Finding behavioral indicators from contextualized commits in software engineering courses with process mining

Mika Pons, Jean-Michel Bruel, Jean-Baptiste Raclet, Franck Silvestre

Finding behavioral indicators from contextualized commits in software engineering courses with process mining

Mika Pons[1][0000−0002−5844−8727], Jean-Michel Bruel[2][0000−0002−3653−0148], Jean-Baptiste Raclet[3][0000−0001−7357−912X], and Franck Silvestre[1][0000−0002−1134−8200]

[1] Université Toulouse 1 Capitole, IRIT, France
[2] Université Toulouse 2 Jean Jaurès, IRIT, France
[3] Université Toulouse 3 Paul Sabatier, IRIT, France

Abstract. Git4School is a dashboard helping teachers to monitor and make decisions during Git-based lab sessions in higher education computer science programs. This tool makes it possible to visualize the commits made by students over time according to the context and, in particular, the type of pedagogical intervention by the teacher (discussions between students on the problem, dissemination of a solution, etc.). Despite its visualizations providing indicators for decision-making, the tool does not provide information about the student's behavior. There are existing studies dealing with Process Mining (PM) in education, specifically in computer science courses and using Git. Through an empirical exploratory study, we explore the possibility of taking advantage of these contextualized commits using PM. We analyzed data from 5 teaching units covering different higher education levels using the bupaR library. Firstly, we discovered promising indicators to predict students' behavior during a lab session. Secondly, we identified several possibilities for future research on PM and contextualized commits. Finally, we have established a set of recommendations to help analyze contextualized commits using PM.

Keywords: Learning analytics · Educational data mining · Git · Process mining · Behavioral patterns.

1 Introduction

In their 2020 study, Raclet and Silvestre [13] introduced the Git4School (G4S) dashboard, which aims to assist teachers in (1) monitoring student activity and in (2) making decisions to trigger a pedagogical intervention (e.g., publication of the solution for a problem, a peer review, etc.). G4S is currently used in software engineering education in higher education computer science programs. The data visualized in G4S is automatically extracted from individual Git repositories in which each learner performs their work, enriched with situational data.

In Git terminology, the result of a source code modification is saved in a unit of information called a *commit*. Triggered by a learner at the end of a unit of work, a commit records the learner's identity, the message provided by the learner describing the unit of work (e.g., `fix question #1`), the date and time of the commit and what was modified by the learner. Combining this information with situational data gives more information about the pedagogical context of the commit; for example, it is possible to know whether it was carried out before or after a particular intervention made by the teacher, during a class, or after, etc. In the remainder of the paper, the term *contextualized commit* refers to the combination of the information contained in a commit with the situational information related to that commit.

Although the G4S dashboard provides relevant indicators for bridging decision-making based on contextualized commits, information needs to be provided on the behavior of learners during the activities offered to them throughout a course. It is, therefore, currently only possible to identify patterns of student behavior that lead to better learning outcomes or, conversely, that lead to failure.

This paper presents an empirical and exploratory study based on data collected from activities supervised with G4S. First, our study aims to answer the following research question:

> RQ1: What indicators about student behaviors can we extract using Process Mining (PM) with contextualized commits?

Moreover, as the learner triggers a commit at the end of a unit of work, it does not contain precise information about the start date and the duration of the work activity. This point represents a limitation for making the best use of PM techniques. Therefore, our study aims to answer the following second research question:

> RQ2: What information should be added to the contextualized commits to make the most of the Process Mining techniques?

The paper is composed of 4 sections. Section 2 presents previous work related to our research questions. Then, section 3 describes the empirical study and the obtained results. Section 4 discusses the results and provides answers to our research questions. Finally, the paper ends with a conclusion and a presentation of future work.

