FrAIm: A What-If Analysis Tool Enabling Framed Autonomy via Automated Planning

Paul H. Wittlinger[1,*,†], Giacomo Acitelli[2,†], Anti Alman[1,†], Fabrizio Maria Maggi[1,†] and Andrea Marrella[2,†]

[1]Free University of Bozen-Bolzano, NOI Techpark via Bruno Buozzi 1, Bolzano, 39100, Italy
[2]Sapienza University of Rome, Via Ariosto 25, 00185, Rome, Italy

Abstract

AI-Augmented Business Process Management Systems (ABPMS) extend existing process-aware information systems through the integration of advanced AI capabilities. A core aspect of ABPMS is *Framed Autonomy*, denoting the capability of the system to autonomously and independently choose how to progress process executions within the given *Process Frame*, which consists of (potentially conflicting) procedural and declarative process specifications. Moreover, an ABPMS should support the completion of partial process executions, even if they conflict with the *Process Frame*. In this paper, we demonstrate FrAIm, a What-If Analysis tool that leverages automata theory and automated planning to explore the behavior induced by framed autonomy.

Keywords

AI-Augmented Business Process Management Systems, Framed Autonomy, Automated Planning, Hybrid Business Process Representations

1. Introduction

The rise of increasingly complex and capable AI systems facilitates the improvement of traditional software systems through the integration of advanced AI models. In the case of Business Process Management Systems (BPMS), this evolution has led to the definition of a new class of AI-augmented Business Process Management Systems (ABPMS), which are equipped with advanced reasoning capabilities that enable them to autonomously operate within their specified operating domains. Dumas et al. describe the components of ABPMS in their research manifesto [1]. The key distinctions of ABPMS are: i) enhancement of the traditional lifecycle phases (model, execute), and ii) addition of novel tasks made possible by AI capabilities (adapt, explain, improve). These tasks are executed within the context of a *Process Frame*, which represents a generalized notion of a process model. A Process Frame may consist of (arbitrarily many and potentially conflicting) procedural and declarative process specifications, which, in combination, define the overall behavior of the underlying process.

Initializing an ABPMS with a Process Frame is a prerequisite for it to act independently of human actors. However, its autonomous behavior is only valid as long as it operates within the boundaries specified by the Process Frame.[1] This key feature is referred to as *Framed Autonomy* [2], and it requires reasoning about the current state of the process, identifying possible violations, and reacting to any deviations from the frame. In other words, an ABPMS must be able to *plan* the ideal execution path, *monitor* ongoing process instances, and reactively *re-plan* in case the initial plan is not followed.

In this paper, we introduce FrAIm, a What-If Analysis tool for exploring the behavior induced by framed autonomy in the presence of (potentially conflicting) process specifications within the Process

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 23rd International Conference on Business Process Management (BPM 2025), Seville, Spain, August 31st to September 5th, 2025.
*Corresponding author.
†These authors contributed equally.
pwittlinger@unibz.it (P. H. Wittlinger); acitelli@diag.uniroma1.it (G. Acitelli); anti.alman@unibz.it (A. Alman); maggi@inf.unibz.it (F. M. Maggi); marrella@diag.uniroma1.it (A. Marrella)
0009-0002-5893-1724 (P. H. Wittlinger); 0000-0002-8194-3611 (G. Acitelli); 0000-0002-5647-6249 (A. Alman); 0000-0002-9089-6896 (F. M. Maggi); 0000-0002-1031-0374 (A. Marrella)

[1]The degree of autonomy can be limited by the designer.

Frame, combined with existing partial process executions that may have violated the Process Frame. More concretely, FrAIm builds on automata-theoretic techniques to compute cost-optimal plans for completing a process execution based on an input Process Frame and a user-defined trace prefix. As such, FrAIm represents an important step toward achieving Framed Autonomy via automated planning.

In the following, Section 2 provides a brief background on automated planning, Section 3 describes the FrAIm tool's functionality, Section 4 evaluates FrAIm's performance, and Section 5 concludes the paper.

2. Automated Planning in AI

Automated Planning is a branch of Artificial Intelligence that aims to solve complex problems in a domain-agnostic way. State-of-the-art planners have demonstrated success across a variety of problems with differing levels of complexity, including successful applications in BPM tasks [3, 4, 5, 6].

The Planning Domain Definition Language (PDDL) [7] is the de facto standard for describing planning problems. A problem consists of an *initial state*, which holds true at the start of the plan, and a *final state*, which must be reached at the end of the execution. Every planning problem is accompanied by a planning *domain*, which defines the relevant objects and states, as well as all possible actions that can be taken to solve the problem.

