Proceedings of the Federated Conference on Computer Science and Information Systems pp. 733–742
DOI: 10.15439/2019F174
ISSN 2300-5963 ACSIS, Vol. 18

Automated Generation of Business Process Models using Constraint Logic Programming in Python

Tymoteusz Paszun, Piotr Wiśniewski*, Krzysztof Kluza* Antoni Ligęza
AGH University of Science and Technology
al. A. Mickiewicza 30, 30-059 Krakow, Poland
*E-mail: {wpiotr,kluza}@agh.edu.pl

***Abstract*—High complexity of business processes in real-life organizations is a constantly rising issue. In consequence, modeling a workflow is a challenge for process stakeholders. Yet, to facilitate this task, new methods can be implemented to automate the phase of process design. As a main contribution of this paper, we propose an approach to generate process models based on activities performed by the participants, where the exact order of execution does not need to be specified. Nevertheless, the goal of our method is to generate artificial workflow traces of a process using Constraint Programming and a set of predefined rules. As a final step, the approach was implemented as a dedicated tool and evaluated on a set of test examples that prove that our method is capable of creating correct process models.**

***Index Terms*—business process management, process composition, workflow logs, constraint programming, BPMN**

I. Introduction

THE purpose of the existence of any organization or company is to carry out its mission effectively and efficiently. Lack of coordination of operational activities may result in the ineffective achievement of goals, and in extreme cases may lead to failure of the entire undertaking. This is a particular threat to enterprises, understood here as organizations whose mission is to provide specific products in the form of goods or services. In their case, permanent failure to meet the clients' needs usually results in bankruptcy or severe difficulties in operating in a competitive market. In order to minimize the risk of such turnover, the activities carried out within the organization in the form of processes are often created. As the processes are often complex, their modeling is a challenge for business analysts. To facilitate this task, it is possible to use some tools to assist analysts in their daily work. This paper combines issues in the areas of management (process approach in organizations) and Information Technology (Constraint Programming, Process Mining).

The main goal of the approach is to provide a method to generate complex process models starting from tasks and constraints obtained from the organization. The construction of the solution was preceded by the analysis of this topic.

This paper is organized as follows: Section II presents an analysis of the Business Process Management approach, including its origins, development, and current trends. The section also includes necessary information related to BPMN and process discovery. In Section III, as the next step to achieving the goal, the analysis and description of the proposed method are included. Next, a project of an IT tool, its assumptions, requirements, and architecture (Section IV) was presented, as well as the technical description of its implementation (Section V). Section VI includes the evaluation of the proposed approach, the description of the developed tool, as well as the results of its application on a set of test data. The work is finished with conclusions and a description of the possible extension of the approach (Section VII).

II. Business Process Management

This Section discusses the issues related to business processes – from their role in management, through the applied notation of their recording, to the description of their research techniques.

A. Overview

Business Process Management (BPM) [1] is one of the most common methods for improving the organization and implementation of the quality system. The ISO 9001 standard [2] introduced the obligation to apply the process approach as one of the key elements of a well-implemented, maintained, and functioning management system.

However, to talk about the process approach, let us look at the concept of a business process per se. There are many definitions of the business process. However, for this study, the definition presented in "Essential Business Process Modeling" will be adopted, where the process is described as step-by-step activities specific to the solution of various problems or business issues [3].

To answer the key question about the purpose of the process approach in enterprises, it is worth taking a closer look at the research carried out in 2009 [4], which shows the organization's goals at various stages of process development. Enterprises in the phase of introducing process-oriented approach set the implementation of the quality system and the creation of a process approach in their structures as the primary goal. In the case of enterprises from this group, the aim is usually to map processes existing in the organization, and less frequently to improve their effectiveness or implement IT tools supporting operational activities. For 38.5% of organizations in the growth phase, the key goal is to improve efficiency, and for 31% of organizations in this group, the most important goal is the development of applied IT systems, which will also improve the organization's efficiency [4]. Enterprises being in the improvement phase during the research indicated three

most important areas of application of the process approach, which are: improvement of quality, improvement of efficiency, and implementation of IT systems [4].

In the mid-nineties, Business Process Management was introduced as the next wave of approach to managing the processes in the organization. BPM postulates, inter alia, mapping, visualization, and analysis of processes in the organization. Thanks to these activities, it is possible to standardize the implemented activities, control their course, and perform much easier analysis of decision situations [5], [6].

Modeling processes in firms can be implemented using a variety of notations. Initially, many organizations used their own methods of describing and modeling processes, which, however, hindered readability and negatively affected cooperation between organizations. In response to this problem, many formal notations and languages of business process modeling were created. The most popular standards used to model business processes are the Unified Modeling Language (UML) [7] and Business Process Model and Notation (BPMN) [8].

In the last three decades, a change in the approach to information systems can be observed. Process-aware systems increasingly replace formerly used data-aware systems. To support business processes implemented within the framework of an organization, enterprise information systems must be somewhat aware of the existence of these processes and the organizational context within which they are implemented. Early examples of process-aware systems were called Workflow Management systems (WFM). In recent years, IT solutions companies have preferred a more precise term of BPM. Business Process Management systems cover a wider range than classical Workflow Management systems and do not focus only on process automation. Business Process Management systems attach more importance to supporting various forms of analysis (e.g. process simulation) and management (e.g. monitoring key performance indicators). Both Workflow Management systems and Business Process Management systems seek to support operational processes, which we refer to as workflow processes or simply workflows [9].

B. Business Process Model and Notation

The BPMN standard consists of a set of graphic elements for constructing diagrams showing components of the process and the way in which it should be executed. Graphic symbols and the way of combining them constitute the syntax of the notation and have defined semantics. In addition to the graphical representation of the model, the BPMN specification [10] also describes the format of the record in the form of XML files.

