# Traceability by design: design of an interactive system to improve the automatic generation of Git traces during a learning activity

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

# Traceability *by design*: design of an interactive system to improve the automatic generation of Git traces during a learning activity

Mika Pons [1], Jean-Michel Bruel [2], Jean-Baptiste Raclet [3], and Franck Silvestre [1]

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

**Abstract.** *Learning Analytics* (LA) is collecting and analyzing traces of learners' activities in order to understand and improve learning. This paper focuses on traces generated using the version control system Git. Existing works on the topic have limitations regarding the quality of the traces they analyze: (1) the quantity and content are not always sufficient for in-depth analysis of student behavior, (2) their limited reliability can lead to a loss of exploitable data, and (3) the method of generating these traces is not generic. We propose a new interactive system based on Git and the observation of file modifications to generate automatically reliable and rich traces. This interactive system will soon be experimented with in an ecological context and is intended for diversified teaching contexts.

**Keywords:** Learning analytics, Git, Traces, Interactive system.

## 1 Introduction

If the use of version control tools, like Git, is widespread in the software industry, it is also used in an educational context, mainly in computer science training [3, 2]. Integrating such tools in teaching constitutes a simple means of obtaining traces of the activity of the students thanks to the actions of *commits* carried out during the progress of the activities to version the work. More particularly, Git works using a decentralized architecture which allows to work with local copies and thus to synchronize with the main remote copy. It helps teachers collect students' work. Also, Git can store in an optimized way the modifications made by the students, called *diff*.

At the crossroads of *Learning Analytics* and version management software, we see the emergence of work based on the analysis of traces produced using version management software in computer learning [8]. Some works exploit *Process Mining* techniques for trace analysis by extracting metrics related to students' learning behavior [4].

However, these studies have limitations concerning the traces generated by version control software (VCS) such as Git. Indeed, the content of these traces

is not sufficient for certain analyses requiring more precise temporal information to be known. For example, we can't make an analysis on the duration of the resolution of exercises because there is not enough information available for that. Between the beginning of a work session and the resolution of an exercise, the time actually spent working is unknown. Also, these traces may lack reliability when their generation is based on a declarative and open process by the student. Finally, these traces are generated with methods that are difficult to apply in fields that do not fall within the field of computer science.

To overcome these limitations, we wonder how to design an interactive Git trace generation system for students, and more specifically: (RQ1) How to increase the quantity of Git traces using an interactive system? (RQ2) How to increase the quality of Git traces using an interactive system? (RQ3) How can an interactive system make the process of generating Git traces usable for an audience unfamiliar with Git?

First, we will provide an overview of works that use Git traces to improve teaching in section 2, while highlighting their limitations. Secondly, in section 3, we will present the interactive system LAWG designed to answer the issues raised by our research questions. Finally, we'll explain how the features of LAWG address this in the section 4.

## 2 State of the art

Several papers [3, 2] have experimented with integrating VCS into computer science learning and show the benefits. The integration of VCS into practical sessions gives the teacher the ability to centralize the distribution of course material for all students and to easily monitor and collect their work[1]. In the field of *Process Mining* (PM), several works [5] focus on analyzing Git traces generated in the context of software development courses. In 2021, Macak et al. [4] proposed a method to analyze Git traces of student projects using PM. This article shows it is possible to identify student learning behaviors using PM technique. However, only the information provided by Git in a commit is considered for analysis. Git traces do not contain contextual information about a student's progress in a learning activity, such as solving a well-identified question. There is no temporal information concerning the dates of the start of resolution activity for a question and therefore does not address (RQ1). Finally, this method of analysis does not deal with the issues of reliability and easy of use of the traces generation, as referred to in (RQ2, RQ3).

In 2018, Silvestre et Raclet [10] developed a learning protocol based on code review and test-guided development. To help orchestrate its different phases, a dashboard, G4S [7], is used. Based on the commits made and the milestones (reviews, corrections) filled in by the teacher, this dashboard provides a number of indicators, such as the progress of students within their group. The dashboard also makes it possible to export the traces enhanced by the context of the resolution of the questions. We exported the dashboard data and analyzed it using PM techniques [6]. Unfortunately, we couldn't extract any indicators of students'

behavior because the traces didn't contain enough information for that. More precise information on students' real-time activity (RQ1) was lacking, the process of marking questions as solved depended too much on the students, leading to errors in the data (RQ2), and the process was difficult to operationalize for an audience not very proficient with Git (RQ3).

