Creating and Updating Personalized and Verbalized Business Process Descriptions

Jens Kolb[1], Henrik Leopold[2], Jan Mendling[3], and Manfred Reichert[1]

[1] Ulm University, Germany
{jens.kolb, manfred.reichert}@uni-ulm.de, www.uni-ulm.de/dbis
[2] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
henrik.leopold@wiwi.hu-berlin.de
[3] Wirtschaftsuniversität Wien, Augasse 2-6, A-1090 Vienna, Austria
jan.mendling@wu.ac.at

Summary. The increasing adoption of process-aware information systems (PAISs) has resulted in large process model collections. To support users having different perspectives on complex processes and related data, a PAIS should enable personalized process views, i.e., user-specific abstractions of process models. Despite the abstraction achieved through views of the graphical process models, many end users still struggle with understanding these graphical models and their details. For selected user groups, therefore, a PAIS should provide verbalized process descriptions describing their role in the process. Existing PAISs neither provide mechanisms for managing process views nor verbalized process descriptions. While process views have been used as visual abstractions for large process models, so far no work exists on how to provide both personalized and verbalized process descriptions based on respective views. This paper presents an approach for creating such personalized and verbalized process descriptions based on process views. Furthermore, textual changes of a personalized and verbalized process description are correctly mapped to corresponding updates of the underlying process model. In this context, all other views and process descriptions related to this process model are migrated to the new version of the process model as well. Overall, our approach enables end users to understand and evolve large process models based on personalized and verbalized process descriptions.

Key words: process model abstraction, updatable process view, process change, natural language, process visualization, human-centered processes

1 Introduction

Process-aware information systems (PAISs) provide support for business processes at the operational level. Usually, a PAIS separates process logic from application code, relying on graphical *process models*. In turn, this enables a separation of concerns, which is a well established principle in computer science to increase maintainability and reduce costs of change [1].

The increasing adoption of PAISs has resulted in large process model collections often including hundreds or thousands of process models [2]. Each of these

process models may refer to different organizational units or user groups, and comprise dozens or hundreds of activities [3]. Usually, the different user groups require customized views on process models, enabling a personalized process abstraction and visualization for them [4, 5, 6]. For example, managers rather prefer an abstract process overview, whereas process participants need a detailed view of those process parts they are involved in. Several approaches for creating such process model views, which are based on well-defined abstraction techniques, have been proposed [7, 8, 9]. However, in many cases providing a customized process view is not sufficient for making the relevant part of a process model understandable for the end user. To be more precise, many domain experts are unfamiliar with process modeling languages and the confidence to understand process models in detail. In such a situation, a verbalization (i.e., textual representation) of both the process model and its corresponding process views would enable domain experts to properly understand process details relevant for them [10].

Generally, real-world processes are frequently subject to change and evolution [1, 11]. In contemporary PAIS, it is not possible to modify a process model through editing and updating one of its view-based process descriptions (i.e., model abstractions). Hence, any process change must be always applied directly to the core process model, which constitutes a complex and error-prone task for domain experts, particularly in the context of large process models. To overcome this limitation, users should be enabled to change large process models through updating their personalized and verbalized process descriptions.

This paper proposes an approach which enables users to create and modify personalized *process descriptions*. Note that a process description is a textual documentation of the real-world process. Therefore, our approach combines existing research on process views [5, 12] and text generation [10]. Furthermore, it is enriched with the possibility to apply changes directly to a personalized process description. In turn, respective changes are then propagated to the underlying core process model. Figure 1 illustrates this approach.

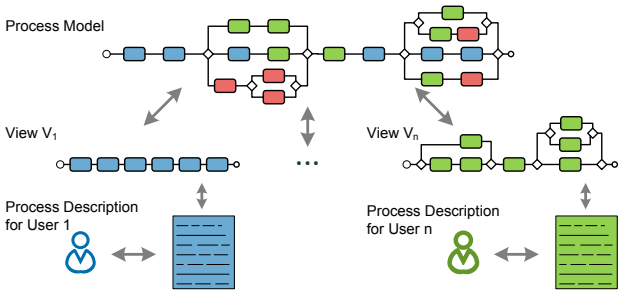

Fig. 1: Providing Personalized and Verbalized Process Descriptions

The remainder of the paper is structured as follows: Section 2 introduces basic notions. Section 3 discusses how process views as well as verbalized and personalized process descriptions may be created from an underlying process model. Section 4 shows how personalized and verbalized process descriptions and the corresponding process model may be modified through text changes. Subsequently, Section 5 presents our proof-of-concept implementation. Section 6 discusses related work and Section 7 concludes the paper.

