Special Issue
of the Infocommunication Journal

Application of Process Discovery Methods for Learning Process Modeling

Erika Baksáné Varga, and Attila Baksa

Abstract—**Process mining encompasses a suite of techniques aimed at analyzing event data to gain insights and improve operational processes. One way of achieving this is to discover the driving process of the activities that occurred in a system. Technically, process discovery algorithms are used to transform an event log into a process model which is representative of the activities registered in the given system. This study explores the application of process discovery methods to better understand the learning processes in an introductory programming course for first-year Computer Science BSc students. A total of 52 practical problems were assigned as out-of-class activities via GitHub Classroom, resulting in 2789 commits from 59 students. These commits, along with the students' exam grades, were recorded in an object-centric event log, subsequently converted into a case-based log for analysis using the PM4Py program library.**

The study had two primary goals: first, to identify the characteristics of successful learning strategies by comparing process models of students who passed versus those who failed the programming exam; and second, to identify bottlenecks that hindered student progress. By employing the Heuristic Miner and Inductive Miner algorithms, we developed and contrasted learning process models, revealing significant patterns and obstacles within the educational process. The findings provide valuable insights into the factors that contribute to effective learning and suggest areas for enhancing our teaching methodologies.

Index Terms—**Process discovery, Learning process modeling, Application of PM4Py package**

I. Introduction

WITHIN the expansive domain of education, the significance of understanding how individuals learn and retain information cannot be overstated. This comprehension is critical for the development and implementation of instructional methods that are truly effective [1]. Historically, traditional teaching approaches have tended to adopt a one-size-fits-all methodology, which unfortunately overlooks the reality that each learner possesses unique preferences and cognitive processes. This oversight can result in learning experiences that are less than optimal, potentially hindering the development of individuals' skills and overall educational performance [2]. Recognizing the diversity in learning styles is thus essential for improving educational outcomes.

To enhance educational practices, it is imperative to identify and understand the processes that lead to successful learning. In this context, the field of process discovery emerges as a promising approach. Process discovery involves uncovering and analyzing the patterns and strategies employed by learners, providing insights that can be used to inform and improve teaching methodologies. By applying process discovery methods specifically to the study of learning procedures in programming, educators can gain a deeper understanding of student behavior. This approach allows instructors to identify and highlight problems that require additional iterations to be resolved, ultimately facilitating a more tailored and effective educational experience.

A. Motivation and Goals

The motivation behind this study lies in the need to better understand the learning behaviors of students in programming education. Programming presents unique challenges, such as mastering problem-solving and abstract thinking, which often leads to high variability in student performance. This study aims to explore how process discovery methods can reveal patterns of effective learning strategies and the obstacles faced by struggling students. By identifying these patterns, educators can improve their instructional methods, offering better support to students.

The study has two main goals: first, to compare the learning processes of students who passed versus those who failed the programming course, using process discovery techniques to reveal key differences in learning behaviors; and second, to identify bottlenecks that hinder student progress. These insights can guide future teaching practices by highlighting the areas where students struggle the most, enabling educators to adapt their teaching strategies accordingly.

B. Challenges and Novelty

Analyzing learning behaviors through event logs presents the challenge of handling diverse and noisy data, especially in educational settings where individual learning paths vary widely. This paper addresses these challenges by applying process discovery techniques, traditionally used in business process analysis, to model and analyze student learning in programming courses. The novelty lies in applying process mining tools – specifically *Heuristic Miner* and *Inductive Miner* – to educational data, which enables a more detailed and dynamic understanding of how students engage with programming tasks.

By capturing event log data from GitHub Classroom, this study provides a unique and fine-grained view of student behavior, going beyond traditional performance assessments. It is one of the first studies to apply process discovery techniques to programming education, thus offering new insights into student engagement and problem-solving strategies.

Authors are working at University of Miskolc, Miskolc-Egyetemváros, Hungary
(E-mail: erika.b.varga@uni-miskolc.hu, attila.baksa@uni-miskolc.hu)
Manuscript received May ..., 2024; revised ..., 2024.

DOI: 10.36244/ICJ.2025.5.4

C. Paper Structure

The paper is structured as follows: Section II outlines the research objectives. Section III reviews related work on the use of GitHub in education and the application of process mining in educational research. Section IV discusses process discovery algorithms, comparing them and selecting the most appropriate for the study. Section V focuses on data modeling and analysis, detailing the data collection process, the OCEL schema used for structuring the event log, and an analysis of the collected data, including key statistics. Section VI presents the results of learning process modeling, while Section VII identifies bottleneck problems that hinder student progress. Finally, Section VIII concludes the paper by summarizing the findings and discussing their implications for educational practices.

II. Research Objectives

Building on the challenges and motivations outlined in the introduction, this study seeks to achieve two main objectives:

1) Create a learning process model that is representative of individual cases to identify bottleneck problems that hinder students' progress.
2) Produce a learning process model of successful students to serve as a showcase for other students.