The basic subset of BPMN symbols used in the developed model generator is presented in Figure 1, its elements are described below.

- Task – atomic activity performed as part of the process which is not subject to further decomposition. It presents actions taken by the end user or software.
- Gateways – elements used to control how sequential flows separate and merge as part of the process. They can support many input and many output flows, although best practices suggest that the gateway should only perform one of these functions. Therefore, in the diagrams, the pair of gates usually serves first to separate and then connect the process flows.
  - OR-gateway – flow object used to create alternative paths within the process flow. A single process instance contains only one of the possible paths selected. This gateway is interpreted as a decision at a given point in the process. It can be understood as a question, and sequence flows from it will be associated with responses.
  - AND-gateway – flow object used to create parallel flows or synchronize them (join). Separation of the flow is not subject to any conditions, the connection, in turn, requires the completion of all input flows of the gate. A single process instance contains all possible paths associated with a given pair of parallel gates.
- Process start and end event – process start event indicates where flows begin within a given process. The process end event symbolizes the end of all flows.
- Sequence flow – indicates the order of flow objects in the process. It always has one source and one target element.

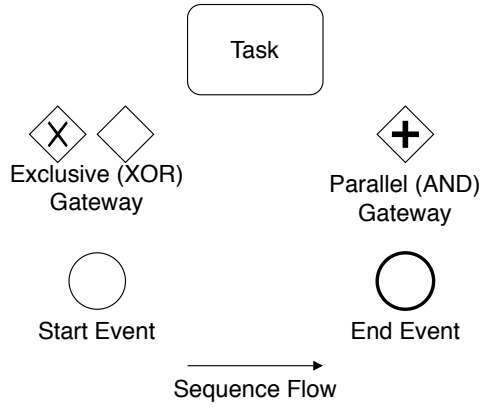

Figure 1. BPMN elements.

C. Process Discovery

Process mining is a relatively new field of research, between machine learning and data exploration on the one hand, and modeling and analysis of processes on the other. Today's information systems store huge amounts of data about activities performed in the form of event logs. The assumptions for the exploration of processes are to discover, control, and streamline real processes by extracting knowledge from read-only event logs in these systems.

In general, process mining methods are often classified into one of the following classes [11]:

- process discovery,
- conformance checking,
- process enhancement.

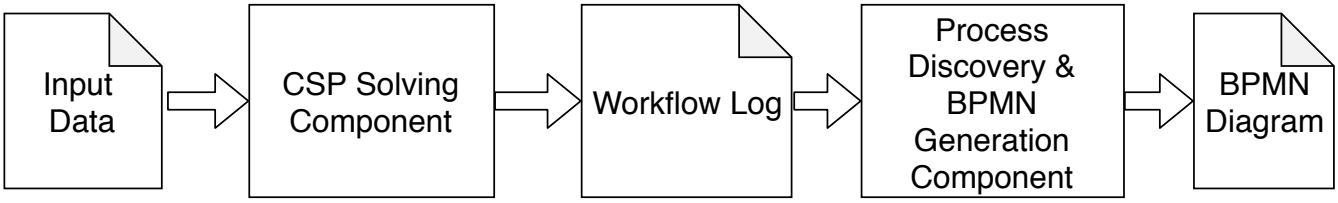

Figure 2. Overview of the process generation method.

For the purpose of this paper, our attention will be focused only on the first group – methods of discovering the process model. Discovery techniques rely on event logs and generate a process model without any known process information. An example is the α algorithm, which results in a Petri net [12] that reflects the behavior recorded in the event log. Using dedicated algorithms, Petri nets can be automatically converted to BPMN [13], [14], [15].

Example algorithms of process discovery include:

- α algorithm [16], used in the developed tool,
- Inductive Miner [17], used in the developed tool,
- Heuristics Miner [18] and its Fodina variant [19],
- Evolutionary Tree Miner [20],
- Structured Miner [21],
- Region-Based Mining [22],
- Split Miner [23].

III. Method

The process model generator presented in this paper uses the adapted method of generating process models based on the one described in the work [24]. It consists of three stages presented in Figure 2. In this Section, the formalization of the method is presented.

The first stage of the method is the preparation and provision of input data in the appropriate format. Such data can be extracted from a system-based source such as data warehouse [25] or acquired directly from business roles [24]. The approach described in this paper requires the provision of input data consisting of:

- matrix for prerequisite tasks,
- matrix of task effects,
- vector of the initial state,
- matrix of acceptable final states of the process,
- the maximum number of executions of each task.

The next stage of the method's operation is the use of a component that uses the Constraint Programming techniques, based on a model built of input data and predefined constraints. The result of its operation is an artificially created log of all possible traces of the process.

The final stage of the method is the part that explores the process (called *process exploration*) directly from the event log and creates a process representation in BPMN.

A. Expected Input Data

In our approach, $\mathbb{T}$ denotes a set of all tasks:

$$\mathbb{T} = \{\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(n)}\}.$$

Table I
Meaning of values in used structures.

Velue	M_{TC}	M_{TE}	M_{ST}	s_0
-1	not relevant	not changed	not relevant	-
0	forbidden	deleted	forbidden	forbidden
1	required	created	required	required

Tasks, or activities, are performed within the course of the process. For our method, the concept of data entity was introduced. Unlike data objects used in BPMN, data entities do not exist in the generated model, but a part of the process specification that is required in the applied method. In other words, the data entity is a variable about a simple or complex type of data that accompanies the execution of the tasks of the process. The set of all data entities is denoted by Δ:

$$\Delta = \{\delta_1, \delta_2, \ldots, \delta_m\}.$$

Cardinality of $\mathbb{T}$ and Δ is equal to n and m, respectively. For the needs of the constraint model, it is necessary to build two matrices of dimensions $n \times m$:

1) M_{TC}: for the prerequisites required for each task,
2) M_{TE}: for the effects caused by each task.