In the context of FrAIm, the planning problem is defined by the given Process Frame (expressed as procedural and declarative process models), while the domain consists of actions that allow the planner to navigate the state space of the process. This includes actions for detecting violations and recovering after a violation has occurred. The details of the underlying formal approach are beyond the scope of this paper and can be found in [8].

3. Tool Description and Features

FrAIm is a Java-based application that combines the strengths of Automated Planning and Hybrid Process Representations to enable interactive exploration of the behavior induced by the ABPMS Process Frame in the presence of different partial process executions. The source code of FrAIm is publicly available online.[2] The tool, along with sample data and a description of how to install the Fast Downward Planning System [9], can be downloaded here.[3] A video demonstration of how to use the tool is provided here.[4]

When launched, the user is expected to provide three inputs: *i)* procedural models in the form of Petri nets, *ii)* DECLARE specifications for the declarative process rules, and *iii)* optionally, a prefix trace. The Process Frame is calculated as the union of the DECLARE specification and the Petri net, with the implicit goal of reaching the final marking of the net while satisfying all constraints.

It is important to note that these models do not necessarily need to be consistent, as potential violations can be accounted for in the plan (by incurring a cost). The consistency of the Process Frame can be verified in FrAIm by providing an empty prefix and clicking "Run Planner" to generate a trace within the given Process Frame. If the Process Frame is internally consistent, the returned plan will have cost 0; if the trace cost is greater than 0, the Process Frame is not consistent.

The process models are visualized, and the trace (including the given prefix and, if generated, its continuation) can be replayed on the currently selected models using the timeline controls at the bottom of the screen or by clicking on the events on the left-hand side. The visualization of both the selected Petri net and DECLARE model is updated during the replay, according to the last processed event of the trace (cf. Figure 1). For the Petri net, the visualization tracks transitions that have fired (blue), caused violations (red), are enabled (white), or are not enabled (grey). For DECLARE, the visualization follows

[2] https://github.com/giacomo1096/FramedAutonomyTool and https://github.com/pwittlinger/framedinterface

[3] https://scientificnet-my.sharepoint.com/:u:/g/personal/pwittlinger_unibz_it/EdDfpSBb2UVDta0TLhl6DsYBV_enMfG3rIWa0yo-qDXbbQ?e=X1c38t

[4] https://youtu.be/Ty2JsRhHewM

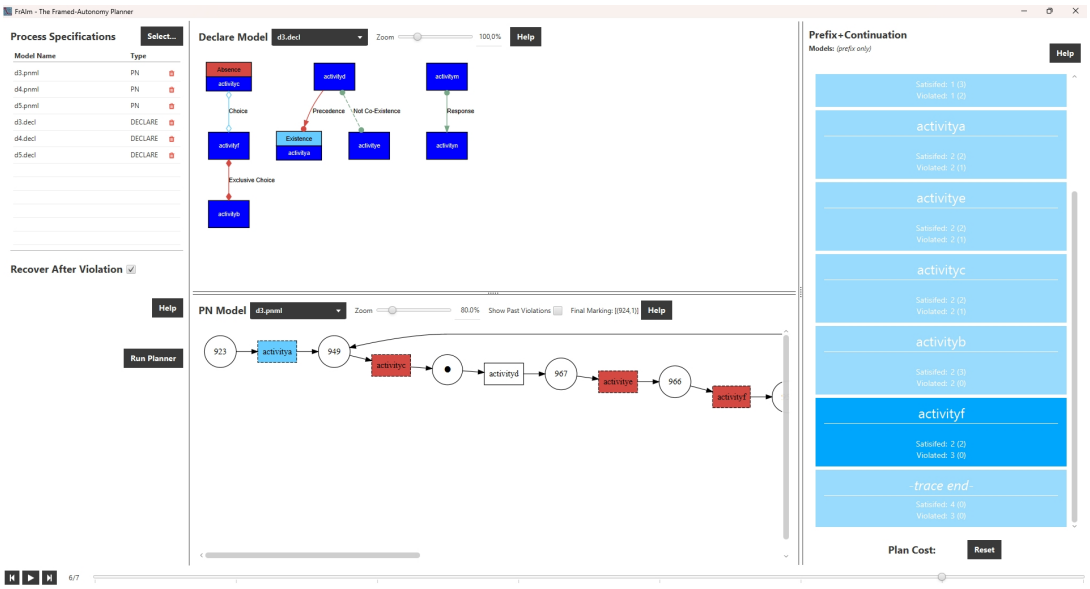


Figure 1: Screencapture highlighting a Process Frame (middle) and a violating trace prefix consisting of the 6 activities *<ActivityC, ActivityA, ActivityE, ActivityC, ActivityB, ActivityF>*. Only the last 5 are visible in the image.

the constraint coloring of [10], with red denoting permanent violations, yellow temporary violations, green temporary satisfactions, and blue permanent satisfactions.[5]

By default, FrAIm will recover from any violations of the Process Frame by resetting the violated element (and incurring a cost).[6] However, the user can also choose to discard permanently violated constraints by deselecting the "Recover After Violation" option.

