HITL-AB-BPM: Business Process Improvement with AB Testing and Human-in-the-Loop

Aaron F. Kurz[1,3], Bennet Santelmann[1], Timo Großmann[1], Timotheus Kampik[3], Luise Pufahl[2,*] and Ingo Weber[2]

[1]*Technische Universitaet Berlin, Berlin, Germany*
[2]*Software & Business Engineering, Technische Universitaet Berlin, Berlin, Germany*
[3]*SAP Signavio, Berlin, Germany*

Abstract
Organizations improve their business processes regularly to react to clients' requests, changing business environments, or regulations. However, applying a changed process design directly in an organization is risky: frequently, process changes do not result in actual improvements. In this demo, we present a tool for AB testing of process versions with reinforcement learning techniques and a new approach to human control of the procedure. The goal of the tool is a fair and fast comparison in the production environment. As the system might not have all information required to make informed decisions, we propose how human experts can be put in control.

Keywords
Process redesign, AB testing, Multi-armed bandit, Human-in-the-loop

1. Introduction

A key challenge of Business Process Management (BPM) is the implementation of continuous improvement practices that ensure business processes do not stagnate. Two core issues when introducing a new version of a process concern risk and fairness. As discussed in earlier research [1, 2], new versions of a process often do not amount to actual improvements, entailing the risk of switching to an inferior version. Fairness of comparison between two process versions is an issue in the traditional BPM lifecycle [3], where the old version is replaced with a new version. Any comparison in such a setting suffers from confounding factors and uncontrolled variables. To counter these issues, in earlier work [1] the AB-BPM methodology was proposed, where two versions of a business process (A and B) are run in production side-by-side, allowing a fair comparison. AB testing is a standard method from DevOps and is nowadays used widely in Business-to-Consumer (B2C) applications. During AB tests, two versions of a product (e.g., a website) are deployed in parallel, and the decision on which version is preferential depends on

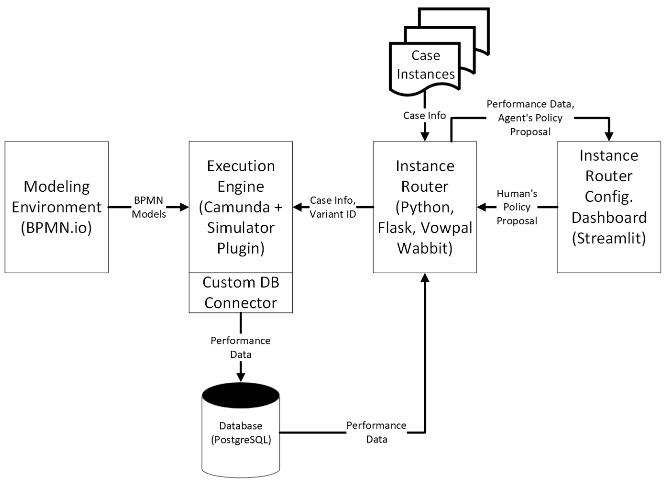
Proceedings of the Demonstration & Resources Track, Best BPM Dissertation Award, and Doctoral Consortium at BPM 2021 co-located with the 20th International Conference on Business Process Management, BPM 2022, Münster, Germany, September 11-16, 2022
*Corresponding author.
aaron.f.kurz@campus.tu-berlin.de (A. F. Kurz); bennet.santelmann@campus.tu-berlin.de (B. Santelmann); timo.grossmann@campus.tu-berlin.de (T. Großmann); timotheus.kampik@sap.com (T. Kampik); luise.pufahl@tu-berlin.de (L. Pufahl); ingo.weber@tu-berlin.de (I. Weber)

the data obtained during these experiments. The application of AB testing to business processes brought about new challenges, which have been addressed in the earlier work. Specifically, in AB-BPM, process instantiation requests are routed dynamically to either one of the versions, using approaches from Reinforcement Learning (RL) and observations of the performance of the instances. This dynamic routing is done to minimize the risk of exposing customers to sub-optimal process versions longer than necessary. The AB-BPM methodology is geared towards iterative process changes, but the authors note that it may also be used for more radical improvements. However, so far, AB-BPM tools have not been released, and the earlier work did not study how to involve humans in the AB testing procedures.

This demo of our Human-In-The-Loop AB-BPM (*HITL-AB-BPM*) tool closes the above two gaps: we present an open-source tool[1] that implements the AB-BPM approach and amends it with human control. In the remainder, we present the architecture and the functionality of the human-in-the-loop AB-BPM tool in Sect. 2 and demonstrate it in Sect. 3 along with a discussion of the tool's maturity.

