

THÈSE de DOCTORAT

de l'UNIVERSITÉ TOULOUSE CAPITOLE

Présentée et soutenue par

Monsieur Mika PONS

Le 2 avril 2026

**exploitation des outils de gestion de version pour
accompagner la décision enseignante dans l'animation des
séquences pédagogiques**

École doctorale : **Mathématiques, Informatique et
Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche : **IRIT : Institut de Recherche en
Informatique de Toulouse**

Thèse dirigée par Monsieur Franck SILVESTRE et Monsieur Jean-
michel BRUEL

Composition du jury

Rapporteur : M. Jean-Marie GILLIOT

Rapporteur : M. Jean-Rémy FALLERI

Examinatrice : Mme Isabelle BLASQUEZ

Examineur : M. Frédéric AMBLARD

Directeur de thèse : M. Franck SILVESTRE

Co-directeur de thèse : M. Jean-Michel BRUEL



**UNIVERSITÉ
TOULOUSE
CAPITOLE**



RÉSUMÉ

L'enseignement supérieur fait face, au cours des dernières décennies, à une augmentation des effectifs rendant difficile le suivi individualisé des étudiant-es. Ce défi est particulièrement prégnant lors des séances de travaux pratiques (TP), où l'enseignant-e doit accompagner un grand nombre d'étudiant-es simultanément. Dans le contexte des TP de programmation, les outils de gestion de version, incontournables dans l'industrie, ont naturellement intégré les cursus pour former les étudiant-es aux pratiques professionnelles. Cependant, les enseignant-es ne disposent pas des outils nécessaires pour exploiter les traces d'apprentissage brutes issues de ces outils, difficilement visualisables et interprétables pour la prise de décision dans leur forme actuelle et pour de grands groupes d'étudiant-es.

Cette thèse, conduite selon une méthodologie de recherche orientée par la conception (*Design-Based Research*), explore la problématique suivante : (QR1) Dans quelle mesure un tableau de bord pour enseignant-es, construit à partir des traces générées par un outil de gestion de version et enrichies par des données contextuelles, permet-il de proposer des visualisations actionnables soutenant la prise de décision enseignante en temps réel? Une première contribution, Git4School, est un tableau de bord proposant des visualisations actionnables de l'activité des étudiant-es en TP à partir de leurs dépôts Git. Une étude qualitative basée sur les retours d'expérience d'enseignant-es l'ayant utilisé dans 5 cours différents (du DUT au Master), a validé sa pertinence aux yeux des enseignant-es pour l'orchestration des séances et l'identification de profils d'étudiant-es. Cependant, une étude quantitative fondée sur le *Process Mining* a révélé les limites des traces générées manuellement par les étudiant-es : elles ne permettent pas de mettre en lumière les comportements d'apprentissage au cours de la résolution des exercices. Ce constat a mené à notre seconde problématique, divisée en deux questions de recherche : (QR2.1) Comment améliorer l'exactitude, la complétude, la cohérence et l'actualité des traces générées, sur la base des événements de modifications de fichiers, de sorte à pouvoir identifier des comportements d'apprentissage complexes? (QR2.2) Comment garantir que la génération de traces améliorée ne perturbe ni les activités de l'étudiant-e ni celles de l'enseignant-e?

Une deuxième contribution, LAWG, propose un système interactif de génération de traces semi-automatiques encapsulant l'environnement de travail des étudiant-es tout en préservant l'intégrité de leur travail. Son évaluation à différents niveaux, une expérimentation en contexte écologique suivie d'analyses statistiques, couplée à une analyse quantitative par *Process Mining*, a permis de détecter le respect ou non de

protocoles de développement attendus, tels que le TDD (*Test-Driven Development*). Au regard du modèle de conscience de la situation, ces avancées correspondent à un approfondissement significatif du niveau 2 (compréhension) : les traces fines rendent observables les processus d'apprentissage et les stratégies des étudiant-es, levant partiellement le plafond de compréhension identifié lors de la première itération.

En conclusion, nos travaux présentent plusieurs contributions. D'une part, la conception et le développement de deux logiciels fonctionnels et open-source, déployables en l'état, dont les expérimentations ont démontré leur intérêt pour l'aide à la prise de décision enseignante dans le contexte de TP de programmation utilisant Git comme logiciel de gestion de version. D'autre part, des recommandations pour la conception d'outils de génération automatique de traces fines s'appuyant sur un logiciel de gestion de version. Enfin, nos perspectives de recherche visent à coupler ces contributions pour offrir un environnement plus complet de supervision, à compléter ces analyses par la collecte intégrée de données subjectives sur l'expérience vécue par les étudiant-es quant à la réalisation de la tâche d'apprentissage, et à envisager la généralisation de ces outils à d'autres contextes d'enseignement.

ABSTRACT

Higher education has faced increasing enrollment numbers over the last decades, making individualized student tracking complex. This challenge is particularly acute during practical sessions, where teachers must support a large number of students simultaneously. In the context of programming practicals, Version Control Systems, essential in the industry, have naturally been integrated into curricula to train students in professional practices. However, teachers lack the necessary tools to exploit the raw learning traces from these systems, which are difficult to visualize and interpret for decision-making in their current form and for large student cohorts.

This thesis, conducted using a Design-Based Research methodology, explores the following problem : (RQ1) To what extent does a learning analytics dashboard for teachers, built from traces generated by a version control tool and enriched with contextual data, make it possible to propose actionable visualizations supporting real-time teacher decision-making ?

A first contribution, Git4School, is a dashboard offering actionable visualizations of student activity during practicals based on their Git repositories. A qualitative study based on feedback from teachers using it in 5 different courses (from undergraduate to master's level) validated its relevance for session orchestration and student profiling. However, a quantitative study based on *Process Mining* revealed the limitations of manually generated traces : they fail to highlight

learning behaviors during exercise resolution. This finding led to our second problem, divided into two research questions : (RQ2.1) How to improve the accuracy, completeness, consistency, and timeliness of generated traces, based on file modification events, so as to identify complex learning behaviors? (RQ2.2) How to ensure that improved trace generation disrupts neither the student's nor the teacher's activities?

A second contribution, LAWG, proposes an interactive semi-automatic trace generation system encapsulating the students' work environment while preserving the integrity of their work. Its evaluation at different levels, an experiment in an ecological context followed by statistical analyses, coupled with a quantitative analysis using *Process Mining*, successfully allowed for the detection of adherence to expected development protocols, such as TDD (*Test-Driven Development*). Through the lens of the situation awareness model, these advances correspond to a significant deepening of level 2 (comprehension) : fine-grained traces make learning processes and student strategies observable, partially lifting the comprehension ceiling identified during the first iteration.

In conclusion, this work presents several contributions. On the one hand, the design and development of two functional and open-source software tools, deployable as is, whose experiments have demonstrated their value for teacher decision support in the context of programming practicals using Git as a version control system. On the other hand, recommendations for the design of fine-grained automatic trace generation tools based on a version control system. Finally, our research perspectives aim to couple these contributions to offer a more complete supervision environment, to complement these analyses with the integrated collection of subjective data on the students' lived experience regarding the learning task, and to consider the generalization of these tools to other teaching contexts.

REMERCIEMENTS

La réalisation de cette thèse n'aurait pas été possible sans le soutien, l'expertise et la bienveillance de nombreuses personnes que je tiens à remercier ici.

Mes premiers remerciements s'adressent aux membres de mon jury de thèse pour l'intérêt qu'elles et ils ont porté à mon travail et pour le temps consacré à la lecture de ce document. Je remercie tout particulièrement Jean-Marie Gilliot pour sa forte implication et ses retours constructifs qui ont grandement contribué à améliorer la qualité de ce manuscrit.

Je tiens bien évidemment à exprimer ma profonde gratitude envers mes directeurs de thèse, Franck Silvestre, Jean-Michel Bruel et Jean-Baptiste Raclet, avec qui j'ai commencé à travailler bien avant ces travaux, et dont la collaboration a toujours été très riche et pleine de sens. Merci pour votre confiance, votre encadrement avisé, votre disponibilité, votre soutien constant et votre grande, si ce n'est extrême, patience tout au long de ces années. Tout cela m'a permis de tenir jusqu'au bout malgré les moments de doute et de difficulté, sur le plan scientifique comme sur le plan personnel.

Je souhaite remercier Emmanuelle Claeys pour son aide précieuse au début de ma thèse. Ses conseils sur les aspects liés au *Process Mining* m'ont permis de démarrer mes travaux sur de solides fondations. Je tiens également à remercier Érik Martin-Dorel pour son enthousiasme concernant le développement de Git4School, ainsi que Benoît Chaluleau de m'avoir fait confiance en intégrant LAWG dans son cours afin de l'expérimenter.

Mes remerciements vont également à l'ensemble des membres de mes équipes de recherche pour nos échanges scientifiques toujours très enrichissants et les moments chaleureux passés durant les séminaires d'équipe ou en dehors du laboratoire. L'atmosphère conviviale dans laquelle j'ai eu la chance d'évoluer a grandement contribué à l'aboutissement de cette thèse. J'associe également à ces remerciements la communauté des EIAH/TEL pour les échanges inspirants et bienveillants lors des diverses rencontres scientifiques.

Le bon déroulement de cette thèse doit aussi beaucoup aux personnels administratifs. Je remercie les équipes administratives de l'IRIT, de l'Université Toulouse Capitole et de l'école doctorale EDMITT pour leur efficacité, leur bienveillance et leur souplesse dans la gestion de mes démarches logistiques et d'inscription.

Cette aventure n'aurait pas eu la même saveur sans mes collègues et ami-es doctorant-es, post-docs, ingénieur-es, alternant-es et stagiaires de TALLIPSE.

Merci d'abord à Rialy, avec qui j'ai partagé mon premier bureau et passé les premiers moments de ma thèse, et qui a toujours su m'apporter les conseils les plus objectifs et rationnels.

Merci également à Alejandra, Aurélien, Elen, Esteban, Gabriel, José, Luis, Maéva, Nathan, Reda, Tanguy et Victor pour l'entraide, l'ambiance de travail, les discussions sur les sujets les plus farfelus et les moments de décompression essentiels, que ce soit au travers des nombreuses parties de baby-foot, des défis journaliers d'échecs et de Cémantix, ou des moments passés autour d'un verre.

Un merci spécial au bureau 258, qui a sans conteste été le meilleur bureau d'accueil grâce aux personnes incroyables qui y ont posé leurs affaires : Malo, Maëlle, Sandra, Anaëlle, Manon, Maria, Rémi et Lisa, par intermittence. L'atmosphère qui y régnait est véritablement l'une des raisons pour lesquelles j'étais toujours heureux de venir travailler.

Je tiens à adresser un remerciement tout particulier à Esther, avec qui j'ai partagé toute cette aventure, devenue au fil du temps une amie très chère sans laquelle la fin de cette thèse n'aurait pas été la même. Nous avons traversé ensemble les meilleurs moments comme les périodes les plus difficiles que l'on peut rencontrer au cours d'un tel parcours. Tu connais la place essentielle que tu as prise dans ma vie depuis notre rencontre, et l'importance de ton soutien dans toutes les étapes de ce long périple qu'est la thèse. Tu es sans aucun doute la meilleure co-bureau que je pouvais avoir.

Enfin, je remercie ma famille et mes amis en dehors du labo. Merci pour votre présence, votre patience à toute épreuve lors de mes moments d'indisponibilité, ainsi que pour votre soutien moral et votre intérêt pour ce que je faisais. Les vacances, les multiples séances d'escalade, de jeux de société, les randonnées et toutes les discussions existentielles m'ont aidé à tenir et à prendre du recul face à cette thèse.