2 Basic Notions

A process model is described in terms of a directed graph whose node set comprises *activities*, *gateways*, and *data elements*. Gateways can be categorized into *AND*, *XOR* and *Loop*, and may be used for modeling parallel and conditional branchings as well as repetition structures. Edges between activities and/or gateways represent precedence relations, i.e., the *control flow* of the process model (cf. Figure 2). Furthermore, *data elements* describe the data perspective of a process model. Based on this, the data flow is defined by a set of directed edges connecting data elements and activities. *Writing* a data element is expressed through an edge pointing from an activity to the data element. In turn, *reading* a data element is expressed through an edge pointing from this data element to the activity.

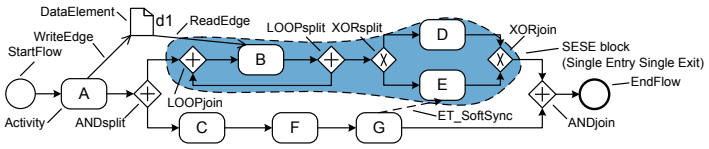

Fig. 2: Example of a Process Model

We presume that process models are *well-structured* [13, 14], i.e., sequences, branchings (of different semantics), and loops are specified as blocks with well-defined start and end nodes having the same gateway type. These blocks, also known as *SESE (single-entry-single-exit)* blocks, may be nested, but are not allowed to overlap (cf. Figure 2).

3 Creating Personalized Process Descriptions

The creation of a personalized process description is a two-step approach. Section 3.1 introduces how process views are derived and Section 3.2 describes how personalized process descriptions are generated.

3.1 Process View Creation

For creating process views, we utilize the *proView*[1] framework. In particular, *proView* is to include alternative process representations (like the textual process description).

The *proView* framework aims at supporting users in intuitively interacting with large business process models as well as evolving them over time. For this purpose, personalized and updatable process views (cf. Figure 3, Part 2) are created for each user (role) abstracting from the overall process model maintained in the central process repository (cf. Figure 3, Part 1). We denote this overall process model as *Central Process Model (CPM)*.

More precisely, a process view abstracts from the CPM by hiding non-relevant process elements (i.e., applying reduction operations) or by combing and abstracting them (i.e., applying aggregation operations). Detailed information about view creation operations and their semantics can be found in [5, 15, 16].

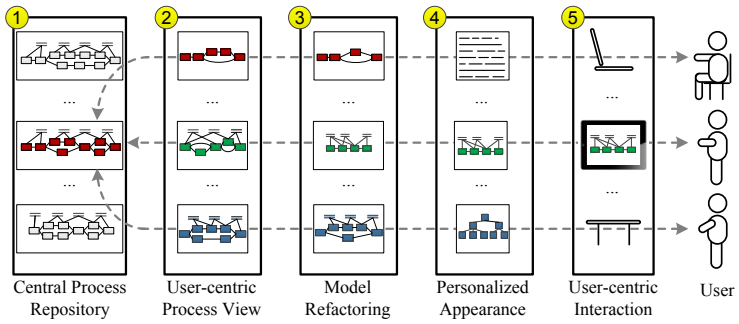

Fig. 3: Overview of proView Framework

When applying view creation operations, non-relevant gateways or empty AND branches might occur. Therefore, the model resulting from the application of such view creation operations is further simplified using behavior-preserving refactorings (cf. Figure 3, Part 3), e.g., AND gateways of a parallel branching with only one remaining branch may be removed.

Part 4 of the *proView* framework (cf. Figure 3) then transforms the process view into a personalized appearance, e.g., a form-, graph-, or tree-based representation [17, 18, 19]. In the context of this paper, Part 4 verbalizes the process view to obtain a textual process description as appearance. This step is described in more detail in Section 3.2. Finally, the result is presented to the user (cf. Figure 3, Part 5). Moreover, for human-centric process management end users should be able to intuitively interact with their processes (e.g., on multi-touch devices) [20].

[1] www.dbis.info/proView

3.2 Creating Verbalized Process Descriptions

In the following, we describe how a process model and a process view respectively can be transformed to a verbalized process description. An overview of the text generation technique applied in this context is depicted in Figure 4. Details of this transformation technique are described in [10]. Note that this transformation is applied in Part 4 of the *proView* framework, which allows creating personalized appearances (cf. Figure 3).

