Process Model Repair Meets Theory Revision - Initial Ideas

Kate Revoredo

Process Model Repair Meets Theory Revision - Initial Ideas

Kate Revoredo[0000−0001−8914−9132]

Vienna University of Economics and Business (WU), Vienna, Austria
kate.revoredo@wu.ac.at

Abstract. Process models are important artifacts to support organizations in documenting, understanding and monitoring their business. Over time, these process models may become outdated and need to be revised to again accurately describe the business, its culture and regulations. *Process model repair* techniques help at automatically revising the existing model from behavior traced in event logs. So far, such techniques have focused on identifying which parts of the model to change and how to change them, but they tend to overlook the potential of using knowledge from practitioners to inform the revision. *Theory revision* techniques are able to revise a logical theory using data. They allow the practitioner to specify which part of the theory is kept unchanged during the revision and which part of the data should or should not be explained by the theory. This paper dives into first ideas on how process model repair techniques can benefit from theory revision. In particular, it elaborates on the use of domain knowledge to identify which data are relevant to be considered and which parts of the model are indeed changeable. Finally, this paper analyzes existing process model repair techniques and discusses challenges that need to be addressed in order to exploit theory revision.

Keywords: Process model repair · process mining · theory revision

1 Introduction

Business Process Management (BPM) [9] relies on process models to support organizations in documenting, understanding and monitoring their processes. These models can be manually specified by stakeholders of the process or automatically discovered from process data using process discovery techniques [1]. Over time, with changes in the regulations, the business or the organization culture these models may become obsolete and less useful for monitoring. Thus, there is a need to revise these process models to meet the new understanding of the business.

In the BPM area, there are some initiatives on process model repair [2] to automatically revise the current model from observed behavior. They use event data collected from information systems to guide the necessary changes in the model. This is typically done by applying conformance checking [6] techniques for identifying non conforming traces and applying local changes to the model to make it compliant to the analyzed traces. These techniques mainly focus on

identifying where to apply the change and how to change the model. However, they overlook the importance of external knowledge in guiding the revisions. The necessary changes to the process model can have the goal to represent new behavior or to prevent undesired behavior. For instance, the organization may decide to switch to a more sustainable business and manage their internal projects digitally. In this scenario, it is important to distinguish in the data between desired and undesired behavior. That is, events from the previous physically managed cases should be marked as undesired while events from the digitally handled process should be marked as desired behavior. The process model repair technique should take these markings into account to guarantee that the final model accurately represents the current business process. Another benefit of considering domain knowledge concerns specific fragments of the process which represent normative work (e.g., safety fallback procedures in case of emergency, organization code of conduct). This kind of work is described by activities which must be done exactly as described in the model. Also, some parts of the model were derived from extensive discussions among the different stakeholders and they share a common understanding among the process participants. It is important that the model repair technique does not change these parts.

Theory revision [25][24] is part of the *Inductive Logic Programming* [16] (ILP) area and it is motivated by concept drifts (i.e., the situation in which properties and relations of the studied data can change over time). It focuses on minimally changing a logical theory in the presence of positive and negative observations with the aim of finding a more accurate theory. Theory revision brings two advantages to the practitioner. First, it allows to distinguish between positive and negative observations (i.e., facts that must or must not be explained by the revised theory). Second, it allows to precisely specify parts of the model to be kept unchanged during the revision. This paper aims at bridging *process model repair* and *theory revision*. More specifically, it provides the first ideas on how to formulate a process model repair problem as a theory revision problem. To this end, it focuses on *i)* the distinction between desired (positive) and undesired (negative) behavior, and *ii)* changeable and unchangeable model fragments to be considered in the model repair. This paper analyzes existing techniques from a theory revision point of view and explores which concepts are already in use and which are further concepts that are useful for model repair.

The rest of the paper is structured as follows. Section 2 reviews preliminary concepts. Section 3 presents the proposal on how to frame process model repair based on theory revision. Section 4 presents an analysis of existing model repair techniques positioning them against derived concepts. Section 5 discusses related work. Section 6 concludes the paper and outlines future work.

