Business Process Design Support with Automated Interviews

Danielle Silva de Castro[1][a], Marcelo Fantinato[2][b] and Mateus Barcellos Costa[1][c]
[1]*Postgraduate Program in Applied Computing (PPComp), Instituto Federal do Espírito Santo, Serra, Brazil*
[2]*School of Arts, Sciences and Humanities, Universidade de Sao Paulo, São Paulo, Brazil*

Keywords: Business Process Design, Conversational Agents, Automated Model Generation.

Abstract: Interviews are widely used to collect and organize decentralized, unstructured, and undocumented information, serving as a valuable tool for business process design and re-design. Conversely, they present several challenges, including high costs related with planning, preparation, and execution, as well as the need for strong engagement from interviewees. Additionally, interviewees often lack a clear and comprehensive understanding of the processes, which complicates their ability to fully articulate them. The advent of intelligent conversational tools presents new opportunities for process elicitation but also introduces challenges in managing the subjective and tacit nature of business process knowledge, while ensuring the generation of consistent and high-quality models. Considering these issues, this paper discusses an automated approach for conducting business process modeling interviews. This approach is designed to capture process behavior using the so-called Situation-Based Modeling Notation (SBMN), which further enables the automated generation of alternative imperative business process models. To evaluate the proposed approach, a conversational agent prototype and a model generator of possible solutions were developed. The experiments conducted demonstrate the ability to construct high-quality models according to the metrics of recall, precision, generalization, and simplicity. The results also indicate that the generated models maintain consistency with process constraints and uncover alternative models effectively.

1 INTRODUCTION

Business process modeling is increasingly adopted by organizations aiming to enhance their efficiency and effectiveness. Its main role is to create models that support mitigating the complexity of planning, managing, executing, controlling, and evolving business processes (Beerepoot et al., 2023).

Modeling occurs either to capture the current behavior and structure of the processes under consideration (as-is) or to introduce modifications and innovations to be implemented in the same processes (to-be), including the design of new processes. In both cases, knowledge about these processes, often manifested only tacitly or subjectively (Dumas et al., 2013; Verner, 2004), must be captured and subsequently transcribed into models using some notation or modeling language, e.g., BPMN – Business Process Model and Notation (OMG, 2013) or EPC – Event-Driven Process Chain (Kim, 1996).

A common practice in modeling involves the participation of a modeling expert who constructs models, e.g., using drawing tools, based on information gathered from domain experts through interviews. Such an approach, although attractive, is prone to errors in conception or interpretation, which can lead to models that deviate from the actual reality they are meant to represent (Kannengiesser and Oppl, 2015; Leyh et al., 2016). According to Leyh et al. (2016), for example, the perspectives of domain experts may be based on fragmented knowledge, lacking individual clarity and/or depth, and originating from multiple sources. The modeler, acting as an intermediary, may produce models based on inaccurate or limited interpretations of reality (Kannengiesser and Oppl, 2015). In addition, it is also recognized that interviewing-drawing-based process modeling involves many manual activities and remains one of the least automated efforts of Business Process Management (BPM) projects. In fact, generally speaking, automating process design and redesign has been pointed out as a major challenge in BPM (Beerepoot et al., 2023).

In Business process modeling, while processing

[a] https://orcid.org/0009-0001-8582-3182
[b] https://orcid.org/0000-0001-6261-1497
[c] https://orcid.org/0000-0002-4235-5411

Silva de Castro, D., Fantinato, M. and Costa, M. B.
Business Process Design Support with Automated Interviews.
DOI: 10.5220/0013362000003929
In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 2*, pages 734-745
ISBN: 978-989-758-749-8; ISSN: 2184-4992

modeling information flows, the modeler may encounter cognitive challenges, including the need to: retain the expected behavior of the process, understand the solution space for modeling and its alternatives, and remain aware of the inherent constraints of the desired process behavior. Since process behavior is typically well-understood by domain experts, this paper presents an approach to support modelers by automating the process of interviewing them. The results of the interview lead to the generation of alternative and possible business process models that capture the specified process behavior and its constraints.

The automated interview is designed to assist modelers in defining the control flow of the process in terms of logical-temporal constraints. In the proposed approach, a computational agent poses a sequence of yes/no questions to a domain expert. The interview results in control flow models that are both consistent and aligned with the reality described by the domain expert.

To evaluate the proposed approach, an empirical study was conducted using a publicly modeling dataset. The experiments demonstrated the ability to generate models with high-quality metrics, as evaluated by typical process mining measures of recall, precision, generalization, and simplicity. It was also observed that the interviews required a relatively small number of questions to produce viable results. However, significant computational effort was observed during this phase to derive the initial solution space.

The remainder of this paper is organized as follows: Section 3 provides an overview of the proposed approach. Section 2 reviews essential conceptual aspects relevant for understanding the proposed solution. In Section 4, aspects of the interviewing model and automated model generation are provided. Section 5 details the experiments conducted. Section 6 reviews related work and Section 7 concludes the article with a discussion on the results obtained, limitations, and directions for future work.

2 BACKGROUND

The control-flow behavior of business processes can be defined through constraints, which can be represented using declarative notations or languages (Fahland et al., 2009). Declarative approaches are designed to focus on specifying the behavioral boundaries of a process by articulating logical and temporal constraints (Pesic et al., 2007). Notable examples of declarative approaches include the Declare language (Pesic et al., 2007) and the Situation-Based Modeling Notation (SBMN), which builds upon the previously established concept of Recommendation Patterns(Costa and Tamzalit, 2017). This study considers the SBMN declarative model, given that this model demonstrates sufficient expressiveness for representing control-flow constraints, while also offering simplicity (Schützenmeier et al., 2023).

In SBMN, a *situation* is a type of relationship between active flow objects that constitute the *Process Domain* (*D*). The situation set of a process represents the mandatory constraints that should be observed during process execution. Table 1 presents the set of situation types that are considered in this work.

Table 1: SBMN Situations Types.

Situation	**Description**	**Usage**
Strict Dependence	Strict precedence between two distinct AFO groups	*b DEP a* – b depends on a.
Circumstantial Dependence	Precedence between two distinct AFO groups in the presence of the depended-upon group	*b DEPC a* – b depends on a, in the presence of a.
Non-coexistence	Mutual exclusion between two distinct AFO groups	*a XOR b* – a excludes b, and b excludes at a same flow.
Union	Logical OR relationship between two distinct AFO groups	*a UNI b* – only a, a and b, or only b may occur in the same flow.