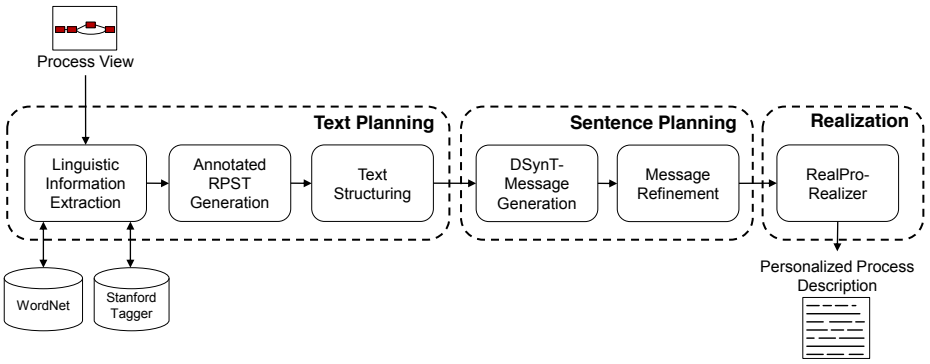

Fig. 4: Architecture for Deriving Verbalized Process Descriptions

Altogether, the transformation consists of five components:

1. *Linguistic Information Extraction*: In this component, we use the linguistic label analysis technique from [21] to recognize the different label patterns that exist for activity labels in the process view. In this way, for instance, we are able to decompose an activity label such as *Choose Contact Type* into the action *Choose* and the business object *Contact Type*.
2. *Annotated RPST Generation*: The *Refined Process Structure Tree (RPST)* generation module derives a tree representation from the given process model in order to provide a basis for a step-by-step process description. In particular, we compute an RPST, which is a parse tree containing a hierarchy of subgraphs derived from the process view [22]. The resulting hierarchy can be visualized as a tree whose root captures the entire tree and whose leaves contain the connections between two elements of the process. After deriving the RPST, we annotate each element with the linguistic information obtained in the previous phase. Thus, for instance, the leave node pointing to activity *Choose Contact Type* is annotated with the action *Choose* and the business object *Contact Type*.
3. *DSynT-Message Generation*: The message generation component maps the annotated RPST elements to a list of intermediate messages. More specifically, each sentence is stored as a deep-syntactic tree (DSynT), which is a

dependency representation introduced by the Meaning Text Theory [23]. Such a deep-syntactic tree facilitates the manageable yet comprehensive storage of the constituents of a sentence. In addition, it can be automatically mapped to a syntactically correct sentence with existing tools [24]. Taking the example of the activity *Choose Contact Type*, the corresponding DSynT consists of a root node pointing to the verb *choose* and two subordinate nodes. The first node specifies *contact type* as object and the second specifies the *clerk* as subject of *choose*.

4. *Message Refinement*: Within the message refinement component, we take care of message aggregation, referring expression generation, and discourse marker insertion. The need for these measures arises if the considered process contains long sequences of tasks. In such cases, for instance, we aggregate messages sharing the same business object. As example, imagine a sequence of the activities *Choose Contact Type* and *Select Contact Type* conducted by a clerk. Instead of generating a sentence for each activity, these activities are aggregated and communicated with a sentence such as "The clerk chooses and selects the contact type." An alternative aggregation strategy is the insertion of referring expressions such as *he* or *it* to ensure lexical variety. For the discourse marker insertion we use an extensible set of connectors to insert markers such as *then* and *afterwards*. In this way, we obtain a well readable text with sufficient variety.
5. *Surface Realization*: The complexity of the surface realization task has led to the development of publically available realizers. Existing tools significantly vary in aspects such as license costs, generation speed, and Java compatibility. Taking these aspects into account, we decided to use the DSynT-based realizer RealPro from CoGenTex [24]. RealPro requires an XML-based DSynT message as input and transforms it to a grammatically correct sentence. As a result, the DSynT for activity *Choose Contact Type* is automatically transformed into the sentence "The clerk chooses the contact type."

After applying these tasks to a process view, the resulting personalized process description may be displayed to the respective user.

Overall, Figure 5 gives an example of how we create such a personalized and verbalized process description based on a CPM. Figure 5a shows the CPM, which describes a simplified *Bank Account Creation* process. The resulting process view in Figure 5b shows the activities of role *Clerk*, i.e., the activities of the manager are reduced. Furthermore, to abstract the process the XOR branching containing activities *Select Customer* and *Create Customer* is aggregated.

Finally, the process view of the clerk is transformed into a verbalized process description. Figure 5c shows the resulting text after this transformation. Note that the AND branching is explicitly documented using bullet points.