2 Related work

2.1 Process Mining in education

Bogarín et al. [3] define Educational Process Mining (EPM) as using of log data gathered specifically from educational environments to discover, analyze, and provide a visual representation of the complete educational process. As such, Process Mining (PM) techniques have already been used to identify patterns of learner behavior. Some studies focus on specific dimensions of the learning

process, such as self-regulation [4] or how students take part in quizzes [8]. Other studies focus on detecting student learning paths by exploring the data collected by Learning Management Systems, such as Moodle [5] or by MOOC platforms [11, 2].

2.2 Process Mining in software engineering

Process Mining has also been applied to software engineering in different ways. In 2007, Rubin et al. [14] proposed a framework to explicit software development processes based on data stored in configuration management systems. Thus, several studies focus on the process mining of software repositories. Poncin et al. [12] used the FRamework for Analyzing Software Repositories (FRASR) to combine logs coming from several kinds of repositories: version control systems, bug trackers, and mail. Then they could classify developers by role or identify some patterns in the bug lifecycle. Gupta et al. [7] analyzed data from three software repositories to improve the process relative to reporting and resolution of issues. Ardimento et al. [1] use the conformance checking technique to test coding behavior, starting with event logs generated from IDE.

2.3 Process Mining in software engineering education

At the crossroads of Education PM and PM applied to software engineering education, some research works present different ways of using PM in the context of software engineering courses. For example, Mittal and Sureka [10], in the context of an undergraduate software development course, mined data from version control systems, wiki, and bug tracking systems to qualify learners' activities in teamwork projects better. In 2020, Eskofier [6] presented same kind of work, mining different kinds of repositories but leveraging the Gitlab[4] platform. Similarly, Shynkarenko and Zhevaho [15] aim to provide visualization resulting from PM to help students and teachers in code review and assessment activities. Finally, the study proposed by proposed by Macak et al. [9] focuses on mining data from students' Git repositories. As such, the authors describe precisely how to convert a Git log into an event log that PM tools can process.

Our exploratory study is in line with the work presented in this section. It differs from it by exploring an aspect not dealt with in the previous work: considering elements of pedagogical context and of the teacher's pedagogical interventions in the activities carried out by the students.

3 Empirical study

In this section, we first present the contexts for the production of our datasets, and the steps followed to transform and analyze the data using the bupaR[5] library. Then, the study results are presented, focusing on relevant items for discussion.

[4] https://about.gitlab.com/
[5] https://bupar.net/

3.1 Datasets description

G4S has been experimented with in several computer science courses from two French higher education institutions, covering ISCED (2011) levels 5, 6, and 7[6] (see tab. 2 for a more detailed description of the datasets):

- Institut Universitaire de Technologie of Rodez, in the context of its Associate Degree and B.Sc. in Computer Science training;
- Université Toulouse 3 Paul Sabatier, as part of its M.Sc. in Software Engineering.

From G4S logs exports, we collected the datasets corresponding to the work done by the students during lab sessions of 6 courses. Concretely, they had to solve sets of questions from a worksheet and commit when a question was done. Out of the six collected datasets, one was not eventually included in the study presented here because it needs to contain the educational interventions we want to analyze. More precisely, this course was organized without review or correction between lab sessions. The questions were corrected after all lab sessions but before the final exam. Therefore, the correction could not have affected the students during the lab sessions.

Thus, five datasets were analyzed, representing about 200 students and 3100 commits over two academic years (from 2020 to 2022). For each course, the lab sessions were supervised, and the order of the questions was fixed. Each of these lab sessions was supervised by a teacher during face to face sessions and the order of the questions was predefined. One course had two optional questions at the end of the worksheet.