NOTE SUR L'ÉCRITURE ÉGALITAIRE

Pour l'écriture du manuscrit, nous avons décidé de suivre les recommandations du *Haut conseil à l'égalité entre les femmes et les hommes* concernant l'écriture égalitaire¹. Cependant, nous avons essayé d'éviter autant que possible l'usage du point médian.

Bien qu'il n'y ait pas de résultats scientifiques indiquant que le point médian pose problème pour les personnes dyslexiques, la Fédération Française des Dys (FFDys) préconise de l'utiliser avec parcimonie par principe de précaution². Son usage peut également constituer une difficulté de lecture pour les personnes malvoyantes utilisant un dispositif de lecture automatisée³.

Pour cela, nous avons d'abord préféré l'utilisation de noms épicènes autant que possible. Quand cela n'est pas possible, nous utiliserons la double flexion lorsque la différence entre la forme féminine et masculine du nom se fait par la substitution d'une syllabe à une autre (comme "acteur et actrice" au lieu de "acteur·rice").

Dans le cas de la double flexion, nous adopterons l'ordre alphabétique pour déterminer l'ordre d'apparition des termes. Nous appliquerons également l'accord de proximité, qui consiste à accorder grammaticalement l'adjectif ou le participe passé avec le substantif le plus proche.

Enfin, pour tous les autres cas, nous utiliserons le point médian, et un seul pour marquer la forme féminine et plurielle d'un nom ("étudiant·es" et pas "étudiant·e·s").

La double flexion consiste à écrire la forme féminine et masculine d'un nom sans point médian ou toute autre contraction du genre, comme "chercheurs et chercheuses".

1. <https://www.haut-conseil-egalite.gouv.fr/guide-pour-une-communication-publique-sans-stereotypes-de-sex>
2. <https://www.ffdys.com/actualites/ecriture-inclusive-et-dyslexie-lavis-de-la-ffdys/>
3. <https://access42.net/langage-non-genre-accessibilite/#:~:text=Tout%20%C3%A9crire,rendre%20le%20propos%20plus%20compr%C3%A9hensible.>

TABLE DES MATIÈRES

1	Introduction	1
1.1	Contexte scientifique	1
1.1.1	Unité et équipes de recherche	1
1.1.2	EIAH et Learning Analytics	3
1.1.3	Motivations et posture épistémologique	4
1.1.4	Méthode de conduite de la recherche	5
1.2	Questions de recherche et contributions	5
1.2.1	Objectifs et questions de recherche	5
1.2.2	Contributions et publications	7
1.3	Organisation du manuscrit	8
1.3.1	Itérations	8
1.3.2	Structure du manuscrit	9
I	Itération 1 : tableau de bord pour l'aide à la prise de décision enseignante au cours d'une séance de TP	
2	État de l'art	13
2.1	Les Learning Analytics au service de l'enseignement	14
2.1.1	Définitions et périmètre	14
2.1.2	Le cycle des Learning Analytics	14
2.1.3	Conception de situation d'apprentissage et alignement pédagogique	15
2.2	Les tableaux de bord d'apprentissage	16
2.2.1	Définition et caractéristiques	16
2.2.2	Panorama des tableaux de bord : tendances et exemples	16
2.2.3	Tableaux de bord destinés aux enseignant·es	18
2.2.4	Orchestration et tableaux de bord en temps réel	19
2.3	De la donnée à l'action : <i>insights</i> actionnables et construction de sens	19
2.3.1	Le fossé entre conscience et action	19
2.3.2	La notion d' <i>insight</i> actionnable	20
2.3.3	Du <i>sensemaking</i> à la conscience de la situation	21
2.4	La conscience de la situation comme cadre théorique	22
2.4.1	Le modèle d'Endsley (1995)	22
2.4.2	Application à l'enseignement et aux tableaux de bord	22
2.4.3	Synthèse du cadre théorique	24
2.5	Traces d'apprentissage et enseignement de la programmation	24
2.5.1	L'enseignement de la programmation : contexte et défis	24

2.5.2	Les traces issues des systèmes de gestion de version	25
2.5.3	Positionnement et lacunes identifiées	26
2.6	Analyse des comportements d'apprentissage par le Process Mining	27
2.6.1	Le Process Mining dans l'éducation	27
2.6.2	Le Process Mining en génie logiciel	28
2.6.3	Le Process Mining dans l'enseignement du génie logiciel	28
2.7	Synthèse et formulation de la question de recherche . .	29
3	Conception de Git4School, tableau de bord	31
3.1	Contexte	31
3.1.1	Git, GitHub et GitHub Classroom	31
3.1.2	Le protocole Peer Review and Test-Driven Development Learning (PRaTDL)	33
3.2	Git4School	36
3.2.1	Éléments d'architecture	36
3.2.2	Configuration	39
3.2.3	Fonctionnalités	41
4	Évaluation qualitative : retours d'expérience des enseignant-es	47
4.1	Contexte des expérimentations	48
4.2	Étude de deux cas représentatifs	48
4.2.1	Étude de cas 1	49
4.2.2	Étude de cas 2	50
4.3	Résultats et discussion	50
4.3.1	Déclenchement des interventions planifiées	50
4.3.2	Obtenir un feedback sur l'efficacité d'une intervention planifiée	51
4.3.3	Identification des étudiant-es en difficulté	51
4.3.4	Information sur l'activité des étudiant-es en dehors des cours	52
4.3.5	Qualification de la difficulté d'un exercice	52
4.3.6	Comparaison de niveaux entre groupes d'étudiant-es	53
4.3.7	Synthèse des observations et limites	53
4.3.8	Analyse au regard de la conscience de la situation	54
4.3.9	Limites de l'étude	55
5	Évaluation quantitative : étude comportementale	57
5.1	Le Process Mining	58
5.1.1	Concepts fondamentaux	58
5.1.2	Mise en œuvre du Process Mining avec BupaR	59
5.1.3	Types de visualisations offertes par BupaR	60
5.1.4	Application à l'apprentissage de la programmation	63
5.2	Description des jeux de données	64
5.2.1	Prétraitement des données	65
5.2.2	Analyse avec BupaR	67

5.3	Résultats	68
5.3.1	Activité : question résolue	69
5.3.2	Activité : type de <i>commit</i>	72
5.4	Discussion	72
5.4.1	Identification de comportements des étudiant-es	72
5.4.2	Facteurs influençant le suivi du processus de résolution de question	74
5.4.3	Nuances dans l'interprétation des données	75
5.5	Bilan de l'itération et nouvelle problématique	75
5.5.1	Synthèse des apports	75
5.5.2	Limites convergentes	76
5.5.3	Problématique induite	77
 II Itération 2 : génération de traces d'apprentissage		
6	État de l'art	81
6.1	Collecte et analyse de traces en programmation	82
6.1.1	Enjeux et état des lieux	82
6.1.2	Niveaux de granularité des traces	82
6.1.3	Cadre de référence pour l'instrumentation	83
6.2	Représentation et gestion des traces d'apprentissage	83
6.2.1	xAPI et les Learning Record Stores	83
6.2.2	Les Systèmes à Base de Traces (SBT) et kTBS	84
6.2.3	ProgSnap2 : un format standardisé pour les don- nées de processus de programmation	85
6.2.4	Git comme générateur et système à base de traces d'apprentissage	85
6.3	Génération de traces pour le suivi fin de l'activité	87
6.3.1	Approches fondées sur les journaux Git	87
6.3.2	Instrumentation d'environnements de dévelop- pement	88
6.3.3	Synthèse et positionnement	89
6.4	Qualité des données : cadres théoriques	90
7	Conception de LAWG	93
7.1	Configuration	94
7.2	Encadrement des sessions de travail	95
7.3	Génération automatique des traces d'activité	96
7.4	Interface de résolution d'un exercice	97
7.5	Historisation des <i>commits</i> automatiques sur une branche dédiée	98
8	Évaluation de LAWG en trois étapes	101
8.1	Évaluation en situation simulée (Vérification)	102
8.2	Évaluation en situation réelle (Validation)	103
8.3	Expérimentation en contexte écologique	104
8.3.1	Méthode	104
8.3.2	Participants	105
8.3.3	Protocole	106

8.3.4	Préparation de l'expérimentation	106
8.3.5	Déroulement de l'expérimentation	107
8.3.6	Matériel	107
8.3.7	Méthodes d'analyse	108
9	Résultats et discussion	111
9.1	Résultats	111
9.1.1	Indicateurs sur la qualité des traces	111
9.1.2	Identification du Test Driven Development (TDD)	131
9.1.3	Formulaire SUS et retours des étudiant-es	138
9.1.4	Entretiens avec les enseignants	139
9.2	Discussion	144
9.2.1	Des nouvelles traces exploitables pour des nouveaux comportements	144
9.2.2	Analyse au regard de la conscience de la situation	146
9.2.3	Un besoin de séparer les traces générées avéré	147
9.2.4	Une adhésion modérée malgré des bénéfices reconnus	148
9.2.5	Recommandations de conception	148
9.2.6	Limites	149
9.3	Bilan de l'itération	151
9.3.1	Synthèse des apports	151
9.3.2	Bilan au regard de la conscience de la situation	151
 III Conclusion et pistes de travaux futurs		
10	Conclusion et perspectives	155
10.1	Rappel du contexte et de la problématique	155
10.2	Synthèse des contributions	156
10.2.1	Itération 1 : Visualiser pour agir avec Git4School	156
10.2.2	Itération 2 : Améliorer la qualité des données à la source avec LAWG	157
10.3	Discussion et mise en perspective	158
10.3.1	Le paradoxe de la richesse des données	158
10.3.2	Dilemme entre automatisation et apprentissage technique	158
10.3.3	L'alignement pédagogique révélé par les traces	159
10.4	Limites	159
10.5	Pistes de travaux futurs	159
10.5.1	Court terme : Travaux en cours et consolidation	160
10.5.2	Moyen terme : Robustesse et extension des capacités d'analyse	162
10.5.3	Long terme : Vers la généralisation et l'auto-régulation	163
 IV Appendix		
	Bibliographie	169

TABLE DES FIGURES

Figure 3.1	Schéma illustrant les interactions entre Git local et GitHub	31
Figure 3.2	Illustration du protocole PRaTDL	33
Figure 3.3	Extrait d'un sujet de Travaux Pratiques (TP) typique pouvant être utilisé dans le cadre du protocole PRaTDL	34
Figure 3.4	Diagramme d'architecture et de communication de Git4School	38
Figure 3.5	Machine à états représentant le processus d'authentification de Git4School	38
Figure 3.6	Capture d'écran de la configuration d'un <i>assignment</i> de Git4School	40
Figure 3.7	Capture d'écran de l'ajout des dépôts Git des étudiant-es dans Git4School	40
Figure 3.8	Capture d'écran du format standardisé du README dans Git4School	41
Figure 3.9	Vue d'ensemble du tableau de bord de Git4School	42
Figure 3.10	Captures d'écran des fenêtres d'ajout de données contextuelles dans Git4School	43
Figure 3.11	Capture d'écran du <i>diff</i> ouvert sur GitHub en cliquant sur un <i>commit</i> d'étudiant-e dans Git4School	44
Figure 3.12	Graphe des <i>commits</i> de chaque étudiant-e dans Git4School	45
Figure 3.13	Graphe du taux d'achèvement des questions dans Git4School	45
Figure 4.1	Capture d'écran de la visualisation des <i>commits</i> par étudiant dans Git4School dans le cadre d'un des cas d'étude	52
Figure 5.1	Schéma illustrant les concepts du Process Mining	59
Figure 5.2	Exemple d'un processus de résolution d'exercices par une étudiante	60
Figure 5.3	Exemple d'une carte de processus	61
Figure 5.4	Exemple d'une matrice de processus	61
Figure 5.5	Exemple d'un diagramme en points	62
Figure 5.6	Exemple d'un explorateur de traces	62
Figure 5.7	Exemple d'un spectre de performance	63
Figure 5.8	Modèles de la structure des jeux de données	66
Figure 5.9	<i>Process map</i> de résolution des questions pendant une séance de travaux pratiques	69