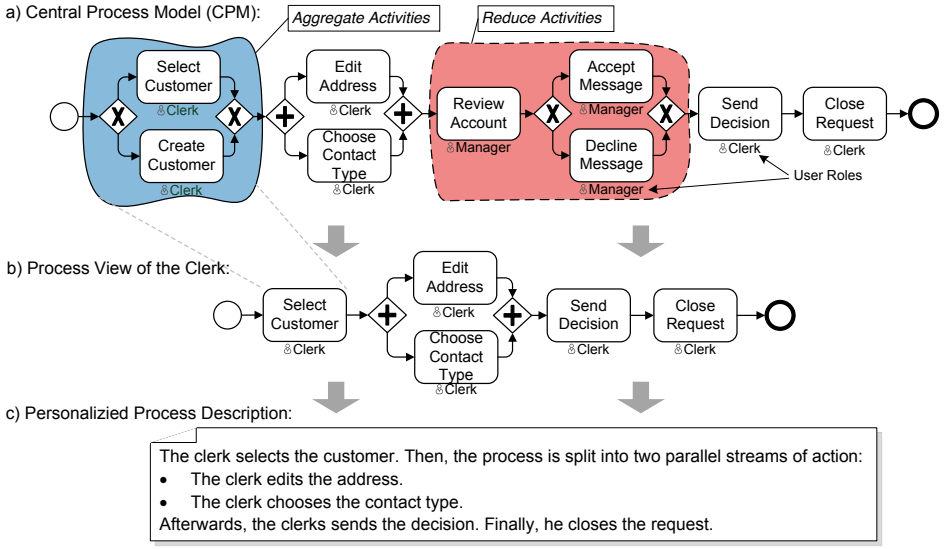

Fig. 5: Creating a Personalized and Verbalized Process Description

4 Process Model Changes through Text Modifications

For a PAIS, it is crucial to support the change and evolution of process models [11]. Consequently, authorized end users should be also enabled to modify the text of their personalized and verbalized process descriptions.

This section describes how such changes of a personalized and verbalized process description (i.e., text modifications) can be mapped to changes of the associated process view and the underlying CPM. More precisely, all modifications of a textual process description must be interpreted and mapped to changes of the corresponding process view. In this context, we build on [20], which provides a core modeling function set comprising functions $F1 - F8$. The latter cover the most common change patterns for process models [25]. Table 1 gives an overview of possible text modifications and their respective mapping to core functions. In the following, we discuss this mapping in detail.

Inserting a Sentence: When adding a new sentence to a verbalized process description, the user wants to express that an action shall be added to the process. In the context of process models, it implies to insert an activity. For this, core modeling function *F1 (Insert Activity)* is triggered to insert, at a certain position in the process view, an activity. Figure 6 shows an example of such an insertion. Sentence "The clerk prints the details." is added by the user to the verbalized process description and analyzed using the Stanford Parser [26]. From the parsing result, we can automatically derive the grammatical relation of the words in the sentence. Relations *nsubj(prints, clerk)* and *dobj(prints, details)* reveal that "clerk" represents the subject and "details" represents the object

Text Modification	Core Modeling Function
New Sentence	F1 Insert Activity
New Enumeration	F2 Insert AND/XOR Branching
New Bullet Point	F3 Insert Branch
Change Object/Verb of Sentence	F4 Renaming Element
Delete (Part of) Sentence	F5 Delete Element
Add Part of Sentence	F6 Insert Data Element
Add Part of Sentence	F7 Insert Data Edge
Change Subject of Sentence	F8 Change User Assignments

Table 1: Mapping of Text Changes to Modification Functions

for predicate "print." Consequently, we extract "clerk" as subject, "details" as object, and "print" as predicate. This information is then used to insert activity "Print Details," into the corresponding graphical process view. Moreover, this activity is performed by role *Clerk*.

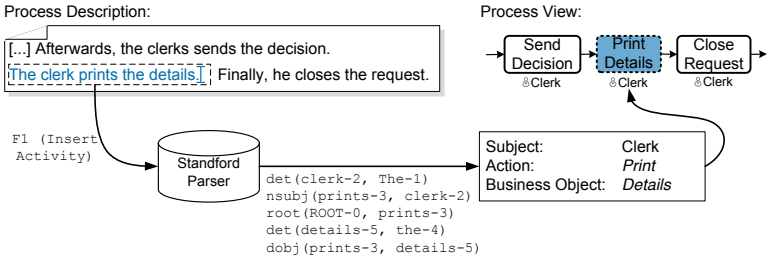

Fig. 6: Inserting a Sentence and Adapting the Corresponding Process View

Inserting an Enumeration: Inserting an enumeration block into the verbalized process description implies that the user wants to insert multiple actions that shall be performed simultaneously or alternatively. Regarding the corresponding process model, the user intends to add an AND/XOR branching, i.e., core modeling function *F2 (Insert AND/XOR Branching)* is applied to the process view. However, the user must manually add the information whether the bullet points of the enumeration should be performed simultaneously (i.e., AND branching is added) or alternatively (i.e., XOR branching is added). Inserting individual sentences to the bullet points triggers again modeling function *F1 (Insert Activity)*. Figure 7 gives an example of inserting a new enumeration block, which performs the bullet points in parallel.

