A Generic Approach for Calculating and Visualizing Differences between Process Models in Multidimensional Process Mining

Carsten Cordes, Thomas Vogelgesang, and Hans-Jürgen Appelrath

University of Oldenburg, 26129 Oldenburg, Germany,
carsten.cordes@uni-oldenburg.de, thomas.vogelgesang@uni-oldenburg.de,
appelrath@offis.de

Summary. Process mining automatically generates process models from event logs. In multidimensional process mining, these models can be analyzed from various viewpoints by clustering event traces according to their attributes, e.g. age or region of the patient for a healthcare process. For each cluster, a distinct process model is calculated. Since these models are supposed to be identical in most parts, differences between them are hard to spot. Therefore, a tool for emphasizing these differences is needed. To face the different challenges presented by multidimensional process mining like the representational bias, such an approach has to be customizable to support different modeling languages and different layout and differencing algorithms. This paper presents a generic approach to calculate and visualize differences between process models which can be used to compare models in multidimensional process mining.

Key words: visualization, differencing, multidimensional process mining

1 Motivation

Process models are important for analyzing and optimizing business processes. While traditional process management makes use of manually created process models, *process mining*[7] allows for the automatic generation and analysis of process models based on *event logs*. Event logs are collections of real process data collected by process aware information systems (PAIS). Whenever an event takes place, it is recorded by the PAIS. These records can be summarized into event logs. Entries in event logs normally contain at least information about the process and the *process instance* they belong to, where an instance describes an actual case (for example the consulting of a particular client). Process mining consists of three different activities:

Process discovery automatically creates a process model from an event log that consists of events recorded during the execution of the process. In the event log, the time-ordered events are grouped by their process instance. While mined models should be as precise as possible, they should also allow for traces which are not in the event log. This is because in general, event logs do not contain all possible traces of a process, but only example behavior (open-world-assumption).

Process conformance determines the compliance of a given process model by comparing it to an event log. Usually, this is done by replaying event sequences on the model.

Process enhancement improves an existing model with information from an event log to add further *perspectives*. In the time perspective for example, timestamps from the event log are mapped to the model to analyze execution times of activities and identify bottlenecks in the process. The organizational perspective focuses on the actors of a process, e.g. to identify social networks while the case perspective examines a single instance of a process e.g. to identify decision rules.

On the one hand, process mining leads to more realistic models, because real data is used instead of assumed workflows. On the other hand, calculated models also tend to be more complex than planned models and can contain a lot of unnecessary detailed or even wrong information due to noise. In general this makes automatically generated process models harder to analyze. Because of that, the decision which modeling language to use and which information to display in the model are very important in the process mining context.

Multidimensional process mining can be used to analyze a given process in relation to particular attributes, for example the patient's age, gender, or region in the domain of health services research (HSR). [24] proposes to use OLAP-techniques to cluster event-logs by relevant attributes and to mine a separate process model for each cluster. When looking at age groups and regions in HSR for example, this approach allows to mine different process models for each age group and region as outlined in the left part of Figure 1. To identify problems like inappropriate healthcare, an analyst could be interested if there are deviations in the treatment process for a particular illness between the young (< 50) and the old patients (> 69) in the region *East*. By comparing these models, as shown in the right part, differences in the healthcare process between these age groups can be identified, indicating possible problems.

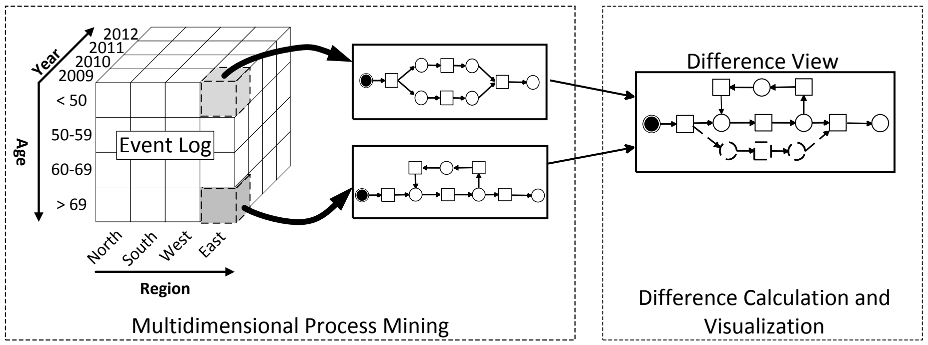

Fig. 1. Difference calculation and visualization in multidimensional process mining.