By focusing on these objectives, this research seeks to provide educators with actionable insights into the learning processes, enabling them to tailor their instructional strategies to better meet the diverse needs of their students. This, in turn, can lead to improved educational outcomes and more effective learning experiences.

III. Related Works

A. GitHub in Teaching Programming

There is a growing need for automated code assessment systems in computer science education due to the increasing number of learners and the limited availability of teaching staff. These systems aim to address the challenges of grading a large volume of code submissions. They help instructors save time, provide timely feedback to learners, and support the learning process. These systems target various types of errors in programming assignments, such as syntax, runtime, logic errors, and code quality issues, and may also address plagiarism concerns through similarity analysis [4]. Some tools use continuous integration for immediate feedback, while others perform symbolic executions and unit test assessments.

Within the array of tools, GitHub is a widely-used software development platform that originally supported version control, collaborative development, and project hosting. It is utilized by many businesses, organizations, and educators. In the context of education, GitHub has gained popularity in programming classrooms, with around $18,000$ educators incorporating it [5]. GitHub in education serves various purposes such as submitting assignments, collaborating on group projects, and receiving feedback. More recently, with GitHub Actions, teachers can automate the testing of code submissions.

The study by Hsing and Gennarelli [6] explores how the implementation of GitHub in programming classrooms affects students' learning outcomes and experiences. The researchers surveyed $7\,530$ students and 300 educators from classrooms using GitHub and classrooms not using GitHub. The findings indicate that incorporating GitHub in programming education yields several benefits, including enhancing students' familiarity with industry tools, facilitating collaboration and teamwork, boosting engagement, and fostering a sense of belonging in the classroom and within the field.

A more recent study [7] summarizes the key lessons learned when using GitHub in the classroom. First of all, the authors recommend providing proper instructions, compatible with students' prior experience, on how to use Git to gain the most benefits from system use. For undergraduate courses, GitHub Classroom is a better choice than GitHub, as GitHub Classroom simplifies the educational use of GitHub. They also found that custom offline automated systems are more effective for assessing students' assignments than using GitHub Actions.

In [8], the authors present two distinct approaches for automatic C code assessment in programming education: one is a custom-designed web-based tool, while the other involves using GitHub Actions and GitHub Classroom. The web-based tool offers a graphical user interface in Hungarian and assesses code based on various criteria, including syntax, behavior, and code quality metrics. The GitHub-based system employs repositories and automation to evaluate student assignments, providing immediate feedback and the ability to track students' progress and activities. The authors conducted experiments at the University of Miskolc to compare the effectiveness of both systems in engaging students and improving their coding skills. The results show that both systems were beneficial for student engagement and learning, with the GitHub-based system offering more comprehensive tracking capabilities and integration with the software engineering community's practices.

B. Mining Educational Data

The field of educational data mining (EDM) has received significant attention in recent years. Researchers and educators are increasingly seeking to utilize the extensive data generated within educational settings to gain valuable insights and drive substantial improvements. This goal can be achieved by applying data mining techniques, which offer a robust set of tools capable of uncovering hidden patterns, associations, and anomalies within educational data. By understanding these patterns, educators can develop targeted interventions and personalized learning strategies that address the specific needs of individual students [9], [10].

Building on the foundation established by educational data mining, the field of educational process mining has emerged as a specialized discipline dedicated to extracting meaningful insights from event logs. Process mining techniques [3] are particularly effective for analyzing the sequences of activities performed by students, which are captured in event logs generated from interactions with learning management systems, online courses, and other educational technologies. By applying process mining techniques to these event logs,

researchers can uncover the actual learning paths taken by students, identify critical points where students struggle or deviations from expected or "ideal" paths, and explore how these deviations impact learning outcomes [11], [12]. These valuable insights allow for educators to tailor their teaching strategies.

In summary, while educational data mining provides broad insights into various types of educational data, educational process mining focuses on the detailed analysis of event logs to uncover the actual learning paths taken by students. The integration of the two fields provides a powerful means to enhance the understanding and improvement of educational processes. This paper aims to contribute to this growing field by exploring how process mining algorithms can be applied to educational data, ultimately paving the way for more effective and personalized education.

C. Event Log Standards

Process mining algorithms work with event logs produced by various information systems. Consequently, these logs can appear in numerous formats and instantiations. Each system architecture with a logging mechanism has historically developed its own solution for recording events. The initial effort to standardize event logs was MXML (Mining eXtensible Markup Language), which organized timestamps, resources, and transactions in a uniform format. This was followed by the introduction of the XES (eXtensible Event Stream) standard, which was published in 2016 as the IEEE 1849-2016 Standard for eXtensible Event Stream.

Despite its comprehensive approach, the XES standard encounters difficulties when handling object-centric data (e.g., database tables) due to the complexities of one-to-many and many-to-many relationships. In response to these challenges, a new method was proposed to extract, transform, and store object-centric data, resulting in the Object-Centric Event Log (OCEL) Standard which was released in 2021 [13].