Inserting a Bullet Point: The user inserts a new bullet point to an existing enumeration in the process description to add a stream of actions, which shall be performed simultaneously or alternatively, to the existing bullet points. This text

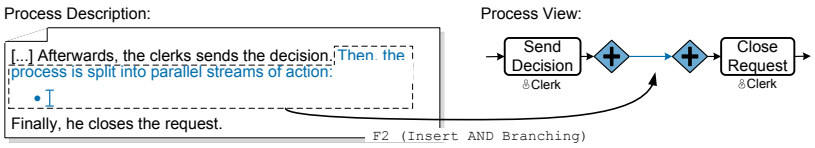

Fig. 7: Inserting an Enumeration Block and Corresponding View Adaption

modification corresponds to core modeling function *F3 (Insert Branch)*, which inserts a new branch to an existing AND/XOR branching in the corresponding process view. Initially, this branch is empty and activities may be added using core modeling function *F1 (Insert Activity)*.

Change the Object or Verb of a Sentence: When the user changes the verb or object of an existing sentence, he wants to adapt the activity described in the sentence. Mapping this to the process view, core modeling function *F4 (Renaming Element)* renames the label of an activity. The changed sentence is again analyzed using the Stanford Parser and the new verb or object is extracted (cf. *F1 (Insert Activity)*). For example, sentence "The clerk prints the order" is changed to "The clerk sends the order." This results in renaming activity "Print Order" in the corresponding process view to *Send Order* using function *F*4.

Deleting a Sentence: Deleting an existing sentence, removes the dedicated action from the personalized and verbalized process description. Thus, the corresponding activity will be deleted in the process view, as well. For this purpose, core modeling function *F5 (Delete Element)* is triggered.

Adding a Part to a Sentence: Adding a new part, like "[...] provides information customer record" to an existing sentence, details the action described through this sentence. In terms of a process model, such information is captured in the data flow. Therefore, core modeling function *F6 (Insert Data Element)* inserts a data element in the process view. Figure 8 shows an example in which the part "[...] requires the information customer record" is added to a sentence. This results in adding data element *Customer Record* to the process view. Furthermore, the verbs "require" or "provide" triggers related modeling function *F7 (Insert Data Edge)* to insert a reading or writing data edge.

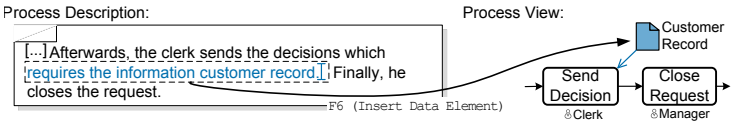

Fig. 8: Example of F6 (Insert Data Element)

Core modeling function *F7 (Insert Data Edge)* adds a reading or writing edge to an existing data element in a process view. Similar to $F6$, phrases using "provides information" and "requires information" are added to a sentence to set the corresponding data edge. Note that the phrases are not predefined in a strict sense. Instead, we parse the added sentence with the Stanford Parser and employ a set of signal verbs and nouns to detect the intention of the user.

Changing the Subject of a Sentence: If the subject of a sentence is changed by the user, he wants to dedicate the action described by the sentence to another person. For this purpose, core modeling function *F8 (Change User Assignments)* is triggered, which changes the user assignment in the corresponding process view. To detect this text modification, the Stanford Parser is used to analyze whether or not the subject is changed.

We have shown how the various text modification are analyzed and mapped to respective core modeling functions. In turn, in [27] we demonstrate how the changes of a process view can be automatically propagated to the CPM. Moreover, all associated process views are updated in this context as well. This is required in order to guarantee that all users work on the current version of the process description.

5 Proof-of-Concept Implementation

The presented *proView* framework was implemented in a proof-of-concept prototype as a client-server application. It enables users to simultaneously edit process models based on updatable process views [28, 12]. Overall, the *proView* prototype demonstrates the applicability of our framework. Figure 9 shows a screen with the models depicted in Figure 5a and 5b. Note that the subprocess activity of the screenshot indicates that an aggregation operation was applied to create the respective node.