A 2022 study [9] explores an alternative way to automatically generate and visualize student activity traces during practical sessions. To automate the generation of the traces, Rodriguez-Rivera et al. integrated the commit commands into a Makefile, generating a Git trace for each compilation. But, compilations are infrequent, leading to coarse tracing granularity (RQ1). Using a Makefile is restrictive for a non-Git-competent audience (RQ3).

In conclusion, to our knowledge, there is a lack of prior work on an interactive system capable of overcoming the identified limitations in quantity, quality, and ease of use. We sought a compromise between G4S and EnCourse. To address this, we have designed an interactive system that generates traces in real-time, considering the contextual information of students' progress, while maximizing reliability and ease of use.

## 3 The interactive system LAWG and its features

To answer our research questions, we designed an interactive system, called LAWG, written in Python and open-source, available for Windows, MacOS, and Linux[4]. Our interactive system offers three main functionalities that we explain in the following sections: (1) Supervise work sessions, saving all resources at the end of each session (2) Observe all changes made to files in the workspace to save them in real-time (3) Provide a command-line interface for students to mark a question as solved.

The context in which LAWG is used is as follows: students work during *work sessions* on *worksheets* made up of several exercises, each consisting of one or more *questions*. Students flag questions as solved by making a *commit* (with an associated message) and synchronize their remote workspace using the `push` command[5]. Finally, the sequence of commits constitutes a *branch*.

Before using the interactive system, the teacher and then each student must configure it[6]. The system manages the student's workspace and automatically traces the start and the end of a work session. When the student opens and closes the system, it opens and closes the workspace, saving the latest changes. These events generate the respective `"Resume"` and `"Pause"` commits.

When LAWG is launched, changes to workspace files are observed. Each event causes a commit to be produced in the `auto` branch. The events observed are: *modification*, *creation*, *deletion*, *move* and *renaming* of a file. For these commits, we have defined a minimalist message form to facilitate further processing:

[4]Available on GitHub at this address: https://github.com/git4school/LAWG.

[5]Local and remote workspaces are managed with the VCS Git

[6]This part is explained in the longer version published here: https://hal.science/hal-04141003

- "[moved] <path_file> -> <path_new_folder>"
- "[renamed] <path_file> -> <new_name_file>"
- "[<event>] <path_file>"

LAWG offers the student a command-line interface. The commands available are easily expandable thanks to a modular design and are suggested as a list to the student. This allows the student to mark a question as solved with the `fix` command, whose questions are also suggested. Using this command generates a commit of the form `"Fix <question>"`.

Last, some limitations of our system have been identified. First, there is still a dependency between the student and the expertise in Git (initial cloning of the repository and SSH key). Moreover, the system is not designed to work with multiple branches as yet.

## 4 What are the benefits of using the interactive system?

Concerning the question (RQ1). The generation of traces when each file is modified provides a more detailed overview of student activity. This gives us greater coverage of their actual activity. To test the system, we generated traces of a student using the system to answer questions on an worksheet from a real course[7]. The figure 1 shows an extract of the generated traces. New traces generated thanks to the system are in red. We can see that the volume of traces is significantly increased and covers a more extensive range of time than without using the system.

Through the generation of a trace for each file modification, it is possible to identify information about students' behavior via the name of these files. A concrete example of this, in the case of the figure 1, is the identification of the good following of the Test-Driven-Development (TDD). We can see in lines 62 and 61 that the student integrates the test `ArticleTest`, then modifies the associated class `Article` thanks to the naming convention of the tests. This is, therefore, representative of TDD. On the other hand, we can see that the student modifies the classes `IndexController` on line 67 before the associated test `IndexControllerTest` on line 66. Here we see that the TDD is not followed, and a simple comparison of the file names would make it possible to automate this identification from these traces.