In Table 2, it is presented a SBMN model from which the Project Approval Model, shown in Figure 1, was derived.

Table 2: SBMN model for the Project Approval process.

```
Domain *****************
Pr6 Project Arrival | Re8 Register Project
Re7 Retrieve Project| Ev0 Evaluate
Fi2 Fill Report     | i4  Visit in Loco
Is3 Issue Permit    | Re6 Request Changes

Situations ***************
Re8 DEP Pr6 | Re7 DEP Pr6  | Ev0 DEPC Pr6
Fi2 DEP Pr6 | Ev0 DEPC Re8 | Ev0 DEP Re7
Fi2 DEP Ev0 | Re8 XOR Re7  | Is3 DEP Ev0
Is3 DEP Fi2 | Re6 DEP Ev0  | Vi4 DEP Re7
Vi4 DEP Re8 | Re6 DEP Fi2  | Is3 XOR Re6
```

The labels DEP, DEPC and XOR are used to represent, respectively, strict dependency, circumstantial dependency, and non-coexistence. It can be noted

that the BPMN model in Figure 1 is a valid *instance* of a BPMN model concerning the constraint model. Even tough this model does not violate any of the constraints defined by the declarative model, it does not provide a high quality score given the reference model for the process problem. It occurs because not enough constraints were given for the *Visit in loco* task and it may occur even after the Issue Permit task is done. However, a better solution was also generated for the same SBMN specification as illustrated in Figure 2.

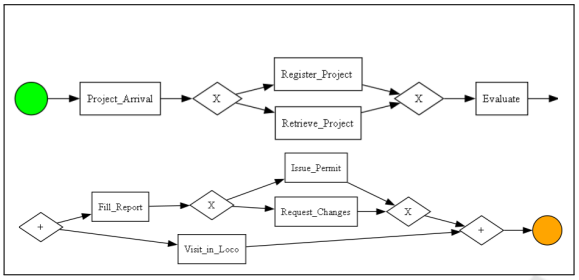

Figure 1: One of the 9 models automatically generated based on the SBMN Specification illustrated in Box 2.

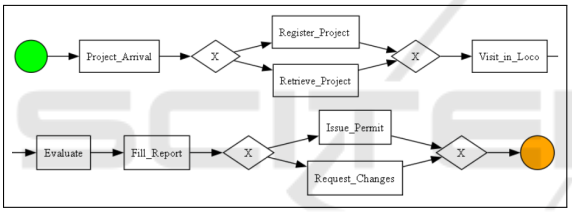

Figure 2: A better BPMN solution based on the SBMN Specification illustrated in Table 2.

3 PROPOSAL OVERVIEW

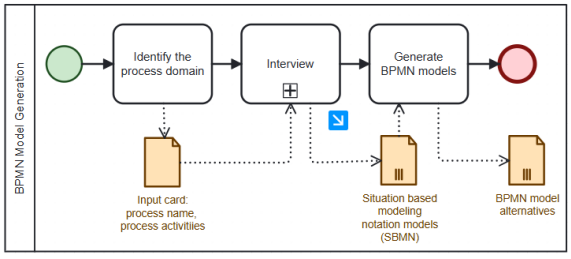

Figure 3: Model Generation Flow.

Figures 3 and 4 provide an overview of the proposed approach, illustrating all stages from identifying the process to be modeled to obtaining alternative BPMN models.

In the flow of model generation, the business process to be modeled must first be identified by a name, along with its *a priori* activities and events. For simplicity, and considering that tasks, events, and subprocesses are indistinguishable in defining the control flow, these elements will collectively be referred to as Active Object Flows (AFOs), or interchangeably as tasks. Ideally, the AFOs that comprise the process should be identified *a priori*; however, additional AFOs may also be introduced during the interview process.

The second stage comprises the interview for the definition of the model based on the perspective of the domain expert. The interview model establishes a flow of questions, considering all plausible hypotheses for the model in the current state of the interview. At the beginning of the interview, the solution space for the modeled process is constructed. This solution space allows inferring questions whose answers lead to new reconfigurations of the solution space to keep it consistent with the interview results. The interview process continues until the solution space is considered satisfactorily sized, meaning the remaining models are appropriate representations of the modeled process given the available information.

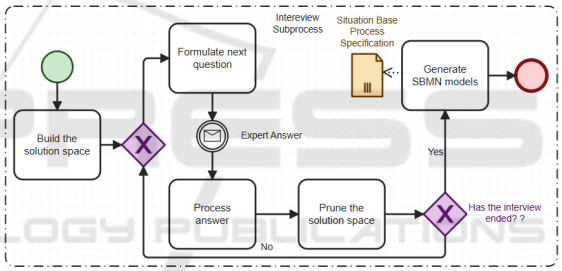

Figure 4: Interviewing Internal flow.

Figure 4 illustrates the interview process in detail. The construction of the Solution Space for the process, as discussed in Section 4.1, encompasses all probable hypotheses of models for the modeled process. These models are built using a declarative notation called Situation-Based Modeling Notation (SBMN), which describes the operational semantics of processes through a logic-temporal constraint-based notation. The solution space is represented as a directed graph, where the vertices correspond to situations and the edges connect valid models. For instance, let us suppose a model containing the following situations exists in the graph: (B Dep A), (C Dep A), (C Xor D), (E Dep C). Then, this graph would have a path of edges connecting the set of vertices that represent these situations.

Figure 5 illustrates the proposed approach, presenting the interview input artifacts, which include basic interview data (interviewee, process name, and application area) and *a priori* tasks; the SBMN model, obtained through the interview, and a BPMN

model generated from the SBMN model.

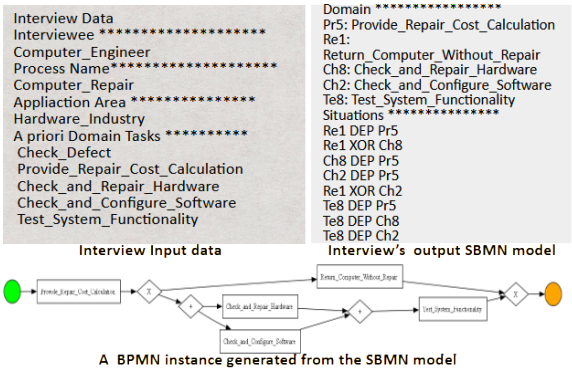

Figure 5: Proposal artifacts. From the interview input to the BPMN result.