Figure 5.10	Un extrait de la <i>process map</i> montrant un processus de résolution de questions bien suivi . .	70
Figure 5.11	Explorateurs de traces des deux jeux de données <i>PDO_MVC</i> et <i>FilmProvider</i>	71
Figure 5.12	<i>Process map</i> du profil de performance (utilisant la fonction <i>moyenne</i>)	72
Figure 5.13	Diagrammes en points des deux jeux de données <i>FilmProvider</i> et <i>M2DL</i>	73
Figure 5.14	Matrice de précédence basée sur le type de <i>commit</i> comme activité	74
Figure 7.1	Exemple de l'utilisation du système interactif LAWG lors de la réalisation d'exercices de travaux pratiques	96
Figure 7.2	Captures d'écran des suggestions de LAWG . .	97
Figure 7.3	Schéma illustrant le mécanisme d'historisation des commits automatiques sur une branche dédiée (auto) sans perturber la branche principale.	99
Figure 8.1	Extrait du sujet posé aux étudiant-es pour l'expérimentation	105
Figure 8.2	Diagramme de Sankey montrant le nombre de participant-es à chaque étape de l'expérimentation	105
Figure 8.3	Calendrier de l'expérimentation	106
Figure 9.1	Frise chronologique des <i>commits</i> d'un-e étudiant-e illustrant deux sessions de travail dans une même journée	114
Figure 9.2	Comparaison des boîtes à moustaches des temps moyens entre deux <i>commits</i> pour les groupes contrôle et expérimental.	116
Figure 9.3	Tableau des statistiques des temps moyens entre deux <i>commits</i> pour les groupes contrôle et expérimental.	117
Figure 9.4	Distribution globale de la densité de commits par question (diagramme en violon)	120
Figure 9.5	Comparaison groupée de la densité de commits par question selon les groupes	120
Figure 9.6	Comparaison visuelle des frises chronologiques des <i>commits</i> des étudiant-es avec et sans LAWG.	122
Figure 9.7	Distribution de la couverture temporelle des commits selon le groupe contrôle et le groupe expérimental	124
Figure 9.8	Distribution de la couverture temporelle pour 10 000 groupes expérimentaux simulés (n=5) en comparaison du score du groupe contrôle .	125
Figure 9.9	Cartes de processus issues de l'analyse des traces selon l'utilisation ou non de LAWG . . .	128
Figure 9.10	Explorateur de traces pour les deux groupes .	130

Figure 9.11	Carte de processus (relative-case) représentant le processus de résolution de question pour le groupe expérimental avec LAWG	133
Figure 9.12	Comparaison des cartes de processus illustrant l'application ou non du TDD	135
Figure 9.13	Carte de processus illustrant un-e étudiant-e suivant le TDD	135
Figure 9.14	Explorateur de traces illustrant un-e étudiant-e suivant le TDD	136
Figure 9.15	Relation entre la fréquence de suivi du TDD et la note obtenue par les étudiant-es.	137
Figure 9.16	Calcul du score moyen SUS (80.42) pour LAWG	138
Figure 9.17	Plan suivi pour les entretiens semi-directifs avec les enseignants	139
Figure 9.18	Matrice à condensés présentant la synthèse des propos de chaque enseignant pour chaque thème identifié lors de l'analyse de contenu des entretiens	140
Figure 10.1	Comparaison des visualisations de commits dans Git4School sans et avec agrégation.	161
Figure 10.2	Capture d'écran de la fenêtre affichée à l'étudiant-e lui demandant quel est son niveau d'émotions positives ressenti	161

LISTE DES TABLEAUX

Table 4.1	Nombre de dépôts en fonction du niveau d'études	48
Table 5.1	Nombre de dépôts en fonction du niveau d'études lors de l'analyse quantitative	65
Table 5.2	Construction des journaux d'événements selon l'activité	67
Table 6.1	Synthèse des approches de génération de traces pour le suivi de l'activité de programmation. .	89
Table 1	Description des jeux de données	167

LISTINGS

Listing 7.1	Structure du fichier settings.yml	94
Listing 7.2	Structure du fichier IDENTITY.json	95

Listing 7.3	Pseudo-code de la synchronisation des commits automatiques sur une branche dédiée	99
-------------	--	----

ACRONYMES

TP	Travaux Pratiques
PM	Process Mining
TDD	Test Driven Development
SUS	System Usability Scale
IDE	Integrated Development Environment
DUT	Diplôme Universitaire de Technologie
LA	Learning Analytics
DBR	Design-Based Research
TBA	Tableau de Bord d'Apprentissage
EIAH	Environnements Informatiques pour l'Apprentissage Humain
LMS	Learning Management System
DAS	Domaines d'Application Stratégiques
PRaTDL	Peer Review and Test-Driven Development Learning
EPM	Educational Process Mining
MVC	Model-View-Controller
MOOC	Massive Open Online Course
ISO	Organisation Internationale de Normalisation

INTRODUCTION

1.1	Contexte scientifique	1
1.1.1	Unité et équipes de recherche	1
1.1.2	EIAH et Learning Analytics	3
1.1.3	Motivations et posture épistémologique	4
1.1.4	Méthode de conduite de la recherche	5
1.2	Questions de recherche et contributions	5
1.2.1	Objectifs et questions de recherche	5
1.2.2	Contributions et publications	7
1.3	Organisation du manuscrit	8
1.3.1	Itérations	8
1.3.2	Structure du manuscrit	9

1.1 CONTEXTE SCIENTIFIQUE

1.1.1 Unité et équipes de recherche

Ce manuscrit documente le travail fait durant trois années de thèse débutant le 1^{er} octobre 2021 sous la supervision de Jean-Michel Bruel, Jean-Baptiste Raclet et Franck Silvestre, à l’Institut de Recherche en Informatique de Toulouse (IRIT).

L’IRIT, Unité Mixte de Recherche (UMR 5505), est l’un des principaux centres de recherche en informatique en France, avec plus de 600 membres, permanents ou non-permanents et des collaborateurs et collaboratrices extérieures. Le laboratoire bénéficie d’une reconnaissance régionale et nationale pour son rôle structurant dans les évolutions du paysage scientifique, en particulier en Occitanie. Il dépend de différentes tutelles (CNRS, Universités de Toulouse) et est actif dans de grands projets régionaux et nationaux tels que LabEx CIMI, IRT Saint-Exupéry, et SAT TTT.

La recherche dans ce laboratoire est organisée autour de cinq grands axes scientifiques :

- Conception et construction de systèmes (fiables, sûrs, adaptatifs, distribués, communicants, dynamiques. . .),
- Modélisation numérique du monde réel,
- Concepts pour la cognition et l’interaction,
- Étude des systèmes autonomes adaptatifs à leur environnement,
- Passage de la donnée brute à l’information intelligible.

L’unité est structurée en 7 départements de recherche, regroupant 24 équipes. Les Domaines d’Application Stratégiques (DAS) sont des

domaines transverses où se concrétisent les recherches du laboratoire. Ces DAS sont ouverts à plusieurs équipes, et les chercheurs et chercheuses sont libres d'y participer en fonction de leurs affinités scientifiques ou de leurs autres intérêts, tels que l'enseignement. L'IRIT compte 6 DAS :

- Santé, Autonomie, Bien-être,
- Ville Intelligente,
- Aéronautique, Espace, Transports,
- Média sociaux numériques et diffusion de l'information,
- e-Éducation,
- Cybersécurité, Sécurité des biens et des personnes.

J'ai rejoint les départements ICI¹ et FSL² au sein desquels j'ai fait partie des équipes TALENT, SM@RT et ACADIE. Si ces équipes travaillent sur des thématiques différentes, elles sont toutes les trois engagées dans le DAS "e-Éducation", sous la responsabilité de Jean-Baptiste Raclet et Franck Silvestre. Ce DAS vise à promouvoir la recherche et les collaborations autour des problématiques de l'e-Éducation, principal domaine de recherche de l'équipe TALENT.

L'équipe TALENT (Teaching And Learning ENhanced by Technologies) développe en particulier 3 axes de recherche dans le domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH) :

Ingénierie d'EIAH pour supporter l'apprentissage actif. Les activités de recherche dans cet axe se concentrent sur la conception et la mise en œuvre d'environnements numériques pour améliorer l'apprentissage, mais aussi sur leur évaluation dans des situations d'apprentissage authentiques. Les recherches portent sur la manière de soutenir les méthodes pédagogiques actives dont l'objectif est de maximiser l'engagement des apprenant-es dans le processus d'apprentissage. En particulier, les études visent à promouvoir un apprentissage en profondeur par l'utilisation de visualisations et d'outils de réflexion, de méthodes telles que l'instruction par les pairs et la confrontation de points de vue, ou d'approches exploratoires fondées sur le développement d'activités pratiques.

Learning Analytics (LA) pour l'apprentissage personnalisé et auto-régulé. L'objectif principal de cet axe est d'étudier comment les données d'apprentissage peuvent être exploitées pour soutenir les processus d'enseignement et d'apprentissage. Des méthodes et des mécanismes innovants sont explorés pour concevoir, développer et promouvoir des solutions technologiques présentant une plus-value éducative du point de vue des enseignant-es et des étudiant-es, et où les méthodes et les techniques issues du *big data* sont utilisées pour fournir des outils et des systèmes efficaces. L'objectif est triple : (1) collecter l'expérience de l'utilisateur dans les environnements éducatifs

1. Intelligence Collective, Interaction

2. Fiabilité des Systèmes et des Logiciels

numériques ; (2) appliquer des techniques de fouille et d’exploration de données (telles que le *process/pattern mining* par exemple) pour identifier les comportements d’apprentissage favorisant la réussite des apprenant-es ; et (3) développer des capacités intelligentes pour guider les apprenant-es dans leurs tâches sur la base des connaissances déduites de la fouille des données comportementales.

Compétences numériques et développement de l’éducation du futur. Cet axe étudie les compétences que les enseignant-es, étudiant-es et institutions devraient développer pour s’adapter aux futurs défis de l’éducation. Le type de questions de recherche posées dans cet axe concerne le soutien à l’acquisition et au développement des compétences du 21^{ème} siècle, ainsi qu’à la transformation des établissements d’enseignements pour y parvenir.

1.1.2 EIAH et Learning Analytics

Nos travaux s’inscrivent dans le domaine des EIAH, dont l’objectif est de concevoir des systèmes informatiques pour soutenir l’apprentissage humain. Il s’agit d’un domaine transversal à plusieurs disciplines puisqu’il mobilise des connaissances, des outils et des théories de l’informatique, de la psychologie, des sciences de l’éducation et de la pédagogie. Les EIAH sont aujourd’hui presque systématiquement intégrés dans l’enseignement supérieur, notamment à travers l’utilisation de Learning Management System (LMS) tels que Moodle, Open EDX [95, 97], ou encore Blackboard [16], très populaire aux États-Unis. Ces plateformes permettent de partager du contenu pédagogique à des groupes d’apprenant-es tout en soutenant l’apprentissage par le biais de la collaboration au sein d’une communauté d’apprenant-es. Elles se retrouvent au cœur de différents projets de recherche car elles permettent notamment de récolter un grand nombre de données sur les étudiant-es en interaction avec les plateformes. Ces données, appelées *traces d’apprentissage* [90], sont au centre de nombreuses recherches. La “mesure, la collecte, l’analyse et la communication de données relatives aux apprenant-es et à leur contexte d’apprentissage, dans le but de comprendre et d’optimiser l’apprentissage et les environnements dans lesquels il se produit” [66] constitue un des champs de recherche majeur en EIAH : les Learning Analytics (LA). La communication des données peut se faire à destination des apprenant-es, notamment dans le cadre d’apprentissage auto-régulé [54, 74], ou des enseignant-es pour l’aide à la régulation des apprentissages par exemple [112]. Nos travaux présentés dans ce manuscrit s’inscrivent dans le champ des LA, et visent plus précisément à répondre à des questions relatives à la présentation d’informations issues de traces d’apprentissage pour aider les enseignant-es à prendre des décisions pédagogiques. Pour cela, nous nous sommes intéressés aux tableaux de bord d’appren-

tissage, ou Tableau de Bord d'Apprentissage (TBA), à destination des enseignant·es.

