world implementation. Figure 8 shows the data model which we reverse-engineered for the sake of our data model development. Shaded elements indicate the recommendation-specific parts; non-shaded parts represent data needed for more general modelling functionality.

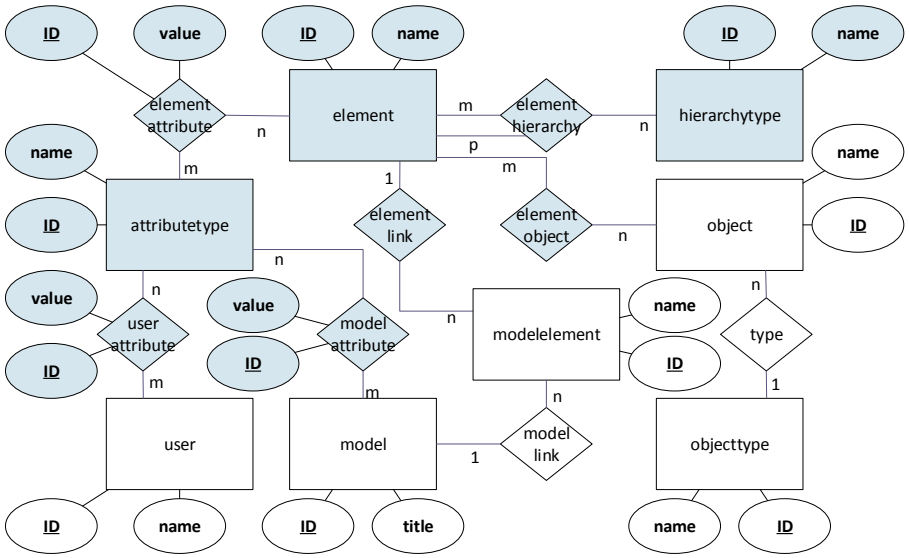

Fig. 8. Data Model of the Real-World PMRs

The following description (in italics) of the data model was given by the person who was involved in the development and was in charge of managing the data model. References to elements of the implemented PMRS inside the description are made using single quotation marks ‘’. We interrupt the description to comment on missing parts in our data model and begin such comments in a new line and enumerate them. For the sake of brevity, we do *not* comment on equivalences between the two models.

“The central aspect of the conceptual data model is the ‘element’ entity. It conduces as the central aspect for saving information about potential elements that could be used in a recommender system. As the recommending is mainly based on names the attribute ‘name’ is important. Furthermore, as in mostly all entities the use of an ID as primary key is compulsory. As one of the requirements has been to allow any kind of hierarchies, the entity ‘hierarchytype’ became part of the conceptual data model. First the information about what kind of hierarchy can be defined such as superior, inferior, equal, parent or child relations. In combination with the relation ‘elementhierarchy’ which combines two entities ‘element’ and an entity ‘hierarchytype’ the saving of generic hierarchies becomes possible. The design with the generic entity and the ternary relation is necessary to make sure individual terminologies as well as different kinds of relations are possible. Potentially the generic construct could be replaced with individual relations named after the kind of hierarchy (e.g. three relations named ‘follower’, ‘parent’ and ‘child’). However, in a conceptual data model generalization is preferred.”

(1) In our current data model, we cannot represent relations such as hierarchies or follower-successor relations. We therefore have to extend the model with an entity *Relation Type* and a ternary relation *Element Relation*.

"Another essential requirement of a generic conceptual data model is the need to save relevant metadata about an element such as the industry for which the element is specific, the context in the enterprise (e.g. production, HR), etc. To make sure that all potential values for arbitrary metadata are savable, this has been included in a generic way. Therefore, a new entity 'attributetype' is introduced. This entity represents meta-information about what types of information could be saved. Consequently, this entity will hold the name of the information (e.g. industry, context, etc.). The information itself will reside in the attached relation between the 'attributetype' and the 'element' which is called 'elementattribute'. It includes the value and, untypical for a relation, an ID as primary key. This makes sure that for a defined 'element' and one 'attributetype' more than one 'value' is saveable (e.g. multiple industries for an activity)."

(2) In our current data model, we cannot store metadata about *Recommendation Elements* and *Element-Sets* in a generic way. We therefore refine our data model and switch *Metadata* from an attribute to a separate entity type that is connected to both *Recommendation Elements* and *Element-Sets* via a relation. In this way, a generic mechanism for storing metadata is possible.

"For a modeling environment, the central entity is the model with its very own data such as 'title' and ID and potentially many more. It is connected through the relation 'modellink' to the entity 'modelelement' which saves the elements within the model with their name and with an ID. However, from the central entity 'element' a relation was built which bridges the recommending to the modeling system. To make sure that certain 'modelelements' that got inserted into a 'model' do not lose their link to an element proposed by the recommender this link is saved within the relation 'elementlink'. So, after inserting a recommended element into a model the conceptual data model will be able to record this."