The prerequisites and effects are understood as the occurrence of the data entity before and after the task.

Additionally, assuming g as the number of allowed final states, it is necessary to define the matrix M_{ST} of dimensions $g \times m$, which describes all acceptable terminal states. The m-element vector s_0 is defined in order to give information about the presence of the data entity before the process is executed. All structures described can contain integer values from the set $\{-1, 0, 1\}$. Table I explains the meaning of values in the context of the data entity in each structure.

The last of the input structures is the n-element vector e_t containing the number of maximum executions for each task. By default, its values should be equal to 1 unless the process contains loops or tasks performed iteratively.

B. Workflow Log

Workflow log $W = \{\sigma_1, \sigma_2, \ldots, \sigma_L\}$ is a multiset of individual workflow traces σ, which can be defined as ordered sequences of activities in the course of the process: $\sigma = (\tau_1, \tau_2, \ldots, \tau_K), \tau_i \in \mathbb{T}$. Although the workflow log definition permits the appearance of identical process traces

many times, the purpose of the described method is to generate a complete log artificially. The generated log contains all acceptable process traces. Therefore, in further considerations, the multiset W will be treated as an ordinary set.

C. Constraints

For the purpose of finding a set of solutions, the concept of the process state S is introduced. It is represented by the state vector of the data entity in every step of the process. The state of the data entity s_i is the vector representing the occurrence of the data entity in i-th step of the process. The values 0 and 1 mean respectively the absence and occurrence of the data entity.

$$S = [s_0, s_1, \ldots, s_K], \text{ where } K > 0, K \in \mathbb{N},$$
$$s_i = [d_{(i,1)}, d_{(i,2)}, \ldots, d_{(i,m)}],$$
$$d_{(i,j)} \in \{0, 1\}, \text{ where } i \in \{1, \ldots, K\}, j \in \{1, \ldots, m\}.$$

Before specifying the constraints needed to generate the correct process flow log, it is necessary to define a predicate that determines whether the data vector state of the data entity s_i meets the requirements of the task to be performed:

$$sat(s_i, TC(\tau^{(i)})) \iff$$
$$\forall j = 1 \ldots m : d_{(i,j)} = TC(\tau^{(i)})_j \vee TC(\tau^{(i)})_j = -1,$$

where $TC(\tau^{(i)})$ is the i-th row of matrix M_{TC} and d_j is the j-th element of state vector s_i.

In addition, a predicate is defined, meaning that the state meets one of the allowed end states:

$$satSet(s_i, M_{ST}) \iff \exists j = 1 \ldots g : sat(s_i, M_{ST_j}),$$

where M_{ST_j} means the j-th row of admissible solution matrix.

To generate a complete workflow log W, the problem being analyzed must be modeled using constraints over variables. This concept is based on three principles:

1) Search space: all completed task sequences.
2) Decision variables: single process flow, process state matrix.
3) Variable constraints: defined by the input data as well as by the set of predefined rules.

Predefined constraints that ensure the correctness of the generated process runs are:

1) The overall limit of task executions MAX_{EX}.
2) The number of executions of each $\tau^{(i)}$ task must be less than or equal to the corresponding value in the vector of the maximum number of executions of the e_t task or the general MAX_{EX} limit.
3) Maximal length of a single workflow trace σ to $K = n \times MAX_{EX}$.
4) The input state of the first completed task is equal to s_0.
5) Every non-idle task $\tau^{(k)}$ in the i-th step changes the elements of its successor state s_{i+1}:

$$s_{i+1} = [d_{(i+1,1)}, d_{(i+1,2)}, \ldots, d_{(i+1,n)}],$$

$$d_{(i+1,j)} = \begin{cases} d_{(i,j)}, & \text{dla } TE(\tau^{(k)})_j = -1, \\ 1, & \text{dla } TE(\tau^{(k)})_j = 1, \\ 0, & \text{dla } TE(\tau^{(k)})_j = 0, \end{cases}$$

where $TE(\tau^{(k)})_j$ means the j-th element of the k-th row of matrix M_{TE}.

6) The process ends when one of the specified final states is reached.

$$satSet(s_i, M_{ST}) \iff \tau_i = \tau^{(0)},$$

where τ_i means the i-th task of a single workflow trace σ and $\tau^{(0)}$ is an idle task.

7) The last process state s_K satisfies one of the admissible goal states M_{ST}:

$$satSet(s_k, M_{ST}).$$

8) The task can be performed only if the current state meets its initial conditions:

$$\tau_i = \tau^{(k)} \iff sat(s_i, TC(\tau^{(k)})).$$

The program built on the basis of the above-mentioned constraints and input data is performed by the system solving the problems of meeting restrictions (called *solver*). Executing a program by a solver to find all solutions results in an artificially generated process log W, needed in the next stage of generating the process model.

D. Generating a BPMN model from a workflow log

In the work [24], two approaches to building a process model based on the delivered process log are listed. The first approach involves the use of process discovery algorithms from the delivered process log. These algorithms were described in more detail in Section II-C. The result of their operation is the process model in the form of a Petri net. One of the methods of converting Petri nets to BPMN was presented in [14]. The second approach is a process composition method based on activity graphs that does not require conversion of the Petri net to the BPMN form because the BPMN composition is directly from the artificially generated workflow log.