We extended *proView* with the ability to create and evolve verbalized process descriptions as described in this paper. Thus, we are able to create personalized and verbalized process descriptions for any well-structured process model and process view respectively.

Figure 10 shows the generated personalized and verbalized process description that corresponds to the process view of role *Clerk*. Pressing the *edit* button on the top right of the paper sheet will enable the user to modify the text. When finishing this editing, in turn, *proView* derives all modifications made in the verbalized process description and highlights them. Once this is accomplished, for all text modifications the respective core modeling function is called to propagate the change to the process view and the underlying CPM. Furthermore, all associated process views and corresponding process descriptions are updated as well. The changed regions in the process descriptions are highlighted again.

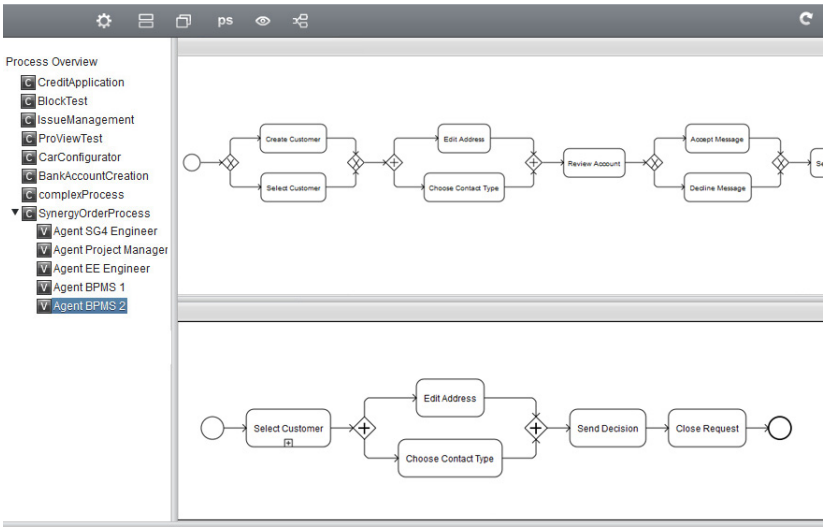

Fig. 9: Proof-of-Concept Prototype

Fig. 10: Personalized and Verbalized Process Description

In future work, our proof-of-concept implementation will be used to involve practitioners in the validation of the presented abstraction approach.

6 Related Work

The work presented in this paper can be related to four major streams of research: natural language generation systems, process model comprehension, process view creation, and process change.

The *generation of natural language texts* has a long tradition and has been applied in various scenarios. Examples include the generation of weather forecasts [29] or the documentation of the activities of planning engineers [30]. The application of text generation techniques on conceptual models is limited to a few examples. The ModelExplainer generates natural language descriptions from object models [31] and the GeNLangUML creates textual specifications from UML class diagrams [32]. However, none of these approaches tackles the specific problems associated with process models.

The field of *process model comprehension* is discussed from different perspectives. For instance, the results from [33] show that the number of arcs has an important effect on overall model understandability. The work in [34] demonstrates the impact the natural language in activity labels has on model comprehension. In general, most authors agree on the fact that text plus diagram provides better comprehension than any of the two in isolation [35, 36, 37]. Hence, the approach presented in this paper builds on these insights as it tries to lower the overall burden of process model comprehension. If users are unable to interpret a given model due to confusing wording or an overwhelming number of arcs, the generated text can guide them through the model.

In prior research, many approaches for *creating process views* have been suggested. Usually, such approaches build on different information to create an abstracted version of the model. Examples include the utilization of a tree decomposition of the process model [38], the semantic similarity of activities [39], or run time data obtained from monitoring [40, 41]. However, these approaches neither enable different user perspectives on a process model nor do they provide concepts for manually creating process views. In [7], the authors introduce an approach providing predefined process view types (i.e., human tasks, collaboration views). As opposed to *proView*, however, this approach is limited to pre-specified process view types. In particular, the latter cannot be used to implement changes in the process model.

For defining and *changing process models*, various approaches exist. [25] presents an overview of frequently used patterns for changing process models. Further, [11] summarizes approaches enabling flexibility in PAISs. In particular, [42] presents an approach for adapting well-structured process models without affecting their correctness. Based on this, [43] presents concepts for optimizing process models over time and migrating running processes to new model versions properly. Still, none of these approaches takes usability issues into account, i.e., no support for user-centered changes of business processes is provided.