The questions are formulated on the basis of the set of valid questions not yet asked and the current state of the solution space. As the questions are answered, the solution space is reconfigured through pruning. For instance, if a given situation, say B Dep A, is indicated as true (present) for the model, the solution space is pruned by removing all models that do not include this situation. If the situation is indicated as false (not present) for the model, the Solution Space is pruned by removing all models that include it.

The interview process can end with one or more valid models reflecting consistent arrangements of the situations indicated as true during the interview. These models are referred to as declarative specifications of the process, described using SBMN. These specifications are then used for the automatic generation of BPMN models consistent with them. For each SBMN model generated, its alternative BPMN models, i.e., the BPMN models that meet the constraints at the SBMN model, can be produced. To analyze the quality of the generated models, experiments were conducted using processes for which a Reference Model was available. The generated models were then compared to the reference models using the typical metrics used to evaluate solutions generated with Process Mining techniques. These metrics were obtained according to the techniques discussed by Fantinato et al. (2023), by using the Processes Mining Python (pm4py) library.

4 INTERVIEWING AND MODEL GENERATION

Interviews aimed at identifying a process's control flow often focus on logical-temporal constructs, which can create a vast possibility space, making it challenging for interviewees to fully disclose process variability. Furthermore, transcribing this information into model constructs involves numerous acceptable combinations of elements. As discussed previously, this combined approach of interviews and modeling can be quite appealing but may be prone to errors and deviations caused by issues in the conception and interpretation of process requirements.

To address these issues, a computational agent-based interviewing and model generator is proposed, aimed at generating procedural model alternatives to describe the process control flow. The interviewing model is designed to simplify interactions through a series of questions with "yes," "no," or "indifferent" answers.

During the interview process, after identifying the preliminary process domain (its set of a priori tasks), the computational agent begins posing questions involving the domain tasks and the constraints permitted by the target notation (SBMN). For example, consider the following question: "Does the **Register project** Depend on **Project arrival**?" In this question, two tasks are linked by a dependency constraint. The response to this question may introduce the constraint into the model, eliminate the possibility of its existence, or leave its inclusion undecided. "Yes" or "No" answers enable the agent to prune the solution space, narrowing it to a set of models that are fully consistent and aligned with the constraints under evaluation by the interviewee.

4.1 Solution Space

The interview sequence of questions is based on the analysis of the Frame of discernment (Schubert, 2012). This structure corresponds to the set of all acceptable hypotheses for process models, referred to as the modeling *solution space*. It was derived from subset trees of the set $\mathcal{P}(S)$ of all possible situations that can be formulated for the process domain. Consequently, the problem assumes time and space complexity with the ceiling function $F(N_S) \propto 2^{N_S}$, where N_S represents the number of possible situations for a process with domain size n in the chosen language.

For an interviewing model that considers the subset of the SBMN notation comprising strict dependence, union, and non-coexistence, as discussed in previously work (Cabral et al., 2021), with S as the set of all situations that can be formulated for the process domain and the process domain having cardinality n, the cardinality of S is given by $2n(n-1)$. The growth of N_S therefore follows the second-order arithmetic progression $\{4, 12, 24, 40, 60, 84, 112, \ldots\}$. Assum-

ing all models in the Solution Space are consistent, the problem size in terms of time and space would grow proportionally to $2^{2n(n-1)}$, becoming intractable even for a small number of tasks.

To mitigate the problem of combinatorial explosion in the solution space, a strategy of gradual introduction of task was adopted. In these cases, the *a priori* set of tasks is reduced to a manageable subset, and the interview is initiated. The remaining tasks are gradually introduced after a significant reduction in the solution space. Another possible strategy is the modularization of the process into phases when applicable. In this case, the interview can be divided into stages, each composed of a manageable number of tasks.

The concept of circumstantial dependence was introduced as a model extension, ensuring that the addition did not increase the size of the solution space. There are two typical scenarios for Depc use. The SBMN meta-model defines the [B Depc A] (A depends circumstantially on B) semantics as: B depends on A if and only if A executes. Otherwise, if A does not run in a process case, B will execute without any constraints related to A. Depc is necessary to preserve the consistency of models in cases where a task should depend on tasks that may not occur.

Figure 6 illustrates a typical case of Depc use. While Ship the product should execute only after Request from manufacturer if it executes, Ship the product can execute without verifying this dependence when it is not necessary to execute Request from manufacturer. In the development of the SBMN model, if a strict dependence were used to represent the relation between these two tasks, the model would become inconsistent, given that a non-coexistence between Request from manufacturer and a void branch should also be formulated to represent the complete specification. That is, instead of modeling with the strict Dep situation ([Request from manufacturer XOR voidbranch], [Ship the product DEP Request from manufacturer]), the DEPC is used and the model remains consistent: ([Request from manufacturer XOR voidbranch], [Ship the product DEPC Request from manufacturer]). The second scenario is quite

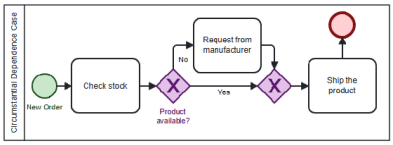

Figure 6: Circumstantial Dependence Out of Stock Example.

common. It is the case where a task depends on two non-coexistent tasks. Again, if strict dependence were used, the model would become inconsistent (e.g., [A DEP B], [A DEP C], [C XOR B]). The correct model should use DEPC ([A DEPC B], [A DEPC C], [C XOR B]).

4.2 Guaranteeing Consistency of Models in SBMN

Given that the solution space consists of models described in SBMN, the validity of these models must be ensured, and only consistent models should be included in the solution space. This consistency evaluation is performed indirectly by considering part of the semantics of the BPMN notation, ensuring the following property:

Validity Property: A SBMN model is considered valid if a BPMN model that satisfies its set of situations can be derived from it.

As an example, consider the model $M_1 = \{D = \{a,\ b,\ c\},\ S = \{b\ DEP\ a,\ a\ XOR\ c\}\}$. This specification is valid because it is possible to construct a BPMN model that includes all elements of the domain of P and does not violate its situations. On the other hand, $M_2 = \{D = \{a,\ b,\ c\},\ S = \{b\,DEP\,a,\ a\,XOR\,b\}\}$ is invalid, as it is impossible to construct a BPMN model that complies with the process constraints, since **b** cannot depend on **a** while **a** is non-coexistent with **b**.

