Conversational Process Model Redesign

Nataliia Klievtsova[1], Timotheus Kampik[2], Juergen Mangler[1], and Stefanie Rinderle-Ma[1]

[1]Technical University of Munich,, TUM School of Computation, Information and Technology,, Boltzmannstraße 3 85748 Garching b. München, Germany, {firstname.lastname@tum.de}
[2]SAP Signavio,, George-Stephenson-Straße 7-13, 10557 Berlin, Germany, timotheus.kampik@sap.com

Abstract

With the recent success of large language models (LLMs), the idea of AI-augmented Business Process Management systems is becoming more feasible. One of their essential characteristics is the ability to be conversationally actionable, allowing humans to interact with the LLM effectively to perform crucial process life cycle tasks such as process model design and redesign. However, most current research focuses on single-prompt execution and evaluation of results, rather than on continuous interaction between the user and the LLM. In this work, we aim to explore the feasibility of using LLMs to empower domain experts in the creation and redesign of process models in an iterative and effective way. The proposed conversational process model redesign (CPD) approach receives as input a process model and a redesign request by the user in natural language. Instead of just letting the LLM make changes, the LLM is employed to (a) identify process change patterns from literature, (b) re-phrase the change request to be aligned with an expected wording for the identified pattern (i.e., the meaning), and then to (c) apply the meaning of the change to the process model. This multi-step approach allows for explainable and reproducible changes. In order to ensure the feasibility of the CPD approach, and to find out how well the patterns from literature can be handled by the LLM, we performed an extensive evaluation. The results show that some patterns are hard to understand by LLMs and by users. Within the scope of the study, we demonstrated that users need support to describe the changes clearly. Overall the evaluation shows that the LLMs can handle most changes well according to a set of completeness and correctness criteria.

Keywords: Process Discovery, Process Models, Large Language Models, Process Redesign, Conversations

1 Introduction

Business process modelling is an approach to describe how businesses execute their operations [1] by using graphical constructs to specify business logic. The utilization of a standardized notation such as Business Process Model and Notation (BPMN 2.0[1]) typically improves operational efficiency, significantly minimizes errors, and enhances communication and collaboration. One of the primary challenges is the extensive training and skill development required for best-practice utilization of BPMN by various stakeholders within an organization, such as domain experts and process designers/modellers. The successful creation of best-practice models [2] can be facilitated either by extensive collaboration between domain experts and modellers, or by investing in training programs for domain experts, so that they can handle modelling tasks themselves.

While collaborations help to avoid the implementation of special training programs and ensure that BPMN models are well designed [2], they can also lead to a "dilemma between process modeller and domain expert" as there is no or only limited knowledge overlap between them, i.e., there exists a communication gap. The process modeller lacks specific domain knowledge, while the domain expert may have only limited knowledge of process modelling notations [3]. The constant need to transfer the domain knowledge to process modellers is especially burdensome for organizations continuously undergoing adaptations caused by internal or external changes, i.e., when business processes need to be designed or redesigned to improve their day-to-day execution performance [4]. Hence, it is crucial to find a simple and effective way to generate, manipulate, and evaluate process models, minimizing the communication effort of domain experts.

Conversational process modelling (CPM) [5, 6] aims to maximize the involvement of domain experts in the creation of process models and hence to minimize the communication effort between domain experts and process modellers [7]. Specifically, CPM refers to the iterative process as depicted in Fig. 1 of creating process models based on process descriptions and conversations between users and chatbots, until the created models reach a certain quality level and become sufficiently mature to fulfil their purpose. This possibly includes several *process redesign* cycles in which the current process model is changed according to user redesign requests (see tasks with bold lines in Fig. 1) by the LLM. We refer to this as *conversational process redesign (CPD)*.

To the best of our knowledge, no CPD approach has been proposed so far. Hence, this work aims at i) providing a CPD approach and ii) evaluating the quality of the redesigned models created through this approach. The basic idea of the presented CPD approach is to pass a user redesign request in natural language (text) to the LLM and to equip the corresponding prompts with well-established change patterns from the literature [8]. The assumption is that the redesign requests stated by the users involve adaptions of the process model, e.g.,

[1] www.omg.org/spec/BPMN/2.0

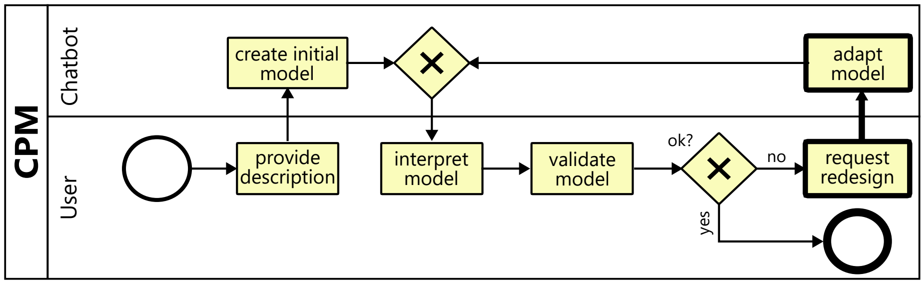

Figure 1: Conversational process modeling including conversational process model redesign

inserting a new task or deleting an existing one, that can be represented based on change patterns. The first study presented in [9] showed that user requests actually involve change patterns. However, the initial assumption of [9], i.e., that the requests mainly refer to basic change patterns such as insert or delete, did not hold. In fact, user requests might also involve more complex change patterns such as embedding tasks into loops. Hence, this work extends the previous work presented in [9] with a systematic and comprehensive evaluation (i) of all 14 change patterns presented in [10], and (ii) of additional patterns that are not part of the original set but emerged as relevant extension to support more complex or diverse process redesign needs. To this end, the change patterns are analysed for usage and representation in a conversational context. Moreover, the CPD approach is formalized and evaluation concepts are provided. The evaluation concepts include the creation of several redesigned process models, i.e., one created by the LLM based on change patterns and user input, and one created manually using change patterns for comparison reasons.

The paper is structured as follows: Section 2 describes the CPD approach and the evaluation concepts. Section 3 puts existing change patterns into a conversational context. Section 4 presents the results of a user study to assess the quality of the LLM-redesigned models regarding user satisfaction, model completeness and correctness, layouting, and the quality of the selected graphical representation. Section 5 discusses related work before Section 6 concludes the paper.

2 Conversational Model Redesign

This section provides the conceptual framework for conversational process model redesign. Section 2.1 provides the overview of the approach, Sect. 2.2 the prompts utilized for redesign, and Sect. 2.3 the concepts for the evaluation of conversational process redesign.

2.1 Conversational Process Redesign: Approach

The conversational redesign problem can be formulated as follows: *Given a process model, a user specifies a redesign request in natural language and the model is adapted based on the request by an LLM.* The conversational redesign approach tackling this problem is depicted in Fig. 2. Its basic idea is to structure the processing of the user request by the LLM, exploiting existing and well-established process change patterns as described in [8].

Change patterns have a formal semantics as defined in [11]. Assume that a process model pm is transformed into process model pm* by applying change pattern cp. Then the formal semantics of pm guarantees that if pm was sound regarding structure and behaviour[2], then pm* is sound.

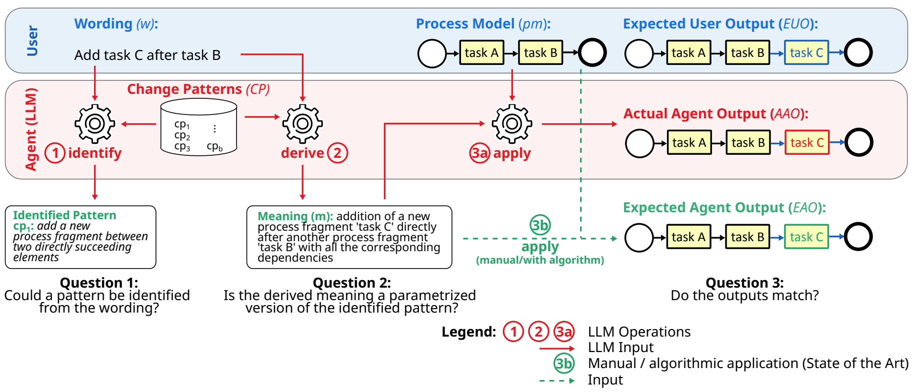

Figure 2: Overview on LLM-based Process Model Redesign.

[10] features 14 patterns, i.e., Insert Process Fragment (cp_1), Delete Process Fragment (cp_2), Move Process Fragment (cp_3), Replace Process Fragment (cp_4), Swap Process Fragments (cp_5), Extract Subprocess (cp_6), Inline Subprocess (cp_7), Embed Process Fragment in Loop (cp_8), Parallelize Process Fragments (cp_9), Embed Process Fragment in Conditional Branch (cp_{10}), Add Control Dependency (cp_{11}), Remove Control Dependency (cp_{12}), Update Condition (cp_{13}), and Copy Process Fragment (cp_{14}).

