Incremental and Interactive Business Process Model Repair in Apromore

Abel Armas-Cervantes[1], Nick R.T.P. van Beest[2], Marcello La Rosa[1], Marlon Dumas[3], and Simon Raboczi[1]

[1] Queensland University of Technology, Australia
{abel.armascervantes,m.larosa,simon.raboczi}@qut.edu.au
[2] Data61, CSIRO, Australia
nick.vanbeest@data61.csiro.au
[3] University of Tartu, Estonia
marlon.dumas@ut.ee

Abstract. This paper presents the integration of a plugin for interactive and incremental business process repair into the Apromore advanced business process analytics platform. Given a model and an event log representing the behavior of a process as input, the plugin describes encountered behavioral differences between them as natural language statements, as well as graphical representations displayed over the model. The graphical representation of the differences provides visual guidance on how to repair the model so that it better reflects the observed behavior. Subsequently, the users have the option to select the discrepancies that need to be resolved and how to resolve them. The users can choose whether to follow the suggestion provided by the tool or to repair the discrepancies differently. The differences are updated with every repair applied over the model, resulting in an iterative procedure where the user has the ultimate control over the extent and the implementation of the changes over the model.

Keywords: Apromore, process model repair, event logs, process model

1 Overview

The behavior of business processes is usually analyzed by means of event logs, which allows analysts to compare the actual execution of a process against its expected execution captured in a model. This model-to-log comparison operation is known as *conformance checking* and aims at identifying the differences between the observed (log) and the specified (model) behavior. Oftentimes analysts want to update the model to resolve the differences identified by a conformance checker in order to better reflect reality. This operation is known as *process model repair*. Process model repair methods take as input a model and an event log, and produce another model that resembles the original one as much as possible, but tries to minimize of eliminate the differences with respect to the log.

A number of techniques have been proposed for automated process model repair (e.g.[1, 2]). Given a model and a log, the automated techniques identify the differences between them, and produce a repaired model that reconciles such differences. However, the repaired models generated by these approaches are generally complex and hard to

interpret, as all small differences with the event log, even exceptional cases, are considered by the repaired model. Nevertheless, in many cases only the most important differences should be incorporated into the repaired model. In this regard, an incremental and interactive process model repair has been proposed in [3]. This approach differs from the automated ones in that it does not seek to fix each and every discrepancy, but instead it suggests the analyst how to reconcile one difference at a time. These suggestions are done by means of graphically representing the differences over the model, such that the user decides which discrepancies to fix and how. A graphical overview of the approach is provided in Fig. 1.

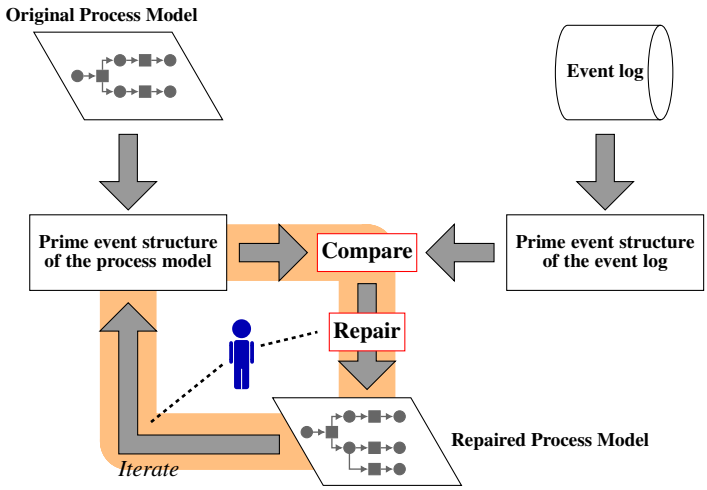


Fig. 1: Overview of the approach.

This paper presents the integration of the technique proposed in [3] into the Apromore advanced business process analytics platform.[4] Apromore is an open-source and extensible online process analytics platform, comprising state-of-the-art capabilities for managing and analyzing large process model collections. The operations for repair complement a wide range of existing capabilities provided by Apromore, such as process model compare, merging, simulation, restructuring and similarity search.

The presented repair technique is an interactive extension that is implemented in the *Compare* plugin in Apromore, as presented in [4]. The repair feature is based on the differences between a model and a log, which are computed by the compare operation. The behavioral comparison performed by the compare operation is based on the approach presented in [5], where the behavior of a model and the behavior of a log are encoded as event structures [6], and then compared using a so called Partial Synchronized Product (PSP) [7]. The encountered behavioral differences are explained as what behavior is observed in the model and not in the log, or vice-versa. The identified differences are based on a predefined set of patterns that capture cyclic behavior, mismatching behavioral relations (exclusive vs. parallel) between tasks, optional behavior, among others.