To verify this property, the following set of assertive tests must be checked as false:

1. **Equivalent Operators** - Two situations of different types have the same operators. For instance, $C = \{b\,DEP\,a,\ a\,XOR\,b\}$.
2. **Cyclic Dependency** - It occurs when a set of operators forms a dependency cycle caused by the situations in a model. For example, a cycle can be found in the specification $C = \{b\ DEP\ a,\ a\ DEP\ b\}$ or $C = \{b\,DEP\,a,\ a\,DEP\,c,\ c\,DEP\,b\}$. Cyclic dependency should be ensured for any linear process. In the presence of loops this assertion should be flexibilized in order to avoid deadlock-style dependencies and, at the same time, to enable loops relation.
3. **Blocking of Indirect Dependency** - Occurs when the operators in an indirect dependency relationship are involved in a choice relation within the same specification. In such cases, no valid BPMN model exists where the operators can depend on each other while satisfying the choice relation. In the specification $C = \{a\,DEP\,b,\ b\,DEP\,c,\ a\,XOR\,c\}$, an indirect depen-

dency blocking occurs because a indirectly depends on c, and both are in a choice relation.

4. **Promiscuity** - Occurs when there is a dependency between two operands, each involved in different choice relations of different types. For instance, $C = \{a\ XOR\ c,\ a\ DEP\ b,\ b\ XOR\ c\}$ has a promiscuous relationship, as a depends on b, making it impossible to have a BPMN model containing a non-coexistence (xor) between a and b and an union (or inclusive) between b and c.
5. **Dual Dependency** - Dual dependency occurs when both operators in a choice relation depend on the same operator in the specification. For example, in the specification $C = \{a\ XOR\ b,\ c\ DEP\ a,\ c\ DEP\ b\}$, there is a dual dependency violation because c depends on both a and b, which are non-coexistent, preventing c from being executed.

As previous discussed, the conversational agent is based on determining the set of possible models for a given process domain, formulating yes/no questions in a manner consistent with the current state of the interview and the responses already provided. To ensure unambiguity, the agent applies the assertive tests and avoids questions that may introduce inconsistencies. The heuristic employed in the interview formulation seeks to minimize the number of questions, prioritizing those that promote convergence toward a smaller set of resulting models. For instance, if at a given stage of the interview a dependency chain $C \rightarrow B \rightarrow A$ is identified, a new question involving task D will first be asked concerning the final element of the chain (C), thereby implicitly establishing the dependency of D on both B and A without requiring additional verifications. However, the interview process does not incorporate mechanisms to guide the interviewee in explicitly detailing the process based on its complexity.

A relevant aspect of this approach is the possibility of decentralized SBPMN model construction, enabling the integration of perspectives from different stakeholders. Although no experiments have been conducted in this regard, the merging of SBPMN models would be straightforward if the typical challenges of semantic integration—common to the fusion of any models—are overcome. If such integration occurs, for instance, through the terminological reconciliation of task names, unification at the SBPMN level is direct, facilitating its application in complex modeling scenarios. From this model, BPMN models are generated in the same manner as individual ones.

SBPMN models can be considered closer representations of business models than BPMN implementations. Declarative models can capture more stable aspects of business processes compared to BPMN diagrams. Although experimental validation of this claim is still required, the proposed approach appears to align with Business Process Management requirements.

4.3 Generation of BPMN Models from SBMN Models

Valid SBMN models can be transformed into BPMN models through the concept of Recommendation Patterns which enable recognizing a situation and providing an imperative-style solution for that situation. As with SBMN situation types, more than one solution may be available; hence, a solution is considered a recommendation. In practical terms, each solution for a situation is represented as a BPMN fragment, and the composition of these solutions results in the construction of valid and complete BPMN model alternatives (Van der Aalst et al., 2004; Costa and Tamzalit, 2017).

A composition algorithm was designed and adapted for the automatic generation of BPMN models. Its general structure is depicted in Algorithm 1. It receives a SBMN model as input and generate a specific BPMN model alternative as output. In order to generate all BPMN alternatives for one SBMN model, the algorithm should be executed repetitively by taking all possible recommendation for each step of generation (that corresponds to the recommendation generation in the While loop of the algorithm). Once a recommendation is selected, its correspondent BPMN fragment can be generated and inserted in the abstract BPMN model. Taking in to account the adopted BPMN subset, the following structures are valid recommendations:

Sequential: A single active flow object. Sequential recommendation is represented by the flow object identifier.

Parallel: A Parallel gateway connecting one or more active objects flow . **Sequential Or:** Sequential inclusive Or gateway connecting one or more active flow objects.

Sequential Xor: An exclusive Xor gateway connecting one or more active flow objects.

PAR_Close: A convergent Parallel gateway recommendation to close parallel branches.

Or_Close: An inclusive convergent Or gateway recommendation to close Or branches.

Xor_Close: An exclusive convergent Xor gateway recommendation to close Xor branches.

The models are progressively constructed in a forward-completion style (Kluza and Nalepa, 2013) by analyzing the situations not yet addressed at spe-

```
Data: SBMN Model sbmn_m
Result: Abstract BPMN model
load(sbmn_m);
preProcess(sbmn_m);
RecommendationPoint rp =
  CreateRecommendationPoint(sbmn_m);
Q.Enqueue(rp);
while (Q.notEmpty()) do
    RecommendationPoint currentRP = Q.dequeue();
    RecommendationList recs =
      determineRecommendations(currentRP);
    if (recs.notEmpty()) then
        rec = selectRecommendation(recs);
        BPMN_fragment = generateBPMNFragment(rec);
        BPMN_model.insertBPMNFragment(
          BPMN_fragment);
        recommmendationPointList new_rps=
          processRecommendation(rec);
        q.enqueue(new_rps);
    end
end
```

Algorithm 1: SBMN to BPMN Algorithm.