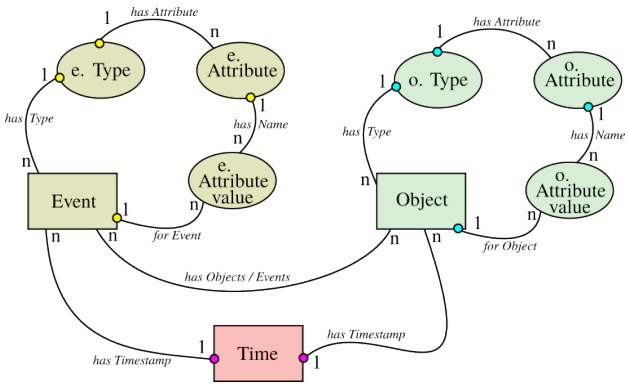

Fig. 1. OCEL schema with $n - n$ Events and Objects connections

OCEL 2.0 represents a modernized approach to capturing and analyzing event data in complex information systems [14]. Traditional event logs typically record sequences of events related to single entities, such as customer orders or individual transactions. However, many real-world processes involve multiple interacting objects, such as customers, orders, products, and payments. OCEL 2.0 addresses this complexity by enabling a richer and more holistic representation of event data (see Figure 1). The key features of OCEL 2.0 [14] are as follows.

- *Multi-Object Support*
  Multiple Objects Per Event: Unlike traditional event logs that often focus on a single case or entity, OCEL 2.0 allows each event to be associated with multiple objects of different types. This capability is crucial for accurately modeling complex processes where events involve interactions between several objects.
- *Enhanced Data Model*
  Object Types and Attributes: OCEL 2.0 defines various object types and allows each object type to have its attributes. This structured approach facilitates detailed analysis and better understanding of the relationships between different objects.
- *Improved Analysis Capabilities*
  Holistic Process Views: By considering the interactions between multiple objects, OCEL 2.0 enables more comprehensive process mining and analysis. This leads to deeper insights into process performance, bottlenecks, and areas for improvement. Multi-Perspective Analysis: Analysts can explore processes from different perspectives, such as the lifecycle of individual objects or the interactions between specific types of objects.
- *Interoperability and Standards*
  Standardized Format: OCEL 2.0 promotes the use of standardized formats for event logs, ensuring compatibility and interoperability between different tools and systems used in process mining and analysis.
- *Tool Support*
  Integration with Process Mining Tools: OCEL 2.0 is supported by various process mining tools, enabling users to leverage its advanced features for process discovery, conformance checking, and performance analysis.

Overall, OCEL 2.0 represents a significant advancement in the field of process mining and event log analysis, allowing for a more nuanced and complete understanding of the processes.

IV. Methods

In this research, we utilized the PM4Py Python program library [15], [16], a sophisticated tool developed by the Fraunhofer Institute, to implement a variety of algorithms and services associated with process mining. PM4Py stands out for its comprehensive suite of functionalities that facilitate the analysis, visualization, and discovery of process models from event logs, making it an ideal choice for our investigations.

Our primary focus was on process discovery algorithms, which are essential for revealing the underlying process models that govern the sequences of events captured in logs. An event log is defined as

$$\mathcal{L} = \{< A, B, C, D >, < A, C, D >, < A, B, D >\} \quad (1)$$

where A, B, C, and D are the events in the log, and $\mathcal{L}$ denotes the possible order of the events. To this end, we

conducted an in-depth examination of three prominent process discovery methods: the Alpha Miner, the Inductive Miner, and the Heuristic Miner [17]. Each of these methods brings distinct methodologies and advantages to the table, allowing us to explore different dimensions of process discovery.

The *Alpha Miner* algorithm [22], one of the earliest and most fundamental process discovery techniques, operates by identifying and interpreting patterns of event sequences to construct a process model. This algorithm works by examining the direct succession of events and determining causal relationships, which it then uses to build a Petri net representation of the process. In Eq. (1) these two cases are possible (which is a sequence pattern):

$$A \rightarrow B \quad \text{or} \quad A \rightarrow B \tag{2}$$

While the Alpha Miner is effective in providing a basic structure of the process, it has limitations when dealing with noise, infrequent paths, and complex dependencies within the event log.

In contrast, the *Inductive Miner* algorithm [19], [20] offers a more advanced and robust approach. It employs a recursive technique that divides the event log into smaller parts, constructs models for these parts, and then combines them to form a comprehensive process model. A directly follows graph is represented mathematically by

$$\mathcal{G}(\mathcal{L}) = (A_L, \rightarrow_L, A_L^s, A_L^e) \tag{3}$$

where A_L is an event in the log, $\rightarrow_L$ means an edge between two events (a directly follows relation), A_L^s is the start and A_L^e is the end event.

This method is particularly suitable for handling noisy and complex data, producing more precise and comprehensible models. Its ability to generate hierarchical models makes it especially useful for understanding complex processes with multiple layers of activities.

The *Heuristic Miner* algorithm [21] takes a different approach by focusing on the discovery of frequent patterns and significant relationships within the event log. This algorithm uses statistical measures to assess the frequency and significance of event connections, thereby identifying the most prominent pathways in the process. The Heuristic Miner is particularly valuable for analyzing real-world data that may contain deviations, exceptions, and variations, providing a realistic and practical view of the process.