[4] http://apromore.org

Given a process model and an event log, the steps in the repair procedure are: 1. computation of the event structures from an event log and a process model, 2. comparison of the event structures, 3. identification of differences, and 4. repair suggestion for each difference. Note that the user can choose whether to apply or ignore the suggested repair for every difference. These four steps are repeated until all relevant differences have been resolved in the repaired model or the user decides that the repaired model is "good enough". That is, the most important differences have been resolved and the remaining differences are exceptions that should not be taken into account in the repaired model. This iterative procedure is highlighted in orange in Figure 1.

The repair plugin is embedded into the compare item in the menu *Analyze* (Fig. 2). It accepts process models in BPMN modeling language and event logs in XES or MXML format; however, Apromore offers conversions from different modeling languages (such as CPF, EPCs, Petri nets and YAWL) to BPMN. Once a model and a log have been selected in Apromore, the *compare* plugin performs a behavioral comparison and outputs the differences as shown in Fig. 3(a). The differences are explained in two different ways: as sentences in natural languages (see the left-hand side of the screen in Fig. 3) and graphically overlaid over the original model (see the right-hand side of the screen in Fig. 3(b)). The sentences contain a textual description of the differences and their impact. The impact represents the percentage of traces affected by a given difference and defines an order between them. Specifically, the differences are listed in a decreasing order of impact. The graphical representations of the differences suggest how to reconcile a given discrepancy by showing what has to be added or deleted from the model, the full range of patterns can be found in [3].

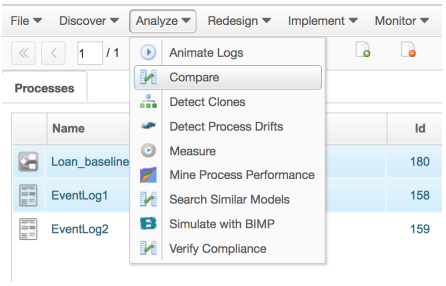
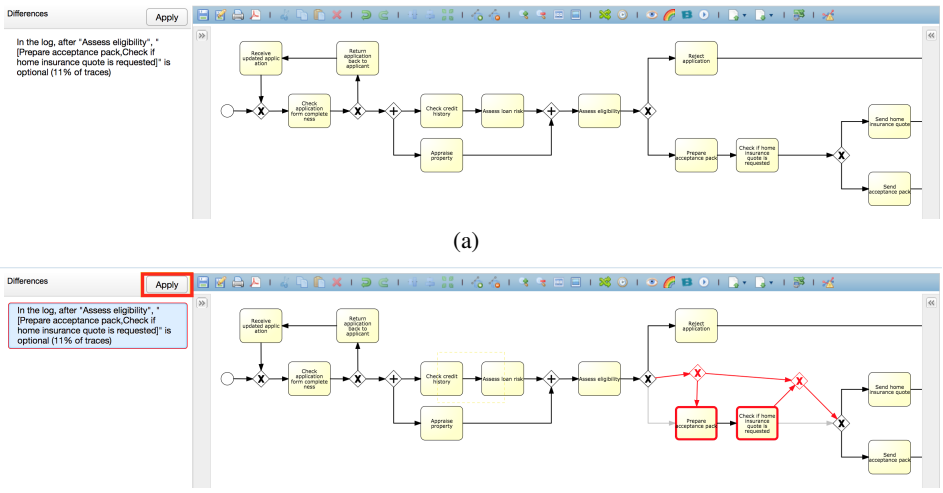


Fig. 2: Compare plugin

In the graphical representation of the differences, the elements to be inserted (gateways, sequence flows and tasks) and tasks affected by the differences are highlighted in red, whereas elements that need to be removed are grayed out, see Figs. 3(a) and 4 for some examples. Once a difference is selected, its graphical representation is overlaid over the model (Fig. 3(b)), then the user can click the button *Apply* (left-hand side of the screen) to make the suggested changes permanent in the model. Everytime a repair is applied, the list of differences is recomputed and refreshed. However, the user does not necessarily need to apply the suggested change, but can also (e.g. based on domain knowledge) decide to repair the model according to his own insights. Observe that the user has access to the full range of features supported by Apromore, for instance for editing, saving, printing, simulating, improving the layout of the models, etc.