By clicking the "Run Planner" button, FrAIm converts the provided inputs into a planning problem (PDDL format) and passes it to the Fast Downward planner to generate a trace continuation that reaches the final marking of the Petri net while satisfying all DECLARE constraints with minimal overall cost. The FrAIm tool then guides the user through the solution step-by-step, highlighting all fired/violated transitions and constraint states after each executed activity. The user can also replay the same trace on any other model by selecting it in the interface, thereby facilitating comparative analysis across multiple models.

Key features of FrAIm include: (i) DECLARE Visualizer, highlighting satisfied and violated constraints; (ii) Petri net Visualizer, showing the current marking, executed transitions, and violations; (iii) Interactive Trace Builder, allowing users to build the prefix by clicking on the corresponding action in the DECLARE or Petri net visualizer.

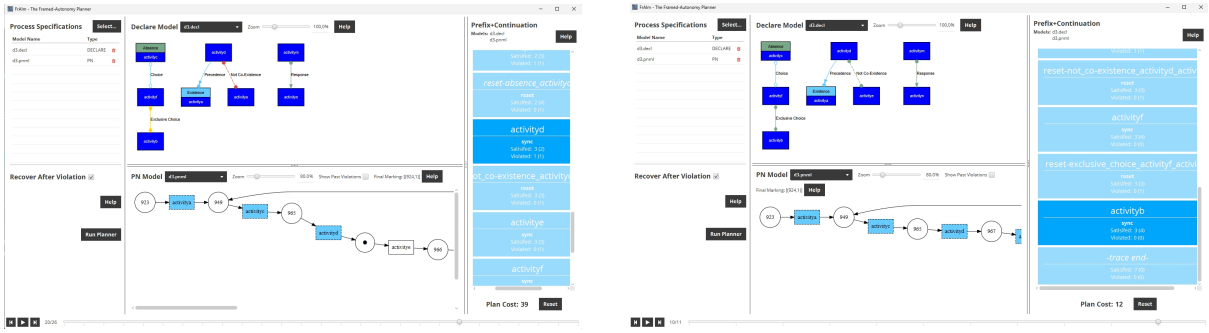
These elements can be seen in Figure 1. On the left-hand side, the user can find the currently uploaded process models, a help button, and a button to run the planner. The currently selected models are visualized in the middle of the screen (DECLARE model above, Petri net below). More information on the color scheme is available via the two "Help" buttons. The visualizations also allow users to build the trace prefix by clicking on activity names. The right-hand side of the tool displays the trace (consisting of the prefix and, after planning, its continuation). The bottom of the screen contains the trace replay controls. The user can step through the trace one event at a time, jump to a specific event (via the timeline or event list), and replay the trace as an animation.

As additional examples of the FrAIm tool in action, Figure 2 depicts two instances of the Process Frame composed of Petri net $\mathcal{N}_1$ and the DECLARE model with seven constraints taken from Table 1. Figure 2a has been run with prefix *<ActivityC, ActivityA, ActivityE, ActivityC, ActivityB, ActivityF>*; Figure 2b has been run with an empty prefix. In both cases, the "Recover After Violation" option has

[5]All temporary satisfactions and violations become permanent at trace termination.

[6]In the current version of FrAIm, every violation incurs a cost of 3, as specified by the cost model utilized. Alternative cost models (e.g. number of violations) are to be supported in future versions.

been ticked. As a result, both plans satisfy all DECLARE constraints and the Petri net. However, we can observe that the Process Frame is not consistent, as an empty prefix leads to a plan with a cost higher than 0. Also, the prefix used in Figure 2a contains violations with respect to the Process Frame, as indicated by the associated higher cost.



(a) Plan for a violating prefix trace with cost 39

(b) Plan for an empty prefix with cost 12

Figure 2: Comparing the results of a given Process Frame for a prefix trace against an empty prefix.

4. Tool Maturity

We tested the tool in [8] and summarize the main results here. The experiments consisted of three synthetic Petri nets from the literature [4], four associated DECLARE models (with 1, 3, 5, and 7 constraints), and prefix lengths ranging from 0 to 4 events, with each resulting combination being tested. For each experiment, the total processing time was measured, representing the time a user would need to wait for the optimal trace continuation to be computed. The results of the experiments are provided in Table 1. As shown in the table, the more components a frame contains and the longer the prefix, the more time is required to find a solution. However, the computation time is primarily affected by the size of the state space induced by the Petri net.