In our investigations we used a real-world event log recorded by Github, which contains few frequent patterns and a multitude of unique cases. For this reason, the Alpha Miner was deemed insufficient for our needs due to its sensitivity to noise and difficulty in handling complex dependencies. Instead, we opted to use the Inductive Miner and Heuristics Miner algorithms.

V. Data Modeling and Analysis

Our experiment covered 52 practical problems propagated using GitHub Classroom within an introductory programming course. These assignments are designed for out-of-classroom practice to help novice programmers in developing their skills in C programming. The tasks are categorized into 8 groups according to the related topics as listed in Table I. The groups represent the level of the acquired skills as the tasks are designed to incrementally build upon one another. The 1st level is completed if the tasks from the first 3 groups are solved. The 2nd level is done after completing tasks from groups 4 and 5. The next level contains tasks from groups 6 and 7, and the top level is represented by the tasks of group 8. Students were asked to complete tasks in the order of these levels, and they were free to choose the tasks to solve and their order within the topic groups. We encouraged them to complete as many assignments as possible and try to deliver more than one acceptable solution for the same problem.

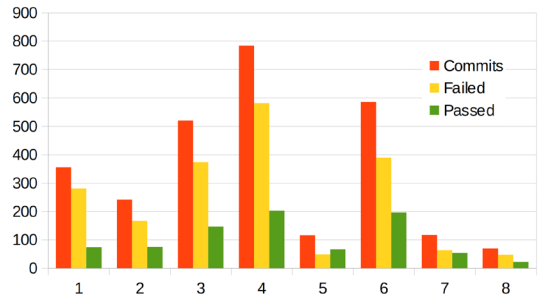

Fig. 2. Commit statistics for tasks groups

In the present study, 59 students registered for GitHub Classroom to access these assignments as extracurricular activities. They collectively produced 2,789 commit events, which were automatically tested by GitHub Actions to provide prompt evaluations as either failed (1,951 commits) or passed (838 commits). The statistics of the executed commits are shown in Figure 2. These data are recorded and stored in an event log, with each entry containing the GitHub identifier of the student (resource), the name of the assignment (activity), the time of the commit event (timestamp), and its result (conclusion). The details of the assignments include their topic group and level. Student data come from the university's learning management system, where their mid-semester performance (signature) and final grade (grade) have been recorded. Since all three sources are utilized to address our research objectives, the data model is described by the OCEL schema in Figure 5.

```
<object>
   <string key="id" value="student1"/>
   <string key="type" value="user"/>
   <list key="ovmap">
     <string key="signature" value="1"/>
     <string key="grade" value="2"/>
   </list>
 </object>
```

```
<object>
   <string key="id" value="1_06"/>
   <string key="type" value="task"/>
   <list key="ovmap">
     <string key="topic" value="1"/>
     <string key="level" value="1"/>
   </list>
 </object>
```

Fig. 3. Objects data in XML format

TABLE I
COMMIT STATISTICS FOR TASK GROUPS

Task group	Programming topic	Level	Num. of tasks	Commits	Failed	Passed	Success rate(%)
1	Use of variables, basic data types, arithmetic operators, and built-in mathematical and input/output functions	1	10	355	281	74	20.85
2	Control structures (selections and iterations), use of relational and logical operators	1	7	242	167	75	30.99
3	Basic algorithms (counting, summing)	1	8	520	373	147	28.27
4	Using one-dimensional numeric and character arrays	2	7	784	581	203	25.89
5	Basic algorithms with one-dimensional arrays (max/min selection, searching)	2	2	116	49	67	57.76
6	Defining functions	3	9	585	389	196	33.50
7	Working with strings and functions	3	4	117	63	54	46.15
8	Using struct data type	4	5	70	48	22	31.43
Total			**52**	**2 789**	**1 951**	**838**	**30.05**

```
<event>
  <string key="id" value="0"/>
  <date key="timestamp" value="2022-09-26T02:19:00"/>
  <string key="activity" value="1_06"/>
  <string key="resource" value="student1"/>
  <list key="omap">
    <string key="object-id" value="student1"/>
    <string key="object-id" value="1_06"/>
  </list>
  <list key="vmap">
    <string key="conclusion" value="failure"/>
  </list>
</event>
```

Fig. 4. Event data in XML format

The data are stored in an xmlocel file. In this format, events are represented as shown in Figure 4, capturing essential attributes such as timestamp, activity, and resource. The event type is not specified because all events in our data set are of the type GitHub commit. However, the result of the commit is important and is stored in the conclusion attribute of the event. Each event is connected to two objects: a student and a task. Student objects serve as the resources for the events and are described by the signature binary flag and exam grade attributes. Task objects represent the activities affected by the event and are described by the topic and level attributes, as shown in Figure 3.

Table I summarizes the commit data for the programming task groups, while Table II presents the same data categorized by the students' exam grades.