Without additional tools however, the differences between these process models representing variants of the same process can be very difficult to find: First of all, these models can be very complex and big, depending on the analyzed process and the mining method. Additionally, when looking at process variants, the resulting models tend to be very similar in large parts, making it even harder to find the differences. The goal of difference calculation in multidimensional process mining is to simplify the analysis of differences between two or more different variants of the same process.

While there are already approaches to compare graphical models, none of them takes into account the special problems in multidimensional process mining. For example, the modeling language of the process variants depends on the used discovery algorithm. Due to this, the difference calculation and visualization should be independent from the modeling language to avoid limitations to particular process mining algorithms. Most existing approaches however, support only a particular modeling language. Therefore, we propose a novel approach to compare model variants in process mining which calculates the differences and visualizes them accordingly. It is kept as generic as possible to be suitable for multidimensional process mining, e.g. by using arbitrary difference calculation algorithms that are independent of a specific modeling language.

This paper presents the challenges for differencing process models in multidimensional process mining and proposes a generic approach to deal with these challenges. Section 2 explains the specific problems in comparing models of process variants and Section 3 presents how these problems can be solved in a generic way. Section 4 describes a prototypical implementation of our approach. The appropriateness of our approach to the problem is discussed in Section 5. Related work is presented in Section 6. Finally, Section 7 concludes the paper.

2 Problem description

In most approaches, calculating differences between graphical models serves as a method to transform models between different versions, for example in a version control system. Enhancing existing methods like textual diff helps software engineers to keep track of their previous work. In contrast to this, in multidimensional process mining there is typically no need to transform between model variants, e.g. the treatment of old and young patients. Therefore, different use cases have to be considered to support the analyst in result interpretation.

Calculating differences could be done either syntactically or semantically. In syntactic analysis only the structure of the models is compared. In semantic analysis, the meaning of different models is compared. In the context of process mining, semantic analysis is normally done by comparing the possible traces between two models, for example all possible variations of the treatment process for old patients with all variations for young patients. When two models produce the same traces, these models represent the same process. Because both approaches are useful when analyzing a process, a software to compare process variants in multidimensional process mining should be able to do semantic as well as syntactic analysis.

A lot of algorithms used in software engineering assume an implicit parent-child relation between the different process models and rely on previously recorded change logs between the different versions. In general, there are no such relationships between process variants, e.g. the treatment processes of old and young patients are mined independently. Furthermore, there is no version control system to record the differences between these variants in multidimensional process mining, hence these algorithms do not work properly here. As a consequence, a differencing tool for process variants should be able to compare models without any model-specific additional information, such as change logs.

In process mining, different modeling languages, for example Petri nets [18], causal nets [21] and BPMN [25] are applied. Each language has its own set of structures which cannot be modeled properly due to restrictions of the language's syntax. In Petri nets, for example, there is no possibility to model a logical OR, only XOR is supported [19]. This problem is referred to as representational bias.

In process mining, where algorithms are used to mine the models, most of these algorithms depend on a specific modeling language. For example the α-algorithm [22] only generates Petri nets. If Petri nets are not able to represent the best fitting model, the algorithm cannot find it either. This is why different languages are used in process mining to represent all types of processes in the best way possible. Because of this, an approach to calculate differences between models in process mining needs to be adjustable to different modeling languages. On the one hand, this means, that the method for calculating differences needs to be independent of the modeling language, while on the other hand it should be able to visualize the model in the corresponding modeling language.

In process mining, models are often enriched with additional information in perspectives, for example how long a patient usually has to wait for an examination. An approach for comparing models in multidimensional process mining should be able to display and to compare this additional information to make process analysis easier, e.g. to find optimization potentials. In the following section, we will present our approach, which addresses the identified problems.

3 Generic Diff Concept

Figure 2 gives an overview of the general workflow, where ellipses denote customizable parts in the process. To be able to deal with different types of process models, a language specific mapping must be present when loading a model. With such a mapping, each process model is transformed into an intermediate model. The mapping itself contains transformation rules between the modeling language and the intermediate data model used throughout the application. Both, the mapping and the models can be provided as XML-files.