As advocated in the introduction, the analyzed data are commits contextualized by their anteriority or posteriority to a pedagogical intervention by the teacher. These interventions are either peer reviews or corrections and address predefined questions. With this information, G4S can type commits as follows:

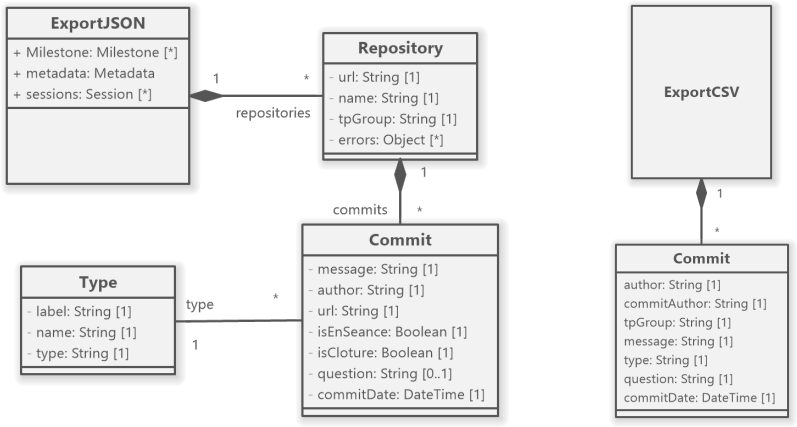
- Intc intermediate: the commit does not resolve a question;
- Bfrr before review: the commit resolves a question before the associated review;
- Brac between review and correction: the commit resolves a question between the associated review and correction;
- Aftc after correction: the commit resolves a question after the associated correction.

We then need to transform these specific logs into event logs in order to be able to use PM techniques.

[6] It corresponds to short-cycle tertiary education, Bachelor's and Master's or equivalent level respectively (https://en.wikipedia.org/wiki/International_Standard_Classification_of_Education)

3.2 Pre-processing

Our datasets include, among other things[7], the list of each student's GitHub repository with a list of commits made during the lab sessions (see fig. 1a).



(a) Model of the JSON G4S export

(b) Model of the CSV used for the analysis

Fig. 1: Datasets structure models

Before analyzing the data, a pre-processing phase was necessary to adapt the data structure for the bupaR library. For this, a Python script[7] was written with two objectives:

- transform the data model structure from a list of GitHub repositories to a list of commits;
- convert the file format to CSV.

The data provided by G4S is in a JSON format which is difficult to manipulate with the R language. Therefore, using a Python library, a CSV file is obtained after picking the data we wanted as columns: The student's name, the commit author identifier (it will be needed for filtering described in the following section), the student's group, the raw commit message, the type of the commit, and the resolved question if there is one (or an empty string otherwise).

In the end, CSV files are produced with a simple structure (see fig. 1b) required for the analysis part presented in section 3.3. This data will be sufficient to build the event logs for two types of activities: (1) the resolution of a question and (2) the work in a pedagogical context identified by the type of commits.

[7]For more details, visit the public repository : https://gitlab.irit.fr/talent/TALENT/around-g4s/G4S-to-PM-scripts-and-data-2022

3.3 Process mining analysis with BupaR

After the pre-processing, the datasets are almost ready to be used with bupaR. Two types of event logs have been constructed corresponding to the two types of activities mentioned before. To this end, an R script has been written (also available in the public repository).

Two columns have been added. First, the column status is needed to indicate when the activity is complete. Because we only get the event linked to the end of the activity in our data, we assign the value "complete" for all the rows. Secondly, we add the column activity_instance with a value incremented on each row to differentiate each activity for the same case. Next, for constructing of the event logs with the resolution of a question as an activity, we excluded the Intc commits as we only want to analyze resolved questions' sequences. Finally, we filter the rows to exclude the automatic commits from GitHub Classroom[8]. Indeed, when the students create their repository, a first commit is done automatically by a bot (whose author is github-classroom[bot]) to have the startup code of the course.

As presented in table 1, we constructed two event logs per dataset according to the above properties.

Table 1: How the event logs are built based on the activity

	Question resolution-based	Commit type activity-based
case_id	author	author
activity_id	question	type
resource_id	type	question
timestamp	commitDate	commitDate

The script then generated two R markdown[9] reports. Both present the same 13 types of graphs, but one uses the event logs based on the resolution of a question, while the other is based on the work in a pedagogical context identified by the type of commits. BupaR uses the heuristicsmineR package. This package uses the frequencies of the transitions between activities to build the precedence matrices (or causal net) and the process maps. The following section presents the more relevant graphs for future discussion.