1.1.3 *Motivations et posture épistémologique*

Lors de mon cursus scolaire, j'ai été confronté à un grand nombre de cours pour lesquels je ressentais que le modèle pédagogique ne fonctionnait ni avec moi, ni avec la plupart de mes camarades, ne suscitant pas d'intérêt particulier autre que l'enjeu de la note. Cependant, certains cours fonctionnant différemment ont éveillé en moi une curiosité et un intérêt concernant les problématiques de pédagogie et de didactique. Ces cours semblaient être articulés de façon à réellement favoriser un apprentissage profond et guidé non plus par la motivation extrinsèque de la note mais bien par une motivation intrinsèque [26] liée à la curiosité et l'intérêt pour les notions à apprendre. C'est avec cela en tête que j'ai abordé la thèse avec une envie de produire des contributions utiles concrètement pour les étudiant·es, plus particulièrement dans l'enseignement supérieur. La question de l'opérationnalisation des résultats de recherche se pose alors. Si les chercheurs et les chercheuses prennent part activement à la diffusion de leur recherche en participant à des conférences, en écrivant des articles de revues, en organisant des ateliers, je pense que l'intégration pérenne de leurs outils s'ils ont montré des résultats positifs devrait être un enjeu important. C'est pour moi d'autant plus vrai dans le domaine de l'éducation et des EIAH dont l'objectif final est l'amélioration de l'apprentissage.

Cette vision se retrouve dans la démarche de recherche-action [67] initiée par Kurt Lewin en 1946. Pour Lewin, il est nécessaire pour les chercheurs et les chercheuses d'appliquer un changement sur le système étudié pour répondre à un problème concret et en déduire des connaissances. Différentes approches en découlent selon que le changement sur le système soit nécessaire en amont pour produire des connaissances ou en aval à la lumière des connaissances produites. Dans le premier cas, on peut trouver, entre autres, la Science Action ou la recherche-intervention. Dans le deuxième cas, des approches psychosociologiques, "militantes" ou encore des approches "coopératives" comme la recherche-action pragmatiste [43]. Je me situe dans la démarche de recherche-action, et particulièrement entre la recherche-intervention qui place l'activité de recherche au centre de la production des connaissances mais aussi du changement du système, et la recherche-action pragmatiste qui a vocation à changer le système de façon pérenne. Pour appliquer ces principes dans ma recherche, je me suis tourné vers la méthode de conduite de la recherche Design-Based Research (DBR).

1.1.4 *Méthode de conduite de la recherche*

Le DBR est une méthode de conduite de la recherche itérative utilisée depuis les années 2000 dans le domaine des EIAH [10, 68]. Elle vise à répondre à une problématique concrète rencontrée par les praticien·nes, c'est-à-dire les bénéficiaires et utilisateurs et utilisatrices des méthodes et outils développés par la recherche. Simultanément, elle produit de nouvelles connaissances scientifiques, des méthodes ou des outils. Elle met en œuvre un processus itératif dont le point de départ de chaque cycle est une problématique co-construite avec les acteurs et actrices de terrain (enseignant·es et/ou étudiant·es, etc.). En fin de cycle, sont produits de nouveaux éléments utiles aux acteurs et actrices sur la problématique, de nouvelles connaissances scientifiques et de nouvelles hypothèses nourrissant le prochain cycle. Comme l'indiquent WANG et HANNAFIN dans [118], chaque cycle se compose des phases suivantes : analyse de la problématique, conception et développement d'un prototype permettant de vérifier les hypothèses énoncées dans la phase d'analyse, et évaluation du prototype.

1.2 QUESTIONS DE RECHERCHE ET CONTRIBUTIONS

1.2.1 *Objectifs et questions de recherche*

Lors de séances de TP, il est important pour l'enseignant·e de pouvoir avoir une idée de la progression des étudiant·es tout au long de la séance. Cela permet d'organiser certaines phases comme faire un point sur une notion cruciale abordée dans une question, faire une revue des solutions trouvées par le groupe ou distribuer des corrections. Il est aussi utile de pouvoir identifier individuellement la progression pour apporter une aide personnalisée à celles et ceux qui en ont besoin au bon moment.

Or, dans une classe de 20 à 30 personnes, il est difficile de suivre la progression de chacun·e en passant derrière chaque étudiant·e. Même s'il existe un moyen de voir l'état courant du travail de chaque étudiant·e, il est fastidieux de le vérifier pour tout le monde un à un à plusieurs reprises.

Les TBA peuvent apporter une solution à ce problème en regroupant un ensemble d'indicateurs sous la forme de visualisations en temps réel, permettant à l'enseignant·e de prendre des décisions éclairées [79, 112].

Dans notre cas, nous nous intéressons aux cas de TP en Génie Logiciel qui se concrétisent par des activités de production de code de la part des étudiant·es. Le contexte de ces TP nous permet de suivre l'avancée des étudiant·es pour chaque question en accédant à leur dépôt distant, synchronisé avec leur travail local.

Dans ce contexte, plutôt que de développer et maintenir une infrastructure spécifique de collecte de données, nous exploitons la brique logicielle Git déjà présente dans l’environnement d’apprentissage. Utilisée initialement pour le versionnement du code et comme objet d’apprentissage, nous la détournons pour générer et structurer les traces nécessaires au suivi des étudiant-es. D’un point de vue technique, ce choix nous fait bénéficier de la robustesse d’outils industriels éprouvés et d’un mécanisme de persistance standardisé, facilitant l’accès aux traces sans nécessiter de déploiement logiciel complexe. Cela permet de suivre la progression en minimisant les perturbations des habitudes de travail et la charge de l’enseignant-e, tout en s’intégrant le plus simplement possible à l’environnement d’apprentissage.

Ce qui nous amène à la question de recherche suivante :

Question de recherche 1

Dans quelle mesure un tableau de bord pour enseignant-es, construit à partir des traces générées par un outil de gestion de version et enrichies par des données contextuelles, permet-il de proposer des visualisations actionnables soutenant la prise de décision enseignante en temps réel ?

Au cours de la première itération d’un cycle de DBR présentée dans cette thèse, un tableau de bord a été conçu répondant en partie à ces besoins. Cependant, elle a aussi révélé que les traces générées manuellement par les étudiant-es manquaient de fiabilité et de richesse pour analyser finement les comportements d’apprentissage. Cela soulève la problématique suivante : **comment améliorer la génération de traces pour raffiner le suivi des étudiant-es en rendant visibles leurs comportements d’apprentissage au cours des résolutions d’exercices de TP de programmation ?**

Cette problématique est étudiée dans la suite sous la forme de deux questions de recherche. La première question se penche sur la qualité des traces nécessaires pour capturer l’activité des étudiant-es. Il s’agit de dépasser la simple observation des résultats (la résolution d’un exercice) pour accéder au processus de construction de la solution. Cela implique de définir une stratégie de collecte capable de saisir les processus d’élaboration en cours, à travers des événements plus fins que les seuls *commits* de résolution. De plus, pour garantir une approche générique, nous privilégions l’usage d’un script observateur indépendant de l’éditeur de code, plutôt qu’un plugin spécifique à un Integrated Development Environment (IDE). Les notions de fiabilité et de richesse évoquées ci-dessus seront formalisées dans l’état de l’art de la seconde itération à l’aide de cadres théoriques établis sur la qualité des données, notamment les dimensions d’exactitude, de complétude, de cohérence et d’actualité.

Question de recherche 2.1

Comment améliorer l'exactitude, la complétude, la cohérence et l'actualité des traces générées, sur la base des événements de modifications de fichiers, de sorte à pouvoir identifier des comportements d'apprentissage complexes ?

Cependant, l'automatisation de la génération de traces et l'augmentation de la granularité des données entraînent une génération massive de traces. Cette profusion d'informations soulève des défis quant à l'intégration du dispositif : il doit rester transparent pour l'étudiant-e afin de ne pas perturber son processus d'apprentissage, tout en évitant de perturber l'enseignant-e par un surplus de données.

Question de recherche 2.2

Comment garantir que la génération de traces améliorée ne perturbe ni les activités de l'étudiant-e ni celles de l'enseignant-e ?

Le défi majeur réside dans la nécessité d'assurer une séparation suffisante entre les traces d'apprentissage générées automatiquement et les traces issues du travail productif de l'étudiant-e, les *commits* de résolution d'exercice. Ces deux types de données partageant le même espace de stockage et le même format, une séparation est indispensable pour garantir que l'historique reste lisible et compréhensible. Cette clarté est essentielle tant pour l'étudiant-e que pour l'enseignant-e, notamment lorsque ce dernier ou cette dernière doit analyser l'historique pour évaluer le processus de développement sans que celui-ci ne soit obscurci par la densité des données collectées.

1.2.2 Contributions et publications

Les contributions de cette thèse sont de deux natures. Premièrement, elles sont logicielles, avec la conception et le développement de deux systèmes dont le code source est ouvert : Git4School, un tableau de bord pour l'orchestration, et LAWG, un système de génération de traces.

Deuxièmement, elles sont scientifiques, avec la diffusion de nos résultats à travers cinq publications dans des conférences et revues avec comité de lecture. Ces publications, réparties en deux ateliers, deux conférences (dont une internationale) et une revue, sont listées ci-dessous :

- [1] Mika PONS, Jean-Michel BRUEL, Jean-Baptiste RACLET et Franck SILVESTRE. « Finding behavioral indicators from contextualized commits in software engineering courses with process mining ». In : *International Workshop on Frontiers in Software Engineering Education*. Springer. 2023, p. 56-68.

- [2] Mika PONS, Jean-Michel BRUEL, Jean-Baptiste RACLET et Franck SILVESTRE. « Traçabilité by design : conception d'un système interactif pour améliorer la génération automatique de traces Git pendant une activité d'apprentissage ». In : *11ème Conférence sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH 2023)*. 2023, p. 68-79.
- [3] Mika PONS, Jean-Michel BRUEL, Jean-Baptiste RACLET et Franck SILVESTRE. « Traceability by design : design of an interactive system to improve the automatic generation of Git traces during a learning activity ». In : *European Conference on Technology Enhanced Learning*. Springer. 2023, p. 611-617.
- [4] Jean-Baptiste RACLET, Franck SILVESTRE et Mika PONS. « Mise en oeuvre d'approches pédagogiques fondées sur des pratiques de l'industrie du logiciel pour l'apprentissage de la programmation ». In : *2020 Colloque Didapro 8 : L'informatique, objets d'enseignements (DidaSTIC 2020)*. 2020, p. 1-12.
- [5] Jean-Baptiste RACLET, Franck SILVESTRE et Mika PONS. « Git4School : un tableau de bord pour assister la prise de décisions de l'enseignant lors des cours de génie logiciel ». In : *STICEF (Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation)* 28.3 (2021), p. 1-20.

1.3 ORGANISATION DU MANUSCRIT

1.3.1 Itérations

Nos travaux se sont déroulés au cours de 2 itérations complètes de DBR.