Using an intermediate model allows for the comparison of a wide variety of modeling languages with generic algorithms. Without an intermediate model,

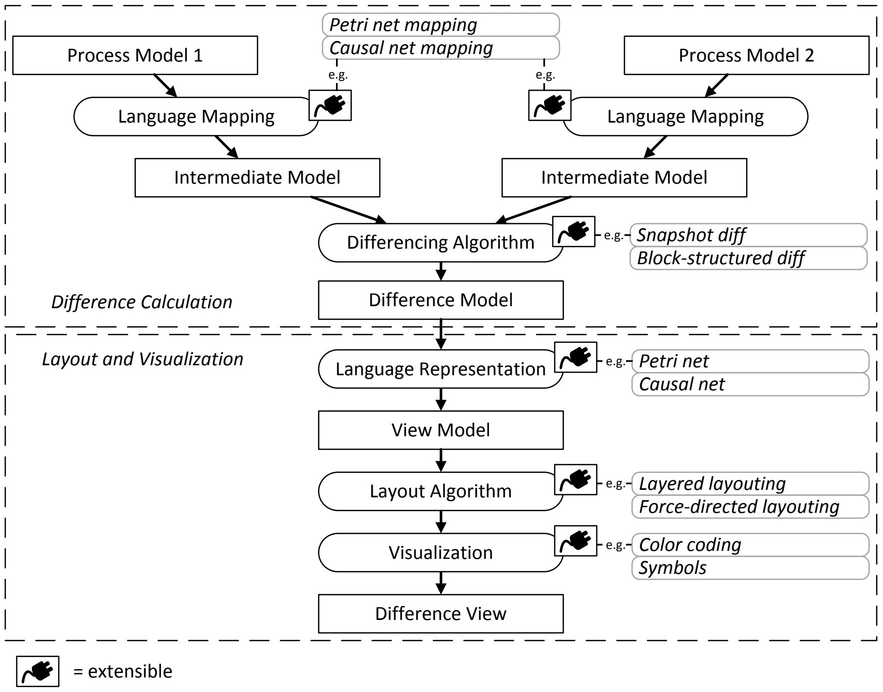

Fig. 2. Overview of differencing workflow and its extensible parts.

each modeling language would need its own set of algorithms. To make implementing generic algorithms easier, the intermediate model should be as simple as possible on the one hand. On the other hand, the intermediate model must not lose any information contained in the original model. Compared models in a single comparison have to be in the same modeling language. This is due to semantic and syntactic differences between different modeling languages, for example events in BPMN which cannot be properly modeled in Petri nets.

The structure of the intermediate model is based upon TGraphs [4]: TGraphs in general consist only of linked nodes. While there is no distinction between different node types or between nodes and edges, these nodes can contain additional information, e.g. provided as key-value-pairs.

Each element of the original model is mapped to a node in the TGraph, nodes like transitions and places of a Petri net as well as edges like the arcs of a Petri net. The original model's structure is retained by linking these nodes accordingly. In doing so, each process model can be transformed into a simple structured intermediate model, regardless of the original model's language. The usage of TGraph simplifies the difference calculation, as the algorithms only have to consider nodes for comparison.

Since language or model specific information is not contained in the intermediate model's structure, this information should be annotated in the according node by providing appropriate key-value-pairs. For example, a node can be marked as a transition by saving the value "transition" for the key "type".

Differencing algorithms can now compare intermediate models, resulting in a difference model. By making these algorithms interchangeable, generic algorithms, as well as specialized algorithms for a certain modeling language can be used. The resulting difference model is generated as an intermediate model itself where the nodes are marked, e.g. as *unchanged*, *added*, *deleted* or *changed*. Depending on the analysis' goals, additional semantic and syntactic differencing algorithms can also be implemented. Some semantic algorithms, e.g. [14], however need information about the logical structure of the process model instead of the syntactical structure. One way to provide this information is a logical block model as shown in Figure 3: The process model (e.g. a Petri net) is interpreted as a set of nested logical blocks, e.g. XOR-blocks, LOOP-blocks or AND-blocks.