3.4 Results

Question resolution. Figure 2 shows the sequences of students' answers to a particular question. Note that each node in all the remaining graphs is a commit and hence a particular step in the worksheet. Ideally, the resolution process

[8] https://classroom.github.com/
[9] https://rmarkdown.rstudio.com/

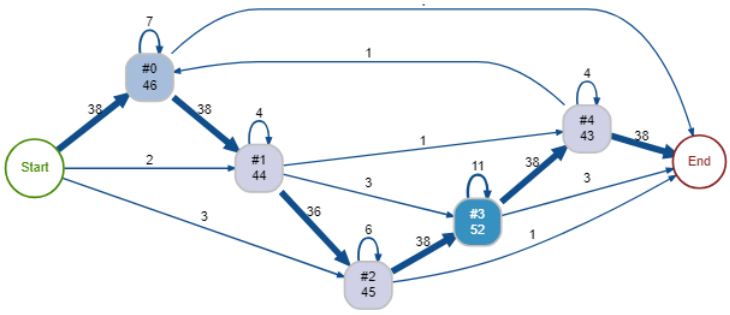


Fig. 2: Question resolution process map during a lab session

should be sequential from the first question to the last. However, the graph shows that the reality is quite different. There are several kinds of transitions between non-consecutive questions:

- loops;
- transitions from Q_n (Question $\#n$) directly to the end (with n not being the last question in the worksheet);
- transitions between Q_n and Q_{n+m} $(m > 1)$;
- transitions between Q_n and Q_{n-m} $(m \geq 1)$.

They are a few loops, but they are always present in all the study datasets. Transitions from Q_n to *End* indicate that the last question resolved by the student was the $\#n$. We can see several transitions between Q_n and Q_{n+m} (m hence representing the number of forgotten commits) and infrequent transitions from Q_n to Q_{n-m}.

All the lab sessions we have analyzed have *chaotic transitions* except for Figure 3. In this one, only two transitions out of 204 (0.98%) skip a question. After asking the teacher in charge of this course, the three reasons for this shallow rate of chaotic commits are:

- an extreme dependence between the questions;
- a small group size making it easier to follow the group individually;
- regular reminders of the question-solving process by commit.

Looking for indicators, we have also analyzed the performance profile with the average time between the resolution of two questions. Figure 4 shows an example of a produced graph. The label on the edge gives the average time between the two activities corresponding to the starting node and the destination node. In our experiment, however, the starting time of the activities is not logged by G4S.

Indeed, for each question, only the commit date is available. This duration could be extrapolated from the time between two consecutive activities, but this introduces too much hazard to be considered. For instance, in Figure 4, we can see that it takes an average of 187.31 hours to complete question #3 (see the

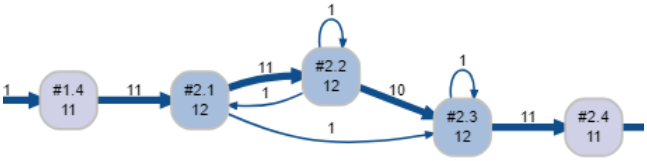


Fig. 3: An extract from the process map showing a well-followed question resolution process

label of the edge between #2 and #3). This duration is not representative of the time taken to solve the question because it includes the time elapsed between the two lab sessions.

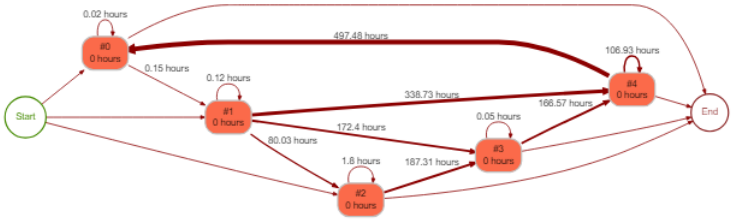


Fig. 4: Performance profile process map (using *mean* function)

Work in a pedagogical context identified by the type of commits. The precedence matrix based on the antecedents was generated (see Figure 5). This graph shows the probability that the next commit will be of a particular type depending on the type of the last commit. Out of all the antecedents, the one most likely to result in an Aftc commit is the Aftc commit itself. This is the same for the Bfrr commits.