(3) In our current data model, we cannot record the information that a user has accepted a recommendation and inserted an element in his or her model. Therefore, we have to extend our data model with the entities *Model* and *Model-Element*.

"Furthermore, to provide for the possibility that not only following 'modelelements' are suggested but also objects that are annotated to a certain 'element' (such as Organizational units and Resources), the entity 'object' has been inserted. With its 'objecttype', the relation 'type' and the relation 'elementobject' it provides the basis for saving complex objects to the suggestable elements. This provides the option to recommend objects.

Another part of the conceptual data model is a relation called 'modelattribute' between a 'model' and an 'attributetype'. The potential of the relation is to save information for a whole 'model' of the same kind an 'element' can have attributes of (e.g. industry, context, etc.). This guarantees that potential recommendation algorithms can take account of model information and adapt their recommendations."

(4) In our current data model, we can only specify settings with the help of the entities *User Settings* or *Global Settings*. We however cannot specify settings re-

lated to a model (e.g. the industry for which a model is constructed). We hence have to extend the data model with an attribute *Model Settings*.

"Equally to the 'modelattribute' the conceptual data model was extended to the relation 'userattribute' which connects the user from the modeling system with the 'attributetype'. For the same reason of saving special informations about a model the system is enabled to save information about the user (e.g. the user is from a special industry, context, etc.). This can also be integrated in the recommending system to make sure the recommendations are adapted accordingly."

4.3 Final Refined Data Model

In the following, we present the final refined data model (cf. Figure 9) that implements the changes identified above. Added/refined elements are depicted with shading and the number of the change identified above is added. Further, simplifications regarding the sub-types of recommendation elements (cf. Figure 6) have been made.

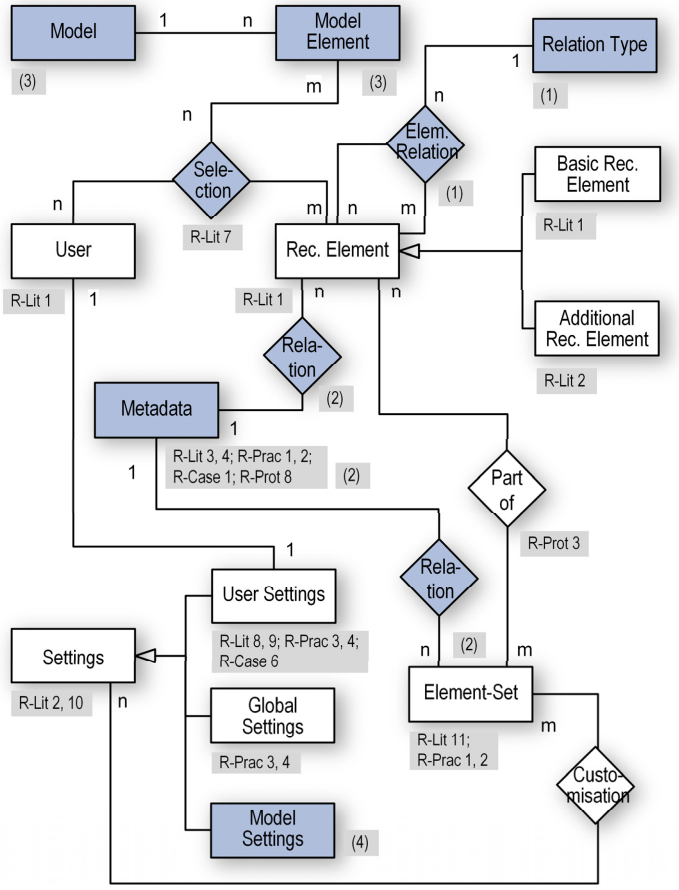

Fig. 9. Overview on Research Process Taken

5 Summary and Outlook

Although sophisticated modeling tools exist, guidance in process modelling in terms of auto-completion and recommendation features is – apart from some few research proposals – largely missing even in today's tools. However, to build such tools is a challenging task since a multitude of data and parameters have to be organized. Unfortunately, existing research works either do not make the data structures underlying recommendation-based systems explicit or the data structures are not developed in a systematic way. In this contribution, we therefore fill this gap in systematically developing a requirements-based data model. We did so using previously elicited requirements from various sources in order to construct partial models. We then combined these partial models into an integrated model. In a last step, we critically reviewed our data model in the light of a data model that was obtained by analyzing a real-world system. This analysis led to interesting insights that in turn caused extensions and revisions of the initial integrated model. These insights would not have been possible without the experiences gained in the implementation of the real-world system. We hope that our model will ease the implementation of new PMRS or informs further development activities around existing systems and that we can help to inspire more research and development in the field of modeling support.