Since most of the patterns can be realized through the *Insert* and *Delete* patterns, the question arises which of the 14 change patterns are relevant in the context of conversational process redesign and how they are supported in a user-LLM interaction.

We conducted a preliminary survey with 10 users in [9] which tested the support of change patterns cp_1, cp_2, cp_9, and cp_{10}. We opted for these patterns as we assumed them as fundamental for process redesign. We found that none of the 10 users referred to the *Insert* and *Delete* patterns (cp_1 and cp_2), but

[2]Soundness of process models typically requires certain structural properties such as connectedness and behavioural soundness requires reachability of an end state, etc. For details see [12].

frequently referred to patterns cp_8 (*Add Loop*), cp_9, and cp_{13} *(Change Condition)*. Furthermore, users referred to splitting of existing activities into several new activities, which could either be considered a special case of the *Replace* pattern (cp_4) or treated as a stand-alone *Split Process Fragment* pattern.

Based on these results and due to the fact that LLM struggles to identify more complex constructs related to the elements and relationships between these elements, in the following, we will systematically test the 14 change patterns and assess

(a) which user requests in textual form (part of the prompt) can be matched to change patterns and if they cannot be matched, possible interpretations.

(b) different user requests (i.e., do they convey the same meaning, and thus refer to the same change pattern).

(c) operational stability, i.e., if a meaning derived from the user request is applied by an LLM to a certain input model, is the output always structurally and semantically identical?

For tackling (a), we analyse how the change patterns as presented in [10] can be utilized in a conversational setting, i.e., with textual input instead of the original "drag & drop" manner. The results of this analysis are presented in Sect. 3. In order to describe the LLM-based tasks in semiformal way, assume the following definitions. Let $\mathcal{W}$ be the set of all wordings where a wording describes a redesign request by a user in natural language. Let PM be the set of all process models. Note that in the following, we assume an imperatively modelled process model in Business Process Modelling and Notation (BPMN) as the de-facto standard. We plan the investigation of declarative process models in future work. Let further CP be the set of all change patterns as described in [8]. Finally, let $\mathcal{M}$ be the set of meanings, i.e., parametrized change patterns. We will describe the change patterns in more details subsequently.

The user provides a redesign request using *wording* w, e.g., `Add task C after task B`. Wording w is then passed to the LLM which checks whether a corresponding change pattern in the set of change patterns CP exists. This is realized by function ① identify, formally:

$$identify : \mathcal{W} \times 2^{CP} \mapsto CP \cup \{\text{false}\} \tag{1}$$

If the LLM identifies a change pattern cp from CP, it returns cp. In the example, cp_1 for inserting new process fragments between directly succeeding process elements. If no change pattern can be identified, identify returns false.

If a predicted change pattern can be identified, i.e., $identify(w, CP) \neq false$, the LLM ② derives a *meaning* m based on wording w and change pattern cp, formally:

$$derive : CP \times \mathcal{W} \mapsto \mathcal{M} \cup \{\text{false}\} \tag{2}$$

Meaning m can be understood as change pattern cp parametrized with the user request wording w.

The LLM 3a applies the parametrized change pattern m to the *process model pm* to be redesigned, i.e., *pm* is transformed into pm^* which can be assumed to be sound, formally:

$$apply : \mathcal{M} \times \mathcal{PM} \mapsto \mathcal{PM} \tag{3}$$

2.2 Prompt Engineering for Conversationally Actionable Model Redesign

To enable the agent to perform the functions `identify`, `derive`, and `apply`, we rely on prompt engineering (i.e., instructions that guide the behaviour of the LLM), where each function is implemented as a distinct prompt template. Each prompt consists of two parts: system instructions on one side and user input on the other.

The system instructions define the LLM's role, set boundaries for its behaviour, and establish parameters for its operation. These instructions provide a clear framework for the task and ensure that the LLM's responses align with the required level of expertise. User instructions contain the description of the specific task and the parameters provided by the user.

While the system instructions remain consistent within a single prompt, the user instructions vary across single prompts based on the user's request.

Identify. This prompt is necessary for classifying the user request into one of the predefined change patterns for business process model redesign. The list of available change patterns, which is part of the system input, and the wording provided by the user are components of the user input.

Identify

System Prompt: Consider following predefined change patterns for business process model redesign: `<List of Existing Change Patterns>`.
Classify the user input into one of the predefined change patterns, if a matching pattern exists. If a match is found, return only the pattern ID. Only one pattern can be matched. If no match is found, return NA. No other information is allowed to be returned!!!
User Prompt: `<Wording provided by a user>`.

If the pattern is identified (i.e., the wording corresponds to one and only one existing change pattern), the agent returns only the pattern ID (e.g., "cp1"). If no match is found (e.g., multiple patterns match, the wording is unclear, misleading, etc.), return "NA".

Derive. The main task of this prompt is to derive the meaning by parametrizing the change pattern with the user request and ensuring that it fits the BPMN modelling rules. The list of available change patterns, which is part of the system input, and the wording provided by the user are components of the user input.

Derive

System Prompt: You are an expert in BPMN (Business Process Model and Notation) modeling. Your task is to evaluate and interpret user-provided modifications to a BPMN process model. Your responsibilities include:
(a) Validating whether the provided modification contains sufficient and unambiguous information to be applied.
(b) Mapping each modification to a predefined change pattern (see list below).
(c) Deriving the actual meaning of the modification based on BPMN semantics and the intent of the change pattern.
(d) Ensuring compliance with BPMN modelling rules and the structure of the existing process.
Final output have to contain only the actual meaning of the user input in natural language without ambiguity and without any additional information. If there is not enough information to perform changes, the agent returns "NA".
Use the following classification of change patterns to interpret user modifications: `<List of Existing Change Patterns>`.
User Prompt: `<Wording provided by a user>`.

If it is possible to derive the meaning, the agent returns the meaning in the form of text expressed in natural language. If, for some reasons, it is not possible to derive the meaning, the agent returns "NA".

Apply. The aim of this prompt is to apply the changes to the input process model pm based on the provided meaning m, while adhering to the syntax rules of the desired output format. The syntax rules of the desired output format are the part of the system input, and the input process model along with the meaning are components of the user input.

Apply

System Prompt: You are an expert in BPMN modelling, specifically in `<Output Format>` format. Your task is to validate and transform BPMN models based on user-provided modifications, ensuring compliance with BPMN rules and `<Output Format>` syntax. The `<Output Format>` syntax for BPMN models is described as follows:
`<Rules for the Process Model in Output Format >`.
Return only mermaid.js as text without any additional information!
User Prompt: Consider following process model: `<Input Process Model>`. Apply these changes to the model: `<Meaning>`.

2.3 Evaluation Concepts

The evaluation is two-staged. In the first phase, we evaluate the outcomes of functions ① identify and ② derive as this provides valuable insights into user behaviour and wording. It helps us understand where users fail to provide meaningful wording, why patterns are sometimes identified but still fail to provide meaning, which aspects of the user request might need clearer expression, and whether the existing change patterns are sufficient to cover user behaviour. In the second stage, we evaluate the output of the approach, i.e., the redesigned process model. To this end, we create three versions of the redesigned process model (cf. Fig. 2), i.e., the Expected User Output (EUO), the Actual Agent Output (AAO), and the Expected Agent Output (EAO). EUO is the process model the user intends to obtain, EAO is the new process model representing correct agent behaviour when applying a change pattern (i.e., after executing function 3b manually or with the algorithm) to a given process model, and AAO is the process model created by the agent, i.e., after executing function 3a apply.

Stage 1: At first, we assess the cases in which function ① yields $false$ (i.e., no change pattern can be identified based on the user wording, and which cases a change pattern is found) and function ② returns $false$ (i.e., change pattern was identified based on the user wording, but it is not possible to derive meaning out of it). Doing so, we analyse whether the change patterns are properly understood by the agent, potentially identify new patterns in user behaviour that have not previously been defined or considered, and detect discrepancies or errors. Table 2.3 summarizes the possible interpretations for $false$ results in functions identify and derive.

Identify	Derive	Interpretation
false	–	**Pattern is not identified:** **(a) No match is found.** The system could not map the wording to any existing rule. Potentially, a new change pattern can be identified. **(b) User input is incomplete.** The wording provided by the user lacks sufficient information to determine the intended change. The system should prompt the user for additional details. **(c) Multiple matches exist.** The request matches multiple patterns, leading to ambiguity. The system should prompt the user to clarify the request.
true	false	**User input is incomplete.** The wording provided by the user lacks sufficient information to determine the intended change (i.e., location, elements, labels, etc.). The system should prompt the user for additional details.