Table 1
Performance results (in seconds) for different Declare constraints and Petri nets.

Prefix length	1 Declare constraint	3 Declare constraints	5 Declare constraints	7 Declare constraints
$\mathcal{N}1$ - Petri net with 36 transitions and 34 places				
0	0.63	0.73	0.96	1.21
1	0.65	0.80	1.03	1.28
3	0.72	0.88	1.12	1.40
4	0.76	0.92	1.18	1.45
$\mathcal{N}2$ - Petri net with 25 transitions and 27 places				
0	18.64	19.09	20.35	21.43
1	19.27	19.90	21.11	22.20
3	20.80	21.45	22.71	24.07
4	21.61	22.23	23.38	24.71
$\mathcal{N}3$ - Petri net with 95 transitions and 88 places				
0	48.39	51.78	56.27	61.27
1	49.73	52.50	57.33	62.45
3	51.29	53.17	57.84	64.77
4	52.14	54.69	59.48	64.97

5. Conclusions and Future Work

In this paper, we presented the core functionality of our planning-based What-If Analysis tool, FrAIm. FrAIm leverages Automated Planning to provide detailed guidance on completing partial process

executions within the boundaries of a given Process Frame. The tool offers insights into the costs of continuing user-defined partial executions by analyzing the interplay between procedural and declarative constraints. By simulating how different prefixes evolve under these constraints, FrAIm helps users understand the trade-offs and implications of various execution paths, particularly regarding violations and recovery costs. The integrated Trace Builder facilitates testing and comparing ABPMS behavior based on different prefixes.

The version described in this paper marks the initial release of the tool. Future versions will offer richer interaction capabilities, including an enhanced Trace Builder for seamless construction and modification of prefixes. Users will also be able to define custom costs for constraint violations, enabling more fine-grained analysis. We also plan to support additional types of constraints, incorporating the data perspective (through data-aware models like MP-Declare and Data Petri Nets), as well as temporal aspects (such as throughput time) and resource perspectives, further broadening the tool's applicability. Moreover, we aim to handle more heterogeneous goals, not only minimizing constraint violations but also optimizing throughput time, outcome quality, and resource costs, while supporting trade-offs among these competing objectives.

Acknowledgments

The work of A. Alman and F. M. Maggi was partially funded by the NextGenerationEU FAIR PE0000013 project MAIPM (CUP C63C22000770006). The work of A. Marrella and G. Acitelli was supported by the project FOND-AIBPM, the PRIN 2022 project MOTOWN, and the PNRR MUR project PE0000013-FAIR.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] M. Dumas, F. Fournier, L. Limonad, A. Marrella, M. Montali, J. Rehse, R. Accorsi, D. Calvanese, G. De Giacomo, D. Fahland, A. Gal, M. La Rosa, H. Völzer, I. Weber, AI-augmented business process management systems: A research manifesto, ACM Trans. Manag. Inf. Syst. 14 (2023) 11:1–11:19.

[2] M. Montali, Constraints for process framing in AI-augmented BPM, in: Business Process Management Workshops, volume 460 of *LNBIP*, Springer, 2022, pp. 5–12.

[3] G. Bergami, F. M. Maggi, A. Marrella, M. Montali, Aligning data-aware declarative process models and event logs, in: BPM, volume 12875 of *LNCS*, Springer, 2021, pp. 235–251.

[4] M. de Leoni, A. Marrella, Aligning real process executions and prescriptive process models through automated planning, Expert Syst. Appl. 82 (2017) 162–183.

[5] S. Agostinelli, F. M. Maggi, A. Marrella, M. Mecella, Verifying Petri net-based process models using automated planning, in: EDOC Workshops, IEEE, 2019, pp. 44–53.

[6] S. Edelkamp, S. Jabbar, Action planning for directed model checking of Petri nets, in: MoChArt@CONCUR/SPIN, volume 149 of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2005, pp. 3–18.

[7] M. Fox, D. Long, PDDL2.1: an extension to PDDL for expressing temporal planning domains, J. Artif. Intell. Res. 20 (2003) 61–124.

[8] G. Acitelli, A. Alman, F. M. Maggi, A. Marrella, Achieving framed autonomy in AI-augmented business process management systems through automated planning, Information Systems 133 (2025) 102573.

[9] M. Helmert, The Fast Downward planning system, Journal of Artificial Intelligence Research 26 (2006) 191–246.

[10] F. M. Maggi, M. Montali, M. Westergaard, W. M. P. van der Aalst, Monitoring business constraints with linear temporal logic: An approach based on colored automata, in: BPM, 2011, pp. 132–147.