2 Preliminaries

This section describes the two main concepts approached in this paper. Section 2.1 reviews the concept of process model repair. Section 2.2 describes the concept of theory revision.

2.1 Process Model Repair

A *process model* (M) is a description of the business process. It represents the sequence of process events (i.e., the activities and control-flow of the process). Usually, this is represented as a directed graph where the nodes represent the events and the edges represent the potential flow of control among the events. An *event log* (L) is a multi-set of traces and each trace (T) represents the execution of a process instance, (i.e., the sequence of process activities). Techniques for automatically discovering a process model from an event log were proposed in the literature [1]. The final process model is expected to conform with the event log (i.e., the model M is expected to replay all the traces T in L).

The event log represents the observed behavior while the model represents the expected behavior. Over time, the expected and the observed behavior may not align anymore, for instance because of concept drift. In this case *process model repair* [10] may be applied. Process model repair aims to improve the quality of a model through process data by applying minimal changes to the initial model. Existing methods take as input a model and an event log, and produce a new model that resembles the original one as much as possible while still guaranteeing that the new model is able to replay the traces in the log. More in detail, process model repair techniques need two inputs: a process model and an event log. A process model typically uses a graph-based notation to express the partial order relation of the activities within the different traces constituting the event log. Such model may have been designed manually by a person using a modeling tool or may have been generated by a process discovery tool. An event log typically comes from an information system (e.g., a BPMS, a database, etc). The data presented in the event log may record events pertaining a large amount of time, including time periods in which the process is enacted differently (i.e., the actual process changed with respect to its model). This means that the initial model is no longer able to accurately describe the behavior recorded in the event log, especially when it comes to newer traces. Thus, the task of model repair is that to produce a new process model that is able to best describe all the facts and relations observed in the event log.

Process model repair can be positioned in between process discovery [1] and conformance checking [6] (i.e. taking a predefined model as the norm and checking whether the event log complies with it). The final model may reflect reality (i.e. observed behavior recorded in the event log) better than the initial model, but may also be very different from it, which can make the final model useless in practice. For instance, practitioners may heavily rely on the initial model to understand how a particular process functions. Presenting to them a model very different from the one they are accustomed with, may result in the final model being ignored by them. To address this issue, a minimality criterion is considered when repairing the initial model guaranteeing that the final model is as similar as possible to the initial one. However, the minimality criterion does not guarantee that still important parts (e.g., commonly agreed pieces of the process that represent a shared understanding of the work) are not changed by the technique. In this paper, I argue that the repair would be more useful to the

practitioners if it takes into account predefined fragments of the model that they do not want to modify.

Furthermore, as stated in [4, 3], existing approaches for process model repair are applied on whole event log. They apply the changes based on all the traces that did not comply with the model, thus including traces that the practitioners do not want to take into account for the repair. As a consequence, the final models is unnecessarily complex and harder to understand by the practitioners. In this paper, I also argue that the repair technique can benefit from a pre-processing step, in which the relevant traces for the repair are identified.

2.2 Theory Revision

A theory revision technique receives as input an initial logical theory (T_i) and a set of factual data (C) [25][24] . The theory is composed by a set of logic rules and can be either specified manually by domain analysts or automatically using an Inductive Logic Programming (ILP) system [16]. Furthermore, the initial theory is divided into two parts: unchangeable, which is assumed to be correct, and a changeable part that can be modified by the revision. The data are split into positive (C^+) and negative (C^-) observations. The final theory (T_f) should logically imply all the positive observations (completeness) ($\forall c^+ \in C^+, T_f \models c^+$), none of the negative observations (consistency)($\forall c^- \in C^-, T_f \not\models c^-$) and satisfy a criterion of minimality [25]. More in detail, theory revision needs two inputs: an initial theory and a dataset. The initial theory is a set of logical rules and it is divided into two parts: a changeable set of rules and an unchangeable one. Rules may be expressed in first-order logic notation (e.g., Horn clauses). These rule sets may have been specified manually by a person or may have been learned via a machine learning technique. A dataset is a collection of facts that may come from an information system (e.g., a database). The data presented in the dataset is divided into two sets: the positive and the negative sets. The positive set represents facts that must be explained by the theory whereas the negative set represent facts that must not be explained. The task of theory revision is to generate a final theory in which all the unchangeable rules of the initial theory are still present and some changeable rules have been replaced by new ones.