Table 1: Stage 1: Assessment of Functions identify and derive

Stage 2 assumes that functions identify and derive do not result in false, but yield a change pattern cp and a meaning m respectively, resulting in AAO after executing function 3a apply on process model pm. Comparing AAO to EUO enables the assessment of how user expectations are met and comparing AAO to EAO assesses the effectiveness of LLM-based redesign, i.e., the agent performance and the effectiveness of the prompt design. It helps identify patterns that may fail due to incorrect agent interpretation or limitations in the existing pattern descriptions. This evaluation also highlights whether all existing patterns are necessary and which patterns can be considered alternatives. Table 2.3 summarizes interpretations of comparing the redesigned process models AAO, EUO, and EAO.

AAO == EUO	**AAO == EAO**	**Interpretation**
true	true	**Correct behaviour.** The expected user output matches the actual system output, and the system behaved as expected.
true	false	**Incorrect pattern implementation.** The system did not behave as expected even though the user's request matched an existing pattern and user's expectations.
false	true	**Incorrect pattern application or identification.** The user misunderstood how the pattern works and need guidance on proper application or the system mapped wording to a wrong pattern, leading to incorrect changes.
false	false	**Critical inconsistency.** The applied pattern produced results that neither match the user's expectations nor align with the expected system behaviour suggesting a fundamental issue with pattern identification, execution or user input.

Table 2: Assessing LLM Performance and Effectiveness

3 Conversational Representation of Change Patterns

In this section, we elaborate on how to utilise change patterns in a conversational setting. The reason is that the change patterns presented in [10] were designed with the assumption (or expectation) that the user interacts with the modelling environment through a graphical user interface. In our case, users interact with the LLM in the CPD approach using as text rather than relying on the typical drag-and-drop behaviour. Communicating with the LLM agent through a conversational user interface (CUI) might provide a more natural and engaging user experience, and as a result, the users are less restricted in their functionality [13]. This can be seen as an advantage or a disadvantage, as user requests can express the same patterns in multiple ways or even exceed the system's intended scope [14]. Consequently, it is possible that the existing low-level primitives and high-level patterns are insufficient or not required for conversational interaction with the LLM and for navigating through the process model.

Therefore, in this section, we introduce and summarize additional patterns, that can serve as extensions to the existing change patterns proposed in [10], based on both existing literature (see Sect. 5) and personal experience gained through interaction with an LLM-agent via the CUI.

As mentioned in Section 5, change patterns are a combination of simple actions on individual model elements, leading to process model modifications. These simple actions (i.e., often called *low-level primitives*) each refer only to a single process model element at a time and have no structural assumptions about the model [10]. Typical low-level primitives are: add node, delete node, add edge, remove edge, and move edge [15]. The problem with these primitives is that when performing more complex changes like, for instance, adding a new task into a process, we are required to add not only a node (i.e., task), but also all the corresponding edges that connect this new element with the existing process model [8].

Since a simple operation like "add task" consists of at least three primitives, like adding a node and adding two edges, operating with low-level primitives quickly accelerates in complexity when applying more complicated changes, and can lead to multiple potential errors even for users who have sufficient modelling skills. Thus, to support users in model redesign, more complex high-level patterns are used. Unlike primitives, high-level change patterns require an understanding of the model's structure and its modelling rules.

Considering existing **low-level primitives** and assuming that the model is generated by the agent, it may be necessary to change the label of an element so that it better aligns with user needs. Therefore, we state that it is necessary to introduce an additional low-level primitive—**"rename node"**. Renaming a node is considered a simple low-level primitive because it does not manipulate the structure or flow of the process model but directly modifies an individual element at the semantic level. Additionally, this primitive cannot be replaced by any other. While one might argue that renaming a node could be achieved by combining the "remove node" and "add node" primitives, this approach results in the substitution of one element with another. In contrast, applying "rename node" ensures that the element remains the same, with no changes to any of its other properties (if such exist), except for its label.

Analysing existing **high-level patterns** firstly we exclude patterns that are not supported within a BPMN 2.0 standard. Both patterns cp_{11} and cp_{12} (Add and Delete Control Dependencies). Secondly, we propose splitting pattern $\mathbf{cp}_8$ (Embed Process Fragment in Loop) into two separate patterns: $\mathbf{cp}_{8.1}$ for Pre-conditional Loop and $\mathbf{cp}_{8.2}$ for Post-conditional Loop since their behaviour have to be expressed in natural language in a different way. Although both patterns express the same intent – to execute a particular process fragment multiple times based on a condition – the timing of the condition check results in different behaviour within the model. In $\mathbf{cp}_{8.2}$, a task must be executed at least once, whereas in $\mathbf{cp}_{8.1}$, a task can either be executed multiple times (similar to $\mathrm{cp}_{8.2}$) or not at all. This variation in behaviour within the same pattern leads to different instructions in wording and meaning and, consequently, differences in output Model. Therefore, the division into two distinct patterns is required.

In addition, we define 5 patterns that are considered to be required during CPD: Split Process Fragment (cp_{15}), Merge Process Fragment (cp_{16}), Delete Entire Branch (cp_{17}), Leave Single Branch (cp_{18}), and Replace Gateways (cp_{19}).

$\mathbf{cp}_{15}$ (Split Process Fragment) allows splitting an existing process fragment into multiple separate process fragments. It is an efficient way to separate tasks, as it also adjusts the control flow between the split tasks. This pattern is useful when multiple tasks that should be performed sequentially are currently combined into a single task. However, these tasks do not have enough complexity or structure to justify forming a separate subprocess.

This pattern affects the granularity of the process model. It differs from cp_7 (Inline Subprocess), which expands an already structured subprocess, making its tasks visible in the main process. In contrast, cp_{15} increases granularity but does not necessarily create a subprocess.

For example, receiving a document and reviewing it for approval might be combined into a single task, "Receive and Review Document". However, since these two activities are sequential and need to be tracked separately, cp_{15} would split them into two distinct tasks: "Receive Document" and "Review Document".

$\mathbf{cp}_{16}$ (Merge Process Fragment) serves for merging multiple existing separate process fragments into one process fragment. It is efficient for merging tasks in a single task, as it also adjusts the control flow. This pattern is useful when activities that are represented separately and considered to be independent are actually a single activity or is a single activity from stakeholder perspective.

This pattern is fundamentally different from cp_6, as cp_{16} combines multiple independent process fragments into a single process fragment within the process. cp_6, on the other hand, takes a set of related process fragments and moves them into a separate subprocess, creating a structured, reusable component.

For example, "Generate Invoice", "Verify Invoice Details" and "Send Invoice" from the perspective of an IT system are three separated tasks, but for managers, simple users, etc. generating, verifying, and sending an invoice happens in one task (i.e., the invoice is created, verified and sent immediately without any manual intervention). In this case, cp_{16} would be used to merge these three activities into "Process Invoice" to better reflect the real-world process.

$\mathbf{cp}_{17}$ (Delete Entire Branch) removes an entire branch inside gateways with all associated tasks, control edges, and dependencies. The pattern adjusts conditions and flattens the hierarchy by removing gateways if only one branch remains. This pattern improves process refinement and clean-up by reducing errors that might occur if multiple elements had to be deleted one by one. It is particularly useful in GUIs where users cannot easily select multiple elements simultaneously using drag and drop.

$\mathbf{cp}_{18}$ (Leave Single Branch) removes multiple branches inside gateways with all associated tasks, control edges, and dependencies, leaving only a single branch. The pattern adjusts conditions and flattens the hierarchy by removing gateways as only one branch remains. This pattern improves process refinement and clean-up by reducing errors that might occur if multiple branches had to be deleted one by one. It is particularly useful in GUIs where users cannot easily select multiple elements simultaneously using drag and drop.

$\mathbf{cp}_{19}$ (Replace Gateways) replaces both splitting and merging components of a gateway simultaneously. This pattern is useful when the control flow behaviour changes (e.g., tasks that were previously executed in parallel are now executed sequentially). Since gateways typically consist of both splitting and merging components, this pattern allows for both parts to be changed at once, rather than modifying them separately to avoid potential inconsistencies.

The list of all existing and potentially required patterns that will be considered further in the paper can be found in Table 3. These patterns are primarily based on existing literature, practical experience, and assumptions, serving as a solid starting point for understanding user needs in chatbot-related scenarios. Next, we systematically test and evaluate these patterns with real users to assess their effectiveness and relevance in practice.