Last, the matrix shows that the lowest probability of resulting in an Aftc commit is an Intc commit. The probability of this happening is 4.56%, while it is 10.93% from the state Bfrr . In other words, a student is less likely to produce an Aftc commit if they made an Intc commit instead of a Bfrr commit.

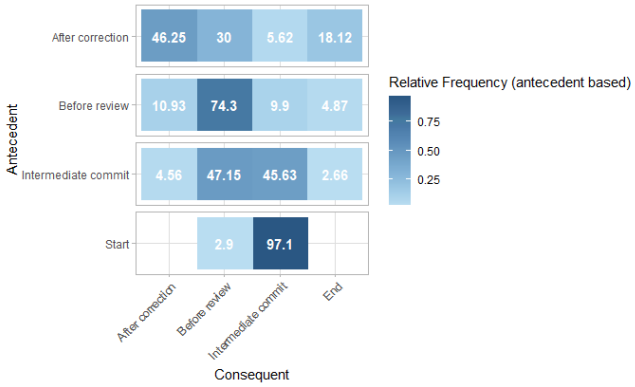


Fig. 5: Precedence matrix based on commit type as activities

4 Discussion

We now propose interpretations of the results presented in the previous section. We also discuss some limits coming from the data and the approach. Finally, we present a work in progress to overcome the identified limits.

4.1 Interpretation

In the first subsection, we address RQ1 by presenting the indicators we found promising for future studies. In the second subsection, we first address RQ2 by identifying data that could enrich the contextualized commits. Then we suggest recommendations to perform a better PM analysis on contextualized commits.

Behavioral indicators. The $Q_n \rightarrow Q_{n-m}$ edges show that students return to previous questions, which is a behavior not easily noticeable in G4S. Two reasons a student goes back to a previous question: (1) they just forgot something, (2) they want to correct or improve their solution (because of a second thought or a teacher's advice or answer, ...). The last reason is showing that the student is engaged in a learning process, but, as it stands, we cannot affirm that this is the main reason for this pattern. Also, it could be interesting to study a possible correlation between that behavior and their grades on the final exam. Future work could be to cluster the students according to whether or not they engage in this behavior in order to see if they tend to make less Aftc commits.

In Figure 5, we saw that Intc commit is the least likely to lead to a Aftc commit. Making Intc commits has a positive influence on the resolution of questions. This hypothesis also needs to be investigated more by looking for a correlation with performance indicators, for instance. It is an exciting indicator as it shows that the student is committed to the work to be done by following an approach common and encouraged in the professional world, i.e., to commit their work regularly.

To finish with this figure, we can see the high probability of transition between two Bfrr commits or two Aftc that we could respectively call the *virtuous loop* and *vicious loop*. Although both behaviors are relatively common (and not only in the context of learning), they can be relevant indicators. For example, the lower the probability of a *vicious loop*, the greater the ability of the group to catch up.

Improvements. To understand the behavioral mechanism behind the $Q_n \rightarrow Q_{n-m}$ and loop transitions, we could ask the students during an interview why they returned to a previous question when they make the commit.

Because our datasets essentially present mandatory questions, we assumed Aftc commits were a negative sign that the student is behind schedule. However, a late resolution for optional questions means the student is trying to understand the solution. It shows that they are engaged in a deep learning process. We need to track which questions are optional to see the difference and have the correct interpretation.

While using bupaR, process map views seem more appropriate for analyzing event logs based on question resolution, as we can see resolved questions' sequences. While the precedence matrix looks quite relevant for event logs based on work identified by the type of commits, we can look at the probabilities of moving from one type of commit to another, which is interesting if we want to maximize the occurrence of one type of commit.