Future research opportunities exist in exploring additional real-world data models of other types of recommendation or assistance systems in order to additionally refine our data model. Moreover, we plan to explore and develop algorithms and parameters working on top of the data model.

References

1. Luftman, J., Zadeh, H.S., Derksen, B., Santana, M., Rigoni, E.H., Huang, Z. (David): Key Information Technology And Management Issues 2012–2013: An international Study. J. Inf. Technol. 28, 354–366 (2013).
2. Sarshar, K., Weber, M., Loos, P.: Einsatz der Informationsmodellierung bei der Einführung betrieblicher Standardsoftware : Eine empirische Untersuchung bei Energieversorgerunternehmen. Wirtschaftsinformatik. 48, 120–127 (2006).
3. Nielen, A., Költer, D., Mütze-Niewöhner, S., Karla, J., Schlick, C.: An Empirical Analysis of Human Performance and Error in Process Model Development. In: Jeusfeld, A., Delcambre, L., Ling T. W. (eds.) Proceedings of the 30th International Conference on Conceptual Modeling (ER 2011), October 31–November 3, Brussels, Belgium. LNCS 6998, Springer. pp. 514–523 (2011).
4. Wilmont, I., Brinkkemper, S., Weerd, I., Hoppenbrouwers, S.: Exploring Intuitive Modelling Behaviour. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., and Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling : Proceedings of the 11th International Workshop, BPMDS 2010 and 15th International Conference, EMMSAD 2010 held at CAiSE 2010, June 7-8, Hammamet, Tunisia. pp. 301–313. Springer, Berlin (2010).
5. Sen, S., Baudry, B., Vangheluwe, H.: Towards Domain-specific Model Editors with Automatic Model Completion. SIMULATION. 86, 109–126 (2010).
6. Kuschke, T., Mäder, P.: Pattern-based Auto-completion of UML Modeling Activities. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software En-

gineering, September 15–19, Vasteras, Sweden. pp. 551–556. ACM, New York, NY, USA (2014).
7. Melville, P., Sindhwani, V.: Recommender Systems. In: Sammut, C. and Webb, G.I. (eds.) Encyclopedia of Machine Learning. pp. 829–838. Springer US (2010).
8. Hornung, T., Koschmider, A., Lausen, G.: Recommendation-based Process Modeling Support: Method and User Experience. In: Li, Q., Spaccapietra, S., Yu, E. (eds.) Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008), October 20-24, Barcelona, Spain. pp. 265–278. LNCS 5231, Springer (2008).
9. Wieloch, K., Filipowska, A., Kaczmarek, M.: Autocompletion for Business Process Modelling. In: Abramowicz, W. (ed.) Proceedings of the 14th International Conference on Business Information Systems (BIS 2011), June 15–17, Poznan, Poland. pp. 30–40. Springer, Berlin (2011).
10. Born, M., Brelage, C., Markovic, I., Pfeiffer, D., Weber, I.: Auto-completion for executable business process models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) Proceedings of BPM 2009 International Workshops, September 7, Ulm, Germany. pp. 510–515. LNBIP 43, Springer, Berlin (2009).
11. Mazanek, S., Maier, S., Minas, M.: Auto-completion for Diagram Editors based on Graph Grammars. In: Bottoni, P., Rosson, M. B., Minas, M. (eds.) IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2008), September 15–19, Herrsching am Ammersee. pp. 242–245. IEEE (2008).
12. Mazanek, S., Minas, M.: Business Process Models as a Showcase for Syntax-Based Assistance in Diagram Editors. In: Schürr, A. and Selic, B. (eds.) Model Driven Engineering Languages and Systems. pp. 322–336. LNCS 5795, Springer Berlin Heidelberg (2009).
13. Clever, N., Holler, J., Shitkova, M., Becker, J.: Towards Auto-Suggested Process Modeling Prototypical Development of an Auto-Suggest Component for Process Modeling Tools. In: Jung, R., Reichert, M. (eds.) Enterprise Modelling and Information Systems Architectures (EMISA 2013), September 5–6, St. Gallen, Switzerland. pp. 133–145. Köllen, Bonn (2013).
14. Li, Y., Cao, B., Xu, L., Yin, J., Deng, S., Yin, Y., Wu, Z.: An Efficient Recommendation Method for Improving Business Process Modeling. IEEE Trans. Ind. Inform. 10, 502–513 (2014).
15. Fellmann, M., Zarvic, N., Metzger, D., Koschmider, A.: Requirements Catalog for Business Process Modeling Recommender Systems. In: Thomas, O. and Teuteberg, F. (eds.) Proceedings of the 12th International Conference on Wirtschaftsinformatik (WI 2015), March 4-6, Osnabrück, Germany. p. 14 (2015).
16. Thomas, O., Fellmann, M.: Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. Bus. Inf. Syst. Eng. 1, 438–451 (2009).