AutoBPMN.AI Execution Modeller: Bring Processes to Life with Conversational Process Execution

Nataliia Klievtsova[1,*], Juergen Mangler[1], Matthias Ehrendorfer[1] and Stefanie Rinderle-Ma[1]

[1]Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany

Abstract

Agentic approaches for generating process models from natural language text have merely focused on control flow aspects so far, including our tool AutoBPMN.AI. However, in order to develop fully executable processes for real-world applications, control flow models must be equipped with data flow, and endpoints. Hence, this demo paper presents 2 additional agents which extend the capabilities of the existing AutoBPMN.AI agent (presented at the BPM demo session 2025) to automatically (1) select functionality from a service registry, (2) create and (3) validate data flow. Users can seamlessly interact with the agents for creating and refining executable process models, which are ready to execute. We will demonstrate this in an IoT scenario, where users can converse with the system to create dynamic processes for dimming the light based on environmental conditions.

1. Introduction

Generative AI has accelerated the creation of process models from textual sources such as process descriptions [1] or regulatory documents [2] by users of different technical background, especially domain experts [3]. Our software AutoBPMN.AI was demonstrated at the BPM demo session in 2025 [4], enabling transforming text into process models and vice versa as well as refining process models based on natural language instructions (conversational process modeling). As such, the AutoBPMN.AI 2025 version supports the full conversational cycle of modeling and refinement, but focused on the control flow of the processes.

One of the major benefits of AutoBPMN.AI is its embedding into the Cloud Process Execution Engine cpee.org which is open source, has been successfully used for process operationalization and execution in different domains, e.g., 16 manufacturing scenarios across 7 companies [5], and has been downloaded more than 570.000 times[1]. Hence, in this 2026 demo we extend AutoBPMN.AI towards the creation of executable processes by connecting the BPMN activities to functionalities (REST endpoints) and by generating data flow for process models. Especially the data flow part is tricky, as it requires ensuring each BPMN activity receives the correct input, while generating code that for each BPMN activity captures its output, if necessary transforms it, and thus makes the output available for XOR conditions and as input for subsequent BPMN activities. This constitutes a major step towards finally reaching the dream of pervasive process thinking and orientation in companies.

While agentic orchestration with MCP (Model Context Protocol) is now routinely utilized to allow LLM directly to automate tasks by sequentially invoking agents, in this work we want to focus on automatically generating BPMN artefacts that serve the same purpose, but can be manually checked by humans before execution, and provide a stable basis for repeated executions.

In order to achieve this goal, we developed the following agentic architecture where 3 different agents interact to provide executable process models:

Proceedings of the BPM 2026 Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum (BPM 2026) , Toronto, Canada, September 27th to October 2nd, 2026.

*Corresponding author.

nataliia.klievtsova@tum.de (N. Klievtsova); juergen.mangler@tum.de (J. Mangler); matthias.ehrendorfer@tum.de (M. Ehrendorfer); stefanie.rinderle-ma@tum.de (S. Rinderle-Ma)

0009-0009-9010-2855 (N. Klievtsova); 0000-0002-6332-5801 (J. Mangler); 0000-0002-7739-9123 (M. Ehrendorfer); 0000-0001-5656-6108 (S. Rinderle-Ma)