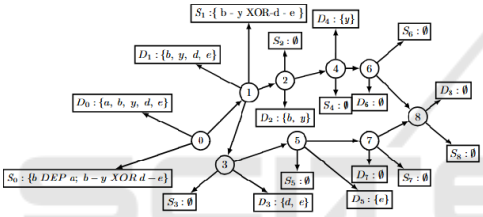

Figure 7: Recommendation Point Graph for the SBMN P_1.

cific analysis points called recommendation points. The recommendation points of a model are used to construct a graph that indicates its generation flow and the corresponding modeling decisions made throughout the generation process. Depending on the model fragment inserted at a given point, new recommendation points may be created, allowing the process to continue until the complete model is built. Each newly generated recommendation point determines its respective subset of recommendations through its associated subset of the domain and the situations that apply to this subset. Figure 7 illustrates the Recommendation Point Graph corresponding to the SBMN model P_1, with Domain $D = a, b, y, d, e$ and Situations $S = [b\ DEP\ c], [b, y\ XOR\ d, e]$. At each recommendation point, its subdomain D_i and the incident set of situations S_i are shown.

5 EXPERIMENTATION AND RESULT EVALUATION

To conduct validation experiments for the proposed approach, prototype implementations of a conversational agent and a model generator were developed. Additionally, an analysis application was created to facilitate tasks related to experiment evaluation. The conversational agent and model generator prototypes were implemented in Java, while the analysis application was developed in Python, utilizing the Process Mining for Python (pm4py) API (Group, 2024).

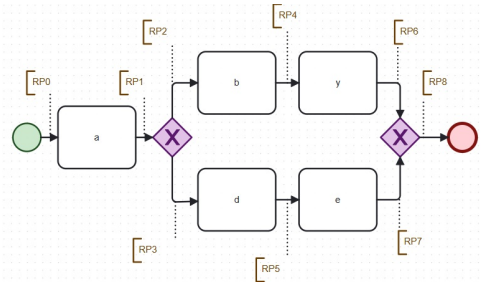

Figure 8: BPMN model with the Recommendation Points of the RPG graph of Figure 7.

The output of the interview application corresponds to one or more SBMN models. These models are represented as JSON files, which serve as input for the Model Generator. The Model Generator produces abstract BPMN files as output, also in JSON format. These files are transformed into full BPMN 2.0 XML based representations by the Result Analyzer, making them ready for analysis.

All data produced and used are available in a public repository on GitHub[1].

5.1 Metrics and Evaluation Procedure

The evaluation of the results produced by the proposed approach was carried out using metrics commonly employed in the context of Process Mining to compare models obtained from event logs. The following metrics were considered (Van Der Aalst, 2016):

- **Fitness or Recall:** Quantifies how well the evaluated model allows the execution of behaviors identified in the reference event log.
- **Precision:** Assesses the extent to which the identified model reproduces behavior not found in the event log;
- **Generalization:** Measures the model's ability to generalize the diverse possible behaviors identified in the reference event log;
- **Simplicity:** Quantifies the simplicity of the generated model. According to Fantinato et al. (2023), simplicity is achieved by using the minimum number of elements necessary to represent behavior, resulting in improved readability.

[1] https://github.com/mbarcosta/ProsaviewDataSets.git

For the evaluated processes, no real logs were available. To address this obstacle and enable the application of these metrics, a BPMN model referred to as the Reference Model was created for each process. These reference models were then used to generate logs, which allowed the application of algorithms for calculating metrics for each solution generated based on the constraint model collected during the interviews. In other words, an event log was simulated from the reference model and subsequently used for analysis.

5.2 Results

During the experiments, good performance was observed even with larger numbers of solutions. Figure 9 presents a comparison of execution time variation for analyzing each solution. It can be observed that there is little variation when comparing the sets of 9, 28, and 87 solutions. For the set of 189 solutions, there was an increase in time, which still approximates the performance of the other sets. However, for the problem containing 774 solutions, the values are more divergent, suggesting a potential exponential behavior.

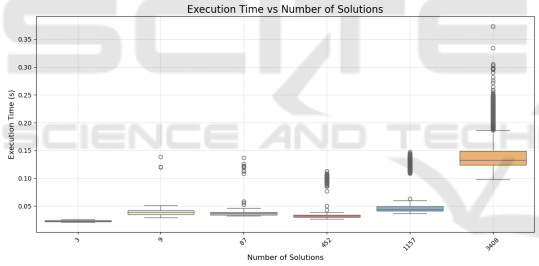

Figure 9: Time X Solutions.

Figures 10 - 14 show the values obtained for the recall, precision, generalization and simplicity metrics for each solution to some of the analyzed problems. It is observed that, as the data sets grow, the variation levels for each metric also increase. This behavior is due to the fact that, in most cases, data sets containing more solutions are obtained from "quick" interviews, where a small number of questions are asked, resulting in fewer constraints for the model. In contrast, interviews conducted with more detail, involving a larger number of questions, produce solutions closer to the ideal. These solutions, when analyzed by the metric algorithm, exhibit lower variation.

Table 3 provides a small example of the behavior described above. For the problem "caseHandling_1", 452 solutions were generated from a declarative model extracted during an interview involving 80 questions out of a total of 144 possible questions,

Table 3: Interviews Informations by Process.

Interviews by Process			
Process Name	**Solution Count**	**Questions Count**	**Possible Questions**
caseHandling_1	452	80	144
complaint_1	1157	72	112
E2_proc	87	32	60

meaning over 55% of the questions were addressed. Conversely, for the problem "E2_proc", only 87 valid solutions were generated from 32 answered questions out of a total of 60, achieving a response rate of approximately 53%. The slight decline in the response rate was sufficient to increase the number of possible solutions generated for this problem.

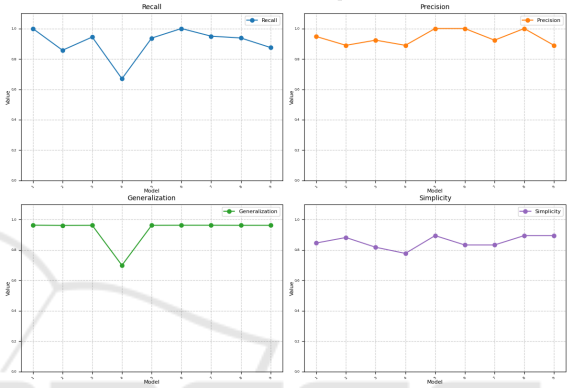

Figure 10: Results for the "permition_2" experiment (9 solutions).