Table 3: Overview of the CAPMoRe Change Patterns CP (where green - existing Primitives and Patterns, yellow - proposed extension, blue - not considered)

ID	Name	ID	Name
Low-level Primitives			
LP1	Insert Node	LP2	Delete Node
LP3	Insert Edge	LP4	Delete Edge
LP5	Move Edge	LP6	Rename Node
High-level Change Patterns			
cp_1	Insert Process Fragment	cp_2	Delete Process Fragment
cp_3	Move Process Fragment	cp_{14}	Copy Process Fragment
cp_4	Replace Process Fragment	cp_5	Swap Process Fragments
$cp_{8.1}$	Embed Process Fragment in Pre-Cond. Loop	$cp_{8.2}$	Embed Process Fragment in Post-Cond. Loop
cp_9	Parallelize Process Fragments	cp_{10}	Embed Process Fragment in Cond. Branch
cp_{15}	Split Process Fragment	cp_{16}	Merge Process Fragment
cp_{11}	Add Control Dependency	cp_{12}	Remove Control Dependency
cp_{13}	Update Condition	cp_{19}	Replace Gateways
cp_6	Extract Sub Process	cp_7	Inline Sub Process
cp_{17}	Delete Entire Branch	cp_{18}	Leave Single Branch

4 Evaluation

The evaluation concepts are presented in Sect. 2.3 and the conversational representation of change patterns in Sect. 3. Based on this, we develop the the evaluation procedure in Sect. 4.1 and present the evaluation results in Sect. 4.2.

4.1 Evaluation Procedure

User preferences, expectations, and actual usage behaviour may differ significantly from the assumptions that formed the basis of the proposed patterns (see Table 3). Therefore, it is essential to conduct comprehensive user studies and gather feedback to validate whether these patterns align with user needs and improve the overall user experience. A closer look at how users interact with the system will provide insights into refining these patterns and introducing new ones if necessary. This approach not only helps improve the user experience by enhancing human-computer interaction, but also contributes to formulating a better rule base, making the chatbot more flexible and improving its performance.

To analyse user behaviour, we conducted a user survey with 64 participants[3],[4]. Users were first asked to answer some general questions about their

[3]https://docs.google.com/forms/d/e/1FAIpQLSerHMjzTR4ll3XJN_vWAZ53KWy1blKLr-fv3Wd0_BvUxUyHWA/viewform

[4]Survey Overview: https://github.com/com-pot-93/cpd/blob/main/survey/survey-overview.pdf

modelling skills, after which they were presented with models pairs (pm, pm*). Their task was to imagine that, using a chatbot, they have to make changes to pm to derive pm*, i.e., using natural language to achieve a transformation (see Fig. 3).

Input process models were presented as simple BPMN models with up to five tasks, parallel and exclusive gateways, and subprocesses for certain patterns. The process models were created with dummy tasks to avoid any potential domain biases, especially since some of the participants had no or limited modelling experience. The output models were derived by applying a list of existing and proposed change patterns (see Tab. 3). The list of all input and output process models, as long as the survey results, can be found in[5], [6], [7].

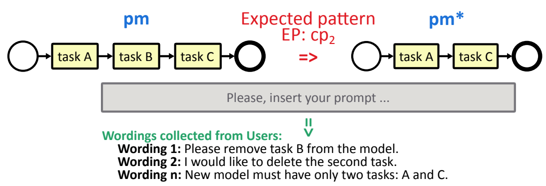

Figure 3: User survey. Example

Thus the setting of the survey is as follows:

- The user is presented with an Input Process Model (pm).
- The user is presented with an Output Process Model (pm*).
- The user is intended to provide a *Wording* that describes how to transform pm into pm*.
- The user had to provide a *Wording* for 18 patterns.
- For each of the 18 transformations an Expected Pattern (EP) was assigned.

The EPs were not communicated to the user. Each user was intended to provide whatever wording he/she deems necessary. The survey itself covered all patterns, so for each pattern 64 different wordings were collected. After collecting the wordings the following steps are then performed:

- For each wording identify a pattern cp_x if possible.
- Compare the pattern cp_x with the expected pattern cp_y.
- Check if m can be derived successfully from cp_x.
- When m is applied to pm, check if the resulting pm' is identical with pm* .

[5] Input Process Models: https://github.com/com-pot-93/cpd/tree/main/input
[6] Output Process Models: https://github.com/com-pot-93/cpd/tree/main/EAO
[7] Survey Data: https://github.com/com-pot-93/cpd/tree/main/survey

The goal is to find out which patterns are useful, which patterns can be replaced, reduced or combined, and if some additional patterns have to be added. For this purpose we follow the procedure, described in Section 2.3. For each of the steps 1a, 1b, 2, 3a yields `True` or `False`, and map obtained results with the Tables 2.3 and 2.3.

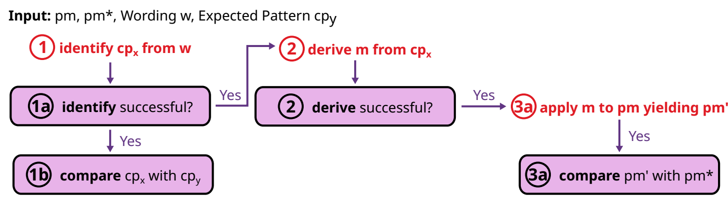

Figure 4: Evaluation Procedure

Figure 4 presents the CAPMoRe-agent stages and illustrates how we interpret the output of these stages in order to establish how efficiently the selected LLMs can map the provided wording to the list of existing patterns, whether the mapped patterns are consistent with expectations, whether the provided wording is sufficient for deriving meaning, and whether a new pattern is required.

1a The status of the identified pattern cp_x is set to `True` if it is matched to one of the patterns in CP. If the wording can not not matched to any pattern in the CP, if it matches multiple patterns, or if the agent provides an output different from the expected one, the status is set to `False`.

1b If cp_x could be identified, we check whether cp_x equals the expected pattern. If they are identical, the status is set to `True`; otherwise, the status is set to `False`.

2 If cp_x could be identified, we also proceed to the next step and derive the meaning from the wording provided by user. If the meaning is successfully derived (i.e., there is enough information in the wording to perform a change), the status is set to `True`; otherwise, the status is set to `False`, and no further steps are performed.

3a If the meaning is successfully derived, we apply it to pm* and generate the AAO as pm', which is then compared to the pm* from the survey (which we define to be the EAO).

To compare two process models, we use element-based semantic similarity, first introduced in [16]. Each element of the process model is considered, and for each element, semantically similar elements in the second model are identified using the *dice score*. The overall score is then weighted by the harmonic mean of the lengths of the elements. The similarity value ranges from 0 to 1.

Since the models in the survey are very simple, and even small differences result in different models, we set the threshold to 1. Thus, if the similarity between the AAO and the EAO is equal to 1, the status is set to `True`; otherwise, the status is set to `False`.

Such a strict threshold ensures that the models are not only structurally identical but also equivalent in terms of completeness (i.e., all elements in one process model are included in the other) and correctness (i.e., the tasks not only have matching labels but also appear in the same sequence). In doing so, it also serves as an indicator of user satisfaction with the outcome, as only fully accurate and complete models are considered equivalent.

In the end, we have six possible combinations introduced in Table 2.3 and 2.3 to evaluate how sufficient the selected change patterns are and, at the same time, how efficient the LLMs are in process model redesign.

4.2 Evaluation Results

For the evaluation, we use three different large language models (LLMs): `gpt-4o`, `gemini-1.5-pro`, and `mistral-large-latest` (hereinafter referred to as gpt, gemini and mistral, respectively).

Based on the results obtained from [9], we adopt the strategy of addressing the tasks by their labels, performing only one change at a time, and passing prompt-related information to the LLM (input process model, change to be performed, and rules for the output format). We also use a zero-shot strategy and provide no examples of process descriptions, input, or output process models. The prompts used during the evaluation along with all generated artifacts and non-average data are available in[8][9].

Correct Behaviour. As a starting point, we consider the cases where both the user and the agent perform well, and the changes provided by the user were successfully implemented. We define a pattern as successfully implemented if that reaches at least 30%.

As can be seen in Table 4, only 8 out of 18 patterns were successfully realized in more than 30% of all cases. For the already well-established change patterns (cp_1–cp_{14}, excluding pattern cp_8), only patterns cp_1, cp_4, cp_5, and cp_{13} reach the 30% threshold. For the proposed patterns (cp_{15}–cp_{19}, $cp_{8.1}$, and $cp_{8.2}$), four patterns (i.e., cp_{15}–cp_{17} and cp_{19}) succeeded.