[1]https://rubygems.org/gems/cpee/

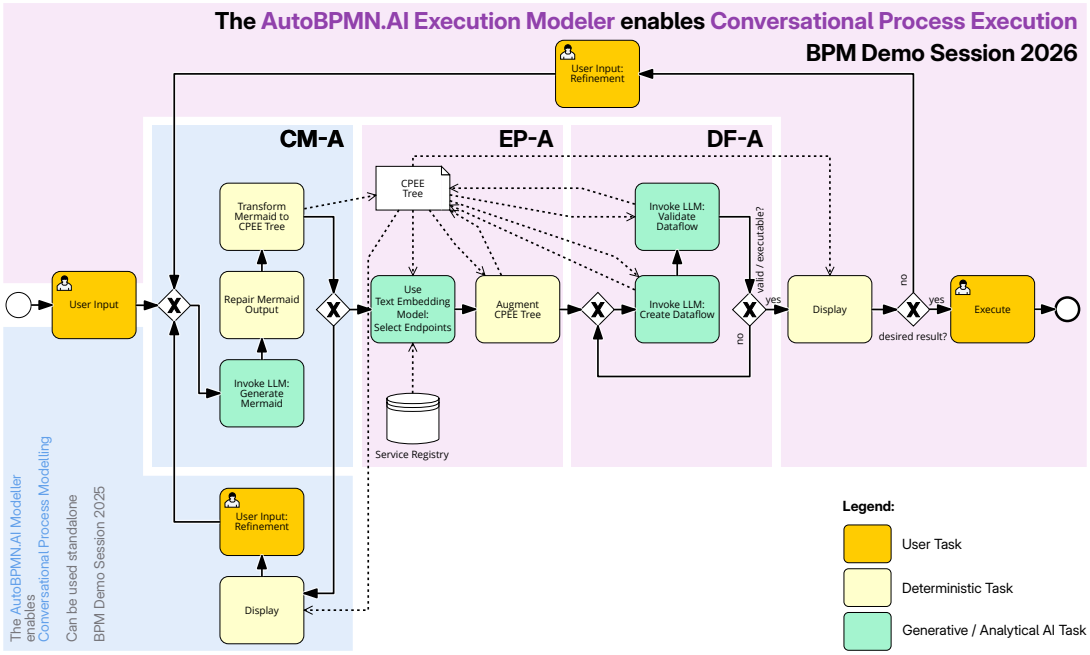

Figure 1: Pipeline to Realize the AutoBPMN.AI Execution Modeler

- **CM-A:** The Conversational Process Modeling Agent presented at the BPM demo session 2025, creates the structure of the basic BPMN, including textual labels for activities and conditions.
- **EP-A:** The EndPoint Agent utilizes the input from the CM-A, and selects REST endpoints from a service repository to best approximate the functionality described by the activity labels, and the sequence of activities. The result is an augmented BPMN model.
- **DF-A:** The Data Flow Agent finally takes the input from the EP-A and creates logic that transforms and assigns (for each BPMN activity) values returned by the REST services to data object available in the process context. These data objects are used to replace XOR labels with functional conditions and also serve as input for functionalities selected for other BPMN tasks. The DF-A also performs **data flow validation**: as the initially generated data flow often has errors and gaps, we repeatedly (in a loop) invoke an LLM to correct these errors, and add Script tasks to fill gaps.

When an existing process model is improved, the same agents are involved, the *CM-A* adds, deletes, or moves tasks, the *EP-A* checks for new endpoints and assigns them if necessary, and the *DF-A* augments and checks the data flow. All interactions with users are stored in logs for subsequent analysis purposes.

Section 2 introduces the novel agents EP-A and DF-A and their interaction with CM-A. The maturity of the software is discussed in Sect. 3 and links to software and video are provided in Sect. 4.

2. Agentic Endpoint Selection and Data Flow Creation

We demonstrate the agentic pipeline for creating process models with endpoints and data flow depicted in Fig. 1 on an IoT predictive lighting process scenario. In the first phase, CM-A[2] is employed by a user, e.g., a domain expert, to create the control flow of the scenario. This can be based on textual process description, an instruction, the import of a BPMN model, or a manually created process model. For the predictive lighting scenario, the following prompt is used to create the control flow model:

[2]In the BPM Demo Session 2025 we presented only the CM-A. The loop in Fig. 1 (blue area) shows how conversational process modeling is realized by utilizing just the CM-A

PREDICTIVE LIGHTING USER INPUT

Imagine I want to implement a smart home system to control the lights in the dining room. Depending on the current lighting conditions, the system would automatically adjust the brightness—either increasing or reducing it as needed—and turn the lights off completely at night.