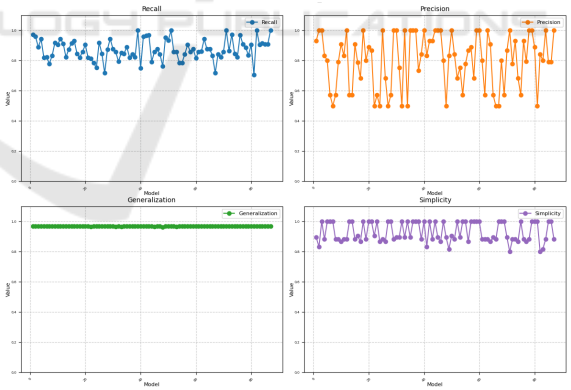

Figure 11: Results for the "E2_proc" experiment (87 solutions).

Table 4 shows statistics of the results obtained for each conducted experiment. It is observed that, as expected, problems containing a greater number of possible solutions exhibit an average rate of results obtained for each metric decreasing compared to values found for problems with fewer generated solutions. However, the metrics for generalization and simplicity show a more linear decline when compared to the results obtained for the recall and precision metrics.

Table 4: Statistics per experiment.

Process Index	Process Name	Solution Count	Metric Statistics							
			Recall		Precision		Generalization		Simplicity	
			Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	caseHandling_1	452	0.8702	0.0423	0.6656	0.0717	0.9187	0.0498	0.8422	0.0295
2	complaint_1	1157	0.8607	0.0508	0.6883	0.0909	0.9616	0.0244	0.8640	0.0475
3	ComputerRepair_1	3	0.9728	0.0235	1.0000	0.0000	0.9593	0.0004	0.9608	0.0679
4	ComputerRepair_2	9	0.9617	0.0235	1.0000	0.0000	0.9581	0.0003	0.9406	0.0593
5	decpm8-200000	3408	0.8067	0.0775	0.4344	0.3052	0.9560	0.0228	0.7857	0.0382
6	E2_proc	87	0.8720	0.0683	0.8019	0.1765	0.9682	0.0007	0.9285	0.0646
7	permition_2	9	0.9078	0.1014	0.9403	0.0488	0.9335	0.0877	0.8528	0.0415
8	Steelmaking_1	9	0.9577	0.0226	1.0000	0.0000	0.9603	0.0005	0.9494	0.0510

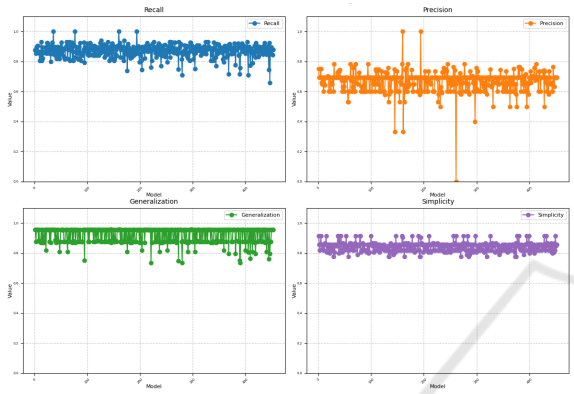

Figure 12: Results for the "caseHandling_1" experiment (452 solutions).

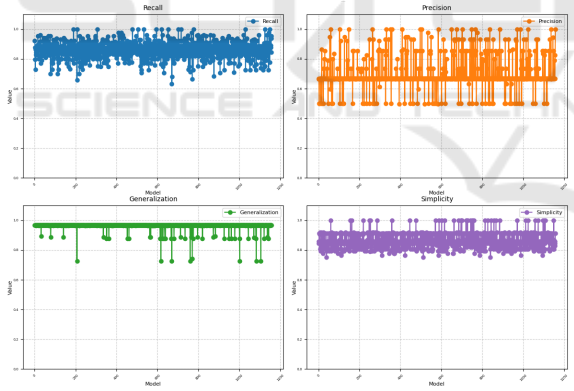

Figure 13: Results for the "complaint_1" experiment (1157 solutions).

5.3 Analysis of Results

In all experiments conducted, the presence of suitable solutions was consistently confirmed, as illustrated in Figures 10 – 14. This behavior was expected, as the solution generator based on SBMN specifications is capable of exhaustively generating all possible solutions for the same model. In the absence of a reference model, identifying the most appropriate solution depends on minimizing solution variability, which, in turn, is influenced by the amount of information provided during the interview.

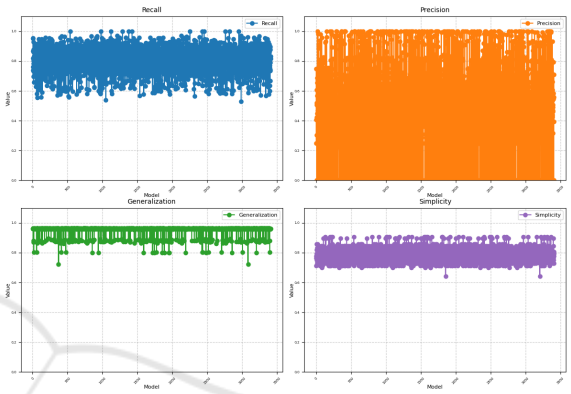

Figure 14: Results for the "decpm8-200000" experiment (3408 solutions).

To illustrate this issue, a synthetic model (DecPM8) was created with minimal information provided, resulting in an SBMN model containing only five situations. The SBMN input model is presented in Table 5.

Table 5: SBMN model for DecPM8 process.

```
Domain **********
Name   a  b  c  d  e  f  x  y
Id     a  b  c  d  e  f  x  y
Situations **********
a XOR b | d XOR c
x XOR c | y XOR c | x DEP d
```

For this process, the experiment generated 774 variations of the model, including the reference model, which is shown in Figure 16. The corresponding generated model is presented in Figure 15. In the absence of a reference model, identifying the most appropriate one would be a challenging task. Conversely, when the information gathered during the interview is more comprehensive, as in the case of the ComputerRepair_1 model, only three solutions are possible, making the identification of the correct model significantly more straightforward.

Among the evaluated processes, the Case Handling process represents an actual interview con-

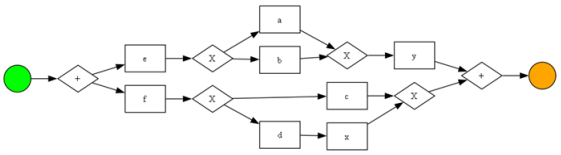

Figure 15: Generated Model for the Decpm8 process with best fitness.