La première avait pour objectif d'offrir aux enseignant-es un moyen de visualiser la progression des étudiant-es engagé-es dans une tâche d'apprentissage, au cœur même d'une séance de TP en tirant parti de l'utilisation de Git déjà mise en place dans un grand nombre de cours de programmation. À cette fin, nous avons développé un prototype de tableau de bord, Git4School, dont la conception et les fonctionnalités sont décrites dans le chapitre 3. L'outil a été déployé dans plusieurs cours de développement logiciel en Diplôme Universitaire de Technologie (DUT), Licence Pro et Master. L'analyse des observations faites par les enseignant-es lors de ces expérimentations a montré que l'outil est bénéfique pour la prise de décision enseignante. L'itération se conclut par une étude quantitative utilisant des techniques de Process Mining (PM) sur les données collectées, qui met en évidence des limites dans la qualité des traces générées manuellement par les étudiant-es.

La seconde itération part de ces constats pour concevoir et développer LAWG, un système de génération automatique de traces évalué en trois étapes (vérification, validation et expérimentation) afin de garan-

tir la fiabilité et la richesse des données collectées. Cette démarche a permis d'aboutir à une solution robuste pour la génération de traces, facilitant l'analyse fine des activités des étudiant-es.

1.3.2 *Structure du manuscrit*

Le manuscrit s'articule autour de trois parties principales, reflétant le déroulement de nos travaux de recherche :

La partie i présente la première itération et se compose de quatre chapitres. Le premier chapitre dresse un état de l'art des travaux liés aux tableaux de bord pour l'apprentissage. Le deuxième chapitre détaille la conception et le développement de Git4School, un tableau de bord destiné à soutenir la prise de décision enseignante pendant les séances de TP. Le troisième chapitre présente une évaluation qualitative de cet outil à travers son utilisation en contexte écologique. Enfin, le quatrième chapitre complète cette évaluation par une étude quantitative des comportements des étudiant-es à l'aide de techniques de PM, mettant en évidence les limites des traces générées manuellement et ouvrant la voie à l'itération suivante.

La partie ii est consacrée à la seconde itération et s'articule autour de quatre chapitres. Elle s'ouvre sur un état de l'art concernant la génération de traces d'apprentissage en génie logiciel, au terme duquel sont formulées les questions de recherche de cette itération. Le deuxième chapitre décrit la conception de LAWG, un système interactif conçu pour générer des traces plus fiables et plus riches. Le troisième chapitre expose le protocole d'évaluation de LAWG en trois étapes : vérification, validation et expérimentation. Enfin, le quatrième chapitre détaille les résultats de cette évaluation et les discute.

Enfin, la partie iii conclut ce manuscrit. Elle synthétise nos contributions, discute des limites de nos travaux et propose des pistes de travaux futurs, notamment autour de l'intégration de données auto-rapportées.

Première partie

ITÉRATION 1 : TABLEAU DE BORD POUR L'AIDE À LA PRISE DE DÉCISION ENSEIGNANTE AU COURS D'UNE SÉANCE DE TP

Cette partie présente la première itération complète du cycle DBR, consacrée à la conception et à l'évaluation d'un tableau de bord destiné aux enseignant·es, exploitant les traces Git des étudiant·es enrichies par des données contextuelles. L'objectif est de proposer des visualisations actionnables de l'activité des étudiant·es, soutenant la prise de décision enseignante en temps réel durant les séances de travaux pratiques. Le cadre de la conscience de la situation permet de situer le niveau de soutien apporté par le tableau de bord, de la perception de l'activité (niveau 1) à la compréhension de la situation pédagogique (niveau 2). Conformément à la méthodologie DBR, cette itération se structure en trois phases : une *phase d'analyse* à travers l'état de l'art, une *phase de conception et de développement* du prototype Git4School, et une *phase d'évaluation* menée selon deux approches complémentaires : une évaluation qualitative fondée sur les retours d'expérience des enseignant·es utilisant l'outil en séance, puis une évaluation quantitative qui analyse les données collectées par l'outil à l'aide du Process Mining afin d'identifier des schémas de comportement des étudiant·es et d'évaluer le potentiel de détection automatisée de ces comportements.

ÉTAT DE L'ART

2.1	Les Learning Analytics au service de l'enseignement .	14
2.1.1	Définitions et périmètre	14
2.1.2	Le cycle des Learning Analytics	14
2.1.3	Conception de situation d'apprentissage et alignement pédagogique	15
2.2	Les tableaux de bord d'apprentissage	16
2.2.1	Définition et caractéristiques	16
2.2.2	Panorama des tableaux de bord : tendances et exemples	16
2.2.3	Tableaux de bord destinés aux enseignant·es . .	18
2.2.4	Orchestration et tableaux de bord en temps réel	19
2.3	De la donnée à l'action : <i>insights</i> actionnables et construction de sens	19
2.3.1	Le fossé entre conscience et action	19
2.3.2	La notion d' <i>insight</i> actionnable	20
2.3.3	Du <i>sensemaking</i> à la conscience de la situation .	21
2.4	La conscience de la situation comme cadre théorique .	22
2.4.1	Le modèle d'Endsley (1995)	22
2.4.2	Application à l'enseignement et aux tableaux de bord	22
2.4.3	Synthèse du cadre théorique	24
2.5	Traces d'apprentissage et enseignement de la programmation	24
2.5.1	L'enseignement de la programmation : contexte et défis	24
2.5.2	Les traces issues des systèmes de gestion de version	25
2.5.3	Positionnement et lacunes identifiées	26
2.6	Analyse des comportements d'apprentissage par le Process Mining	27
2.6.1	Le Process Mining dans l'éducation	27
2.6.2	Le Process Mining en génie logiciel	28
2.6.3	Le Process Mining dans l'enseignement du génie logiciel	28
2.7	Synthèse et formulation de la question de recherche . .	29

Ce chapitre constitue la *phase d'analyse* de la première itération du cycle DBR. Son objectif est de dresser un état de l'art des travaux qui sous-tendent notre question de recherche : dans quelle mesure

un tableau de bord d'apprentissage peut-il soutenir la conscience de la situation de l'enseignant-e pour favoriser la prise de décision en temps réel ? Pour y répondre, nous suivons un fil conducteur allant des fondements des LA aux tableaux de bord d'apprentissage, en passant par les défis liés à la production d'*insights* actionnables à partir des données, le processus de construction de sens (*sensemaking*) et le cadre théorique de la conscience de la situation. Nous terminons par le positionnement de notre travail dans le contexte spécifique de l'enseignement de la programmation et de l'exploitation des traces issues des outils de gestion de version.

2.1 LES LEARNING ANALYTICS AU SERVICE DE L'ENSEIGNEMENT

2.1.1 Définitions et périmètre

Le terme *Learning Analytics* (LA) a été popularisé au début des années 2010 [106], dans un contexte de massification des données numériques en éducation. La définition la plus largement acceptée est celle de la *Society for Learning Analytics Research* (SoLAR), formulée lors de la première conférence LAK en 2011 : les LA désignent « la mesure, la collecte, l'analyse et le compte-rendu de données sur les apprenant-es et leurs contextes, dans le but de comprendre et d'optimiser l'apprentissage et les environnements dans lesquels il se produit » [66]. Cette définition met en avant deux caractéristiques essentielles : l'ancrage dans les données empiriques et l'orientation vers l'amélioration des pratiques éducatives.

FERGUSON [34] retrace l'émergence du domaine en identifiant ses racines dans plusieurs disciplines : l'informatique (fouille de données éducatives), les sciences de l'éducation (évaluation formative, apprentissage autorégulé) et les sciences de gestion (*business intelligence*). Cette hybridité disciplinaire confère aux LA un potentiel d'intégration riche, mais aussi un risque de fragmentation si les outils développés ne s'ancrent pas dans des cadres pédagogiques explicites.

Plus spécifiquement, VAN HARMELEN et WORKMAN [113] soulignent que l'efficacité des LA réside dans leur capacité à aider les parties prenantes (étudiant-es, enseignant-es, institutions) à prendre des décisions éclairées, qu'il s'agisse d'identifier des étudiant-es à risque, d'adapter des ressources pédagogiques ou d'améliorer les stratégies d'enseignement.

2.1.2 Le cycle des Learning Analytics

CLOW [21] propose un modèle cyclique des LA articulé en quatre étapes : les apprenant-es génèrent des *données* par leur activité, ces données sont transformées en *métriques* ou visualisations, qui alimentent des *interventions* (*feedback*, alertes, adaptations pédagogiques) destinées

à agir sur les apprenant-es, bouclant ainsi le cycle. L'apport principal de ce modèle est de souligner que les LA n'ont de valeur que si la boucle est effectivement fermée : des données qui ne conduisent à aucune intervention restent lettre morte. Cette insistance sur la *boucle de rétroaction* est centrale pour notre travail : le tableau de bord que nous concevons doit précisément permettre à l'enseignant-e de transformer les traces d'apprentissage en interventions pédagogiques.

Pour autant, GAŠEVIĆ, DAWSON et SIEMENS [37] avertissent que les LA ne doivent pas se limiter à une collecte de données computationnelle : elles doivent être ancrées dans les cadres pédagogiques issus de la recherche en éducation. Sans cet ancrage théorique, les analyses risquent de mesurer des indicateurs sans rapport avec les processus cognitifs ou pédagogiques réels, limitant ainsi leur capacité à expliquer ou améliorer l'apprentissage.

2.1.3 Conception de situation d'apprentissage et alignement pédagogique

C'est dans cette optique que LOCKYER, HEATHCOTE et DAWSON [70] soulignent l'importance d'aligner l'analyse sur la *conception de situation d'apprentissage (learning design)*. Ils définissent cette dernière comme la description documentée de la séquence des tâches, des ressources et des supports prévus par l'enseignant-e. Selon eux, cette conception fournit le contexte indispensable pour interpréter les données brutes : les objectifs d'apprentissage et le plan pédagogique servent de grille de lecture, permettant non seulement de suivre la participation des étudiant-es, mais aussi d'évaluer si la conception de la situation d'apprentissage produit les effets attendus.

Dans le prolongement de ces travaux, BAKHARIA et al. [5] proposent un cadre conceptuel liant explicitement la conception de situation d'apprentissage à l'analytique, avec pour objectif de « créer des représentations plus significatives des données pour les enseignant-es ». Parmi les cinq dimensions qu'ils identifient, nous retenons dans cette thèse la dimension **temporelle** (analyser quand les événements se produisent par rapport au plan prévu) et la dimension **dynamique de cohorte** (observer la progression du groupe), en combinant les traces issues de l'outil de gestion de version avec les données contextuelles fournies par l'enseignant-e.

Ces fondements posés, la question qui se pose est celle de la *matérialisation* de ces analyses : comment rendre les données et indicateurs visibles et compréhensibles pour les enseignant-es ? C'est le rôle des tableaux de bord d'apprentissage.

2.2 LES TABLEAUX DE BORD D'APPRENTISSAGE

2.2.1 Définition et caractéristiques

Dans son acception générale en informatique décisionnelle, un *tableau de bord* (*dashboard*) désigne une représentation visuelle consolidant les informations les plus importantes pour atteindre un ou plusieurs objectifs, organisée de sorte que l'information puisse être surveillée d'un coup d'œil [35]. Transposée au domaine éducatif, cette définition a été affinée par plusieurs auteurs et autrices. VERBERT et al. [115] définissent un TBA comme un affichage unique qui agrège différents indicateurs sur les apprenant·es, les processus d'apprentissage et le contexte, en des visualisations compréhensibles et orientées vers l'action. SCHWENDIMANN et al. [100], dans leur revue systématique portant sur 55 articles, précisent qu'un TBA capture et visualise les traces des activités d'apprentissage dans le but de promouvoir la prise de conscience (*awareness*), la réflexion et l'action de la part des différents acteurs éducatifs.