Figure 2 depicts the result of creating the process model with CM-A based on the prompt above and using gemini (alternative LLMs in different versions are supported, i.e., gpt, mistral, gemma, qwen). We can see from the control flow model that augmentation with endpoints and data flow, including the decision logic at the XOR split, is required in order to make this model executable.

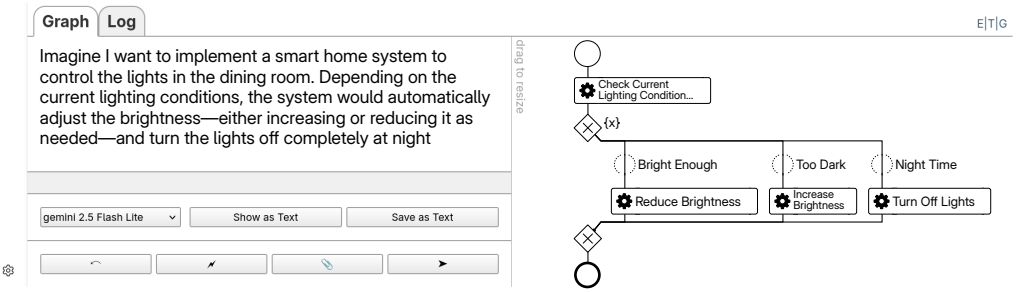

Figure 2: Creation of Control Flow Model in CM-A

Hence, following the pipeline in Fig. 1, the control flow model is passed to the EP-A. The EP-A accesses a service registry that contains a list of REST services with semantic annotations (which we automatically generated from OpenAPI specifications). The EP-A then utilizes the mxbai-embed-large[3] to find suitable functionalities to connect to each BPMN activity. It performs label-based matching between service annotations (cf. `<endpoint/>`) and BPMN task labels by computing the semantic similarity over their embedding vectors. Listing 1 shows an example of a service with an endpoint url (XML attribute) and various annotations for getting the ambient light that can be invoked by task `Check Current Lighting Conditions` in Fig. 2.

Listing 1: Service Repository Entry With Endpoint And Annotations.

```
<resource endpoint="https-get%3A%2F%2Fecho.bpm.in.tum.de%2Flight%2Fsensor%2F">
  <endpoint>Get ambient light</endpoint>
  <functionality>
    Get the ambient light in lux. This endpoint accepts an HTTP GET without any parameters.
        A rough guideline for how lux can be interpreted:
      * 0 means it is pitch black outside.
      * Below 100: Very low light (early morning, dusk, dawn, or city glow).
      * From 100 to 600: Low outdoor light (typically overcast).
      * Above 600, often around 1200 means, that it is sunny outside.
  </functionality>
  <output>
    The result is a single float value (lux) which can be accessed through 'result'.
  </output>
</resource>
```

The result of augmenting the control flow model with endpoints in EP-A is shown in Fig. 3. The task "`Check Current Lighting Conditions`" has been assigned the best matching endpoint.

The endpoint-augmented process model is then used by DF-A to create the corresponding data flow (cf. Fig. 1) using an LLM. In detail, this means to extract the information about input and output parameters of the endpoints (i.e., from an OpenAPI specification if available), and the creation of necessary data objects that hold (sometimes parts of - e.g., JSON keys) the services output. In Fig. 3, we can see that the access variable of the service call by task `Check Current Lighting Conditions` writes the process data object `ambient_light_lux` with the value returned by the endpoint (→ `Finalize`).