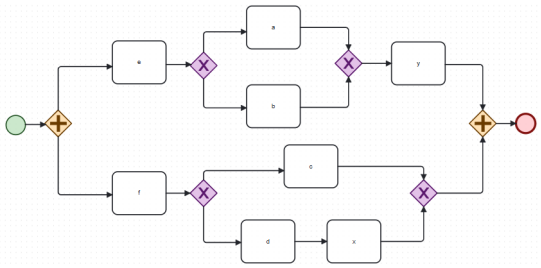

Figure 16: Reference Model for the Decpm8 process.

ducted in the academic registry department of a University. The reference model for this process was obtained from this department. The remaining processes are sourced from the literature (Complaint_1 (Mendling et al., 2010), Permit_2, ComputerRepair_1 and 2 (Dumas et al., 2013)) or developed by our team (Steelmaking_1, Decpm8_20000 and E2_proc). In these cases, the interviews were conducted with individuals knowledgeable in BPMN, who were provided with a textual description of the process and asked to develop the reference model.

For the current version of the implemented prototypes, the following considerations were made:

1. Linear processes without loops were considered, although the addition of loops is already a defined aspect of the research, as discussed in the Extensibility section.
2. For the generation of BPMN models, only control flow elements and exclusive or parallel gateways were considered.
3. Aspects of communication and message exchange were not yet taken into account.
4. The construction of the model was limited to producing structured models, where each divergent gateway has a corresponding convergent gateway. Additionally, only gateways with two outgoing branches were produced as recommendations. This limitation does not affect the expressiveness of the solution, as gateways with any number of outgoing branches can be equivalently represented using a set of gateways with a maximum of two branches. These aspects, while limiting the variability of the models created, do not impose representation limitations. They were also adopted as they are recognized as best practices for model creation (Mendling et al., 2010).

6 RELATED WORK

Klievtsova et al. (2023) provide a literature review on the subject of Conversational Process Modeling. The authors highlight that the application of Conversational Agents or Chatbots for both gathering modeling information and supporting Model Generation remains a little explored theme, despite its recognized potential. During their investigation, no substantial work was found when searching for the terms “chatbots”, “conversational agents”, and “process modeling”. Moreover, most related studies have focused on directions such as quality and analysis, primarily employing natural language processing methods, while neglecting approaches designed to address specific aspects of process modeling or to support its particular tasks. In particular, the use of such agents for information gathering and model construction remains poorly addressed.

In the literature review conducted for this work, two perspectives were considered to identify initiatives related to the proposed approach: the perspective of process control flow determination and the perspective of information gathering and conversational agents. Works with some proximity to the approach, such as models based on NLP, are discussed. However, as also highlighted byKlievtsova et al. (2023), to the best of our knowledge, no proposals involving the direct application of conversational agents to control flow acquisition have been identified.

R’bigui and Cho (2017) discuss Process Mining as a means of obtaining control flow. One aspect highlighted is the inconsistency in the results, which are often not easily interpretable and have usability and accessibility limitations. The authors propose suggestions for greater user involvement. This procedure, called User-Centered Process Mining, is based on ISO Standard 9241-210: 2010. More recently, the use of more suitable metrics has led to better results, as presented by Fantinato et al. (2023), who discuss the application of a genetic algorithm to obtain process models from event logs. These models, evaluated using metrics such as recall, simplicity, generalization, and precision, achieved significantly superior results compared to alternative approaches, such as those proposed by Augusto et al. (2019) and Leemans et al. (2014).

Business process recommendation is another approach widely adopted for obtaining control flow. Wang et al. (2019) discuss a proposal focused on incorporating behavioral characteristics of processes

into search keys for similarity-based search methods to find recommendations. The authors suggest transforming process models into representations based on the so-called Task-Based Process Structure Tree (TPST). Sola et al. (2021) address process recommendation by generating activity suggestions for specific positions in the model during the modeling process, treating it as a knowledge graph completion problem. Khider et al. (2018) propose an unique recommendation method that involves associating process models with potential users. In this subject-based approach, process models are recommended in their entirety to promote reuse. Wang et al. Wang et al. (2022) suggest a model for obtaining recommendations that complements a graph similarity-based method with a cost model. Using the cost representation associated with the retrieved models, a constraint-based cost method is applied to identify more efficient recommendations. Shreya et al. (2023) present a generic recommendation model enabling ordinary users to design process models simply and quickly.

Considering the perspective of conversational agents and information gathering, van der Aa et al. (2019) presented an approach using Natural Language Processing (NLP) for the automated extraction of declarative process models from textual descriptions. According to the authors, this approach resulted in high accuracy compared to manually established constraints, but many challenges related to the extraction of declarative constraints inherent to the use of NLP had to be overcome. Alman et al. (2020) proposed a *chatbot* for defining declarative constraints using natural language instructions provided via voice or text interactions with users. The supported constraints employ *Multi-Perspective Declare* (MP-Declare), complementing control flow constraints with data and temporal perspectives. Additionally, van der Aa et al. (2019) and Van der Aa et al. (2018) developed customized NLP techniques to identify activities and their interrelationships based on textual descriptions of constraints.

7 CONCLUSION

This work discussed the use of conversational agents to assimilate specialized knowledge about business processes, aiming at the automated generation of control flow models from this knowledge. The results demonstrate the generation of models with satisfactory metrics when compared to reference models. In the experiments, the presence of a small number of solutions for a given process could be interpreted as the result of a better specification of the process through the interview data. Conversely, a large number of generated solutions may indicate either a lack of specificity in the interview data or a high flexibility in the process. In both scenarios, the absence of a reference model requires determining which solution is most appropriate. For cases involving numerous solutions, a potential approach is the development of classifiers capable of evaluating, in light of the interview data, which solutions are better or worse.

The proposed approach seeks to simultaneously facilitate access to tacit knowledge about processes, typically held by domain experts, and support the modeling task. The finding of good models demonstrate the approach's potential to address typical challenges in process design and leverage intelligent conversational tools to improve the efficiency of business process management. With the establishment of AI tools based on Large Language Models (LLMs), such as GPTs, the use of conversational agents has become a widely accepted and scalable approach. Since the proposal is designed independently of the algorithms developed, it becomes possible, for instance, to use SBMN model validation rules to guide GPTs in formulating questions. Future work is expected to include the incorporation of loops and the development of a classification method for the solutions based on the obtained SBMN models.