Le TBA se distingue d'un simple rapport statistique par plusieurs caractéristiques : son orientation vers l'action, sa synthèse visuelle, sa mise à jour régulière ou en temps réel, et sa conception pour une audience spécifique ayant des objectifs pédagogiques identifiés [116].

La conception des TBA s'appuie sur des principes issus de la visualisation de l'information et des sciences cognitives. Le *Visual Information Seeking Mantra* de SHNEIDERMAN [104] – « vue d'ensemble d'abord, zoom et filtrage ensuite, détails à la demande » – est particulièrement pertinent pour les TBA enseignants : l'enseignant·e doit d'abord saisir la situation globale de la classe, puis pouvoir cibler un·e étudiant·e ou un exercice spécifique. Par ailleurs, la théorie de la charge cognitive [110] impose des contraintes de parcimonie : un TBA surchargé d'indicateurs ou de visualisations complexes peut imposer une charge cognitive extrinsèque qui nuit à l'objectif même de l'outil. Ces principes guident la conception de tout TBA efficace, et tout particulièrement ceux destinés à un usage en temps réel pendant une séance, où l'enseignant·e ne dispose que de quelques secondes pour consulter l'interface.

2.2.2 Panorama des tableaux de bord : tendances et exemples

Plusieurs revues systématiques permettent de dresser un panorama du domaine. SCHWENDIMANN et al. [100] observent que la majorité des TBA ciblent les apprenant·es, tandis que ceux exclusivement destinés aux enseignant·es sont moins fréquents. BODILY et VERBERT [13] confirment cette tendance et montrent que les tableaux de bord orientés enseignant·es et ceux orientés étudiant·es diffèrent significativement en termes d'indicateurs présentés et de fonctionnalités offertes. MATCHA, GAŠEVIĆ, PARDO et al. [74] notent par ailleurs que la majo-

rité des TBA publiés souffrent d'un ancrage théorique insuffisant et que leurs évaluations se limitent souvent à la perception et à l'utilisabilité, sans mesurer l'impact réel sur les pratiques ou les résultats d'apprentissage. JIVET et al. [54] renforcent ce constat en identifiant quatre lacunes spécifiques : le manque d'alignement entre les objectifs déclarés et les fonctionnalités offertes, le manque de soutien à l'interprétation des données par les utilisatrices et utilisateurs, le manque d'orientation vers l'action, et le manque de rigueur dans les protocoles d'évaluation. Des revues plus récentes nuancent et prolongent ces observations. PAULSEN et LINDSAY [84] identifient une trajectoire vers des TBA de plus en plus orientés vers l'apprentissage plutôt que vers la seule analytique, avec un ancrage pédagogique plus explicite. Toutefois, KALIISA et al. [59], dans une revue systématique de 38 études portant sur l'impact des TBA, concluent qu'il n'existe pas de preuve solide que ces outils améliorent les résultats académiques des étudiant·es, la plupart des effets mesurés étant négligeables ou faibles. Ce constat renforce l'idée que l'efficacité des TBA ne réside pas tant dans l'affichage de données aux apprenant·es que dans le soutien qu'ils apportent aux enseignant·es pour adapter leurs interventions pédagogiques.

Parmi les systèmes existants, plusieurs méritent d'être mentionnés pour leur contribution au domaine et leur pertinence pour notre travail :

- **CourseVis** [75] est l'un des premiers tableaux de bord enseignants, proposant des visualisations de l'activité étudiante sous forme de matrices et de graphiques chronologiques. Il a démontré qu'une visualisation simple pouvait améliorer la conscience situationnelle de l'enseignant·e.
- **Course Signals** [3], développé à l'université Purdue, combine un modèle prédictif de risque d'échec avec un système de signalisation visuelle (feux tricolores). Il a établi le paradigme de l'*alerte précoce* (*early warning system*).
- **OnTask** [82] représente une avancée significative vers l'aide à la décision : plutôt qu'un simple affichage de données, ce système permet aux enseignant·es de définir des règles conditionnelles pour générer des messages personnalisés aux étudiant·es en fonction de leurs données d'activité. Il illustre le passage de la visualisation passive à l'action outillée.
- Les travaux de HOLSTEIN et al. [47, 48] sur les tableaux de bord en temps réel pour les enseignant·es dans des classes utilisant des systèmes tutoriels intelligents (*Lumilo*) sont particulièrement proches de notre contexte. En utilisant une méthodologie de co-conception avec des enseignant·es, ils ont identifié que ceux-ci et celles-ci souhaitent principalement savoir quels étudiant·es sont bloqué·es, comprendre la nature de la difficulté, voir la progression globale de la classe et détecter les comportements

hors tâche. Leur étude contrôlée a montré que l'outil conduisait les enseignant-es à distribuer leur attention de manière plus équitable, avec un impact positif mesurable sur les résultats d'apprentissage des étudiant-es les plus faibles.

- Dans le contexte spécifique de la programmation, **CodeOpticon** [44] affiche en mosaïque les éditeurs de code de tous les étudiant-es en temps réel, permettant à l'enseignant-e de voir littéralement ce que chacun-e écrit. Si cette approche illustre le besoin profond des enseignant-es de « voir » l'activité de la classe, elle pose des questions de surcharge informationnelle et de respect de la vie privée. **OverCode** [41] adopte une approche complémentaire en regroupant automatiquement les solutions par similarité structurelle, permettant d'identifier les familles d'approches adoptées par la classe plutôt que chaque étudiant-e individuellement.

2.2.3 Tableaux de bord destinés aux enseignant-es

Les enseignant-es présentent des besoins et des contraintes spécifiques en matière de tableaux de bord. Alors que les apprenant-es ont besoin d'informations sur leur propre activité pour s'autoréguler, les enseignant-es doivent monitorer simultanément l'activité de l'ensemble des apprenant-es de leur classe, identifier les situations nécessitant une intervention et prendre des décisions pédagogiques en conséquence.

VAN LEEUWEN et RUMMEL [114] proposent une taxonomie des outils de support à l'enseignant-e en distinguant trois types :

- les outils de **mirroring** (miroir) présentent les données telles quelles, laissant l'interprétation à l'enseignant-e ;
- les outils d'**alerting** (alerte) signalent les situations critiques dépassant certains seuils ;
- les outils d'**advising** (conseil) recommandent des actions spécifiques.

Leurs résultats suggèrent que les outils d'alerte sont les plus efficaces pour les enseignant-es ayant peu d'expérience avec les tableaux de bord, tandis que les outils de mirroring sont préférés par les enseignant-es expert-es qui souhaitent conserver leur autonomie d'interprétation.

Les contraintes temporelles sont particulièrement aiguës en contexte synchrone : l'enseignant-e ne peut consacrer que quelques secondes à la consultation du tableau de bord entre deux interactions avec les étudiant-es. Le tableau de bord doit donc fonctionner comme un outil de *réduction de la complexité* plutôt que comme une source d'information supplémentaire [47].

2.2.4 Orchestration et tableaux de bord en temps réel

Plus axée sur la prise de décision en temps réel, la communauté de recherche sur l'apprentissage collaboratif supporté par l'informatique (CSCL) a développé le concept de *technologie d'orchestration des activités pédagogiques* (*orchestration technology*). TCHOUNIKINE [111] définit l'orchestration comme la régulation en situation (*conducting*) du scénario pédagogique prévu (*scripting*), impliquant une adaptation constante aux contraintes et aux imprévus. Plus concrètement, les technologies d'orchestration sont « des technologies qui fournissent aux enseignant-es des moyens de surveillance ou d'intervention » [111]. Ces technologies prennent souvent la forme de tableaux de bord : TISSENBAUM et SLOTTA [112] montrent qu'un *feedback* en temps réel aide les enseignant-es à savoir quand et où intervenir, réduisant ainsi la charge cognitive liée à la gestion de classes complexes.

Dans cette perspective, WISE [122] a proposé le concept de *conception d'intervention pédagogique fondée sur les LA* (*pedagogical learning analytics intervention design*), défini comme des « efforts systématiques pour intégrer l'utilisation des LA comme partie productive des pratiques d'enseignement et d'apprentissage dans un contexte éducatif donné ». Ce concept articule explicitement LA et orchestration : les visualisations ne sont pas une fin en soi, mais un moyen de soutenir les interventions pédagogiques, qu'elles soient planifiées ou déclenchées en réaction à la situation observée.

Le tableau de bord en temps réel apparaît ainsi comme un outil d'orchestration à part entière. Des travaux récents confirment cette orientation : POSSAGHI et al. [88] montrent que l'intégration de données multimodales (expressions faciales, rythme cardiaque) dans un TBA améliore la conscience de la situation des enseignant-es et leur prise de décision en classe. Cependant, la littérature met en évidence un constat récurrent : afficher des données ne suffit pas à garantir qu'elles conduiront à des actions pédagogiques pertinentes. C'est le défi de la production d'*insights actionnables*.

2.3 DE LA DONNÉE À L'ACTION : *insights* ACTIONNABLES ET CONSTRUCTION DE SENS

2.3.1 Le fossé entre conscience et action

Un constat récurrent dans la littérature sur les TBA est celui du *fossé entre conscience et action* (*awareness-to-action gap*) : les tableaux de bord informent, mais ne conduisent pas forcément à l'action [123]. Comme évoqué en section 2.2.4, le concept de conception d'intervention pédagogique fondée sur les LA [122] insiste sur la nécessité d'intégrer l'analytique directement dans le flux d'activité. Cela implique non

seulement de concevoir des outils d'orchestration, mais aussi de soutenir le dialogue autour des données et de faciliter leur interprétation.

MOLENAAR et KNOOP-VAN CAMPEN [79] étudient empiriquement comment les enseignant-es transforment l'information d'un tableau de bord en actions pédagogiques. Leurs résultats montrent que la capacité à agir sur les données dépend fortement de la manière dont l'information est présentée, du contexte pédagogique et de l'expertise de l'enseignant-e.

Notre proposition s'inscrit directement dans la lignée de ces travaux puisque nous utilisons les LA et plus précisément les visualisations afin d'aider les enseignant-es à réaliser des interventions pendant leurs cours en face à face.

2.3.2 La notion d'*insight* actionnable

Pour franchir ce fossé, la notion d'*insight* actionnable (*actionable insight*) a émergé comme concept central. Selon COOPER [23], le terme « actionnable » indique que l'analytique « s'intéresse au potentiel d'action pratique plutôt qu'à la description théorique ou au simple *reporting* ». Plus récemment, JØRNØ et GYNTHNER [56] formalisent l'*insight* actionnable comme « une donnée permettant d'établir une procédure corrective, ou boucle de rétroaction, pour un ensemble d'actions », reliant ainsi explicitement le concept à la boucle de Clow évoquée en section 2.1. Dans leur revue de littérature dédiée au concept en LA, ces auteurs identifient trois propriétés d'un *insight* actionnable :

Le terme
« analytique »
traduit ici *analytics*,
dont les *learning*
analytics constituent
un sous-domaine.

- **Pragmatique** : l'information est orientée vers « que faire » plutôt que vers « pourquoi ».
- **Contingent** : les données informent la décision mais ne la prescrivent pas ; l'enseignant-e conserve son jugement professionnel.
- **Substantié** : l'information est qualifiée par des mesures de validité et de fiabilité.

Ces propriétés ont des implications directes pour la conception de TBA : un tableau de bord efficace doit aller au-delà de l'affichage de données pour guider l'interprétation et soutenir la décision, tout en respectant l'autonomie professionnelle de l'enseignant-e. Dans la suite de ce manuscrit, nous utiliserons le qualificatif « actionnable » pour désigner des indicateurs, des visualisations ou des tableaux de bord qui satisfont ces propriétés, c'est-à-dire qui fournissent à l'enseignant-e une information suffisamment contextualisée et orientée vers l'action pour soutenir sa prise de décision pédagogique et son pouvoir d'agir.