We can see from Table I that students achieved the lowest success rate in the first task group, although these tasks are the simplest ones. The reason behind this is that the use of Git was new to the students, and they used these simple tasks to get acquainted with the technique of taking an assignment and then committing and pushing the solution back to GitHub while producing a high rate of unsuccessful commits. In task groups 5 and 7 less number of commits occurred with the highest success rate, which means that most students quit the game after completing the 1st level and those who carried on were eager to solve the problems. The least number of commits were executed in task group 8. Here the success rate is low, which can be attributed to the fact that this group contains advanced-level tasks that are not included in the end-semester

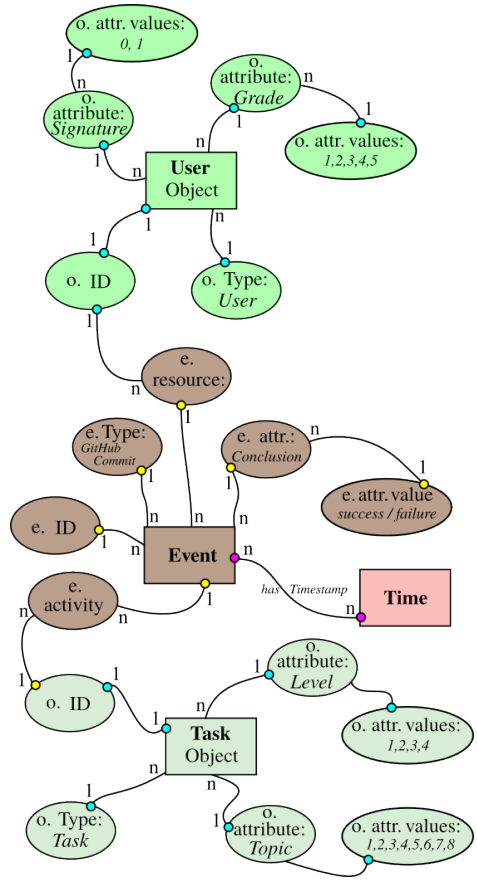

Fig. 5. OCEL schema for User-Task-Event log

exam. Therefore few students worked on these problems and they were not motivated to make more effort.

Table II summarizes the commit events from the students' aspect. The data show that the success rate of GitHub commits and the number of successful commits per student increase in line with the grade, so we can conclude that students who were more insistent in finding the right solution achieved better results at the end of the semester. It is also worth noting that

TABLE II
COMMIT STATISTICS FOR STUDENT GRADES

Exam grade	Num. of students	Commits		Failed		Passed		Success rate (%)
		Total	*Per student*	*Total*	*Per student*	*Total*	*Per student*	*Total passed / total commits*
1 – failed	14	489	34.93	391	27.93	98	7.00	20.04
2 – passed	17	766	45.06	564	33.18	202	11.88	26.37
3 – satisfactory	9	526	58.44	364	40.44	162	18.00	30.80
4 – good	6	307	51.17	197	32.83	110	18.33	35.83
5 – excellent	13	701	53.92	435	33.46	266	20.46	37.95
Total	59	2 789	47.27	1 951	33.07	838	14.20	30.05

the total number of trials is significantly higher in the case of students passing the exam than for those who failed. This number is also remarkable for excellent students compared to those who completed the exam with less success.

By applying process discovery methods to this event log, we first created and compared learning process models of successful and failing students. The second experiment focused on creating a learning process model that is representative of individual cases to identify bottleneck problems that hinder students' progress.

VI. Learning process modeling

Analyzing the event log, the aggregated values in Table III do not show a significant difference between the students who completed the C programming course with an exam grade > 1 and those who failed the exam in terms of the number of commits. We applied independent t-tests to investigate this hypothesis with 95% significance, and we obtained a p-value of 0.20 for the total number of commits and a p-value of 0.50 for the failed commits, so there is no evidence to reject the null hypothesis of equal means in these cases. On the other hand, there is a difference between the number of successful commits, as indicated by a p-value of 0.0056. Also, there is a significant difference between the means of the number of tasks solved, as indicated by a p-value of 0.0077.

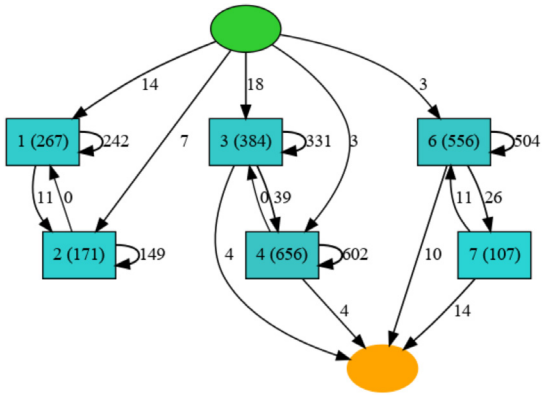

Fig. 6. Learning process model of successful students

For modeling the learning process of the two student groups, the event log was first filtered to include only those students who completed the C programming course. This resulted in an event log containing 2 300 committed events from 45 students. We applied the Heuristic Miner algorithm in the PM4Py package to produce the heuristics net in Figure 6, as opposed to Figure 7, which was generated from the event log containing the 489 commits of 14 students who failed the course exam. This heuristics net provides a comprehensive visualization of the processes, highlighting the critical paths, frequent events, and transitions.