Table 4: Correct Behaviour across Change Patterns

Method	**cp_1**	cp_2	cp_3	**cp_4**	**cp_5**	cp_6	cp_7	$cp_{8.1}$	$cp_{8.2}$
gemini	0.72	0.03	0.08	0.64	0.84	0.09	0.08	0.19	0.02
gpt	0.64	0.45	0.45	0.66	0.84	0.11	0.03	0.03	0.11
mistral	0.52	0.03	0.05	0.48	0.31	0.02	0	0	0
average	**0.63**	0.17	0.19	**0.59**	**0.67**	0.07	0.04	0.07	0.04

Method	cp_9	cp_{10}	**cp_{13}**	cp_{14}	**cp_{15}**	**cp_{16}**	**cp_{17}**	cp_{18}	**cp_{19}**
gemini	0.13	0.03	0.91	0.11	0.39	0.61	0.58	0.02	0.53
gpt	0.06	0.03	0.86	0.08	0.53	0.59	0.61	0.08	0.56
mistral	0.05	0	0.31	0.03	0.16	0.06	0.03	0	0.31
average	0.08	0.02	**0.69**	0.07	**0.36**	**0.42**	**0.41**	0.03	**0.47**

To understand why so many patterns fall below this threshold, we examine more closely the cases where either identification was not possible, or it was not possible to derive a meaning from the provided wording (Stage 1, see Sect. 2.3),

[8]Prompts: https://github.com/com-pot-93/cpd/tree/main/prompts
[9]Generated Data: https://github.com/com-pot-93/cpd/tree/main

or where the wording and the meaning were considered sufficient by the agent but the user applied the wrong pattern, or where the agent implemented the pattern incorrectly (Stage 2, see Sect. 2.3).

Pattern is not identified. First, we analysed all cases where the identification received a `False` status (i.e., the pattern was not identified or the identified pattern did not match any existing pattern in CP). According to Table 2.3, there are three possible reasons for such behaviour, and only in one of these cases a new pattern can be required.

In Table 5, we can see that, generally, for each of the patterns in CP, between 2% and 20% of the cases had wording that was not matched to existing patterns. In detail, we will consider only the patterns that reach at least 10% of unidentified cases, e.g., patterns **cp_4, cp_6, cp_7, cp_9, cp_{10}, cp_{14}–cp_{16}, and cp_{18}**.

Table 5: Unidentified Change Patterns

Method	cp_1	cp_2	cp_3	**cp_4**	cp_5	**cp_6**	**cp_7**	$cp_{8.1}$	$cp_{8.2}$
gemini	0.05	0.03	0.02	0.25	0.09	0.16	0.11	0.06	0.09
gpt	0.02	0.02	0.02	0.19	0.03	0.06	0.05	0.06	0.05
mistral	0.02	0.02	0.14	0.14	0.05	0.11	0.14	0.06	0.06
average	0.03	0.02	0.06	**0.19**	0.06	**0.11**	**0.1**	0.06	0.07

Method	**cp_9**	**cp_{10}**	cp_{13}	**cp_{14}**	**cp_{15}**	**cp_{16}**	cp_{17}	**cp_{18}**	cp_{19}
gemini	0.13	0.14	0.05	0.17	0.2	0.13	0.05	0.2	0.09
gpt	0.05	0.08	0.05	0.08	0.06	0.08	0.05	0.08	0.09
mistral	0.13	0.08	0.05	0.08	0.11	0.2	0.05	0.14	0.09
average	**0.1**	**0.1**	0.05	**0.11**	**0.13**	**0.14**	0.05	**0.14**	0.09

In most of these cases, the user input (i.e., wording) did not align with any existing pattern, because the user was unable to describe the transformation from the input process model pm to the output process model pm* (e.g.,"I don't know", empty wording, or irrelevant content).

In the case of patterns **cp_6**, **cp_7**, **cp_9**, and **cp_{18}**, the user was not referring to the patterns themselves, but to multiple patterns, such as insert, delete, etc., to realize these patterns. As a result, the agent was not able to match the patterns properly.

For pattern **cp_4**, users often referred to an operation that is not defined as a pattern but as a proposed low-level primitive — "Rename node" (e.g., in the provided example: renaming a label or task name). Its utilization and frequency of occurrence suggest that extending the existing set of low-level primitives to include this action is appropriate.

Regarding pattern **cp_{15}**, users frequently described it not just as splitting a task, but as transforming a single task into a sequence of multiple tasks (e.g., "Make task B and E sequential, with task B followed by task E."). Since this is exactly what cp_{15} represents, it highlights the need to revise the definition of cp_{15} to better align with how LLMs interpret this pattern.

A similar issue exists for pattern **cp_{16}**. In multiple cases, instead of using common terms like merge, combine, or join, users chose to describe the transformation using the term "summarise" (e.g., "Summarize task B and task E into a single task B&E."). While this can be interpreted as cp_{16}, it is not immediately clear, since the initial meaning of summarizing is *"to express the most important facts or ideas about something or someone in a short and clear form"*[10].

Meaning is not derived. Next, we examine the cases in which the pattern was identified but the meaning was not derived. On average, up to 12% of all cases in all patterns failed to derive a meaning (detailed information can be found in [ADD git]). In most cases, the failure was due to incorrect wording, which was insufficient to capture the proper meaning of the user's request.

Incorrect Pattern Application or Identification. In this case, the user selects a pattern different from the expected one but still got the same result as expected. In this situation, we can talk either about pattern alternatives, i.e., where one pattern can be used to realize another, either consistently or in some particular scenario, or about agent pattern misidentification, i.e., when the user's wording is correct, but for some reason, the agent matches the wording to the wrong pattern.

To do so, we further examine which other pattern appears most frequently in those misidentification cases, i.e., we look for the predominant alternative pattern. This helps us develop an intuition about how users interpret and understand patterns, and identify which patterns tend to co-occur, serve as alternatives, or are conceptually related to the expected ones.

Table 6: Incorrect Pattern Application or Identification across Change Patterns

Method	$\mathbf{cp_1}$	$\mathbf{cp_2}$	cp_3	cp_4	cp_5	$\mathbf{cp_6}$	$\mathbf{cp_7}$	$cp_{8.1}$	$cp_{8.2}$
gemini	0.09	0.23	0.08	0.09	0.03	0.16	0.22	0.02	0.02
gpt	0.19	0.28	0.05	0.08	0.02	0.2	0.2	0.06	0.05
mistral	0.08	0.05	0	0.02	0.08	0	0.03	0.02	0
average	**0.12**	**0.19**	0.04	0.06	0.04	**0.12**	**0.15**	0.03	0.02
predominant pattern	cp_{10}	cp_{17}	-	-	-	cp_4	cp_4, cp_{10}	-	-

Method	cp_9	cp_{10}	cp_{13}	$\mathbf{cp_{14}}$	cp_{15}	cp_{16}	cp_{17}	$\mathbf{cp_{18}}$	$\mathbf{cp_{19}}$
gemini	0.09	0	0.05	0.61	0.06	0.11	0.06	0.27	0.19
gpt	0.03	0	0.05	0.77	0.16	0.03	0.05	0.23	0.13
mistral	0.02	0	0.05	0.58	0.05	0.02	0.02	0.05	0.11
average	0.05	0	0.05	**0.65**	0.09	0.05	0.04	**0.18**	**0.14**
predominant pattern	-	-	-	cp_1	-	-	-	cp_{17}	cp_9, cp_{13}

In Table 6, we present a summary of the results for the cases where the wording was mismapped to the wrong pattern, as well as the pattern to which it was mismapped (i.e., see the predominant pattern in Table 6) in cases where, in more than 10% of instances, all three LLMs consistently predict one or two alternate patterns that differ from the expected one. In other cases, further examination is not conducted, as we consider such pattern variation to be random and not significant.

For each of these predominant patterns, we assess whether the issue arises from user wording (i.e., misunderstanding of the pattern) or incorrect reasoning by the LLM (see Tab. 7).

[10] https://dictionary.cambridge.org/dictionary/english/summarizing

Wrong user wording **(User)** means that, in this case, the application of the provided wording cannot result in the expected output process model or violates BPMN 2.0 rules. Wrong LLM reasoning **(LLM)** means that the user's wording was correct; however, the LLM's understanding of it is inconsistent with the true pattern definition and thus needs to be adjusted.

In all cases, we provide an example of user wording in Tables 7 and 8. Additionally, when the likely cause of the mismatch is an LLM error or pattern ambiguity, we offer suggestions for why the LLM might fail to match the expected pattern or where clarifying the pattern definitions may help guide both LLM reasoning and user interpretation more effectively.

Incorrect Pattern Implementation. Here, we examine how often the agent fails to perform the changes correctly, even though the wording was complete, the correct pattern was identified, and the meaning was extracted.

As can be seen, there are many patterns that exceeded the 10% threshold, reaching failure rates as high as 62%. This indicates that our agent, first, can't easily handle more complicated patterns; second, suffers from issues with meaning extraction from the wording; and third, that the high threshold we set for model comparison may be too severe for some cases, which should be examined in more detail.

A detailed examination of these cases can help to determine whether adjustments to the threshold, the prompt itself, or the logic and sequence of the steps (e.g., the analysis of user input, the extraction of meaning from that input, and its application to adjust the initial process model) are necessary.