[3]Mixedbread AI 2024, mxbai-embed-large-v1, https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1, accessed 2026-07-02

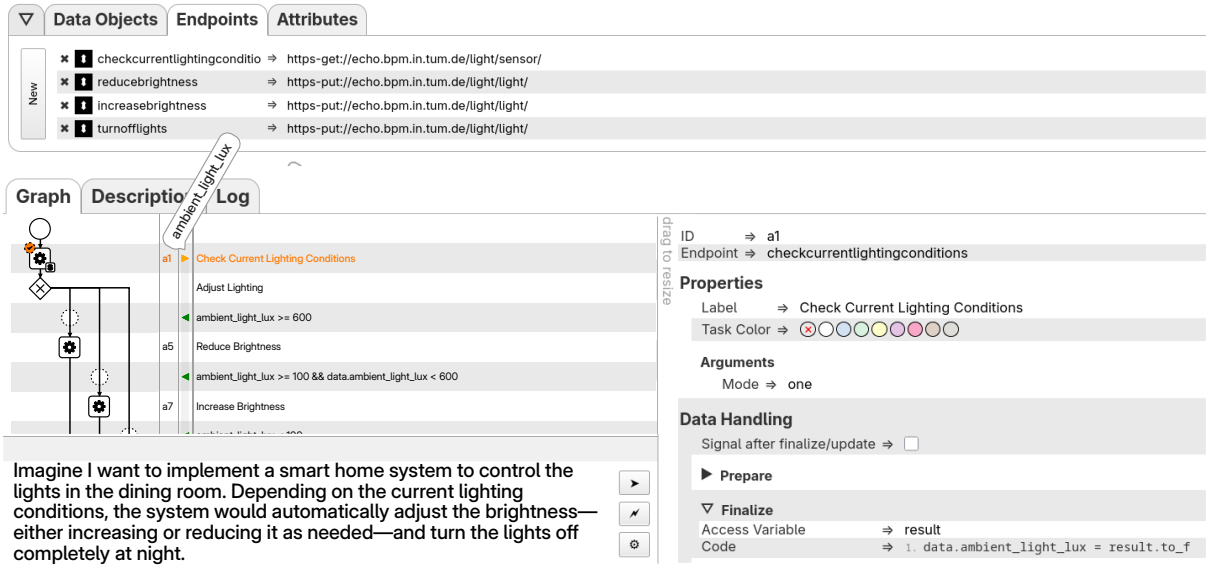

Figure 3: Augmenting Control Flow with Endpoints in EP-A

We can also see that the fuzzy decision conditions extracted from the text (e.g., "`Bright Enough`", cmp. Fig. 2) now utilize the extracted data object. Figure 3 displays the complete data flow: task `Check Current Lighting Conditions` writes data element `ambient_light_lux` which is then read by all branches of the subsequent alternative branching, building the basis for the decision. For `ambient_light_lux` $>=$ 600, task `Reduce Brightness` is executed. For 100 $<=$ `ambient_light_lux` $<$ 600, task `Increase Brightness` is executed. Otherwise, if `ambient_light_lux` $<$ 100, the lights are turned off. On top of Fig. 3, we can see all endpoints that are invoked by tasks, e.g., endpoint `reducebrightness` being invoked by task `Reduce Brightness`. As such the process model can be executed right away. To allow demo participants to experience the effects of running the process, we build a physical demo box, that simulates the IoT scenario (see Fig. 4).

3. Maturity

The underlying **process technology** realized by the engine cpee.org has been used for more than 15 years in different application domains ranging from office automation to manufacturing. The latter comprises 16 operationalized processes running across 7 different companies. It is open source and has been downloaded more than 570.000 times. For the executable processes realized with the agentic pipeline described in this demo this means an application readiness of 100%.

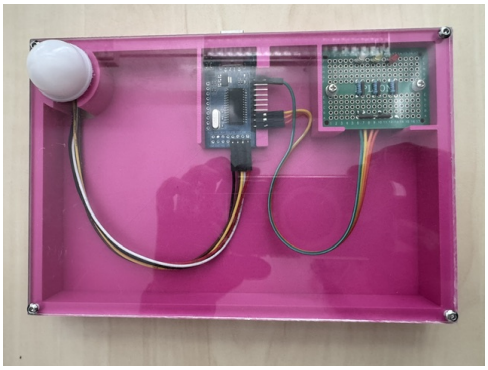

Figure 4: IoT Lighting Scenario Demo Box

The underlying conversational process modeling and automation **research** has been published in several papers and was partly conducted in a joint project with SAP Signavio which has, in turn, already led to product innovations.