2. Tool Description

Architecture To realize the HITL-AB-BPM tool, we have developed an instance router that decides, using a Reinforcement Learning (RL) algorithm and under consideration of feedback from a human expert, which case is executed through which process version. It interacts with the Camunda[2] execution engine as shown in Fig. 1. The HITL-AB-BPM tool consists of three main components: (1) the front end for the interaction with the process expert/analyst, (2) the back end, and (3) the database (PostgreSQL[3]). The front end is the single point of interaction with the user (human in the loop) and is built using Streamlit[4]. It communicates with the back end via RESTful API endpoints. The back end is built using the micro-framework Flask[5]. It handles the logic of i) communicating with the process engine through Camunda's API to deploy process models, start instances, and retrieve relevant execution information; ii) storing meta-information about the processes and their performance in the


Case Instances
Case Info
Performance Data, Agent's Policy Proposal
Modeling Environment (BPMN.io)
BPMN Models
Execution Engine (Camunda + Simulator Plugin)
Custom DB Connector
Case Info, Variant ID
Instance Router (Python, Flask, Vowpal Wabbit)
Human's Policy Proposal
Instance Router Config. Dashboard (Streamlit)
Performance Data
Database (PostgreSQL)
Performance Data
Figure 1: Architecture and used technologies.

[1]https://github.com/aaronkurz/hitl-ab-bpm, accessed 2022-06-22; this paper refers to release v0.1.0
[2]https://camunda.com/platform-7/workflow-engine/
[3]https://postgresql.org, last accessed 2022-03-19
[4]https://streamlit.io/, last accessed 2022-03-19
[5]https://flask.palletsprojects.com/en/2.0.x/, last accessed 2022-03-19

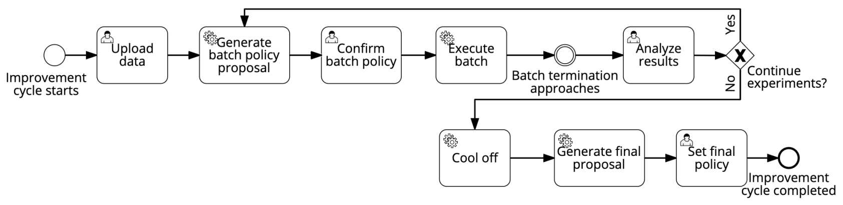


Figure 2: Operational flow of the application. User tasks denote activities that have to be executed by the process expert, whereas service tasks are executed by the application itself.

database; iii) training the RL agent and extracting relevant information from it; iv) hosting the routing algorithm used to map requests from process instances to process versions v) handling the requests from the front end. We use an RL-based recommender system algorithm (more specifically, a so-called contextual multi-armed bandit algorithm) that is part of Vowpal Wabbit, an open-source online-learning machine learning library[6] with Python bindings. The Camunda engine –with simulator plugin for demonstration purposes– executes process instances and provides necessary data to train the RL agent.

Functionalities The HITL-AB-BPM tool incorporates a set of functionalities that aim to strike a balance between exploration and exploitation when evaluating process versions while supporting human supervision and interference. The operational flow of the application can be seen in a process-orientated view in Figure 2. The user of the tool is a process expert. Therefore, the terms user, process expert, and human expert are used interchangeably.

First, the process expert uploads BPMN files for each process version and historical process execution data about the default ("old") version. The execution data is used to evaluate new instances that are part of the experiment, *i.e.*, the reward function judges the duration of experimental instances in relation to the production duration data from the old version. This reward strategy is in line with prior work [1]. Note that although the framework supports different performance indicators in theory, the current version of the tool uses only duration for the evaluation. Duration is relevant in most scenarios and available for any process. The process versions are then deployed to and executed in Camunda. The routing of incoming process instantiation requests to either version is based on so-called *batch policies*. A *batch* contains a certain amount of process instances, that are created in response to incoming process instantiation requests, and then monitored and evaluated. In a batch policy, the user can specify how many of the following instances will be part of the next batch and with which probability each version will be instantiated. To allow for more control, the user may differentiate these probabilities by contextual factors. For this prototype, the customer category is used as the relevant contextual factor. In theory, different contexts could be used here.

Batch policy proposals are provided by the RL agent and suggest how instances should be routed so as to most efficiently explore and then exploit the knowledge about which process version is preferential. The first batch policy proposal is naïve, suggesting a 50/50 split for each customer category, since no experimental data has been obtained to inform a more specific

[6]https://vowpalwabbit.org/, last accessed 2022-03-19

decision. It is presented to the user immediately after the upload of the data. Then, the user has to submit the first batch policy: setting the batch size and adjusting the routing probabilities. The following process instances are then routed according to this policy until the batch size is reached. Instantiation requests between batch policies (when the user has to check the new batch policy proposal) or before the first batch policy are not part of the experiment; they are routed to the default version and not evaluated by the RL agent. After all instances of one batch policy have been started, the process engine is polled for information about the already finished instances, which is then used to train the RL agent. Subsequently, the agent creates a new batch policy proposal for the user to review. Along with this proposal, the process expert can also view additional data about the past experimental instances. After the process expert analyzed the results, they then have the following choices: i) continuing with another experimental batch, *i.e.*, to either accept or modify the proposal and to then set the batch policy; ii) ending the experiment and starting the cool-off phase. When they decide to continue the experiment, they can either accept the proposal presented by the RL agent or modify it before setting the new batch policy. This means that another batch of incoming process instantiation requests is routed according to this batch policy and considered for the RL agent and the analysis. This loop of reviewing the batch policy proposal, setting the batch policy, and running and learning from the new batch can be repeated as often as the user chooses. If they decide to end the experiment, the cool-off period starts, which lasts until all experimental instances terminated.