In addition, the management of work sessions makes integrating the beginning and end of these sessions in the logs possible. This can be observed in the figure 1 at the lines 44 and 74 with the commits `"Resume"` and `"Pause"`. Also, the associated mechanism, which empties and restores the workspace, allows us to ensure the systematic use of the interactive system for each work session and, therefore, the completeness of the traces.

[7]The anonymized dataset is available on OSF at this address: https://osf.io/t3wqr/?view_only=69907570f39046edba382a5e855ed26a.

| | | | |
|---|---|---|---|
| 44 | fbf16ec | Mon, 10 Apr 2023 18:28:55 +0200 | Pause |
| 45 | ea3f9d1 | Mon, 10 Apr 2023 18:26:46 +0200 | Fix #2b |
| 46 | 19c7009 | Mon, 10 Apr 2023 18:26:06 +0200 | [modified] src/test/java/doremi/doremi/BandTest.java |
| 47 | 6890448 | Mon, 10 Apr 2023 18:23:36 +0200 | [modified] src/main/java/doremi/domain/Band.java |
| 48 | 89a6b70 | Mon, 10 Apr 2023 18:23:31 +0200 | [modified] src/main/java/doremi/domain/Band.java |
| 49 | 1157e96 | Mon, 10 Apr 2023 18:23:15 +0200 | [modified] src/test/java/doremi/doremi/BandTest.java |
| 50 | fa4054a | Mon, 10 Apr 2023 18:23:13 +0200 | [created] src/test/java/doremi/doremi/BandTest.java |
| 51 | fa0c6fa | Mon, 10 Apr 2023 18:22:39 +0200 | [modified] src/main/java/doremi/domain/Album.java |
| 52 | ce3c3bd | Mon, 10 Apr 2023 18:22:23 +0200 | [modified] src/main/java/doremi/domain/Album.java |
| 53 | 1c5090f | Mon, 10 Apr 2023 18:21:47 +0200 | [modified] src/test/java/doremi/AlbumTest.java |
| 54 | 2312a58 | Mon, 10 Apr 2023 18:21:34 +0200 | [modified] src/test/java/doremi/AlbumTest.java |
| 55 | 533580e | Mon, 10 Apr 2023 18:21:33 +0200 | [created] src/test/java/doremi/AlbumTest.java |
| 56 | ce3dd1d | Mon, 10 Apr 2023 18:19:26 +0200 | Fix #2a |
| 57 | be88b27 | Mon, 10 Apr 2023 18:19:11 +0200 | [modified] src/main/java/doremi/domain/Article.java |
| 58 | 6e9899f | Mon, 10 Apr 2023 18:19:07 +0200 | [modified] src/main/java/doremi/domain/Article.java |
| 59 | 6b853e4 | Mon, 10 Apr 2023 18:18:21 +0200 | [modified] src/main/java/doremi/domain/Article.java |
| 60 | 6da0f26 | Mon, 10 Apr 2023 18:17:59 +0200 | [modified] src/main/java/doremi/domain/Article.java |
| 61 | ecb0a7a | Mon, 10 Apr 2023 18:17:29 +0200 | [modified] src/test/java/doremi/ArticleTest.java |
| 62 | 151e5c2 | Mon, 10 Apr 2023 18:17:20 +0200 | [created] src/test/java/doremi/ArticleTest.java |
| 63 | c13b362 | Mon, 10 Apr 2023 18:16:09 +0200 | Fix #1 |
| 64 | fe51014 | Mon, 10 Apr 2023 18:15:52 +0200 | [modified] src/main/java/doremi/controllers/IndexController.java |
| 65 | e42f65f | Mon, 10 Apr 2023 18:14:54 +0200 | [modified] src/test/java/doremi/IndexControllerTest.java |
| 66 | 81bfb45 | Mon, 10 Apr 2023 18:14:47 +0200 | [created] src/test/java/doremi/IndexControllerTest.java |
| 67 | 55f182b | Mon, 10 Apr 2023 18:14:22 +0200 | [modified] src/main/java/doremi/controllers/IndexController.java |
| 68 | 9f3e898 | Mon, 10 Apr 2023 18:14:05 +0200 | [modified] src/main/java/doremi/controllers/IndexController.java |
| 69 | 424e099 | Mon, 10 Apr 2023 18:13:43 +0200 | [modified] src/main/java/doremi/controllers/IndexController.java |
| 70 | dff1b1d | Mon, 10 Apr 2023 18:13:28 +0200 | [created] src/main/java/doremi/controllers/IndexController.java |
| 71 | 3191ec0 | Mon, 10 Apr 2023 18:12:33 +0200 | [modified] src/main/resources/templates/index.html |
| 72 | 9c8158d | Mon, 10 Apr 2023 18:12:24 +0200 | [modified] src/main/resources/templates/index.html |
| 73 | 79bbdbd | Mon, 10 Apr 2023 18:12:17 +0200 | [created] src/main/resources/templates/index.html |
| 74 | 8a0b7c7 | Mon, 10 Apr 2023 18:10:07 +0200 | Resume |