Critical Inconsistency. Here, we examine how often the agent fails to perform the changes correctly, even though the wording was complete and the meaning was derived. However, in comparison with the **Incorrect Pattern Implementation**, we cannot be sure who or what is responsible for the failure: the user or the LLM.

Looking at Table 10, we can see that many patterns failed, reaching up to 70% failure. The analysis of patterns $\mathbf{cp}_2$**,** $\mathbf{cp}_6$**,** $\mathbf{cp}_7$**,** $\mathbf{cp}_{18}$**, and** $\mathbf{cp}_{19}$ can be found in **Incorrect Pattern Application or Identification** (see Tab. 7).

For the patterns $\mathbf{cp}_3$**,** $\mathbf{cp}_{8.1}$**,** $\mathbf{cp}_{8.2}$**,** $\mathbf{cp}_9$**,** $\mathbf{cp}_{10}$**,** $\mathbf{cp}_{16}$**, and** $\mathbf{cp}_{17}$ we perform another round of evaluation (see Tabs. 11 and 12).

Summary: As can be seen above, there are multiple reasons why conversational model redesign failed. The reasons for that are not only related to the agent and its pattern understanding, but also to the users themselves, their understanding of the patterns, and their application in various scenarios.

Despite the fact that in most cases, either the user or the LLM is fault when a pattern fails, in some cases it is not straightforward to define whose fault it is. In such cases, we are dealing with ambiguity in the pattern itself, and clarification of the pattern is necessary.

Pattern ambiguity means that the wording can be interpreted as multiple patterns, as the definition of the pattern provided to the LLM is not completely clear and can be interpreted in multiple ways. As a result, the identified pattern may differ from the expected one but cannot be considered a truly wrong classification, since this information was not provided to the LLM from the be-

Table 7: Incorrect Pattern Application or Identification: Detailed Analysis – 1

Pattern	Misclassified As	Interchangeable?	Cause	Example and Explanation
cp_1	cp_{10}	No	LLM	*"Add the task E after task C on the false branch."* The misclassification seems to appear by the user specifying a particular location within a conditional structure, such as referencing a branch with a condition. This reference introduces contextual information that may lead the model to overemphasize the structural aspect of the condition, even though the user's intent is simply to insert a task.
cp_2	cp_{17}	No	LLM	*"Remove task C such that no task is executed in the false-branch."* This result is highly dependent with the selected example, where the user's intent is to delete a specific task, which would result in the false branch being empty.
cp_6	cp_4	No	LLM, User	*"After 'task A' and before the 'end place', replace all elements with a transition 'subprocess F'."* The distinction between "extracting a subprocess" and "replacing elements with one" is subtle and often blurred. This suggests that clearer definitions and separation between cp_4 and cp_6 are necessary to avoid such ambiguity in both user understanding and model classification.

Table 8: Incorrect Pattern Application or Identification: Detailed Analysis – 2

Pattern	Misclassified As	Interchangeable?	Cause	Example and Explanation
cp_7	cp_{10}	No	LLM	*"Expand subprocess F and execute B, if task A returns true, or C if it returns false. After both cases, execute D."* The misclassification is caused by the LLM overemphasising the conditional logic in the prompt, incorrectly interpreting a clear subprocess inlining as a conditional insertion.
cp_{14}	cp_1	Yes	User	"Insert task A' before task D."
cp_{18}	cp_{17}	Yes	User	*"Delete the branches with condition b and with condition c."*
cp_{19}	cp_9	Partially	LLM, User	*"Replace OR with AND", "Instead of XOR condition, make the tasks be performed in parallel (AND condition)."* The misclassification results as changing an XOR to an AND gateway can be interpreted as either modifying decision logic or enabling parallel execution.
cp_{19}	cp_{13}	No	User	*"Make no more conditions for the task B and C."*

Table 9: Incorrect Pattern Implementation across Change Patterns

Method	cp_1	$\mathbf{cp_2}$	$\mathbf{cp_3}$	cp_4	cp_5	$\mathbf{cp_6}$	$\mathbf{cp_7}$	$cp_{8.1}$	$\mathbf{cp_{8.2}}$
gemini	0.08	0.39	0.63	0.02	0	0.44	0.36	0	0.52
gpt	0.02	0	0.23	0.02	0.03	0.36	0.19	0.2	0.41
mistral	0.19	0.27	0.5	0	0.14	0.47	0.2	0.06	0.45
average	0.09	**0.22**	**0.45**	0.01	0.06	**0.42**	**0.25**	0.09	**0.46**

Method	$\mathbf{cp_9}$	$\mathbf{cp_{10}}$	cp_{13}	cp_{14}	$\mathbf{cp_{15}}$	$\mathbf{cp_{16}}$	$\mathbf{cp_{17}}$	$\mathbf{cp_{18}}$	$\mathbf{cp_{19}}$
gemini	0.25	0.64	0	0.02	0.23	0.06	0.13	0.16	0.11
gpt	0.38	0.66	0	0	0.17	0.05	0.03	0.11	0.08
mistral	0.3	0.56	0.36	0.08	0.39	0.25	0.31	0.05	0.22
average	**0.31**	**0.62**	0.12	0.03	**0.27**	**0.12**	**0.16**	**0.1**	**0.14**

Table 10: Critical Inconsistency across Change Patterns

Method	cp_1	cp_2	cp_3	cp_4	cp_5	cp_6	cp_7	$cp_{8.1}$	$cp_{8.2}$
gemini	0.06	0.31	0.2	0	0.03	0.16	0.23	0.73	0.36
gpt	0.08	0.09	0.17	0.02	0.03	0.11	0.33	0.61	0.34
mistral	0.06	0.22	0.13	0	0.02	0.14	0.44	0.75	0.31
average	0.07	0.21	0.17	0.01	0.03	0.14	0.33	0.7	0.34
predominant pattern	-	cp_{17}	cp_{10}	-	-	cp_4	cp_4, cp_{10}	$cp_{8.2}$, cp_{10}	cp_{10}

Method	cp_9	cp_{10}	cp_{13}	cp_{14}	cp_{15}	cp_{16}	cp_{17}	cp_{18}	cp_{19}
gemini	0.41	0.19	0	0.09	0.11	0.09	0.19	0.36	0.08
gpt	0.22	0.13	0.02	0.06	0.02	0.09	0.08	0.31	0.06
mistral	0.34	0.13	0.02	0.13	0.11	0.2	0.14	0.5	0.2
average	0.32	0.15	0.01	0.09	0.08	0.13	0.14	0.39	0.11
predominant pattern	cp_{10}	cp_{13}, cp_{19}	-	-	-	cp_9	cp_{10}	cp_{17}	cp_9, cp_{13}

ginning. In some cases, the change can be realized through multiple patterns, but this is not a consistent setup; rather, it is a coincidence based on the current example or settings.

To summarise the evaluation, we categorise all the cases discussed above based on the most frequent reason for failure (i.e., No Failure ("Correct Behaviour"), User ("User input is incomplete" and "Incorrect pattern application or identification"), LLM ("Incorrect pattern implementation"), or Pattern Ambiguity ("Pattern is not identified"and "Critical Inconsistency")).

As shown in the evaluation, only eight patterns performed correctly in most cases (see Tab. 13). Since four of these patterns were proposed by us and did not exist previously, we can conclude that these patterns are valid candidates to be considered as change patterns for conversational model redesign.

The remaining patterns failed due to issues related to the user, the LLM, or pattern ambiguity (see Tab. 13). However, we cannot immediately suggest the exclusion of these patterns from the CP set based on some thresholds. Each pattern requires further examination to determine whether it still appears to be necessary and, depending on the reason for failure, may need to be clarified either on the agent or the user side.

For instance, in the case of cp_{14}, the misclassification appears to be caused by the user wording, which referred to cp_1 instead of cp_{14}. Since the suggested alternative, cp_1, leads to the same outcome as the expected pattern cp_{14}, cp_1

Table 11: Critical Inconsistency: Detailed Analysis – 1

Pattern	Misclassified As	Interchangeable?	Cause	Example and Explanation
cp_3	cp_{10}	No	LLM	*"Move task C to be done in both cases if status after task a is true or false, it should still be done before task D."* The LLM associates "move" with something more complex like embedding in a conditional branch due to the presence of conditional references in the wording.
$cp_{8.1}$	cp_{10}	No	User	*"The task D could only be conducted if the last condition is false."*
$cp_{8.1}$	$cp_{8.2}$	No	LLM	*"Before Task D, add a conditional Split. If true then end, if false then Task D and loop back to the conditional Split."* In most cases the user intent is to conditionally enter a loop where particular task is executed only if a specific condition is met (clearly describing a pre-condition loop structure). The misclassification likely results from the complexity of the user request, where the looping behaviour is described in multiple partial sentences, making it difficult to accurately identify the position of the condition evaluation.
$cp_{8.2}$	cp_{10}	No	User	*"Do task D only when condition is true."*
cp_9	cp_{10}	No	User	"Create task D and create a new branch after task A, so that task D can be executed instead of task B and C."