In the process model generator described in this paper, the implemented implementations of process discovery and conversion algorithms for BPMN contained in library *Numberjack* were used. The tool uses the α and Inductive-Miner algorithms described in [26] and [17] respectively.

IV. Tool

This section presents the tool design – from specifying functional and non-functional requirements, by specifying the input data format specification, to the architecture description of the developed process model generator.

At the initial stages of work, functional (Section IV-A) and non-functional requirements (Section IV-B) were defined.

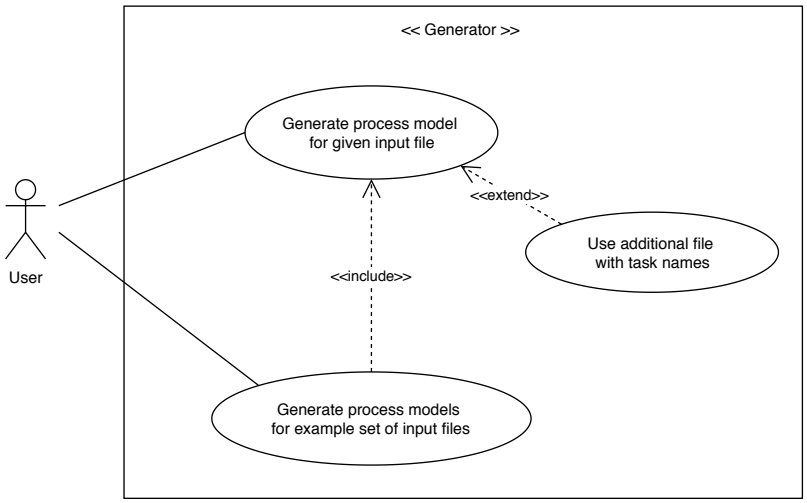

Figure 3. Use case diagram.

A. Functional Requirements

1) The tool should accept a set of input data in a specified format at the input.
2) The tool should optionally accept a file with the specification of task names at the input.
3) The tool as a result of the action should generate:
   a) BPMN diagrams in graphic form,
   b) BPMN diagrams in XML format according to the standard,
   c) artificially generated process log,
   d) Petri net diagrams resulting from the operation of process discovery algorithms.

Functional requirements are presented in the use case diagram (Figure 3).

B. Non-functional Requirements

1) The tool is distributed in an easy-to-use form, in particular without having to manually install all dependent libraries.
2) The tool is implemented with division into independent modules so that it can be further expanded.

C. Input File Format

The input tool accepts text files with the input data value definition described in Section III-A. The file should contain in sequence:

- initial state vector,
- matrix of task prerequisites,
- task results matrix,
- matrix of allowed final states,
- vector of the possible number of task executions.

The vectors and matrices should be separated from each other by at least one empty line. The line in the file may contain a comment beginning with the # character. Comments are ignored by the input data parser. The values of subsequent elements of the introduced vectors and matrices are separated by commas. An example input file is shown in Listing 1.

Listing 1. Example input file of the tool.

```
# s_0
 0, 1, 0, 0

# m_tc
 0,  1,  0,  0
 0,  1,  1, -1
 0,  1,  1,  0

# m_te
-1, -1,  1, -1
 1, -1, -1, -1
-1, -1, -1,  1

# m_st
 1, -1, -1, -1

# e_t
 2, 2, 2
```

D. Architecture

While working on the tool, three functional areas were identified as separate modules. The modular division of the tool is aimed at introducing a clear structure of responsibility for individual parts, as well as facilitating further development by providing other implementations. The identified software modules are:

1) Parser module for input files.
2) Log module of the process log generator (based on programming techniques with limitations).
3) Module for process discovery and generation of process model diagrams.

The modules are presented in the component diagram (Figure 4). Coordination of the use of these modules by the tool is presented in the sequence diagram (Figure 5).

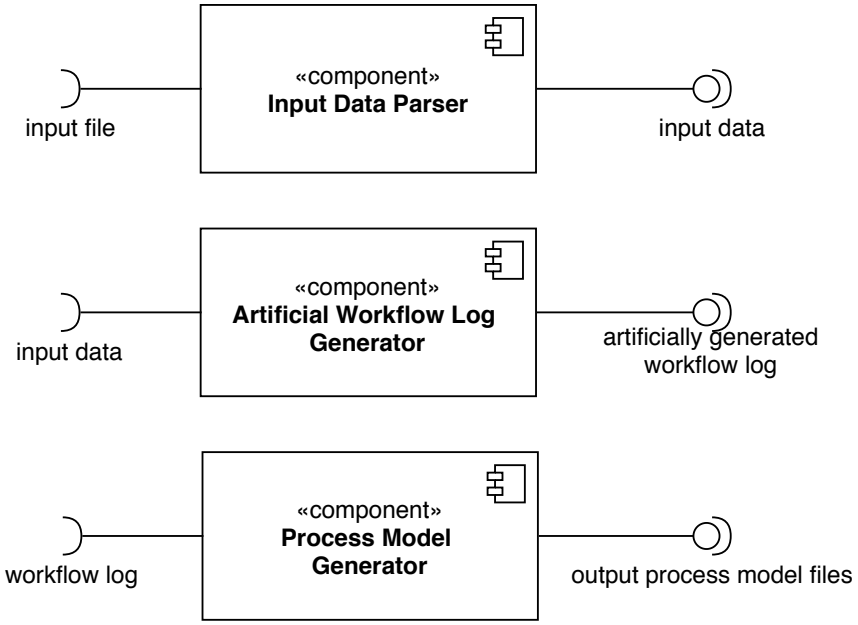

Figure 4. Components diagram.