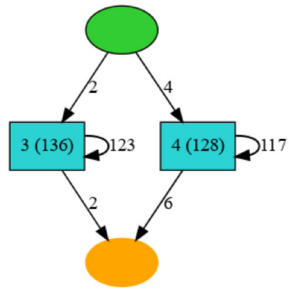

Fig. 7. Learing process modell of failing students

In Figure 6, we can see that in the group of successful students, 14 started the learning process with a task from topic No. 1, 7 started with topic No. 2, and 18 started with topic No. 3. Most frequently, they stopped after completing tasks in topics No. 6 or No. 7. Students typically solved the tasks in the order of the topics. The only exception is when some of them stepped back from topic No. 7 to topic No. 6. Tasks from topic No. 5 were less frequently solved, which is why this is not depicted in the figure. The numbers in the brackets after the topics indicate the related commit events. In this respect, topics No. 4 (Using one-dimensional arrays) and No. 6 (Defining functions) contain the tasks that required the most practice, as evidenced by the highest number of repetitions. This indicates a high number of unsuccessful commits, which motivated students to keep on practicing with another task from the same topic.

Comparing this learning process model to the one in Figure 7, we can conclude that failing students solved fewer tasks from a smaller number of topics, typically from No. 3 or No. 4, and then stopped practicing. This resulted in a smaller number of successful commits, which discouraged them from continuing this kind of practice. This explains why their learning model is less complex than the model of the other group.

TABLE III
AGGREGATED VALUES OF COMMIT EVENTS FOR STUDENT GROUPS

		Num. of commits	Successful commits	Failed commits	Num. of tasks
All students	mean	47.27	14.20	33.07	16.58
	max	221	36	185	42
	min	2	0	0	1
Successful students	mean	51.12	**16.45**	35.67	**18.76**
	max	137	35	110	42
	min	2	0	0	1
Failed students	mean	34.93	**7.00**	27.93	**9.57**
	max	221	36	185	35
	min	2	0	0	1
T-test p-value		0.20	**0.0056**	0.50	**0.0077**

VII. Detecting bottleneck problems

A bottleneck problem is defined as a task that is the last task attempted by a student during their learning process within any topic except for topics No. 6, No. 7, and No. 8, and concluded with failure. Identifying these problems is crucial for instructors when determining their teaching methods. If they recognize the most significant problem areas, they can focus on these during practice in class. We do not consider tasks from the last topics as problematic, even if most students stop practicing after completing them, because these topics represent the highest level we cover for novice programmers.

In this examination, we identified 18 tasks where students quit the practice process. From these, we filtered out the ones that meet our bottleneck problem definition and visualized the learning processes that most frequently failed at these tasks. In this case, the filtered event log contained a significantly reduced number of cases showing infrequent behavior, rendering the Heuristics Miner algorithm inapplicable. As an initial approach, we used the Directly Follows Graph (DFG), where the nodes represent the activities in the log, and directed edges exist between nodes if there is at least one trace in the log where the source task is directly followed by the target task. Frequency values are represented on these directed edges. Figures 8 and 9 highlight two examples of bottleneck problems for instructors of C programming fundamentals.

A. Topic No. 3: Basic algorithms

Task 3_02: Count leap years between two years that you have to read from the standard input while checking their validity (integer numbers between 1 900 and the current year).

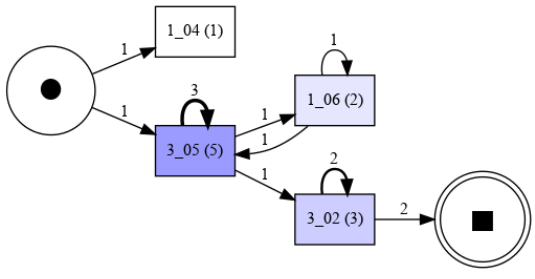

Fig. 8. DFG graph of learning processes ending with Task 3_02.

B. Topic No. 4: Using one-dimensional arrays

Task 4_03: Read float numbers from the standard input, store them in an array, and decide whether the values follow a monotonic-increasing order or not.

Since a DFG-based process map is created without generalization, it is typically much more complex than other process models. For this reason, we also experimented with the Inductive Miner algorithm of the PM4Py package, which is optimized for handling infrequent cases. This method generated the process tree on Figure 10 for the problem at hand, which was converted to a BPMN model for better comprehension.

We can conclude that students following the learning process in Figure 8 were not persistent and stopped practicing after a few attempts. On the other hand, students following the learning process in Figure 9 got stuck at Task 4_03 after solving several tasks.