Table 12: Critical Inconsistency: Detailed Analysis – 2

Pattern	Misclassified As	Interchangeable?	Cause	Example and Explanation
cp_{10}	cp_{13}	No	LLM , User	*"If a = 1, the XOR fragment of task B and C should be skipped."* The user did not clearly express the intent to insert a new conditional branch. The LLM interpreted the prompt as a condition update rather than a new conditional structure, due to the absence of explicit wording and lack of LLM's familiarity with the process context.
cp_{10}	cp_{19}	No	LLM, User	*"Please add XOR in front and after the decision option of false and true."* The distinction between "adding" and "modifying" decision logic likely needs to be better defined in the pattern descriptions or clarified through examples.
cp_{16}	cp_{9}	No	User	*"In the new model, the system executes tasks B and E simultaneously."*
cp_{17}	cp_{10}	No	User	*"Add condition c to the xor with task E followed by task F in the branch"*

Table 13: Average Distribution of Reasons for Change Pattern Failures

Reason	cp_1	cp_2	cp_3	cp_4	cp_5	cp_6	cp_7	$cp_{8.1}$	$cp_{8.2}$
Pattern Ambiguity	0.1	0.23	0.23	0.2	0.09	0.25	**0.43**	**0.76**	0.41
User	0.19	**0.38**	0.13	0.2	0.19	0.26	0.28	0.08	0.09
LLM	0.09	0.22	**0.45**	0.01	0.06	**0.42**	0.25	0.09	**0.46**
No Failure	**0.63**	0.17	0.19	**0.59**	**0.67**	0.07	0.04	0.07	0.04

Reason	cp_9	cp_{10}	cp_{13}	cp_{14}	cp_{15}	cp_{16}	cp_{17}	cp_{18}	cp_{19}
Pattern Ambiguity	**0.42**	0.25	0.06	0.2	0.21	0.27	0.19	**0.53**	0.2
User	0.2	0.11	0.13	**0.69**	0.17	0.19	0.25	0.33	0.19
LLM	0.31	**0.62**	0.12	0.03	0.27	0.12	0.16	0.1	0.14
No Failure	0.08	0.02	**0.69**	0.07	**0.36**	**0.42**	**0.41**	0.03	**0.47**

can be considered a valid substitute. A similar situation applies to cp_{18}, which was effectively realized through cp_{17}.

In the case of cp_{18}, the same result can be indeed achieved by applying cp_{17}; thus, these patterns are completely interchangeable. However, in the case of pattern cp_{14}, there is a significant difference between copying and inserting a new process fragment, even though at first glance these patterns may appear to produce the same result. When copying, the process fragment remains unchanged, preserving all of its original properties. In contrast, inserting creates a new process fragment from scratch, without inheriting any properties or characteristics from the original.

Comparing performance across LLMs, we achieve better results using gemini and gpt, and observe more cases where the agent demonstrates correct behaviour. Additionally, gemini and gpt adhere more closely to the instructions and return output that is consistent with the provided guidelines. Mistral, on the other hand, tends to return output (e.g., multiple identified patterns, comments, and explanations) that was not requested, making evaluation more difficult.

However, despite the fact that in some cases mistral performs worse compared to gemini and gpt, the average case distribution in the evaluation remains similar across all three LLMs. This means that, by applying the selected procedure and prompts, we are still able to obtain consistent results, the comprehension capabilities of the LLMs are similar, and our findings can be considered valid.

4.3 Discussion

A high percentage of patterns that failed due to incorrect or incomplete user wording highlights the necessity of a recommender system implemented by the agent to support the user in clarification wording and achieve better results.

A high number of patterns that failed due to the LLM suggests that we need to improve the utilised prompts, clarify or even formalise the pattern descriptions, or modify the architecture of the current pipeline. This could involve combining LLM capabilities with traditional deterministic approaches to improve model redesign performance. For instance, the LLM could be used to identify the pattern and extract relevant parameters from the wording, while the pattern application itself could be carried out using deterministic methods.

The presence of pattern ambiguity in some cases indicates that pattern clarification is required not only for the agent but also for the user, in order to minimise misunderstandings and incorrect pattern application.

Interestingly, when considering user wording in general regardless of the specific pattern, there are two common user tendencies that occurred. First, when the changes required for the models were too complex or lengthy to describe, users tended to create a new model from scratch (e.g., "Delete everything and create a new process with task A, then task B"). Second, in some cases, users employed the concept of reverting changes, referring back to a previous model state (e.g., "Reverse the action from the previous question/step"). These two behaviours could also be considered as patterns, not in terms of change patterns for model refinement, but rather as patterns in user behaviour that have to be supported by the agent implementation.

4.4 Threats to validity

This study presents several limitations that may affect the generalizability and validity of the findings.

The process models used in the evaluation are small and simple, consisting of generic task labels lacking domain-specific semantics (e.g., Task A, Task B, etc.). While this design choice ensures clarity and consistency across all participants and LLMs, it limits the applicability of the results to more complex scenarios as our examples may not fully represent the challenges users face when working with larger or more complex process models.

Participants were not engaged in an interactive setting with the conversational agent. Instead, they were asked to provide wording based on a static input-output model pair. Consequently, user behaviour in this study may differ from that observed in real deployment contexts.

Users were required to express their intent using a single prompt per model transformation. This constraint does not reflect natural user-agent interaction patterns, where clarification and follow-up are common.

In addition, the output models used in the evaluation were constructed based on a predefined set of change patterns. While this approach allowed for better evaluation and comparison, it may not fully reflect the variety of real-world user requests. In real-world scenarios, users may express additional or more complex transformation goals that were not anticipated or covered by the selected patterns.

LLM behaviour can vary based on version, update timing, and underlying training data. Future replications may observe different results due to changes in LLM behaviour.

5 Related Work

Business process modelling requires the accurate representation of intricate workflows, decision points, and interactions between multiple stakeholders within an organization. This complexity is further compounded by the need for effective communication between domain experts, who possess the contextual knowledge, and process modellers, who translate that knowledge into formal models, as well as the need to update and redesign the model in the future.

Several studies address the communication gap between domain experts and process modellers, e.g., [17, 18, 19, 3, 20]. With recent advancements in Natural Language Processing (NLP) and Generative AI, which are transforming transforming classical BPM systems into AI-augmented Business Process Management systems [21], the use of natural language and chatbots can be a realistic scenario to overcome the communication gap.

These systems become conversationally actionable, meaning they can proactively communicate with human agents about process-related actions, goals, and intentions using natural language [22]. This interaction can be enhanced via the integration of intelligent chatbot functions for improved communication within the BPM framework, promoting collaboration [21]. The systems can lead conversations in a multi-turn nature, considering context and incorporating utterances from previous turns to achieve a higher degree of user engagement [23]. Currently, as mentioned in [24, 25, 26, 27, 28], there is an increasing interest in the potential benefits for the entire BPM domain arising from employing LLMs, particularly in process model generation. However, most existing approaches focus solely on single-time interactions, where the user is able to receive a final artifact from the system, but is not able to adjust and redesign it. So far, the multi-turn conversational capabilities of LLMs for process modelling have received little attention and have not yet been thoroughly explored in the Business Process Management domain.

A process model can be redesigned by rearranging various elements to satisfy predefined business rules and constraints. The primary goal of process redesign is process improvement, which can be categorized into two levels: (a) Functional goals, such as ensuring desired or acceptable process behavior, and (b) Non-functional goals, including cost reduction, time optimization, quality of service enhancement, and increased flexibility [29].

To facilitate process redesign, researchers have introduced process improvement/redesign patterns (PIPs)—generic concepts aimed at enhancing specific aspects of business processes [30]. Since the early 2000s, numerous studies have focused on process redesign strategies, process redesign patterns and frameworks. A comprehensive overview of these patterns can be found in [31]. These works demonstrate that redesign patterns cover multiple aspects of process models, ranging from structural changes and data transformations to quality, compliance, and risk-related modifications.

However, identifying the most suitable redesign pattern for a given scenario remains challenging, as most researchers focus on only a subset of available patterns, driven by specific requirements and constraints. Given that the scope of our paper is to explore structural changes in process models using process redesign patterns, we refer specifically to change patterns. Several studies have examined change patterns in the context of structural modifications and model variability [32, 33, 31, 34].

In our work, we adopt change patterns as a foundation, specifically referring to the adaptation patterns introduced by Weber et al., apms these patterns are widely recognized and well-established in the literature [35]. Change patterns support users in performing structural modifications on process models by encapsulating multiple low-level actions on individual model element (e.g., add node, delete node, add edge, remove edge, move edge) into a single, semantically meaningful transformation, simplifying process modifications for users. These high-level change operations provide a higher level of abstraction, enabling complex transformations that maintain process integrity [10].