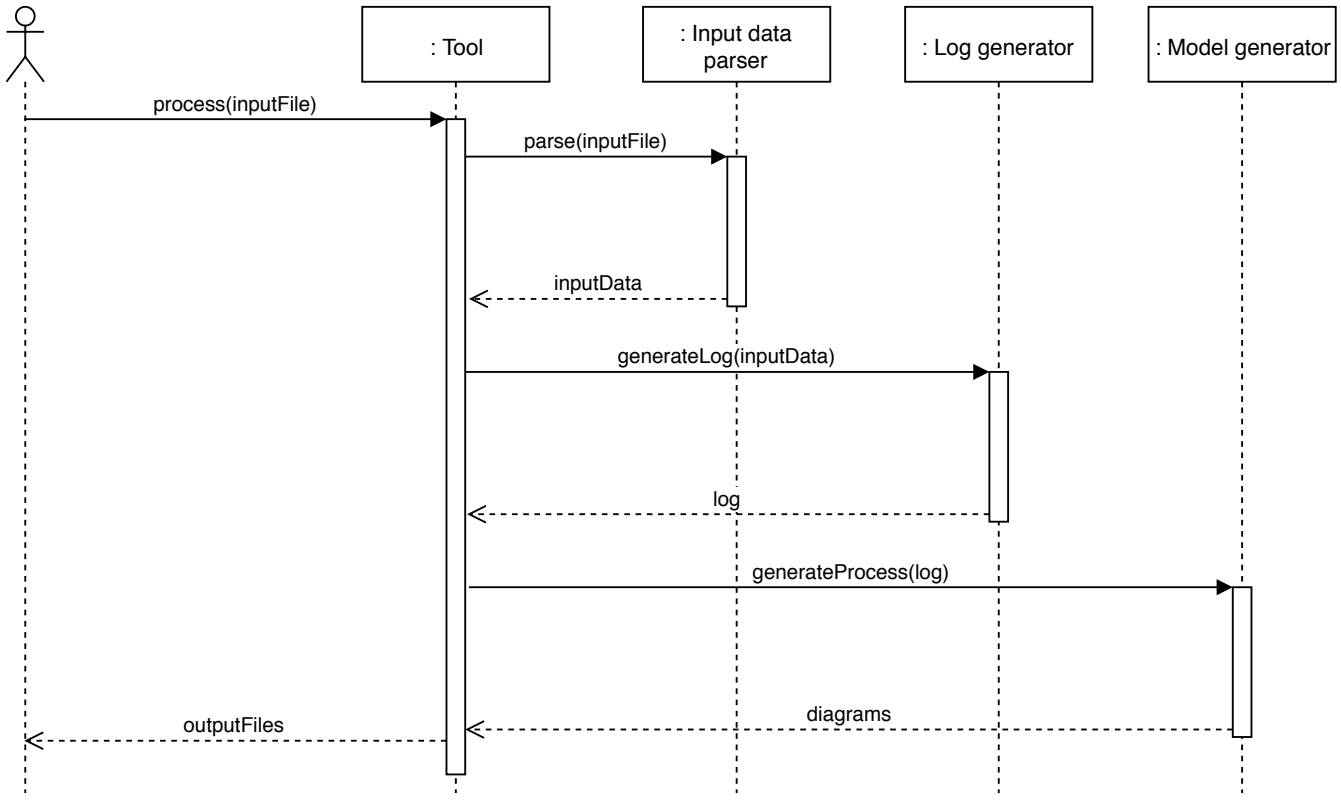

Figure 5. Sequence diagram of model generation.

V. Implementation

In this section, we present the details of the implementation of the process model generator.

A. Constraint Programming

Constraint Programming is a technique for solving the problems of satisfying constraints. These problems can be defined using variables that take values from their domains and the constraints over variables. The solution to the problem is a set of variable value assignments that meet the given constraints [27].

In addition, the problems expressed in Constraint Programming languages are characterized by declarativeness, i.e. a description of the problem becomes a program solving this problem [28].

The Constraint Programming technique can be used in imperative languages by means of libraries that allow building a problem model (variables and constraints) with the help of structures appropriate to a given language. These libraries can themselves implement solvers or provide an interface to solvers implemented in other languages. Examples of widely used solvers are: CP-SAT Solver from the Google OR-Tools package, Gecode, Mistral, Mistral2, ILOG Solver.

B. Used Techniques

Python was chosen as the implementation language of the developed tool. It is a popular language in the academic environment, as well as widely used in the IT industry. Support for programming with restrictions is ensured by the library *Numberjack* (https://github.com/eomahony/Numberjack), which allows for high-level modeling of problems and the use of several solvers. Discovering the process is carried out using the library *pm4py* (http://pm4py.pads.rwth-aachen.de/), which provides the implementation of the algorithms: $alpha$ and Inductive Miner. It enables the presentation of discovered processes in the form of Petri nets, as well as their conversion to the BPMN diagram in XML and graphical format. Functions related to BPMN diagram support are currently in the implementation phase (they are not shared with a stable version of the library) and have been taken from the appropriate branch of the library version control repository. The result of their use is described in more detail in the Section VII.

For the purpose of easily recreating the entire working environment of the tool and simplifying its use on other computers, the *Docker* software (https://www.docker.com/) used for virtualization at the operating system level was used. The application image, containing the generator of process models and the configured environment for its launch and proper operation, was prepared.

VI. Evaluation

As part of the research, tests were carried out on synthetic examples. Simple processes have been selected that contain:

- a single task,
- b sequential submission of two tasks,
- c two tasks covered by the XOR-gateway,
- d two tasks performed concurrently,
- e two tasks covered by XOR-gateway preceded and completed by one task,
- f two tasks performed concurrently preceded and ended with one task,
- g three tasks performed sequentially, in which the middle one is optional.