When analyzing the performance profiles in section 3.4, we have seen that having no start date for the activity makes this analysis irrelevant. We need to add this data with more accurate tracking of student activity.

The datasets with very few *chaotic transitions* show that only analyzing the data without having the context of its production is a mistake. To avoid this, interviews with the involved teachers are necessary.

As the last recommendation, it is essential to check that datasets represent the behaviors to study. This could be expected in any analysis, but it is not easy to assess when analyzing contextualized commits with PM. For example, in the precedence matrix, a dataset could have transitions to an Aftc commit simply because no student has committed after the correction (instead of having no correction).

4.2 Limits

The first limitation we faced when analyzing the data was the lack of reliability. Indeed, as shown in Figure 2, a certain number of students do not strictly follow the process of producing a commit when they finish a question. Some students may work, resolve several questions and make one commit, or make the commits way after, so they appear as Aftc commit when it is not the case.

Also, all the processes rely on Git-based lab sessions, which brings two issues: (i) it makes this analysis hard for first-year students as it requires a minimal degree of expertise with Git; (ii) it limits this type of study to computer science

students. We are working on a script[10] to help reduce or avoid some of these limitations.

4.3 Script : G4S-automation

The G4S-automation script is written with Python; it has two objectives: (i) to make the question resolution process based on contextualized commits accessible to learners not familiar with Git; (ii) to make the learner activity tracking (through the commits) more accurate. To this end, the script proposes three main features.

Firstly, it tracks any operations on the files inside the repository it has been launched in. An automatic commit identifies each operation. We will have the start of a question resolution and will be able to perform an accurate performance profile analysis.

Secondly, it provides a console for a student to perform a commit to resolve a question without using Git. For example, the command fix #1 will commit and push all the changes since the last question was resolved so that the teacher can inspect the progress with a dashboard like G4S. This will ease the use of this tool for first-year students in computer science curricula because it abstracts the use of Git.

Thirdly, it allows the opening and closing of the workspace. When the student stops the script, all the files in his Git repository become inaccessible locally. Then, when they restart the script, the files are accessible again, and the student can resume their work. This feature guarantees the complete tracking of the student's activity, even outside the lab session.

The use of Github to manage practical work is very common in computer science. To orchestrate the resolution of questions on an exercise sheet, "issues" are sometimes used. It is then possible to close these issues, marking them as solved, directly in the commit message, by indicating the character "#" followed by the issue number to close. We have designed this interactive system so that it is possible to close these issues via the "fix" command. To do this, you just need to define the questions to be resolved in the same way as the close issue key, i.e.: "#1" for the question and issue 1, ...

Finally, to address the issue of GDPR compliance, we plan to soon integrate data anonymization directly into the dashboard. This will ensure that only teachers have access to personal data, while researchers wanting to process data extracted from the dashboard will only have access to anonymized data, by design.

5 Conclusion

In this study, we proposed to analyze five datasets, with more than 3100 contextualized commits for about 200 students in higher education, using Process

[10] https://github.com/git4school/git4school-automation

Mining (PM). We extracted several indicators of student learning behavior. Although we could not determine their effectiveness, we have listed relevant data to add and recommendations to improve the analysis of contextualized commits with PM. Finally, we have introduced an outgoing work with a script that allows for partial automation of the question commit process, which will reduce some limitations encountered in this study. This script will be tested in a real-life classroom setting during the 2023 school year. Future work will be helpful to verify the relevance of the indicators, looking for a correlation between them and the marks obtained by the students at the final summative assessment.