Currently, despite recent interest in AI-augmented BPM systems and the conversational capabilities of LLMs, the potential of generative AI to support multi-turn, user-guided process model redesign using change patterns remains under-explored.

6 Conclusions

In this work, we explore whether an LLM-based agent can effectively support domain experts during the redesign of process models in continuous interaction via a conversational user interface, aiming to overcome the communication gap between domain experts and process modelers. The continuous interaction is based on redesign tasks of the models. To this end, we propose a conversational process model redesign approach that incorporates and adapts existing change patterns for business process models into a conversational context. The approach is systematically evaluated against process models that are redesigned by the user and manually using the change patterns.

I dont know how much the submission will take We not only evaluate existing change patterns, but we also propose the potential extension of already existing ones, which are necessary due to the conversational nature of the user interface of the LLM-based agent. In addition, we introduce the multifaceted evaluation concept, which allows us to grasp not only the fact of the failure of the model redesign, but also the reason why the failure happened, and show, in practice, how to utilize this methodology through an example with three different LLMs.

Model redesign evaluation is a complex task that depends on multiple factors: the user, the LLM, and the nature of the pattern. These include aspects such as how the redesign task is described, whether the description is complete, whether the user applies the correct pattern for the intended change, whether the LLM explicitly understands the provided patterns, which patterns appear similar even if they are not, and the complexity of the change itself.

On average, 9% of cases failed at the stage of identification and 12% at the stage of meaning derivation due to poor wording, highlighting the need for mechanisms that support users in minimizing ambiguity, improving clarity, and selecting the appropriate pattern for their request.

On the other hand, the high level of failures due to the agent during pattern application, especially in more complex cases, indicated the necessity of improved methods of pattern application (i.e., hybrid approaches leveraging the strengths of both LLMs and traditional approaches).

Future research will focus on three directions. The first will explore more complex datasets to address the increasing complexity of real-world scenarios. The second direction will focus on the implementation of change patterns utilizing formalization, i.e., deriving meaning not in a free natural language form but rather in a formal way, to perform its implementation via traditional approaches. This will minimize deviations between expected and actual agent output, helping to focus on conversational and human-agent interaction aspects. The third direction will emphasize evaluating and integrating knowledge about user behaviour to improve the quality of human-chatbot communication, better meeting the needs of domain experts. Additionally, observing this communication as a learning process for domain experts may help develop their modelling skills and foster *process thinking* through active engagement in process model creation.

References

[1] C. M. DaSilva and P. Trkman, Business model: What it is and what it is not, *Long Range Planning* **47**(6) (2014) 379–389.

[2] A. Mursyada, The role of business process modeling notation in process improvement: A critical review, *Advanced Qualitative Research* (2024).

[3] D. Ley, Approximating process knowledge and process thinking: Acquiring workflow data by domain experts, *2011 IEEE International Conference on Systems, Man, and Cybernetics* (2011) 3274–3279.

[4] D. Beverungen, Exploring the interplay of the design and emergence of business processes as organizational routines, *Bus. Inf. Syst. Eng.* **6**(4) (2014) 191–202.

[5] N. Klievtsova, J. Benzin, T. Kampik, J. Mangler and S. Rinderle-Ma, Conversational process modelling: State of the art, applications, and implications in practice (2023).

[6] N. Klievtsova, J. Mangler, T. iik, J.-V. Benzin and S. Rinderle-Ma, How can generative ai empower domain experts in creating process models? (2024), (accepted).

[7] H. Leopold, J. Mendling and A. Polyvyanyy, Supporting process model validation through natural language generation, *IEEE Trans. Software Eng.* **40**(8) (2014) 818–840.

[8] B. Weber, M. Reichert and S. Rinderle-Ma, Change patterns and change support features - enhancing flexibility in process-aware information systems, *Data Knowl. Eng.* **66**(3) (2008) 438–466.

[9] N. Klievtsova, T. Kampik, J. Mangler and S. Rinderle-Ma, Conversationally actionable process model creation, in *Cooperative Information Systems*, (Springer, 2024), pp. 39–55.

[10] B. Weber, S. Rinderle and M. Reichert, Change patterns and change support features in process-aware information systems, in *Advanced Information Systems Engineering*, eds. J. Krogstie, A. Opdahl and G. Sindre (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007), pp. 574–588.

[11] S. Rinderle-Ma, M. Reichert and B. Weber, On the formal semantics of change patterns in process-aware information systems, in *ER*, (Springer, 2008), pp. 279–293.

[12] M. Weske, *Business Process Management - Concepts, Languages, Architectures, Third Edition* (Springer, 2019).

[13] X. Liu, T. Rietz and A. Maedche, Conversational versus graphical user interfaces: the influence of rational decision style when individuals perform decision-making tasks repeatedly, *Universal Access in the Information Society* (06 2024) 1–16.

[14] L. A. Flohr, S. Kalinke, A. Krüger and D. P. Wallach, Chat or tap? – comparing chatbots with 'classic' graphical user interfaces for mobile interaction with autonomous mobility-on-demand systems, in *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, *MobileHCI '21*, (Association for Computing Machinery, New York, NY, USA, 2021).

[15] M. A. Xydis, Comparing change primitives versus change patterns support using think aloud.

[16] M. Voelter, R. Hadian, T. Kampik, M. Breitmayer and M. Reichert, Leveraging generative ai for extracting process models from multimodal documents (2024).

[17] J. Mendling, H. Reijers and J. Recker, Activity labeling in process modeling: Empirical insights and recommendations, *Information Systems* **35**(4) (2010) 467–482.

[18] J. Mendling, H. Reijers and W. Aalst, Seven process modeling guidelines (7pmg), *Information and Software Technology* **52** (02 2010) 127–136.

[19] A. Doren, A. Markina-Khusid, M. Cotter and C. Dominguez, A practitioner's guide to optimizing the interactions between modelers and domain experts (2019).

[20] E. Mickeviciute, R. Butleris, S. Gudas and E. Karciauskas, Transforming bpmn 2.0 business process model into sbvr business vocabulary and rules, *Inf. Technol. Control.* **46** (2017) 360–371.

[21] D. Hildebrand, S. Rösl, T. Auer and C. Schieder, Next-generation business process management (bpm): A systematic literature review of cognitive computing and improvements in bpm (05 2024).

[22] M. Dumas and et al., AI-augmented business process management systems: A research manifesto, *ACM Transactions on Management Inf. Syst.* **14** (2022) 1 – 19.

[23] A. Casciani, M. L. Bernardi, M. Cimitile and A. Marrella, Conversational systems for ai-augmented business process management (2024).

[24] T. Kampik, C. Warmuth, A. Rebmann, R. Agam, L. N. P. Egger, A. Gerber, J. Hoffart, J. Kolk, P. Herzig, G. Decker, H. van der Aa, A. Polyvyanyy, S. Rinderle-Ma, I. Weber and M. Weidlich, Large process models: Business process management in the age of generative AI, *CoRR* (2023).

[25] K. Busch, A. Rochlitzer, D. Sola and H. Leopold, Just tell me: Prompt engineering in business process management (2023).

[26] A. Beheshti, J. Yang, Q. Z. Sheng, B. Benatallah, F. Casati, S. Dustdar, H. R. Motahari-Nezhad, X. Zhang and S. Xue, Processgpt: Transforming business process management with GenAI (2023).

[27] M. Vidgof, S. Bachhofner and J. Mendling, LLMs for business process management: Opportunities and challenges (2023).

[28] U. Jessen, M. Sroka and D. Fahland, Chit-chat or deep talk: Prompt engineering for process mining, *CoRR* **abs/2307.09909** (2023).

[29] A. Kumar and R. Liu, Business workflow optimization through process model redesign, *IEEE Transactions on Engineering Management* **69**(6) (2022) 3068–3084.

[30] M. Lohrmann and M. Reichert, Effective application of process improvement patterns to business processes, *Software and Systems Modeling* **15** (01 2015).

[31] G. Zellner, Towards a framework for identifying business process redesign patterns, *Business Process Management Journal* **19** (07 2013).

[32] A. Yousfi and R. Saidi, Variability patterns for business processes in bpmn, *Information Systems and e-Business Management* **14** (08 2015).

[33] D. Kim, M. Kim and H. Kim, Dynamic business process management based on process change patterns (2007).

[34] A. Kumar and P. Indradat, Optimizing process model redesign, in *Service-Oriented Computing*, (Springer International Publishing, Cham, 2016), pp. 39–54.

[35] M. Fellmann, A. Koschmider, R. Laue, A. Schoknecht and A. Vetter, Business process model patterns: State-of-the-art, research classification and taxonomy, *Business Process Management Journal* **25** (08 2018).