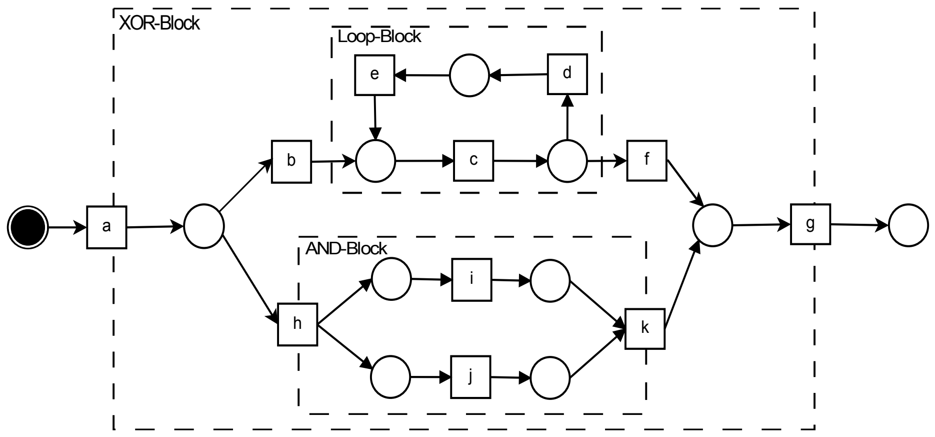

Fig. 3. Logical block model.

Our approach allows to define simple logical patterns, for example XOR-splits and XOR-joins, in a modeling language's mapping. These patterns are described in the corresponding modeling language to avoid a representational bias between the model and the patterns. Logical blocks in a process model can be identified by searching for these patterns and deriving the block model.

After calculating a difference model, this model is visualized. To maintain an individual look for each modeling language, a view model is generated from the difference model. The view model's structure is analogous to the difference model but instead of similar nodes, the view model contains different kinds of representation nodes. These depend on the respective modeling language and are defined in the mapping. To differentiate between different node types, e.g. transitions, places and arcs in a Petri net, each node in the intermediate model contains the representation type used for drawing this node. Depending on the situation and goals of an analysis, representations can be changed (e.g., using different colors or changing the size of labels in the Visualisation). By customizing the representation, additional information stored in the internal model can also be displayed. For example, this allows to add the time perspective by storing timestamps in the intermediate model and changing the appearance of a node dependent on these timestamps. Before drawing the model, the view model is laid out. Layout algorithms are interchangeable to allow for generic as well as specialized algorithms.

Sometimes it is desirable to consider more than two models in comparison, e.g. an analyst wants to find the differences of all age groups (indicated in Figure 1) in relation to the young patients. Our approach allows to compare more than two models by relating them to a selected model serving as a reference. All changes are marked in relation to this reference model. While the comparison of more than three process models is possible, the results are harder to visualize with each additional model, due to the exponential growth of possible states a single node can have. A node could have been added to the second model and been deleted in the third, for example. When trying to highlight these different states by coloring them, the visualization's complexity increases. Hence, to our experience, visualizing differences between more than three models is not useful.

4 Implementation

We implemented our approach as described in section 3 in a prototype. As a proof-of-concept, three modeling languages have been implemented in the prototype: Petri nets, causal nets and process trees [12]. They represent different types of process visualization and are widely used in process mining. Moreover, a tool processing Petri nets should also be able to process similar structured languages, like BPMN with little additional effort.

For difference calculation, three different approaches were implemented: A simple snapshot-diff algorithm [11] provides a way to compare the structure of two process models without considering the models' semantics. This algorithm can be used for additional modeling languages without adapting it or the models in any way as long as both models contain corresponding unique identifiers for each model element in both models. An extended snapshot-diff algorithm distinguishes between edges and nodes. While nodes are compared by their identifier as in the first algorithm, edges are considered equal when the identifiers of their previous and following nodes match. By doing so, only node identifiers need to be equal, but edge identifiers can differ between the two models. This is important for multidimensional process mining where edge ids are typically automatically generated and do not match between different models. The block-structured algorithm explained in [14] has also been implemented in combination with the algorithm in [13], to allow for a semantic analysis. As these algorithms need a process model to be structured as logical blocks, pattern templates for different logical splits and joins were defined for petri-nets.