La notion d'*insight* actionnable définit ainsi la *finalité* du tableau de bord, le « quoi » qu'il doit produire. Reste à comprendre le « comment » : par quel processus cognitif l'enseignant-e transforme-t-elle ou il les données affichées en une compréhension qui appelle l'action ? C'est ce processus que nous explorons dans les deux sections

suivantes, d'abord à travers la construction de sens, puis à travers un cadre structurant les niveaux de traitement de l'information.

2.3.3 *Du sensemaking à la conscience de la situation*

Pour qu'un insight soit actionnable, l'enseignant-e doit d'abord *construire du sens* à partir des données affichées. Ce processus de *construction de sens* (*sensemaking*) constitue un maillon cognitif essentiel entre la perception de l'information et la prise de décision.

WEICK [121] définit le *sensemaking* comme un processus rétrospectif de construction de sens face à l'ambiguïté, guidé par la plausibilité plutôt que par la certitude. Appliqué à notre contexte, cela signifie que l'enseignant-e qui consulte un tableau de bord ne reçoit pas passivement l'information : l'enseignant-e construit activement une interprétation de la situation pédagogique en mobilisant son expertise, ses attentes et le contexte.

VERBERT et al. [115, 116] ont explicitement intégré le *sensemaking* dans leur modèle d'évaluation des TBA, en proposant quatre étapes : *Awareness* (prise de conscience) → *Reflection* (réflexion) → *Sensemaking* (construction de sens) → *Impact* (action). Le *sensemaking* y apparaît comme l'étape charnière entre la réflexion sur les données et l'action informée.

KLEIN et al. [65] proposent un modèle complémentaire, le modèle *Data/Frame* : les données perçues activent des cadres d'interprétation (*frames*) qui, à leur tour, guident la recherche de nouvelles données et orientent la compréhension. Appliqué à l'enseignant-e face à un tableau de bord, ce modèle éclaire la manière dont les visualisations activent les schémas pédagogiques de l'enseignant-e : une barre de progression en retard active le cadre « étudiant-e en difficulté », qui oriente l'attention vers d'autres indicateurs confirmant ou infirmant cette hypothèse.

Le *sensemaking* décrit ainsi le processus cognitif de construction de sens. La question de savoir comment soutenir ce processus dès la conception des TBA a fait l'objet de travaux récents. SADALLAH et al. [99] proposent un outil participatif permettant aux parties prenantes de prototyper des TBA qui favorisent la construction de sens, en étendant la participation des utilisateur·ices à la phase de prototypage. SADALLAH et GILLIOT [98] prolongent cette approche en proposant une méthode générative de TBA intégrant explicitement le soutien au *sensemaking* et à la prise de décision. Pour opérationnaliser ce processus dans le contexte de la prise de décision en temps réel – comme lors d'une séance de TP – un cadre plus structuré est nécessaire. Le modèle de conscience de la situation d'Endsley fournit cette structure en distinguant trois niveaux de traitement de l'information.

2.4 LA CONSCIENCE DE LA SITUATION COMME CADRE THÉORIQUE

2.4.1 *Le modèle d'Endsley (1995)*

Le modèle de conscience de la situation (*situation awareness, SA*) a été formalisé par ENDSLEY [29] dans le domaine des facteurs humains, initialement pour décrire la cognition des pilotes dans des environnements dynamiques. La SA y est définie comme « la perception des éléments de l'environnement dans un volume de temps et d'espace, la compréhension de leur signification et la projection de leur état dans un futur proche ».

Ce modèle distingue trois niveaux :

- **Niveau 1 – Perception** : la détection des éléments pertinents de l'environnement. C'est la base sur laquelle reposent les niveaux supérieurs.
- **Niveau 2 – Compréhension** : l'intégration et l'interprétation des éléments perçus pour en dégager une signification globale. Ce niveau va au-delà de la simple perception pour construire une représentation cohérente de la situation.
- **Niveau 3 – Projection** : l'anticipation des états futurs à partir de la compréhension de la situation courante. Ce niveau est le plus exigeant cognitivement et permet de planifier les actions.

Un point essentiel est que la SA est un état de connaissance, distinct de la prise de décision et de l'action qui en découlent [30]. Une bonne SA ne garantit pas une bonne décision (l'enseignant·e peut comprendre la situation mais choisir une intervention inadaptée), et inversement une mauvaise SA rend probable une décision inappropriée (intervenir auprès d'un·e étudiant·e qui n'en a pas besoin, ignorer celui ou celle qui est en difficulté).

2.4.2 *Application à l'enseignement et aux tableaux de bord*

Transposé au contexte de l'enseignement, et en particulier à la supervision de séances de TP, le modèle d'Endsley permet de caractériser ce que les différents niveaux de traitement de l'information impliquent pour la conception d'un TBA :

- **Niveau 1 (perception)** : rendre visible l'activité des étudiant·es, c'est-à-dire qui travaille, qui est inactif, qui a terminé un exercice, qui n'a pas encore commencé.
- **Niveau 2 (compréhension)** : aider l'enseignant·e à comprendre la situation pédagogique, notamment pourquoi un·e étudiant·e est bloqué·e, quel est l'état d'avancement global du groupe, quels exercices posent le plus de difficultés.
- **Niveau 3 (projection)** : permettre d'anticiper, par exemple qui va avoir besoin d'aide dans les prochaines minutes, si le groupe

va finir l'exercice dans le temps imparti, ou s'il faut adapter le rythme de la séance.

Cette transposition ne doit pas être interprétée comme l'obligation pour un TBA de soutenir explicitement chacun des trois niveaux. Endsley souligne que la SA est le produit de l'interaction entre les informations disponibles dans l'environnement et les connaissances internes de l'opérateur, notamment ses *modèles mentaux* et ses schémas d'expertise [29, 30]. Ces représentations internalisées permettent à un opérateur expérimenté d'inférer les éléments non perçus directement : un enseignant expert peut construire une projection (niveau 3) à partir des données de perception et de compréhension que le tableau de bord lui fournit, sans que l'outil n'ait à calculer ou afficher cette projection explicitement. Ce mécanisme est cohérent avec le modèle RPD de KLEIN [64] : les experts reconnaissent des configurations situationnelles typiques et activent des réponses adaptées sans analyse systématique de chaque niveau. Le soutien qu'un TBA doit apporter n'est donc pas absolu : il est conditionné par le niveau d'expertise de l'utilisateur et la complexité des décisions requises.

Le modèle de décision fondée sur la reconnaissance (*Recognition-Primed Decision*, RPD) de KLEIN [64] éclaire la manière dont un-e enseignant-e expert-e exploite cette conscience de la situation. Selon ce modèle, issu de la *Naturalistic Decision Making*, les expert-es en situation dynamique ne comparent pas systématiquement des options : ces personnes reconnaissent des *schémas récurrents* dans la situation qui activent des réponses typiques apprises par l'expérience. En TP, l'enseignant-e expérimenté-e reconnaît des schémas comportementaux (un-e étudiant-e inactif ou inactive depuis longtemps, une succession d'erreurs de compilation, un retard marqué par rapport au groupe) et active des schémas d'intervention adaptés. Le tableau de bord, en rendant visibles des comportements qui seraient autrement inaccessibles (activité de commit, progression dans les exercices), enrichit la base de schémas disponibles pour cette reconnaissance.

VERBERT et al. [115] ont adopté ce cadre pour structurer l'évaluation des TBA en y ajoutant un quatrième niveau : l'*action informée*, c'est-à-dire l'intervention pédagogique prise en conséquence de la compréhension de la situation. Ce cadre à quatre niveaux (awareness, réflexion, sensemaking, impact) est devenu un modèle de référence dans le domaine.

Ce modèle peut être articulé avec la taxonomie de VAN LEEUWEN et RUMMEL [114] : les outils de *mirroring* soutiennent principalement le niveau 1 (rendre les données visibles), les outils d'*alerting* facilitent le passage au niveau 2 (signaler les situations nécessitant une compréhension), et les outils d'*advising* visent le niveau 3 et au-delà (recommander des actions anticipées). Plus récemment, GILLIOT et SADALLAH [39] ont proposé le cadre DEFLAD, qui intègre les niveaux de conscience de la situation d'Endsley comme dimension d'expres-

sion des objectifs lors de la co-conception de TBA. Ce cadre articule trois dimensions : l'expression des objectifs comme niveau de SA, la visualisation et les interactions associées, et le soutien au processus de prise de décision. DEFLAD confirme ainsi la pertinence du modèle d'Endsley comme grille de lecture pour la conception et l'évaluation des TBA destinés aux enseignant-es.

2.4.3 Synthèse du cadre théorique

Les cadres présentés dans les sections précédentes s'articulent de manière complémentaire. La notion d'*insight* actionnable (section 2.3) identifie le problème : les tableaux de bord doivent conduire à l'action, pas seulement informer. Le *sensemaking* (section 2.3.3) décrit le processus cognitif par lequel l'enseignant-e construit du sens à partir des données. La conscience de la situation fournit la structure : trois niveaux de traitement (perception, compréhension, projection) qui permettent de caractériser ce que le tableau de bord peut soutenir et d'évaluer dans quelle mesure, combiné à l'expertise de l'utilisateur ou l'utilisatrice, il y parvient.

Un TBA efficace ne se réduit donc pas à l'affichage de données brutes : il doit transformer ces données en représentations qui soutiennent la compréhension et la prise de décision. Ce cadre théorique structure l'évaluation de Git4School dans les chapitres suivants : nous examinerons comment les visualisations proposées soutiennent la construction de la conscience de la situation par l'enseignant-e, en tenant compte de son expertise comme facteur déterminant du niveau de soutien requis.

2.5 TRACES D'APPRENTISSAGE ET ENSEIGNEMENT DE LA PROGRAMMATION

2.5.1 L'enseignement de la programmation : contexte et défis

L'enseignement de la programmation, et plus largement du génie logiciel, occupe une place centrale dans les formations en informatique. Les séances de TP constituent un lieu privilégié d'apprentissage : les étudiant-es y développent leurs compétences par la pratique, en résolvant des exercices de complexité croissante. L'apprentissage de la programmation est reconnu comme une activité cognitivement exigeante : la méta-analyse de BENNEDSEN et CASPERSEN [11] rapportait un taux d'échec moyen de 33 % dans les cours d'introduction à la programmation. Sur le plan conceptuel, les étudiant-es peinent notamment à construire un modèle mental de l'exécution du code, ce que BOULAY [15] appelle la *notional machine*, à décomposer les problèmes et à adopter des stratégies de débogage efficaces.

Ce contexte présente plusieurs spécificités pertinentes pour notre travail :

- L'activité est *productive* : les étudiant-es produisent du code, c'est-à-dire des artefacts observables et analysables.
- La démarche de résolution est *progressive* : les étudiant-es passent par des étapes identifiables (lecture de l'énoncé, conception, codage, test, débogage).
- Les difficultés sont *variées* et souvent *silencieuses* : un-e étudiant-e peut être bloqué-e sans que cela soit visible de l'extérieur, contrairement à un exercice sur papier où l'enseignant-e peut voir la feuille. L'activité de programmation est essentiellement *opaque* pour l'observateur : un-e étudiant-e devant son écran peut être en train de résoudre efficacement un problème ou être complètement bloqué-e, sans que l'enseignant-e ne puisse le distinguer à distance.

Ces caractéristiques créent à la fois un besoin et une opportunité pour les LA : un besoin parce que l'enseignant-e a du mal à suivre 20 à 30 étudiant-es travaillant à des rythmes différents ; une opportunité parce que l'activité de programmation génère naturellement des traces exploitables.