They are presented in Figure 6. For the needs of the tests, input data corresponding to these processes was prepared.

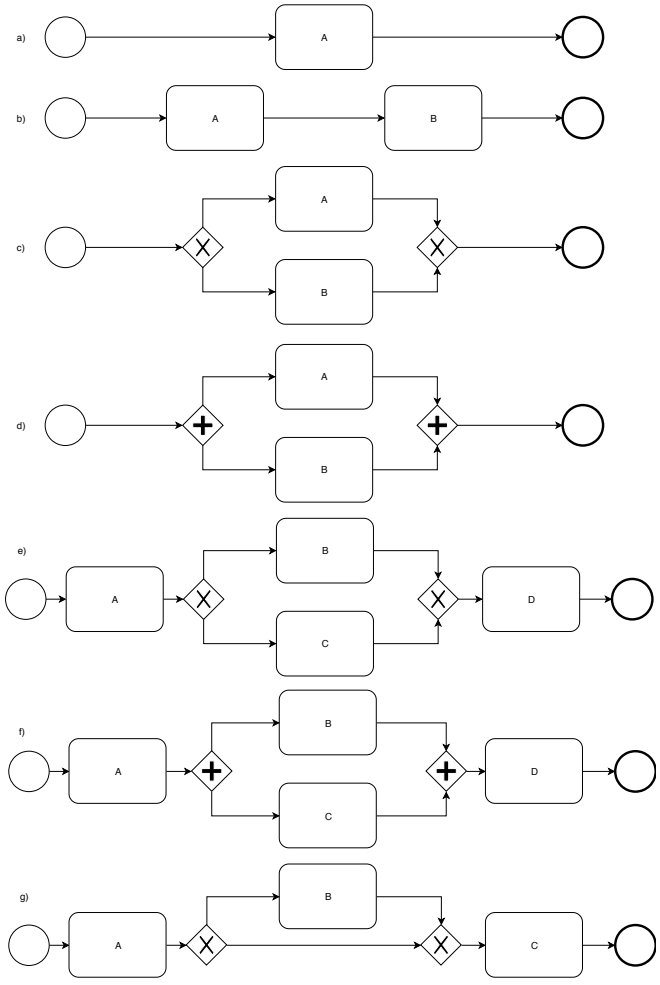

Figure 6. Example test cases in BPMN.

Analysis of test results was divided into two parts. The first one included verification of the correctness of the generated process logs. The second one focused on checking the generated model diagrams.

A. Generated Workflow Logs

The first of the test models – a process containing a single task – has only one possible trace consisting of this task. Also, the second test model, which is the composition of two tasks, should generate one pass. The generated process logs (respectively `"A"` and `"A""B"` confirm the correctness of this stage of the tool for given models.

In the case of two successive process models (two tasks connected with the OR gate, two tasks performed concurrently), we expect logs consisting of two process runs. The correct behavior of the tool was also found here - logs were created with the forms `"A","B"` and `"A""B","B""A"`.

The next two test cases are the extension of the previous ones by adding tasks at the beginning and end of the process. Here too, the proper generation of logs has been observed (`"A""B""D","A""C""D"` and `"A""B""C""D","A""C""B""D"`).

The last example (optional execution of the task in the middle of the process) gives the tool the expected log in the form `"A""B""C","A""C"`.

The above analysis confirms the correctness of the method used to generate an artificial process log for each of the test cases.

B. Generated Process Models

The result of the comparison of generated process model diagrams in the form of BPMN for test cases is shown in Table II.

The following deviations from the expected results were found during the analysis:

1) algorithm α for example c) generated BPMN diagram without XOR-gateways (Fig. 7),
2) Inductive Miner for example c) generated a BPMN diagram containing doubled XOR-gateways (Fig 8),
3) algorithm α for example d) generated a BPMN diagram without parallel gateways (Fig. 9),
4) algorithm α for example f) generated a BPMN diagram not containing initial and final tasks (Fig. 10),
5) algorithm α for example g) generated a BPMN diagram without the first task and with the wrong XOR-gateway instead of a parallel one (Fig. 11).

Table II
GENERATED MODELS AND THE EXPECTED RESULTS.

Test case	α Algorithm	Inductive Miner
single task (a)	correct	correct
two tasks (b)	correct	correct
XOR-gateway (c)	incorrect	incorrect
parallel gateway (d)	incorrect	correct
XOR-gateway in a process (e)	correct	correct
parallel gateway in a process (f)	incorrect	correct
optional task (g)	incorrect	correct

C. Example

Apart from test cases, the tool was also evaluated on the basis of a more complicated example – the process of opening a bank account. Synthesis of the input file was made on the basis of the process initially presented in [29]. The results of BPMN generation using the Inductive Miner algorithm are shown in Figure 12.