When applying theory revision three considerations must be made. First, it must be clear where the theory should be modified (*revision points*). Second, it must be clear how the theory should be revised (*revision operators*). Third, it must be clear what *evaluation function* is going to be considered in order to choose the best revision.

Revision points are defined through the data. Positive observations define *generalization revision points* while negative observations define *specialization revision points*. The first is the literal in a rule responsible for the failure of proving a positive example (failure point) and other antecedents (contributing points) that may have contributed to this failure. The second is defined by clauses used in successful proofs of negative examples. The specification of the revision point determines the type of revision operator that will be applied to make the theory consistent with the data. Generalization operators are used when a

positive observation is not proved by the theory, i.e. the theory must be more generic in order to explain a positive observation. The second group is applied when a negative observation is proved by the theory, i.e. the theory should be more specific in order to not explain a negative observation.

Theory revision relies on operators that propose modifications at each revision point. Any operator used in machine learning (first-order) can be used in a theory revision system. For instance, the specialization operator *delete-rule*, that deletes the rule that is causing the proof of a negative observation and the operator *add-antecedent*, that adds antecedents to a rule in an attempt to make negative observations unprovable. As examples of generalization operators we can consider the *delete-antecedent* operator that deletes antecedents from a rule making this rule more generic and therefore allowing positive observations previously not proved by the theory to be proved. Another operator is the *add-rule* operator. This operator leaves the original rule in the theory and generates new ones based on the original. The process is made in two ways. First it copies the original rule and, using hill-climbing antecedent deletion, deletes antecedents without allowing any negative observation to be proven, and also those that allows one or more previously unprovable positive observation to be proven (even if doing so allows proofs of negatives). Then it creates one or more specializations of this core rule using the add-antecedents operator, to allow proofs of the desired positives while eliminating the negatives. An evaluation function such as accuracy is used to select the best proposed revision to be implemented.

3 Proposal

This section discuses how to frame the task of model repair as a theory revision task. Figure 1 introduces the overarching method.

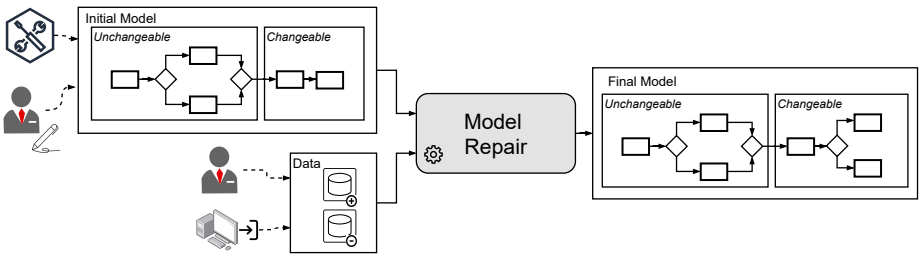

Fig. 1: Process model repair framed as a theory revision problem schema

As input, a model repair technique receives an initial model and an event log and outputs a revised model. The initial model is divided into two parts. The first one (BK) represents the part of the process model that should not be

changed during the repair procedure. The specification of this part is done by the practitioners, based on their knowledge about the domain. For instance, it can represent some external or internal regulations that should be kept. Once the protected part of the process model is defined, all the rest is associated to the repairable part (M_i). For the repair to happen it is necessary that M_i is defined. The BK can be empty meaning that the practitioners chose to allow repair to be considered in the whole structure of the process model. The event log is also divided in two: positive (L^+) and negative (L^-) event logs. L^+ corresponds to the traces representing acceptable behavior while L^- corresponds to behavior that should be avoided. The model repair technique implements changes in the model (M_i) guided by the event logs generating a final model that includes the unchangeable part (BK) plus the repaired model (M_f). The choice of which change to implement follows a minimality criterion.