The CM-A agent of AutoBPMN.AI has been demonstrated at the BPM 2025 conference and won the "tech wizardry" award[4]. Since then, we have showcased it in numerous demonstrations for partners

[4] https://www.bpm2025seville.org/awards/

from university hospitals, bioengineering, auditing, and higher education. AutoBPMN.AI has also been used by students of our lab courses for 3 semesters now.

Figure 4 shows the IoT demo box for the predictive lighting scenario. It can be plugged into any computer, connected to a publicly hosted REST service, and used together with the executable process model developed with the agentic pipeline depicted in Fig. 1. This combination of the executable process model (software) and the IoT demo box (hardware) provides a compact demonstration of physical AI [6], where AI agents not only process data but also interact with and control the physical world through connected sensors and actuators. We believe that AutoBPMN.AI can significantly lower the barrier of developing executable process models for users with non-technical background. In particular, AutoBPMN.AI fosters the rapid development of process models for i) communication and documentation [7], ii) simulation and development of digital process twins [8], and full-fledged process execution. In addition, to support and accelerate this development, future work will focus on reusability and integration of existing MCP specifications and tools into the proposed service registry.

4. Videos and Tool

The tool is available at autobpmn.ai. A new process can be created by selecting "Execution Modeler". The process model can be designed/improved as demonstrated in Sect. 2. User input examples can be found at autobpmn.ai. The user interface is realized as a CPEE cockpit[5]. The REST service exposing the 3 agents invoked by the user interface, as well as the prompts utilized by the agents are available on github[6]. A video demonstrating the functionality of the tool is available at https://autobpmn.ai/autobpmn_exec.webm (accessed 2026-07-01).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools in writing this paper, creating its figures, or creating the code for this demo. However, Generative AI was used to convert the video transcript into spoken voice.

References

[1] N. Klievtsova, T. Kampik, J. Mangler, S. Rinderle-Ma, Conversational process model redesign, Int. J. Cooperative Inf. Syst. 35 (2026) 2650004:1–2650004:37.

[2] C. Sai, S. Rinderle-Ma, Agentic generation of process models from regulatory texts, in: Cooperative Information Systems, 2025, pp. 370–387.

[3] N. Klievtsova, J. Mangler, T. Kampik, J. Benzin, S. Rinderle-Ma, How can generative AI empower domain experts in creating process models?, in: 19th International Conference on Wirtschaftsinformatik (WI 2024), Würzburg, Germany, September 16-19, 2024, Proceedings, AISeL, 2024, p. 66.

[4] N. Klievtsova, M. Ehrendorfer, J. Mangler, S. Rinderle-Ma, Autobpmn.ai: Conversational process modeling and automation, in: BPM Demos, 2025. URL: https://ceur-ws.org/Vol-4032/paper-40.pdf.

[5] S. Rinderle-Ma, J. Mangler, Process automation and process mining in manufacturing, in: Business Process Management, 2021, pp. 3–14.

[6] Y. Li, Z. Li, Y. Duan, A.-B. Spulber, Physical artificial intelligence (pai): the next-generation artificial intelligence, Frontiers of Information Technology & Electronic Engineering 24 (2023) 1231–1238.

[7] N. Klievtsova, J. Mangler, T. Kampik, S. Rinderle-Ma, Utilizing process models in the requirements engineering process through model2text transformation, in: RE, 2024, pp. 205–217.

[8] J. Benzin, J. Mangler, S. Rinderle-Ma, A service-oriented digital twin architecture for seamless reliability verification of iot systems, in: Service-Oriented Computing, 2025, pp. 175–190.

[5] https://cpee.org/, accessed 2026-07-01

[6] https://github.com/etm/cpee-llm