VIII. Conclusions

This study aimed to enhance the understanding of successful learning processes in programming by applying process discovery methods to student commit data. In an introductory programming course for first-year Computer Science BSc students, 52 practical problems were distributed via GitHub Classroom as out-of-class assignments. GitHub recorded 2 789 commits from 59 students, whose exam grades were extracted from the learning management system of the university. All these data were incorporated into an object-centric event log, which was converted to a case-based log to execute the process discovery algorithms implemented in the PM4Py library. The primary objectives were to identify the distinguishing features of the learning processes of successful students and to detect bottlenecks that hinder student progress.

By applying the Heuristics Miner and Inductive Miner process discovery methods to the event log, we first created and compared learning process models of successful and failing students. This comparison revealed specific patterns and strategies that contributed to student success. Subsequently, we conducted a second experiment focusing on identifying bottleneck problems within the learning processes. These bottlenecks were found to be significant barriers to student progress, providing crucial insights for improving educational practices.

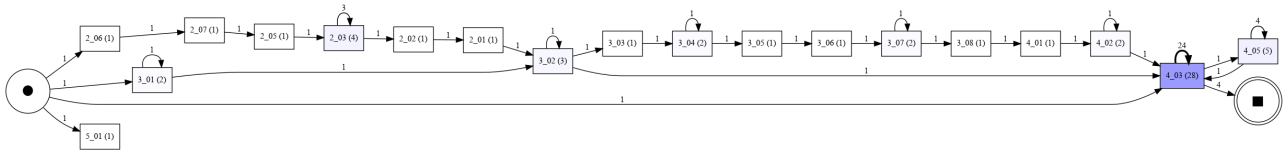

Fig. 9. DFG graph of learning processes ending with Task 4_03.

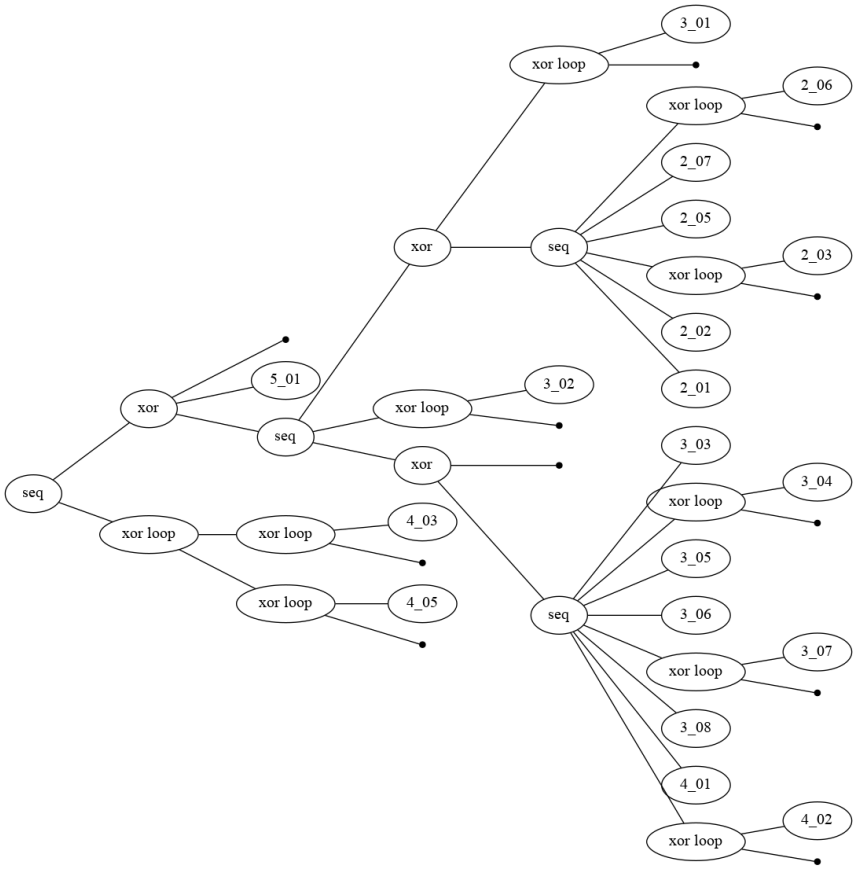

Fig. 10. Process tree of learning processes ending with Task 4_03.

Overall, the findings from this research contribute to a deeper understanding of how students interact with programming assignments. The results highlight the importance of tailored instructional methods that address individual learning challenges, ultimately enhancing educational outcomes.