The changes are made in a batch mode, i.e. all the traces are received at once and the changes to the model are made considering all of them. There are some approaches for model repair that work in an incremental manner. In [13], an approach for incrementally learn declarative process models was proposed. The constrains are represented in a fragment of first-order logic consisting of Datalog Horn clauses. The approaches implements changes in the model based in one trace. It can learn from scratch as well as implementing modifications in an existing model. It does not required the definition of positive and negative observations implementing the modifications only based on positive observations. Incremental approaches are not in the scope of the present work.

4 Analysis of Existing Process Model Repair Techniques

This section reports the results of the analysis of the existing approaches for process model repair with respect to our proposal of framing the problem as a theory revision problem. All the approaches rely on using conformance checking techniques for identifying traces not conforming with the model and to guide the necessary repair in the model. They vary mainly on the alignment technique used and in the proposed repair. Therefore, revision points and revision operators are considered in an implicitly manner. This paper then focused the analysis on (i) specification of fragments of the initial model defining changeable and unchangeable fragments; (ii) partitioning of the event log into positive and negative traces; (iii) application of minimality criterion. This paper considered approaches where the revision procedure is fully automatized. Table 1 summarizes our findings.

In [5] the alignment of a trace and a process model is used to propose changes in the model. The approach uses a metric to calculate the alignment of the proposed repaired model with the initial one, aiming for repairs that provide minimal repair, thus a minimality criterion is used.

In [19] an impact-driven process model repair was proposed. The model repair problem was addressed as an optimization problem where each possible repair had a cost and the task was to find the repaired model that maximize fitness

Table 1: Summary of the model repair approaches

Approach	Model Fragmentation	Event log Partitioning	Minimality Criterion
[5]			✓
[19]			✓
[26], [27], [20]			✓
[15]	✓		✓
[23]			✓
[22]			✓
[12]		✓	✓

constrained by the cost. A maximum degree change was considered, in this way minimality criterion is met.

In [26] an approach for repairing a process model described in logic Petri Net was proposed. It focuses on repairing a choice structure to make the model replay activities in different branches. Variations of the work considering non-free-choice structures [27] or process models with choice structures [20] were also considered. In all the approaches, a minimality criterion was considered.

In [15] the principle of divide and conquer was used to decompose the initial process model in several fragments. Then, each fragment is classified in good or bad fragment, depending whether they conform with the event log or not, respectively. For the bad fragments, repair operations are applied and the generated repair fragments are then composed with the good one generating a final repaired model. This work aligns partially with (i) decomposing the model in changeable and unchangeable fragments. However, the choice is based on conformance with the event log and not as a prior decision based on the understanding of the business and the needs of the practitioners as stated in this paper. Moreover, it can be the case that a part of the model that conforms with the data should not. By focusing on changing only the parts that did no conform with the event log keeping as much as possible the initial model, the minimality criterion is considered.

In [23], the process model as a workflow net and the event log are both represented as footprint matrix. The repair approach implements modification in the model based on differences found between the two footprint matrix. The approach searches for a minimal change in the model and consider all the event log as desirable traces.

In [22], the authors presented the task of generalized conformance checking. A level of quality trust is associated to the log and to the model and this quality is used to repair both the log and the model. Although the authors acknowledge the possibility of the event log not representing all the possible behaviors or representing undesirable behavior, the model repair does mot consider these issues in a different way, also because they are not distinguished in the event log. Therefore the consideration of positive and negative event logs is not take into account by this approach. The alignment between the original model and the

final model is calculated, but the trust value associated to the model defines the amount of change accepted. If the trust on the model is high which cope with the idea of theory revision (i.e., the model is approximately correct), then the model will be changed minimally, therefore the approach follows a minimality criterion.

The approach proposed in [12] is the most related to the concepts presented in this paper. It uses conformance checking to find the sequence of the model most similar to the traces that did not conform. Then the parts that did not conform are separated and process discovery technique are used to learn the correspondent sub-process that are then included in the initial model repairing it. They separated in a different log traces that should not be replayed by the model. Therefore, they partially considered negative and positive traces. The approach satisfies a minimality criterion.