Afterward, the user is asked to set a *final policy*. This decision entails setting a winning version for each customer category. Subsequently, all incoming instantiation requests will be routed per this decision. The user can consult two primary resources while deciding: the data insights (overview of collected data) and the final proposal by the RL agent. The *final proposal* is the routing decision the RL agent would make if it would have to route an additional instance. It is important to note here that this proposal by the agent does not necessarily only contain the way the agent would exploit the collected knowledge, but might also be based on the need for more exploration. Additionally, we added the functionality to conclude the experiment at any time and make a *manual decision*, *i.e.*, static routing to either version of the process. The manual decision terminates the experiment. This functionality corresponds to a manual override, which might become necessary upon discovering severe flaws in one version, vocal customer complaints, or regulatory reasons. HITL-AB-BPM can be seen as an Augmented BPM approach [4], where the user provides a clear *frame* for the RL agent in terms of batches and manual override, thus enabling joint human/agent decision-making.

3. Demonstration and Future Work

The exemplary *helicopter licensing* process from Saytal *et al.*'s evaluation of AB-BPM [1] is used to demonstrate the prototype's functionality. For simulation purposes, two process versions are created based on data from the relevant business domain. The initial version – Version A – works through the required steps (tests and exams) of attaining a license sequentially. If one of the steps fails, the process ends in rejection. These steps are parallelized in the supposedly improved business process version (Version B). A person attempting to obtain a helicopter license can, for example, already take the medical exam while waiting for the eligibility test results. This parallelization should, in theory, reduce the execution time. To provide historic data of the default version, 100 instances of Version A are simulated beforehand, and the results are uploaded to the HITL-AB-BPM tool at the beginning of the experiment. For demonstration purposes, process execution is simulated in the process engine. One day is scaled down to one second in the simulation. Customer categories are introduced as contextual attributes that may affect process performance. A process instantiation request can either come from a private sector customer (*priv*) or a government entity (*gov*). Since our demonstration does not serve real client requests, the front end offers access to a client simulator (*development mode*). For the evaluation, the client simulator sends bundles of requests at regular intervals.

To demonstrate HITL-AB-BPM, we ran the following concrete experiment. Initially, the routing to the new version (Version B) was set to be lower for government customers, to reduce the potential risk of losing this important customer. For priv 80% and for gov 20% of instantiation requests were routed to Version B. After the first batch finished, the RL agent presents a new batch policy proposal. The proposal indicated that Version B was faster and preferential to Version A. For priv, the RL agent proposed to route 91% of instances to version B, and 79% for gov. The performance data supports the proposal. The average duration of Version A was 72 seconds, while the average duration of B was only 19 seconds. Due to the small batch size, the RL agent might opt to require more exploration before fully committing to B. For the second batch, the human expert accepts the proposed batch policy. The final proposal is in line with the expectations: the RL agent's probability to choose B is 96% for the priv category and 95% for the gov category. The performance data matches the final proposal.

The presented software system is a proof-of-concept (research) prototype; automated tests have been developed and run in a continuous integration pipeline, including process-level tests with the above helicopter licensing process models. The prototype has generic capabilities – *i.e.*, it integrates with a BPMN 2.x standard-compliant process execution engine and it can be used to optimize any process that can run on the engine. Still, the following conceptual and technical challenges need to be addressed to further facilitate its adoption. i) HITL-AB-BPM would benefit from more lose coupling regarding technical execution aspects, *i.e.*, not only from the specifics of a particular engine, but from BPMN-based model execution in general. In alignment with this, future work can detach the optimization core component of HITL-AB-BPM for example, by making it available as a Python library (PyPI package). ii) The current implementation focuses on AB testing: it always experiments with exactly two process versions. An extension could expand testing capabilities to larger numbers of processes, *i.e., multi-variate testing*.

Acknowledgments. We thank Diana Baumann, Omar Sharif, Jiefu Zhu, and Konstantinos Poulios for their contributions to the implementation.

References

[1] S. Satyal, I. Weber, H. Paik, C. Di Ciccio, J. Mendling, Business process improvement with the AB-BPM methodology, Information Systems 84 (2019) 283–298.

[2] S. Satyal, I. Weber, H. Paik, C. Di Ciccio, J. Mendling, AB testing for process versions with contextual multi-armed bandit algorithms, in: Advanced Information Systems Engineering, 2018, pp. 19–34.

[3] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, 2nd ed., Springer, 2018.

[4] M. Dumas, F. Fournier, L. Limonad, A. Marrella, M. Montali, J.-R. Rehse, R. Accorsi, D. Calvanese, G. D. Giacomo, et al., Augmented business process management systems: A research manifesto, 2022. arXiv:2201.12855.