2 Significance and Maturity

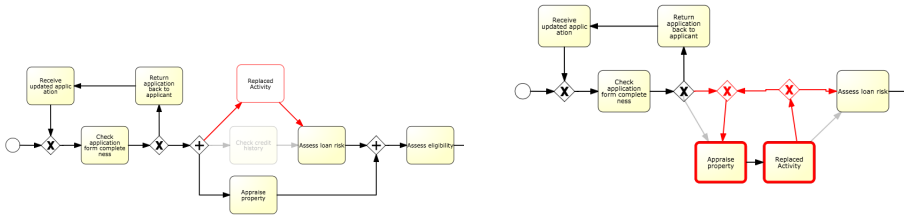
Our toolchain exhibits a novel approach for repairing process models, such that it better reflects the behavior observed in an event log. The presented approach is semi-automated and incremental, such that the user can decide which differences to re-



(a)

(b)

Fig. 3: Differences detected from a model to log comparison.



(a) (b)

Fig. 4: Example of repair changes suggested by the repair plugin.

pair based on their visual representation and their importance. As such, the presented toolchain is particularly useful for business analysts who want to maintain as much as possible the process model that is already in place, but would like to adjust it to better reflect reality, without ending up with an unreadable or highly generalized model.

This approach provides for the first time a tool that allows for a semi-automated repair where the users have full control over which differences to repair, the extent to which all differences should be repaired, and how to repair them. As opposed to existing approaches, the repair feature of Apromore prevents having repaired models with a very low precision and a very low similarity to the original model.

The Apromore features presented in this paper have been evaluated extensively with respect to accuracy, scalability and advantages over existing approaches. An empirical evaluation had been conducted on a collection of synthetically modified model-log pairs, and a real-life model-log pair [3]. The results showed that the proposed approach leads to repaired models with a much better precision and higher structural similarity relative to two state-of-the-art automated process model repair methods.

Apromore is the result of over seven years of ongoing development and is currently in version 5.0. The platform is implemented and deployed as a Software as a Service via a service-oriented architecture. Apromore combines several technologies. It uses ZK as the AJAX front end, Spring as the Java development framework, Maven as the dependency manager, OSGi as the plugin architecture, and EclipseVirgo as the OSGi-based application server. As a result, Apromore is an extensible framework, allowing new plugins to be easily added to an ecosystem of advanced capabilities for analyzing and managing process models. These plugins include presentation capabilities with respect to process model restructuring, filtering of models based on process-related aspects, searching and querying for specific process patterns, and advanced design and repair of process models, including configuring and merging of existing models. Additionally, the plugins comprise evaluation capabilities to assess the quality, correctness and compliance of models, along with simulation and conformance checking techniques for benchmarking. Furthermore, Apromore provides full import and export functionality to a large variety of business process modeling languages and log formats, such as BPMN, XPDL, EPML, ARIS, YAWL, PNML, XES and MXML.

3 Screencast

A screencast of Apromores compare feature can be found at `https://youtu.be/3d00pORc9X8`. The public release of Apromore is available at `http://apromore.org` and its source code can be downloaded under the GNU LGPL license version 3.0 from `https://github.com/apromore/ApromoreCode`.

References

1. Fahland, D., van der Aalst, W.M.: Model repairaligning process models to reality. Information Systems **47** (2015) 220–243
2. Polyvyanyy, A., Van Der Aalst, W.M., Ter Hofstede, A.H., Wynn, M.T.: Impact-driven process model repair. ACM Transactions on Software Engineering and Methodology **25**(4) (2016) 28
3. Armas-Cervantes, A., La Rosa, M., Dumas, M., García-Bañuelos, L., van Beest, N.R.T.P.: Interactive and incremental business process model repair. (2017) Tech. report, http://eprints.qut.edu.au/106611/.
4. Armas-Cervantes, A., van Beest, N.R.T.P., Dumas, M., García-Bañuelos, L., La Rosa, M.: Behavior-based process comparison in apromore. In: BPM (Demos). (2016) 34–38
5. García-Bañuelos, L., van Beest, N.R.T.P., Dumas, M., La Rosa, M.: Complete and interpretable conformance checking of business processes. IEEE Transactions on Software Engineering (2017) Accepted February 2017.
6. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri Nets, Event Structures and Domains, Part I. TCS **13** (1981) 85–108
7. Armas, A., Baldan, P., Dumas, M., García-Bañuelos, L.: Behavioral comparison of process models based on canonically reduced event structures. In: BPM 2014, Springer (2014) 267–282