To be able to easily identify differences between two models, a proper layout and visualization is needed. To layout a model, the layered layout algorithm by Sugiyama et al. [17] is used. This algorithm lays out the model's nodes in layers and then tries to minimize crossing edges by reordering these layers. To minimize crossings, nodes in each layer are positioned as close as possible to the median of their previous nodes' positions. By doing so, the node placement is fast and deterministic. This is necessary because similar process models should be drawn as similar as possible to avoid confusion. Furthermore, the drawing needs to be as fast as possible to be suitable for large models.

To visualize the differences between two models, their difference model is visualized using different colors by default. Elements marked as deleted are drawn in red, elements marked as added are drawn in green and changes are drawn in yellow. However, the view model can be easily adapted to alternative visualization concepts (e.g. dashed lines, symbols etc.). Figure 4 shows the difference model of two process trees from the healthcare domain. The differences are highlighted using color coding and dashed lines. By coloring different parts of an element instead of the same part, the complexity of comparing more than two process models can be slightly reduced. Which part of the according model element (e.g. the background or the border) is colored depends on the definition of the element's representation and can be customized as needed.

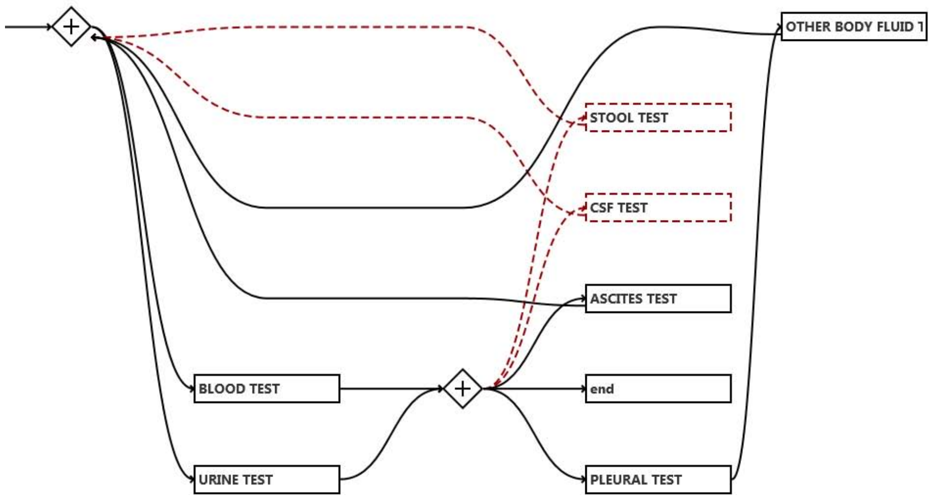

Fig. 4. Visualization of the differences between two process models.

5 Discussion

In the following, we discuss our approach and show how it addresses the challenges in multidimensional process mining. As pointed out in Section 2, difference calculation in multidimensional process mining must not require additional, model-specific information, such as change logs. While our approach requires a language-specific mapping, it does not need any additional information besides the compared models.

To deal with the problem of representational bias, the approach supports different modeling languages by transforming these languages into an intermediate model. Thus the user can deal with representational bias (as long as the compared models are in the same language) by properly configuring the modeling language. Differencing algorithms are interchangeable, thus specialized algorithms, as well as generic algorithms are possible. To allow for semantic analysis, patterns can be defined which help identifying logical blocks in the model which can be utilized by semantic algorithms. Hence, syntactical and semantic analysis are supported.

Once compared, differences are marked according to the kind of difference, e.g. addition, deletion or change. In doing so, common structures in the compared models are unobtrusive, whereas differences between these models are emphasized. To maintain a specific appearance for each modeling language, each language's representation can be customized as needed and it is possible to represent additional information (e.g. perspectives) in the model. To ensure a clear visualization, the layout algorithm can be exchanged, e.g. to minimize crossings at the cost of run-time performance.

6 Related work

Process mining in general is summarized in [7] and explained in detail in [20].

While calculating differences between process models is not often necessary in normal process mining, it is very useful in multidimensional process mining. [24] explicitly motivates the use of difference models to emphasize differences between model variants. This work also suggests different kinds of visualizations for these differences, including the one used in this approach.

In software engineering, a lot of research has already been done about comparing graphical models in the context of software evolution, for example in [26], [8] and [5]. However, these approaches focus on differences between different versions of one model. Furthermore, they mostly rely on previously recorded change logs which are not available when comparing models mined from event data.