As a result of the analysis of the existing techniques for process model repair it is possible to observe that the techniques use a minimality criterion to guarantee that the final model resembles as much as possible the initial model. However, the techniques can be improved by the involvement of the practitioners for fragmentation of the initial model and partitioning of the event log. The approaches proposed in [15] and [12] partially fulfilled these two points, respectively.

5 Related Work

Given the analyzed sources of the literature, it can be observed that the concept of theory revision has not yet been considered for process model repair when the model follows an imperative paradigm. When the process model is represented with a declarative paradigm, work such as [7] can be mentioned. Theory revision concepts were used to improve DECLARE [18] rules, where a set of positive and negative event logs were built and used for the revision of the set of DECLARE rules. In the context of process discovery, the approach presented in [8] used a partitioning of the event log into positive and negative event logs to learn declarative process model.

The idea of framing an existing task as a theory revision task has been used in other areas such as learning game rules [17], updating social network in the presence of stream data [14] and discovering links in real biological networks [21]. This paper is a first attempt to provide formal foundations for process model repair. Especially, it sheds light on the necessity of the fragmentation of the initial process model and the partition of the event log involving practitioners' knowledge.

6 Conclusion

BPM relies on process models to support the practitioners in documenting, understanding and monitoring their processes. If these models become outdated over time, the monitoring of the process becomes inaccurate. Process model repair proposes to change the model in order to cope with the latest changes in

the data recorded in event log. From the area of ILP, theory revision techniques allow the revision of a logical theory, i.e. a set of rules, guided by positive and negative observations. The logical theory is minimally changed in order to explain all the positive observations and none of the negative observations.

This paper explored the area of process model repair framing it as a theory revision problem. It identified two main points from theory revision that can contribute to process model repair, the fragmentation of the initial process model in a way that the practitioners may indicate fragments that should not be considered for repair and the definition of two event logs, namely positive and negative logs. The first one includes behavior that should be replayed by the model and the second includes behavior that should be avoided by the model. To use theory revision for process model repair, two main challenges should be addressed, namely the fragmentation of the initial process model and the partitioning of the event log.

As future work, I intend to i) delve deeper into formalizing other aspects of theory revision and ii) explore other facets of the broader area of *theory refinement* [25]. For what concerns i), as existing works are already implicitly using the concepts of revision points and revision operators, I plan to formally define them. For what concerns ii), the automatic improvement of logic knowledge bases, known as *theory refinement*, can be divided into two classes: *theory revision* and *theory restructuring*. Both aim at improving the quality of the logical theory. The revision task involves changing the answer set of the given theory, i.e., improving its inferential capabilities by adding previously missing answers (generalization) or by removing incorrect answers (specialization). On the other hand, the task of restructuring does not change the answer set of the given theory; its objective is to improve performance and/or user understandability of the theory. As a follow-up work, I intend to investigate how concepts of theory restructuring can support the task of process model repair, e.g. process simplification [11].

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Armas-Cervantes, A.: Process model repair. In: Encyclopedia of Big Data Technologies. Springer (2019)
3. Armas-Cervantes, A., van Beest, N.R.T.P., Rosa, M.L., Dumas, M., García-Bañuelos, L.: Interactive and incremental business process model repair. In: OTM Conferences (1). Lecture Notes in Computer Science, vol. 10573, pp. 53–74. Springer (2017)
4. Armas-Cervantes, A., van Beest, N.R.T.P., Rosa, M.L., Dumas, M., Raboczi, S.: Incremental and interactive business process model repair in apromore. In: BPM (Demos). CEUR Workshop Proceedings, vol. 1920. CEUR-WS.org (2017)
5. Buijs, J.C.A.M., La Rosa, M., Reijers, H.A., van Dongen, B.F., van der Aalst, W.M.P.: Improving business process models using observed behavior. In: Cudre-Mauroux, P., Ceravolo, P., Gašević, D. (eds.) Data-Driven Process Discovery and Analysis. pp. 44–59. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
6. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)