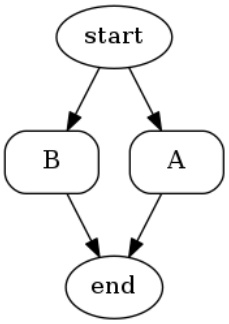

Figure 7. Diagram generated by the α algorithm for case c)

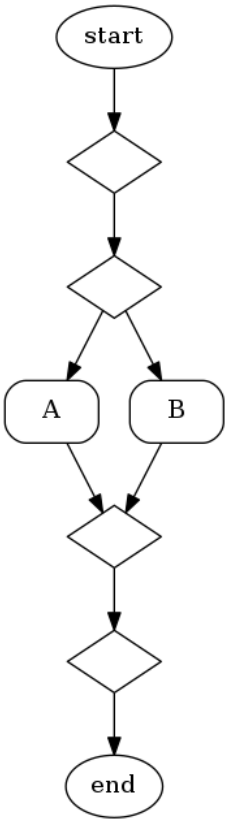

Figure 8. Diagram generated by Inductive Miner for case c)

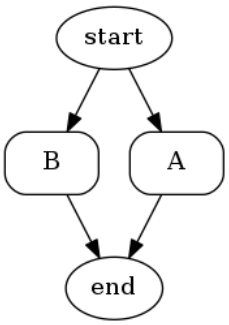

Figure 9. Diagram generated by α algorithm for case d)

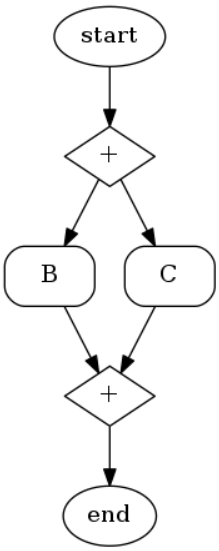

Figure 10. Diagram generated by α algorithm for case f)

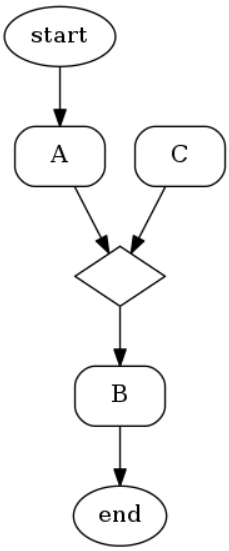

Figure 11. Diagram generated by α algorithm for case g)

VII. Conclusions and Future Works

The works on the generator of process models presented in this paper are topped off with half-hearted success. The module for generating an artificial log of the process flow provides the correct results. They can be the basis for discovering the process. However, the process discovery and BPMN diagram generation module generates diagrams far from expectations. Inconsistencies appear at the stage of testing the test cases. If the observed behavior is recorded in an event log, it is possible to repair such a model [30]. However, our goal is to provide a prototype model based on the provided input data. Thus, in order to accurately diagnose the reason for this behavior of the module, a broader review of process discovery algorithms

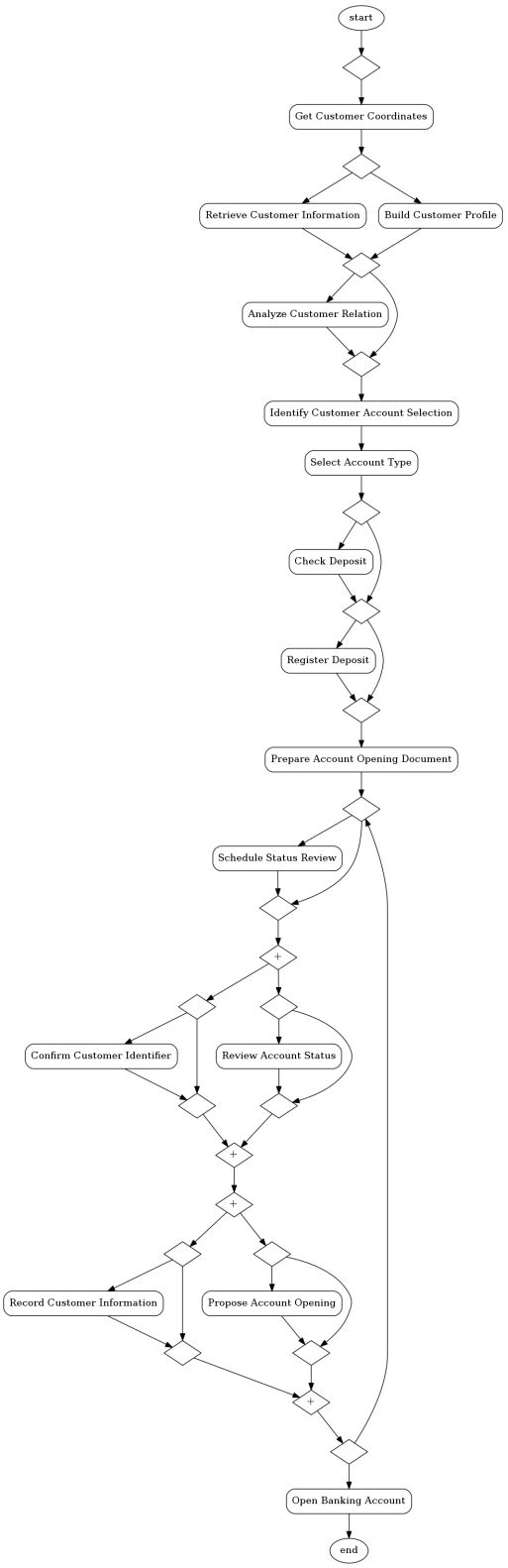

Figure 12. Diagram generated by Inductive Miner for the example process of bank account opening.

and a comparison of their properties should be made. The implementation of the Petri net conversion algorithm to BPMN provided in the experimental branch of *pm3py* can also be one of the causes of the problem.

The possibilities of the process model generator extensions are as follows:

- The use of other process discovery algorithms mentioned in the section II-C, in particular the implementation of the process composition method based on the activity graphs described in work [24]. The use of this approach should give better results during the process discovery phase from the artificially generated log flow of the process than the α and Inductive-Miner algorithms used.
- Performing a GUI facilitating the input of data, or enabling cooperation with the organization's business roles in order to collect information about tasks, data entities and their relationships.
- Improving the layouting of the generated BPMN diagrams, arranging elements on the diagram in a way similar to how they are visualized in commercial tools.
- Extension of the tool to include information about various departments within the organization, linking them to tasks and extending the generated model with pools and lanes.
- Adding the possibility of exporting the generated process log to the standardized event log format. The XES format is the standard for text-event logs for further analysis using tools that implement the process discovery functions (for example, the ProM framework [31]).