7 Summary and Outlook

In this paper, we introduced an approach for creating personalized and verbalized process descriptions based on process views. Such process descriptions support end-users to easily understand their tasks within a business process, which increases their process knowledge. Furthermore, modifying the text of personalized process descriptions triggers changes of the underlying process model. This enables users to perform changes and optimizations of the process without requiring process modeling knowledge. Our approach will increase process-awareness in companies especially for users without in-depth knowledge on process modeling. Furthermore, it includes such users in optimizing and evolving business processes.
We have implemented the described view mechanism in a prototype based on the *proView* framework. In future work, we plan user studies to evaluate whether personalized and verbalized process descriptions are easier to understand and maintain compared to "regular" process models and process views.

References

1. Weber, B., Sadiq, S., Reichert, M.: Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-Aware Information Systems. Computer Science - Research and Development **23**(2) (2009) 47–65
2. Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal **12**(2) (2006) 249–254
3. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring Large Process Model Repositories. Computers in Industry **62**(5) (2011) 467–486
4. Streit, A., Pham, B., Brown, R.: Visualization Support for Managing Large Business Process Specifications. In: Proc 3rd Int'l Conf. Business Process Management (BPM'05). (2005) 205–219
5. Kolb, J., Reichert, M.: Data Flow Abstractions and Adaptations through Updatable Process Views. In: Proc 27th Symposium On Applied Computing (SAC'13), Coimbra, Portugal (2013) 1447–1453
6. Kabicher-Fuchs, S., Rinderle-Ma, S., Recker, J., Indulska, M., Charoy, F., Christiaanse, R., Dunkl, R., Grambow, G., Kolb, J., Leopold, H., Mendling, J.: Human-Centric Process-Aware Information Systems (HC-PAIS). CoRR **abs/1211.4** (2012)
7. Tran, H.: View-Based and Model-Driven Approach for Process-Driven, Service-Oriented Architectures. TU Wien, Dissertation (2009)
8. Bobrik, R., Bauer, T., Reichert, M.: Proviado - Personalized and Configurable Visualizations of Business Processes. In: Proc 7th Int'l Conf Electronic Commerce & Web Technology (EC-WEB'06), Krakow, Poland (2006) 61–71
9. Chiu, D.K., Cheung, S., Till, S., Karlapalem, K., Li, Q., Kafeza, E.: Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment. Inf. Techn. and Mgmt. **5**(3 & 4) (July 2004) 221–250
10. Leopold, H., Mendling, J., Polyvyanyy, A.: Generating Natural Language Texts from Business Process Models. In: Proceedings 24th Int'l Conf on Advanced Information Systems Engineering. (2012)
11. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)