7. Cattafi, M., Lamma, E., Riguzzi, F., Storari, S.: Incremental declarative process mining. In: Smart Information and Knowledge Management, Studies in Computational Intelligence, vol. 260, pp. 103–127. Springer (2010)
8. Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. Trans. Petri Nets Other Model. Concurr. **2**, 278–295 (2009)
9. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
10. Fahland, D., van der Aalst, W.M.P.: Repairing process models to reflect reality. In: BPM. Lecture Notes in Computer Science, vol. 7481, pp. 229–245. Springer (2012)
11. Fahland, D., van der Aalst, W.M.P.: Simplifying discovered process models in a controlled manner. Inf. Syst. **38**(4), 585–605 (2013)
12. Fahland, D., van der Aalst, W.M.P.: Model repair - aligning process models to reality. Inf. Syst. **47**, 220–243 (2015)
13. Ferilli, S.: Incremental declarative process mining with woman. In: EAIS. pp. 1–8. IEEE (2020)
14. Guimarães, V., Paes, A., Zaverucha, G.: Online probabilistic theory revision from examples with proppr. Mach. Learn. **108**(7), 1165–1189 (2019)
15. Mitsyuk, A.A., Lomazova, I.A., Shugurov, I.S., van der Aalst, W.M.P.: Process model repair by detecting unfitting fragments. In: AIST (Supplement). CEUR Workshop Proceedings, vol. 1975, pp. 301–313. CEUR-WS.org (2017)
16. Muggleton, S.: Inductive logic programming. New Gener. Comput. **8**(4), 295–318 (1991). https://doi.org/10.1007/BF03037089, https://doi.org/10.1007/BF03037089
17. Muggleton, S., Paes, A., Costa, V.S., Zaverucha, G.: Chess revision: Acquiring the rules of chess variants through FOL theory revision from examples. In: ILP. Lecture Notes in Computer Science, vol. 5989, pp. 123–130. Springer (2009)
18. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: full support for loosely-structured processes. In: EDOC. pp. 287–300. IEEE Computer Society (2007)
19. Polyvyanyy, A., van der Aalst, W.M.P., ter Hofstede, A.H.M., Wynn, M.T.: Impact-driven process model repair. ACM Trans. Softw. Eng. Methodol. **25**(4), 28:1–28:60 (2017)
20. Qi, H., Du, Y., Qi, L., Wang, L.: An approach to repair petri net-based process models with choice structures. Enterp. Inf. Syst. **12**(8-9), 1149–1179 (2018)
21. Raedt, L.D., Kersting, K., Kimmig, A., Revoredo, K., Toivonen, H.: Compressing probabilistic prolog programs. Mach. Learn. **70**(2-3), 151–168 (2008)
22. Rogge-Solti, A., Senderovich, A., Weidlich, M., Mendling, J., Gal, A.: In log and model we trust? A generalized conformance checking framework. In: BPM. Lecture Notes in Computer Science, vol. 9850, pp. 179–196. Springer (2016)
23. Sun, Y., Du, Y., Li, M.: A repair of workflow models based on mirroring matrices. Int. J. Parallel Program. **45**(4), 1001–1020 (2017)
24. Taylor, C., Nakhaeizadeh, G.: Learning in dynamically changing domains: Theory revision and context dependence issues. In: ECML. Lecture Notes in Computer Science, vol. 1224, pp. 353–360. Springer (1997)
25. Wrobel, S.: First Order Theory Refinement. In: De Raedt, L. (ed.) Advances in Inductive Logic Programming. IOS Press (1996)
26. Xu, Y., Du, Y., Qi, L., Luan, W., Wang, L.: A logic petri net-based model repair approach by constructing choice bridges. IEEE Access **7**, 18531–18545 (2019). https://doi.org/10.1109/ACCESS.2019.2896079
27. Zheng, W., Du, Y., Wang, S., Qi, L.: Repair process models containing non-free-choice structures based on logic petri nets. IEEE Access **7**, 105132–105145 (2019)