REFERENCES

[1] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers *et al.*, *Fundamentals of business process management*. Springer, 2013, vol. 1.

[2] V. Jovanovic and D. Shoemaker, "Iso 9001 standard and software quality improvement," *Benchmarking for Quality Management & Technology*, vol. 4, no. 2, pp. 148–159, 1997.

[3] M. Havey, *Essential business process modeling*. O'Reilly Media, Inc., 2005.

[4] W. Cieśliński, "Procesowa orientacja przedsiębiorstw: wyniki badań empirycznych," *Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu*, no. 52 Podejście procesowe w organizacjach, pp. 41–48, 2009.

[5] S. Lusk, S. Paley, and A. Spanyi, "The evolution of business process management as a professional discipline," *BP Trends*, vol. 20, pp. 1–9, 2005.

[6] E. Kucharska, "Heuristic method for decision-making in common scheduling problems," *Applied Sciences*, vol. 7, no. 10, p. 1073, 2017.

[7] K. Kluza, P. Wiśniewski, K. Jobczyk, A. Ligęza, and A. Suchenia (Mroczek), "Comparison of selected modeling notations for process, decision and system modeling," in *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 11. IEEE, 2017, pp. 1095–1098.

[8] P. Pasamonik, "Modelowanie procesów biznesowych zorientowane na czynności," *Zeszyty Naukowe Wyższej Szkoły Informatyki*, vol. 9, no. 2, pp. 102–116, 2010.

[9] W. M. van der Aalst, "Process-aware information systems: Lessons to be learned from process mining," in *Transactions on petri nets and other models of concurrency II*. Springer, 2009, pp. 1–26.

[10] OMG. (2011) Business process model and notation. [Online]. Available: https://www.omg.org/spec/BPMN/2.0

[11] W. M. van der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs *et al.*, "Process mining manifesto," in *International Conference on Business Process Management*. Springer, 2011, pp. 169–194.

[12] M. Szpyrka, *Sieci Petriego w modelowaniu i analizie systemów współbieżnych*. Wydawnictwa Naukowo-Techniczne, 2008.

[13] A. A. Kalenkova, M. De Leoni, and W. M. van der Aalst, "Discovering, analyzing and enhancing BPMN models using ProM," in *BPM (Demos)*, 2014, p. 36.

[14] A. A. Kalenkova, W. M. van der Aalst, I. A. Lomazova, and V. A. Rubin, "Process mining using BPMN: relating event logs and process models," *Software & Systems Modeling*, vol. 16, no. 4, pp. 1019–1048, 2017.

[15] A. Kalenkova, A. Burattin, M. de Leoni, W. van der Aalst, and A. Sperduti, "Discovering high-level BPMN process models from event data," *Business Process Management Journal*, 2018.

[16] W. M. van der Aalst, A. Weijters, and L. Maruster, "Workflow mining: Which processes can be rediscovered," BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven, Tech. Rep., 2002.

[17] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Scalable process discovery with guarantees," in *International Conference on Enterprise, Business-Process and Information Systems Modeling*. Springer, 2015, pp. 85–101.

[18] A. Weijters and J. Ribeiro, "Flexible heuristics miner (fhm)," in *2011 IEEE symposium on computational intelligence and data mining (CIDM)*. IEEE, 2011, pp. 310–317.

[19] S. K. van den Broucke and J. De Weerdt, "Fodina: a robust and flexible heuristic process discovery technique," *decision support systems*, vol. 100, pp. 109–118, 2017.

[20] J. C. Buijs, B. F. Van Dongen, and W. M. van der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*. Springer, 2012, pp. 305–322.

[21] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and G. Bruno, "Automated discovery of structured process models: Discover structured vs. discover and structure," in *International Conference on Conceptual Modeling*. Springer, 2016, pp. 313–329.

[22] W. M. van der Aalst, *Process mining: discovery, conformance and enhancement of business processes*. Springer, 2011, vol. 2.

[23] A. Augusto, R. Conforti, M. Dumas, and M. La Rosa, "Split miner: Discovering accurate and simple business process models from event logs," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 1–10.

[24] P. Wiśniewski, K. Kluza, and A. Ligęza, "An approach to participatory business process modeling: BPMN model generation using constraint programming and graph composition," *Applied Sciences*, vol. 8, no. 9, p. 1428, 2018.

[25] M. L. Owoc *et al.*, "Benefits of knowledge acquisition systems for management. an empirical study," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2015, pp. 1691–1698.

[26] W. M. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[27] E. Tsang, *Foundations of constraint satisfaction: the classic text*. BoD–Books on Demand, 2014.

[28] A. Niederliński, *Programowanie w logice z ograniczeniami: Łagodne wprowadzenie dla platformy ECLiPSe*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2010.

[29] P. Wiśniewski, K. Kluza, M. Ślażyński, and A. Ligęza, "Constraint-based composition of business process models," in *Business Process Management Workshops*, E. Teniente and M. Weidlich, Eds. Cham: Springer International Publishing, 2018, pp. 133–141.

[30] A. A. Cervantes, N. R. van Beest, M. La Rosa, M. Dumas, and L. García-Bañuelos, "Interactive and incremental business process model repair," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2017, pp. 53–74.

[31] B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. Van Der Aalst, "The ProM framework: A new era in process mining tool support," in *International conference on application and theory of petri nets*. Springer, 2005, pp. 444–454.