A simple method to compare the structure of two models is a Snapshot differential analysis, shown for example in [11]. Although this method is normally used to calculate differences in data warehousing, it can also be applied for comparison of process models. Model elements are matched by their identifiers and differences are annotated. This algorithm is easy to implement and only requires model element identifiers to be equal in both models. The comparison itself however is not very powerful and thus not suitable for advanced (e.g. semantic) analysis. Furthermore, unique identifiers cannot be guaranteed in multidimensional process mining. Apart from that, some specialized algorithms for comparing process models in a non-software-engineering context exist:

[3], [2] and [23] provide and evaluate different metrics and algorithms to determine the degree of similarity between two process models (e.g., to search

for a model in a repository). In contrast to this, we focus on identifying and highlighting the differences between process models.

[1] introduces a method to compare process models in scientific workflows. First, these workflows are transformed into series-parallel graphs. Then basic operations, such as path insertion and path deletion are used to calculate the minimal edit distance between these graphs and a specification containing all possible workflows. This algorithm can be used to find differences between process variants by assuming the overall process model as process specification. However, it only identifies structural differences and lacks support of semantic analysis. Besides, it implies that a complete specification of the analyzed process exists which contradicts the open world assumption. This could be a problem in the context of multidimensional process mining.

[14] is a semantic approach, where different process models are compared by the traces they allow. This is done by creating an order matrix (as explained in [13]) for the intersection of each model and comparing these matrices. Their differences are minimized, resulting in a minimal set of changes between the models. This method however needs a model to consist of semantic structures like logical splits and joins to create the order matrix. In most cases, mined process models do not provide these structures by themselves.

[15] and [16] demonstrate a method to mine the changes between different process variants directly from the event log and treat these variants as differences to a calculated reference model. This reference model and the differences change dynamically with each new trace added to the model. However, this method cannot always be applied to other multidimensional process mining approaches, for example [24], since it relies on the output of the presented mining algorithm.

[10] introduces a method to merge different process models into a configurable workflow model [6]. A configurable workflow model is a process model, where paths can be enabled and disabled as needed. This allows to merge two models while keeping the original models' information. However, this work focuses on the merging algorithm itself. Hence, no advice on visualizing differences and commonalities in a generic way is given.

[9] presents a visualization concept for displaying differences between process models which supports different modeling languages. Compared models are merged and differences colored accordingly. By showing instance traffic in the difference model, this method intends to find optimization potentials in a process. As opposed to this, the approach presented in this paper focuses on identifying the differences between process models in multidimensional process mining. Therefore, we chose a more generic approach to be adaptable to different circumstances, e.g. to allow for semantic and syntactic analysis of a process. Furthermore, our approach is extendable to allow for comparing and visualizing different process mining perspectives, while [9] focuses on the control-flow perspective.

7 Conclusion

In process mining, process models are automatically mined from event logs. While these models are normally more realistic than manually designed models, they also tend to be much more complex. In multidimensional process mining, multiple process models are mined from clustered sets of traces. Differences between these models can help to identify problems in the process. In addition to their complexity, they are supposed to be identical in most parts which makes finding differences between them very difficult. Thus, a tool to compare these models and emphasize their differences can significantly ease the analysis.

In this paper, we presented a generic approach for calculating and visualizing differences between process model variants in the context of multidimensional process mining. As the application implies several challenges like the representational bias, our approach supports arbitrary modeling languages by mapping the original language to an intermediate representation based on TGraphs. Apart from the modeling language's mapping, no additional model-specific information (such as change logs) is required. This allows the comparison of not hierarchically related models as required in multidimensional process mining. Furthermore, the concept also allows for syntactic and semantic differentiation between models and additional, specialized algorithms can be added as needed.

By using a layered layout, models are neatly arranged and colors are used to emphasize differences. To be suitable for different analytical purposes, the visual representation of each model element can be changed and additional layout algorithms can be provided. Additional perspectives can be visualized, too. Hence, by using a suitable configuration, the complexity of difference analysis in multidimensional process mining can be reduced significantly. Thus, the presented approach provides a highly customizable method for comparing different process models in the context of multidimensional process mining.

To further improve our approach, a method to visualize differences between multiple models in a more user friendly way would be useful. This would allow for the comparison of even all mined process model variants at once. Furthermore, our approach should be evaluated in a user case study.