Fig. 1: Extract of the traces generated with the system

Concerning the question (RQ2). With the question resolution interface, we went from declarative and open trace generation to declarative but closed trace generation. Indeed, the command `fix` generates a trace whose message is normalized. Validating the command prevents the student from marking as solved a question that is not in the list given to the configuration. In this way, errors from the students are avoided. Thanks to its partial automation, traces generation becomes semi-automatic and closed. Many of the traces generated by LAWG are no longer dependent on student declaration, reducing the possibility of omissions.

Concerning the question (RQ3). The interactive system makes it easier to integrate into non-computer science areas. It abstracts the use of Git for trace generation so that it can be used for an audience unfamiliar with Git. The triggering of the automatic generation of traces when simply modifying a file makes it usable in all areas that require working on a computer, especially when the expected contributions are text productions.

## 5 Conclusion

We addressed the limitations of trace generation tools through Git repositories and introduced our interactive system, LAWG. This system automatically generates traces when students modify files and includes a command for validating

questions resolution. Our contribution improves the quantity, quality and easy of use of activity trace production. In future experiments, we plan to apply process mining techniques or analyze student behavior in relation to target behaviors. Our interactive system will be tested in computer learning courses this year, and we are prioritizing the collection of personal data, ensuring compliance with GDPR regulations. Access to personal data is restricted to teachers via the G4S dashboard, and anonymized data is made available to researchers. To simplify the process and ensure GDPR compliance, data anonymization will be integrated into the export feature of G4S.

## Bibliography

[1] Clifton, C., Kaczmarczyk, L.C., Mrozek, M.: Subverting the fundamentals sequence: using version control to enhance course management. ACM SIGCSE Bulletin (2007)
[2] Laadan, O., Nieh, J., Viennot, N.: Teaching operating systems using virtual appliances and distributed version control. In: Proceedings of the 41st ACM technical symposium on Computer science education (2010)
[3] Lawrance, J., Jung, S., Wiseman, C.: Git on the cloud in the classroom. Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13 (2013)
[4] 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 (2021)
[5] 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 (2014)
[6] Pons, M., Bruel, J.M., Raclet, J.B., Silvestre, F.: Finding behavioral indicators from contextualized commits in software engineering courses with process mining. In: Frontiers In Software Engineering Education, Springer (2023), to be published
[7] Raclet, J.B., Silvestre, F.: Git4School: A dashboard for supporting teacher interventions in software engineering courses. In: European Conference on Technology Enhanced Learning, Springer (2020)
[8] Robles, G., González-Barahona, J.M.: Mining student repositories to gain learning analytics. an experience report. In: 2013 IEEE Global Engineering Education Conference (EDUCON) (2013)
[9] Rodriguez-Rivera, G., Turkstra, J., Buckmaster, J., LeClainche, K., Montgomery, S., Reed, W., Sullivan, R., Lee, J.: Tracking large class projects in real-time using fine-grained source control. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (2022)
[10] Silvestre, F., Raclet, J.B.: Développement dirigé par les tests et revue de code par les pairs pour l'apprentissage de la programmation. In: Ludovia CH: 1ère édition sur le thème" Émanciper l'école et la société avec le numérique?" (2018)