References

[1] M. A. Ghazal, O. Ibrahim and M. A. Salama, "Educational Process Mining: A Systematic Literature Review." *2017 European Conference on Electrical Engineering and Computer Science (EECS)*, Bern, 2017, pp. 198–203, DOI: 10.1109/EECS.2017.45

[2] I. Y. Kazu, "The Effect of Learning Styles on Education and the Teaching Process.", *Journal of Social Sciences*, 5(2), pp. 85–94, 2009, DOI: 10.3844/jssp.2009.85.94

[3] W. van der Aalst, Process Mining – Data Science in Action, *Springer Berlin*, Heidelberg, 2016, DOI: 10.1007/978-3-662-49851-4

[4] S. Combéfis, "Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools." Software 2022, 1, pp. 3–30. DOI: 10.3390/software1010002

[5] Empowering the next generation of developers, [Online], Available: https://github.com/edu

[6] C. Hsing, V. Gennarelli. "Using GitHub in the Classroom Predicts Student Learning Outcomes and Classroom Experiences: Findings from a Survey of Students and Teachers." In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, pp. 672–678. 2019, DOI: 10.1145/3287324.3287460

[7] Yu-Cheng Tu, V. Terragni, E. Tempero, A. Shakil, A. Meads, N. Giacaman, A. Fowler, K. Blincoe. "GitHub in the Classroom: Lessons Learnt." ACE '22: *Proceedings of the 24th Australasian Computing Education Conference*, pp. 163–172, 2022, DOI: 10.1145/3511861.3511879

[8] E. Baksáné Varga, K.A. Fekete. "Applications for Automatic C Code Assessment." In: *Proceedings of the 2023 24th International Carpathian Control Conference (ICCC)*, pp. 21–26, 2023, DOI: 10.1109/ICCC57093.2023.10178987

[9] C. Romero, S. Ventura. "Educational data mining and learning analytics: An updated survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

[10] Y. S. Mitrofanova, A. A. Sherstobitova, O. A. Filippova. "Modeling Smart Learning Processes Based on Educational Data Mining Tools." In: Uskov, V., Howlett, R., Jain, L. (eds) *Smart Education and e-Learning 2019. Smart Innovation, Systems and Technologies*, vol 144. Springer, Singapore. DOI: 10.1007/978-981-13-8260-4_49

[11] E. M. Real, E. Pinheiro Pimentel, L. V. de Oliveira, J. Cristina Braga, I. Stiubiener. "Educational Process Mining for Verifying Student Learning Paths in an Introductory Programming Course." *2020 IEEE Frontiers in Education Conference (FIE)*, Uppsala, Sweden, 2020, pp. 1–9, DOI: 10.1109/FIE44824.2020.9274125

[12] M. A. Ghazal, O. Ibrahim and M. A. Salama. "Educational Process Mining: A Systematic Literature Review." *2017 European Conference on Electrical Engineering and Computer Science (EECS)*, Bern, pp. 198–203, 2017, DOI: 10.1109/EECS.2017.45.

[13] A. Ghahfarokhi, G. Park, A. Berti, W. Aalst. "OCEL: A standard for object-centric event logs." ISBN 978-3-030-85081-4, 2021, pp. 169–175.

[14] OCEL 2.0 Specification, [Online] Available: https://www.ocel-standard.org/specification/overview/

[15] State-of-the-art-process mining in Python, Fraunhofer Institute for Applied Information Technology (FIT). [Online] Available: https://pm4py.fit.fraunhofer.de/

[16] A. Berti, S. van Zelst, D. Schuster, "PM4Py: A process mining library for Python", *Sowtware Impacts*, 17, 100556, 2023, DOI: j.simpa.2023.100556

[17] L. Numminen, "Process Mining Algorithms Simply Explained", 2023, [Online] Available: https://www.workfellow.ai/learn/process-mining-algorithms-simply-explained

[18] W. van der Aalst, A. J. Weijters, L. Maruster. "Workflow Mining: Discovering Process Models from Event Logs." *IEEE Trans. on Knowledge and Data Engineering*, 2004

[19] S.J. Leemans, D. Fahland, D., W. van der Aalst. "Discovering block-str uctured process models from event logs – a constructive approach." In: *Petri Nets. Lecture Notes in Computer Science,* vol. 7927, pp. 311–329. Springer, 2013

[20] S. J. Leemans et al. "Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour." *Business Process Management Workshops*, 2013

[21] A. J. Weijters, J. T. Ribeiro. "Flexible Heuristics Miner (FHM)", *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Paris, France, 2011, pp. 310–317, DOI: 10.1109/CIDM.2011.5949453

[22] W. van der Aalst, A.J. Weijters, L. Maruster. "Workflow Mining: Discovering Process Models from Event Logs." *IEEE Trans. on Knowledge and Data Engineering*, 2004

Erika Baksáné Varga was born in 1976. Got MSc degree in Information Engineering in 2000 and MSc degree in Economics for Engineers in 2003 from the University of Miskolc. PhD student at József Hatvany Doctoral School for Computer Science and Engineering from 2003 till 2006. Holds PhD degree since 2011 for defending the thesis entitled "Ontology-based Semantic Annotation and Knowledge Representation in a Grammar Induction System". Currently associate professor at the Institute of Information Technology. Research interests: programming languages, natural language processing, data and knowledge engineering, process mining.

Attila Baksa was born in 1976. Received his MSc degree in Information Engineering from the University of Miskolc in 2000. Holds PhD degree since 2006 for defending the thesis entitled "Numerical Analysis of Mechanical Contact". Currently associate professor at the Institute of Applied Mechanics. Research interests: programming, numerical simulations, nonlinear problems, algorithms, finite element method.