References

1. Z. Bao et al. Differencing Provenance in Scientific Workflows. In *Proc. of the 25th Int. Conf. on Data Engineering (ICDE), IEEE*, pages 808–819, 2009.
2. R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *Proc. of BPM'09*, pages 48–63, Berlin, Heidelberg, 2009. Springer-Verlag.
3. R. Dijkman et al. Similarity of Business Process Models: Metrics and Evaluation. *Inf. Syst.*, 36(2):498–516, April 2011.
4. J. Ebert and A. Franzke. A Declarative Approach to Graph Based Modeling. In *Graphtheoretic Concepts in Computer Science*, pages 38–50. Springer, 1995.
5. C. Gerth et al. Precise Mappings between Business Process Models in Versioning Scenarios. In *IEEE SCC*, pages 218–225. IEEE, 2011.

6. F. Gottschalk, W. van der Aalst, M. H. Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *Int. J. Cooperative Inf. Syst.*, 17(2):177–221, 2008.
7. IEEE Task Force on Project Mining. Process Mining Manifesto. In *BPM 2011 Workshops, Part I*, pages 169–194. Springer Verlag, 2012.
8. E. Eli Jacobsen, B. B. Kristensen, P. Nowack, and T. Worm. Software Evolution: Prototypical Deltas. In *TOOLS (31)*, pages 14–30. IEEE Computer Society, 1999.
9. S. Kriglstein, G. Wallner, and S. Rinderle-Ma. A Visualization Approach for Difference Analysis of Process Models and Instance Traffic. In *BPM*, pages 219–226. Springer, 2013.
10. M. La Rosa, M. Dumas, R. Uba, and R. Dijkman. Merging Business Process Models. In *Proc. of OTM'10*, pages 96–113, Berlin, Heidelberg, 2010. Springer.
11. W. J. Labio and H. Garcia-molina. Efficient Snapshot Differential Algorithms for Data Warehousing. In *Proc. of VLDB*, pages 63–74, 1996.
12. S. J. J. Leemans, D. Fahland, and W. van der Aalst. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In José Manuel Colom and Jörg Desel, editors, *Petri Nets*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer, 2013.
13. C. Li, M. Reichert, and A. Wombacher. Representing Block-structured Process Models as Order Matrices: Basic Concepts, Formal Properties, Algorithms. Technical report, University of Twente, Enschede, The Netherlands, December 2009.
14. C. Li, M. U. Reichert, and A. Wombacher. On Measuring Process Model Similarity Based on High-Level Change Operations. In *ER'08*, pages 248–264, London, 2008. Springer.
15. C. Li, M.U. Reichert, and A. Wombacher. Discovering Process Reference Models from Process Variants Using Clustering Techniques, March 2008.
16. Chen Li, Manfred Reichert, and Andreas Wombacher. Mining Process Variants: Goals and Issues. In *IEEE SCC (2)*, pages 573–576. IEEE Computer Society, 2008.
17. K. Sugiyama, S. Tagawa, and M. Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man & Cybernetics*, 11(2):109–125, 1981.
18. W. van der Aalst. The Application of Petri Nets to Workflow Management, 1998.
19. W. van der Aalst. On the Representational Bias in Process Mining. In *WETICE*, pages 2–7. IEEE Computer Society, 2011.
20. W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer, 1st edition, 2011.
21. W. van der Aalst, A. Adriansyah, and B. van Dongen. Causal nets: a modeling language tailored towards process discovery. In *CONCUR'11*, pages 28–42. Springer, 2011.
22. W. van der Aalst, A.J.M.M. Weijter, and L. Maruster. Workflow Mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16:2004, 2003.
23. B. van Dongen, R. Dijkman, and J. Mendling. Measuring Similarity Between Business Process Models. In *Proc. of CAiSE '08*, pages 450–464. Springer, 2008.
24. T. Vogelgesang and H.-J. Appelrath. Multidimensional process mining: a flexible analysis approach for health services research. In *Proc. of the Joint EDBT/ICDT 2013 Workshops*, pages 17–22, New York, NY, USA, 2013. ACM.
25. S. A White. Introduction to BPMN. *IBM Cooperation*, pages 2008–029, 2004.
26. Z. Xing and E. Stroulia. UMLDiff: An Algorithm for Object-oriented Design Differencing. In *Proc. of ASE '05*, pages 54–65. ACM, 2005.