12. Kolb, J., Reichert, M.: Supporting Business and IT through Updatable Process Views: The proView Demonstrator. In: Demo Track of the 10th Int'l Conf on Service Oriented Computing (ICSOC'12), Shanghai, China (2013) 460–464
13. La Rosa, M., Wohed, P., Mendling, J., ter Hofstede, A.H.M., Reijers, H.A., van der Aalst, W.M.P.: Managing Process Model Complexity Via Abstract Syntax Modifications. IEEE Transactions on Industrial Informatics **7**(4) (2011) 614–629
14. Mendling, J., Strembeck, M.: Influence Factors of Understanding Business Process Models. In: Proc. BIS'08. (2008) 142–153
15. Kolb, J., Reichert, M.: A Flexible Approach for Abstracting and Personalizing Large Business Process. ACM Applied Computing Review **13**(1) (2013)
16. Reichert, M., Kolb, J., Bobrik, R., Bauer, T.: Enabling Personalized Visualization of Large Business Processes through Parameterizable Views. In: Proc 26th Symposium on Applied Computing (SAC'12), Riva del Garda (Trento), Italy (2012) 1653–1660
17. Kolb, J., Reichert, M.: Using Concurrent Task Trees for Stakeholder-centered Modeling and Visualization of Business Processes. In: Proc S-BPM ONE 2012, CCIS 284. (2012) 237–251
18. Lanz, A., Kolb, J., Reichert, M.: Enabling Personalized Process Schedules with Time-aware Process Views. In: Proc. CAiSE 2013 Workshops, 2nd Int'l Workshop on Human-Centric Information Systems (HCIS 2013), Valencia, Spain (2013) 205–216
19. Kolb, J., Hübner, P., Reichert, M.: Automatically Generating and Updating User Interface Components in Process-Aware Information Systems. In: Proc 10th Int'l Conf on Cooperative Information Systems (CoopIS 2012). (2012) 444–454
20. Kolb, J., Rudner, B., Reichert, M.: Towards Gesture-based Process Modeling on Multi-Touch Devices. In: Proc 1st Int'l Workshop on Human-Centric Process-Aware Information Systems (HC-PAIS'12), Gdansk, Poland (2012) 280–293
21. Leopold, H., Smirnov, S., Mendling, J.: On the Refactoring of Activity Labels in Business Process Models. Information Systems **37**(5) (2012) 443–459
22. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified Computation and Generalization of the Refined Process Structure Tree. In: Web Services and Formal Methods. Volume 6551 of LNCS. Springer (2011) 25–41
23. Mel'cuk, I., Polguère, A.: A Formal Lexicon in the Meaning-Text Theory (or How to Do Lexica with Words). Computational Linguistics **13**(3-4) (1987) 261–275
24. Lavoie, B., Rambow, O.: A Fast and Portable Realizer for Text Generation Systems. In: Applied natural language processing, ACL (1997) 265–268
25. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data & Knowledge Engineering **66**(3) (2008) 438–466
26. Klein, D., Manning, C.D.: Accurate Unlexicalized Parsing. 41st Meeting of the Association for Computational Linguistics (2003) 423–430
27. Kolb, J., Kammerer, K., Reichert, M.: Updatable Process Views for User-centered Adaption of Large Process Models. In: Proc 10th Intl. Conf. on Service Oriented Comp. (ICSOC'12), Shanghai, China (2012)
28. Kolb, J., Kammerer, K., Reichert, M.: Updatable Process Views for Adapting Large Process Models: The proView Demonstrator. In: Proc of the Business Process Management 2012 Demonstration Track, Tallinn, Estonia (2012)
29. Goldberg, E., Driedger, N., Kittredge, R.: Using Natural-Language Processing to Produce Weather Forecasts. IEEE Expert **9**(2) (1994) 45–53
30. McKeown, K., Kukich, K., Shaw, J.: Practical Issues in Automatic Documentation Generation. In: Applied natural language processing, ACL (1994) 7–14
31. Lavoie, B., Rambow, O., Reiter, E.: The ModelExplainer. In: Proceedings 8th Int'l Workshop on Natural Language Generation. (1996) 9–12

32. Meziane, F., Athanasakis, N., Ananiadou, S.: Generating Natural Language Specifications from UML Class Diagrams. Requirements Engineering **13** (2008) 1–18
33. Mendling, J., Reijers, H., Cardoso, J.: What Makes Process Models Understandable? In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management. Volume 4714 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2007) 48–63
34. Mendling, J., Reijers, H.A., Recker, J.: Activity Labeling in Process Modeling: Empirical Insights and Recommendations. Information Systems **35**(4) (2010) 467–482
35. Ottensooser, A., Fekete, A., Reijers, H.A., Mendling, J., Menictas, C.: Making Sense of Business Process Descriptions: An Experimental Comparison of Graphical and Textual Notations. Journal of Systems and Software (2012) (to appear).
36. Mayer, R.E.: Multimedia Learning. Second edition edn. Cambridge University Press (2009)
37. Hipp, M., Michelberger, B., Mutschler, B., Reichert, M.: A Framework for the Intelligent Delivery and User-Adequate Visualization of Process Information. In: Proc. 28th Symposium on Applied Computing (SAC'13), Coimbra, Portugal (2013)
38. Polyvyanyy, A., Smirnov, S., Weske, M.: The Triconnected Abstraction of Process Models. In: Proc 7th Int'l Conf. Business Process Management. (2009)
39. Smirnov, S., Reijers, H.A., Weske, M.: A Semantic Approach for Business Process Model Abstraction. In: Advanced Information Systems Engineering. (2011) 497–511
40. Shan, Z., Yang, Y., Li, Q., Luo, Y., Peng, Z.: A Light-Weighted Approach to Workflow View. APWeb 2006 (2003) (2006) 1059–1070
41. Schumm, D., Latuske, G., Leymann, F., Mietzner, R., Scheibler, T.: State Propagation for Business Process Monitoring on Different Levels of Abstraction. In: Proc. 19th ECIS, Helsinki, Finland (2011)
42. Reichert, M., Dadam, P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. Journal of Intelligent Inf. Sys. **10**(2) (1998) 93–129
43. Rinderle, S., Reichert, M., Dadam, P.: Flexible Support of Team Processes by Adaptive Workflow Systems. Distributed and Par. Databases **16**(1) (2004) 91–116