Bibliography

[1] Ardimento, P., Bernardi, M.L., Cimitile, M., Maggi, F.M.: Evaluating coding behavior in software development processes: A process mining approach. In: 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), pp. 84–93, IEEE (2019)

[2] Arpasat, P., Premchaiswadi, N., Porouhan, P., Premchaiswadi, W.: Applying Process Mining to Analyze the Behavior of Learners in Online Courses. In: Int. J. of Inf. and Education Technology, vol. 11, pp. 436–443 (2021)

[3] Bogarín, A., Cerezo, R., Romero, C.: A survey on educational process mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(1), e1230 (2018)

[4] Cerezo, R., Bogarín, A., Esteban, M., Romero, C.: Process mining for self-regulated learning assessment in e-learning. Journal of Computing in Higher Education **32**(1), 74–88 (2020)

[5] Dolak, R.: Using process mining techniques to discover student's activities, navigation paths, and behavior in lms moodle. In: International Conference on Innovative Technologies and Learning, pp. 129–138, Springer (2019)

[6] Eskofier, B.M.: Exploration of process mining opportunities in educational software engineering-the gitlab analyser. In: Proc. of The 13th International Conference on Educational Data Mining (EDM 2020), pp. 601–604 (2020)

[7] Gupta, M., Sureka, A., Padmanabhuni, S.: Process mining multiple repositories for software defect resolution from control and organizational perspective. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 122–131 (2014)

[8] Juhaňák, L., Zounek, J., Rohlíková, L.: Using process mining to analyze students' quiz-taking behavior patterns in a learning management system. Computers in Human Behavior **92**, 496–506 (2019)

[9] Macak, M., Kruzelova, D., Chren, S., Buhnova, B.: Using process mining for git log analysis of projects in a software development course. Education and Information Technologies **26**(5), 5939–5969 (2021)

[10] Mittal, M., Sureka, A.: Process mining software repositories from student projects in an undergraduate software engineering course. In: Companion proceedings of the 36th international conference on software engineering, pp. 344–353 (2014)

[11] Mukala, P., Buijs, J.C., Leemans, M., van der Aalst, W.M.: Learning analytics on coursera event data: A process mining approach. In: SIMPDA, pp. 18–32 (2015)
[12] Poncin, W., Serebrenik, A., Van Den Brand, M.: Process mining software repositories. In: 2011 15th European conference on software maintenance and reengineering, pp. 5–14, IEEE (2011)
[13] Raclet, J.B., Silvestre, F.: Git4school: A dashboard for supporting teacher interventions in software engineering courses. In: European Conference on Technology Enhanced Learning, pp. 392–397, Springer (2020)
[14] Rubin, V., Günther, C.W., Van Der Aalst, W.M., Kindler, E., Dongen, B.F.v., Schäfer, W.: Process mining framework for software processes. In: International conference on software process, pp. 169–181, Springer (2007)
[15] Shynkarenko, V., Zhevaho, O.: Application of constructive modeling and process mining approaches to the study of source code development in software engineering courses. Journal of Communications Software and Systems **17**(4), 342–349 (2021)

Appendix A Description of the datasets

Datasets	Course name	Institution	Learning outcomes	ISCED level	Number of students
FilmProvider	XML and Web Services	IUT de Rodez	Know how to use XPATH and create an XML schema	5	2 groups of 20+ students
M1 SDL	Engineering of dynamic web applications	Université Paul Sabatier	Design (multi-layer architecture, role of a framework) and technology (J2EE: servlet, JSP, ORM, etc)	7 (first year M.Sc.)	49 students in 4 groups
M2 SDL	Optimization of dynamic web applications	Université Paul Sabatier	Be able to implement the advanced notions of JPA in the context of a Java project accessing a relational database. Be able to use Spring Boot to develop a backoffice application exposing a RESTFull API	7	1 group of 32 students
OurBusiness	Data persistence	IUT de Rodez	Understand the main principles of ORM approaches and how to use the JPA API	6	1 group of 11 students
PDO_MVC	Server-side Web programming	IUT de Rodez	Know how to use SQL from a programming language (PHP in our case) and how to develop a Web application (in PHP in our case) respecting the MVC pattern	5	3 groups of 20+ students
db_my_activities	Database Programming	IUT de Rodez	Know how to develop functions and procedures in SQL	5	3 groups of 20+ students

Table 2: Description of datasets