ACKNOWLEDGMENTS

We would like to thank the Graduate Program in Applied Computing at Ifes–Serra (PPCOMP) and the Capixaba Open University Program (UnAC) of the Secretariat for Science, Technology, Innovation, and Professional Education (SECTI) of the Government of the State of Espírito Santo, Brazil, for their support in the development of this work.

REFERENCES

Alman, A., Balder, K. J., Maggi, F. M., and Aa, H. v. d. (2020). Declo: A chatbot for user-friendly specification of declarative process models. In *CEUR Workshop Proceeding*, volume 2673, pages 122–126. RWTH.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., and Polyvyanyy, A. (2019). Split miner: automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems*, 59.

Beerepoot, I., Di Ciccio, C., Reijers, H. A., Rinderle-Ma, S., Bandara, W., Burattin, A., Calvanese, D., Chen, T., Cohen, I., Depaire, B., et al. (2023). The biggest business process management problems to solve before we die. *Computers in Industry*, 146:103837.

Cabral, F. B., Coutinho, B. C., de Castro, F. Z., and Costa, M. B. (2021). Interações automatizadas para apoio à modelagem de processos de negócio. In *Brazilian Symposium on Intelligent Automation (SBAI)*, volume 1. Original in Portuguese.

Costa, M. B. and Tamzalit, D. (2017). Recommendation patterns for business process imperative modeling. In *Proceedings of the Symposium on Applied Computing*, pages 735–742. ACM.

Dumas, M., La Rosa, M., Mendling, J., Reijers, H. A., et al. (2013). *Fundamentals of business process management*, volume 1. Springer.

Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., and Zugal, S. (2009). Declarative versus imperative process modeling languages: The issue of understandability. In *Enterprise, Business-Process and Information Systems Modeling*, pages 353–366. Springer.

Fantinato, M., Peres, S. M., and Reijers, H. A. (2023). X-processes: Process model discovery with the best balance among fitness, precision, simplicity, and generalization through a genetic algorithm. *Information Systems*, 119:102247.

Group, P. M. (2024). Pm4py – process mining for python. https://pm4py.fit.fraunhofer.de. Accessed: (11/12/2024).

Kannengiesser, U. and Oppl, S. (2015). Business processes to touch: Engaging domain experts in process modelling. In *BPM (Demos)*, pages 40–44.

Khider, H., Hammoudi, S., Benna, A., and Meziane, A. (2018). Social business process model recommender: An mde approach. In *2018 Fifth Int. Conf. on Social Networks Analysis, Management and Security (SNAMS)*, pages 106–113. IEEE.

Kim, Y.-G. (1996). Process modeling for bpr: event-process chain approach. In *Proceedings of The Korea Society of Management Information Systems*, pages 41–47. The Korea Society of Management Information Systems.

Klievtsova, N., Benzin, J.-V., Kampik, T., Mangler, J., and Rinderle-Ma, S. (2023). Conversational process modelling: state of the art, applications, and implications in practice. In *International Conference on Business Process Management*, pages 319–336. Springer.

Kluza, K. and Nalepa, G. J. (2013). Automatic generation of business process models based on attribute relationship diagrams. In *International Conference on Business Process Management*, pages 185–197. Springer.

Leemans, S., Fahland, D., and Aalst, W. (2014). Discovering block-structured process models from event logs containing infrequent behaviour. volume 171, pages 66–78.

Leyh, C., Bley, K., and Seek, S. (2016). Elicitation of processes in business process management in the era of digitization–the same techniques as decades ago? In *Int. Conf. on Enterprise Resource Planning Systems*, pages 42–56. Springer.

Mendling, J., Reijers, H. A., and van der Aalst, W. M. (2010). Seven process modeling guidelines (7pmg). *Information and software technology*, 52(2):127–136.

OMG, O. M. G. (2013). Bpmn 2.01 specification. https://www.omg.org/spec/BPMN/2.0.1.

Pesic, M., Schonenberg, H., and Van der Aalst, W. M. (2007). Declare: Full support for loosely-structured processes. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 287–287. IEEE.

R'bigui, H. and Cho, C. (2017). The state-of-the-art of business process mining challenges. *Int. Journal of Business Process Integration and Management*, 8(4):285–303.

Schubert, J. (2012). Constructing and evaluating alternative frames of discernment. *International Journal of Approximate Reasoning*, 53(2):176–189.

Schützenmeier, N., Jablonski, S., Käppel, M., and Ackermann, L. (2023). Comparing the expressiveness of imperative and declarative process models. In *International Workshop on Model-Driven Organizational and Business Agility*, pages 16–31. Springer.

Shreya, J., Saini, A., Kumari, S., and Jain, A. (2023). Generic recommendation system for business process modeling. In *Int. Conf. On Innovative Computing And Communication*, pages 267–282. Springer.

Sola, D., Meilicke, C., van der Aa, H., and Stuckenschmidt, H. (2021). A rule-based recommendation approach for business process modeling. In *Int. Conf. on Advanced Information Systems Engineering*, pages 328–343. Springer.

Van der Aa, H., Carmona Vargas, J., Leopold, H., Mendling, J., and Padró, L. (2018). Challenges and opportunities of applying natural language processing in business process management. In *COLING 2018: The 27th Int. Conference on Computational Linguistics: August 20-26, 2018 Santa Fe, NM, USA*, pages 2791–2801. ACL.

van der Aa, H., Di Ciccio, C., Leopold, H., and Reijers, H. A. (2019). Extracting declarative process models from natural language. In Giorgini, P. and Weber, B., editors, *Advanced Information Systems Engineering*, pages 365–382, Cham. Springer.

Van Der Aalst, W. (2016). *Process Mining - Data science in action*. Springer.

Van der Aalst, W., Weijters, T., and Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142.

Verner, L. (2004). The challenge of process discovery. *BPM Trends, May*, pages 05–04.

Wang, J., Gui, S., and Cao, B. (2019). A process recommendation method using bag-of-fragments. *International Journal of Intelligent Internet of Things Computing*, 1(1):32–42.

Wang, Q., Shao, C., Fang, X., and Zhang, H. (2022). Business process recommendation method based on cost constraints. *Connection Science*, 34(1):2520–2537.