2.5.2 Les traces issues des systèmes de gestion de version

Dans le contexte de l'enseignement de la programmation, les systèmes de gestion de version constituent une source de traces particulièrement intéressante. Chaque *commit* enregistre un état du code à un instant donné, accompagné de métadonnées (auteur, date, message descriptif). L'historique des commits fournit ainsi une chronologie de l'activité de l'étudiant-e.

Il existe dans la littérature plusieurs travaux concernant l'extraction de métriques à partir des journaux d'un outil de gestion de version. La plupart des travaux existants considèrent les outils de gestion de version centralisés : CVS [69, 77] ou SVN [42, 55, 62, 86]. Les objectifs de ces travaux sont variés : le suivi des étudiant-es [42], l'analyse de la collaboration entre les membres d'un groupe de travail [55, 62, 78, 93], la corrélation avec les résultats obtenus [69] ou la prédiction des performances [77].

Plus récemment, plusieurs travaux ont exploré l'exploitation de *Git* en contexte éducatif. ROBLES et GONZÁLEZ-BARAHONA [93] utilisent des techniques de fouille de dépôts pour l'évaluation et la détection de plagiat. KERTÉSZ [63] montrent que les régularités dans les *commits* (fréquence, distribution temporelle) sont corrélés avec les notes finales : les étudiant-es qui commitent régulièrement et commencent tôt obtiennent de meilleurs résultats que celles et ceux qui concentrent leur travail juste avant l'échéance. HAARANEN et LEHTINEN [46] confirment cependant que les pratiques de commit sont très variables : certain-es

étudiant-es commitent à chaque modification mineure, d'autres n'effectuent qu'un commit final, ce qui complique l'interprétation des indicateurs.

Quelques outils ont été développés pour exploiter ces traces. **GitInspector** [71] calcule des statistiques par auteur ou autrice (commits, lignes ajoutées/supprimées) mais reste un outil en ligne de commande générant des rapports statiques. **Gitcavator** [49] propose des visualisations du processus de développement à partir de dépôts Git, mais vise l'auto-réflexion de l'étudiant-e plutôt que le monitoring enseignant. **Deveo** [96] est un prototype de tableau de bord basé sur GitHub Classroom présentant des métriques de progression pour la classe, mais se concentre sur le suivi entre les séances plutôt que sur le temps réel.

Cependant, contrairement à notre approche qui analyse les messages de *commits* porteurs d'informations sémantiques (étape de conception de l'apprentissage au moment de la réalisation d'un exercice identifié), ces travaux s'intéressent plutôt aux informations quantitatives sur la volumétrie des commits, leur fréquence, ou leur temporalité par rapport à une échéance. À notre connaissance, aucun de ces systèmes ne combine spécifiquement traces Git, visualisation en temps réel, conception orientée enseignant et usage pendant une séance de TP.

2.5.3 Positionnement et lacunes identifiées

L'état de l'art met en évidence plusieurs lacunes que notre travail vise à combler :

1. **Sous-exploitation de Git** : la majorité des travaux s'appuient sur des systèmes de gestion de version centralisés (CVS, SVN), alors que Git est devenu le standard de l'industrie et est de plus en plus utilisé dans l'enseignement.
2. **Absence de combinaison avec des données contextuelles** : aucun des travaux identifiés ne combine les données extraites des dépôts de code avec d'autres données contextuelles (exercice en cours, plan de la séance) pour la prise de décision pédagogique.
3. **Absence de suivi en temps réel** : les travaux existants utilisent les données de gestion de version de manière rétrospective (après la séance, en fin de semestre), et non en temps réel pendant une séance de TP.
4. **Absence de lien avec les cadres théoriques** : aucun des travaux identifiés ne s'inscrit explicitement dans un cadre théorique de conscience de la situation ou d'*insights* actionnables pour guider la conception du tableau de bord.

2.6 ANALYSE DES COMPORTEMENTS D'APPRENTISSAGE PAR LE PROCESS MINING

Pour être utile, un tableau de bord doit fournir des indicateurs non seulement informatifs mais véritablement actionnables, c'est-à-dire directement reliés à des décisions ou ajustements possibles dans la pratique pédagogique. Cette exigence est cohérente avec les travaux de SHARPLES sur l'orchestration et l'importance d'outils qui soutiennent concrètement l'action des acteurs et actrices humaines dans des situations d'enseignement complexes [103]. Dans le champ des TBA, plusieurs travaux récents insistent de façon explicite sur cette dimension d'*actionable insights*, en montrant que la valeur des tableaux de bord réside dans la capacité des indicateurs à guider des interventions ou des régulations (par exemple [58, 109, 124]).

Cependant, se limiter à une perception statique des traces ne permet de déduire des comportements qu'à un niveau de base. Par exemple, la détection des difficultés chez les étudiant-es peut se révéler tardive si l'on ne se fie qu'à la date des rendus. Pour aller plus loin et franchir le fossé vers des *insights* plus profonds (niveau 2 de la conscience de la situation), il est nécessaire d'analyser les traces en tant que processus et enchaînements d'événements, afin de déceler des comportements d'apprentissage plus complexes au moment où ils émergent. C'est ici qu'interviennent les techniques de Process Mining (PM), qui permettent d'analyser des journaux d'événements bruts contenant des cheminements variés.

2.6.1 Le Process Mining dans l'éducation

BOGARÍN, CEREZO et ROMERO [14] définissent le Educational Process Mining (EPM) comme l'utilisation des données de journalisation recueillies spécifiquement dans des environnements éducatifs pour découvrir, analyser et fournir une représentation visuelle du processus éducatif complet.

Ainsi, les techniques de PM ont déjà été utilisées pour identifier des schémas de comportement des apprenant-es. Certaines études se concentrent sur des dimensions spécifiques du processus d'apprentissage, comme l'autorégulation [19] ou la façon dont les étudiant-es participent aux quiz [57]. D'autres études se focalisent sur la détection des parcours d'apprentissage des étudiant-es en explorant les données collectées par les systèmes de gestion de l'apprentissage, comme Moodle [28] ou par les plateformes de Massive Open Online Course (MOOC) [4, 81].

2.6.2 Le Process Mining en génie logiciel

Le PM a également été appliqué au génie logiciel de différentes manières. En 2007, RUBIN et al. [94] ont proposé un cadre pour expliciter les processus de développement logiciel basé sur les données stockées dans les systèmes de gestion de configuration.

Ainsi, plusieurs études se concentrent sur le PM des dépôts logiciels. PONCIN, SEREBRENİK et VAN DEN BRAND [87] ont utilisé le *FRamework for Analyzing Software Repositories* (FRASR) pour combiner les traces provenant de plusieurs types de dépôts : systèmes de contrôle de version, gestionnaires de bogues et courrier électronique. Ils ont ensuite pu classer les développeurs et développeuses par rôle ou identifier certains motifs dans le cycle de vie des bogues. GUPTA, SUREKA et PADMANABHUNI [45] ont analysé les données de trois dépôts logiciels pour améliorer le processus relatif au signalement et à la résolution des *issues*. ARDIMENTO et al. [2] utilisent la technique de vérification de conformité pour tester le comportement de programmation, en commençant par les journaux d'événements générés par l'IDE.

2.6.3 Le Process Mining dans l'enseignement du génie logiciel

À l'intersection du EPM et du PM appliqué au génie logiciel, certains travaux de recherche présentent différentes façons d'utiliser le PM dans le contexte des cours de génie logiciel. Par exemple, MITTAL et SUREKA [78], dans le cadre d'un cours de développement logiciel de premier cycle, ont exploité les données des systèmes de contrôle de version, des wikis et des systèmes de suivi de bogues pour mieux qualifier les activités des apprenants dans les projets d'équipe. En 2020, ESKOFIER [31] ont présenté un travail similaire, exploitant différents types de dépôts mais en s'appuyant sur la plateforme Gitlab¹. De même, SHYNKARENKO et ZHEVAHO [105] visent à fournir des visualisations résultant du PM pour aider les étudiant-es et les enseignant-es dans les activités de revue de code et d'évaluation. Enfin, l'étude proposée par MACAK et al. [73] se concentre sur l'exploitation des données des dépôts Git des étudiant-es. Ainsi, les auteurs décrivent précisément comment convertir un journal Git en un journal d'événements que les outils de PM peuvent traiter.

Notre approche, présentée au chapitre 5, s'inscrit dans la lignée des travaux présentés dans cette section. Elle s'en distingue en explorant un aspect non traité dans les travaux précédents : la prise en compte des éléments du contexte pédagogique et des interventions pédagogiques de l'enseignant-e dans les activités réalisées par les étudiant-es.

1. <https://about.gitlab.com/>

2.7 SYNTHÈSE ET FORMULATION DE LA QUESTION DE RECHERCHE

Ce chapitre a tracé un parcours allant des fondements des LA au contexte spécifique de l'exploitation des traces Git en enseignement de la programmation. Chaque étape a contribué à construire le cadre théorique de notre question de recherche :

- Les **Learning Analytics** (section 2.1) fournissent le socle disciplinaire : mesurer, analyser et agir sur les données d'apprentissage, en ancrant les analyses dans des cadres pédagogiques.
- Les **tableaux de bord d'apprentissage** (section 2.2) constituent l'outil de matérialisation : ils rendent les données visibles et compréhensibles pour les enseignant-es, avec des enjeux spécifiques de conception pour le temps réel et l'orchestration.
- Les **insights actionnables et le sensemaking** (section 2.3) identifient le défi central : passer de la donnée affichée à la décision pédagogique, en soutenant le processus cognitif de construction de sens.
- La **conscience de la situation** (section 2.4) fournit le cadre d'analyse structurant : trois niveaux de traitement (perception, compréhension, projection) qui permettent de caractériser et d'évaluer le soutien qu'un tableau de bord peut apporter à la prise de décision.
- Les **traces Git et l'enseignement de la programmation** (section 2.5) situent notre contribution dans un contexte où les lacunes sont identifiées : pas de tableau de bord exploitant Git en temps réel, pas de combinaison avec des données contextuelles, pas de lien avec les cadres théoriques de la SA.
- L'**analyse des comportements par le Process Mining** (section 2.6) apporte les méthodes nécessaires pour dépasser la simple perception et atteindre un niveau de compréhension supérieur, en modélisant les enchaînements d'actions pour déceler des stratégies ou des difficultés.

L'ensemble de ces éléments converge vers la question de recherche suivante, formulée dans l'introduction (p. 6) :

Dans quelle mesure un tableau de bord pour enseignant-es, construit à partir des traces générées par un outil de gestion de version et enrichies par des données contextuelles, permet-il de proposer des visualisations actionnables soutenant la prise de décision enseignante en temps réel ?

Le chapitre suivant présente la conception de Git4School, le tableau de bord que nous avons développé pour répondre à cette question.

CONCEPTION DE GIT4SCHOOL, UN TABLEAU DE BORD À DESTINATION DES ENSEIGNANT·ES

3.1	Contexte	31
3.1.1	Git, GitHub et GitHub Classroom	31
3.1.2	Le protocole PRaTDL	33
3.2	Git4School	36
3.2.1	Éléments d'architecture	36
3.2.2	Configuration	39
3.2.3	Fonctionnalités	41

L'état de l'art ayant permis d'identifier les fondements théoriques et les travaux existants, ce chapitre aborde la *phase de conception et de développement* de la première itération du cycle DBR.

Dans cette thèse, nous défendons la pertinence de Git comme outil permettant d'obtenir des informations sur le comportement d'un·e étudiant·e au cours d'une séance de TP. Pour cela, nous présentons dans ce chapitre le tableau de bord Git4School qui assiste l'enseignant·e dans sa prise de décision en offrant des visualisations à partir des traces Git des étudiant·es.

3.1 CONTEXTE

3.1.1 Git, GitHub et GitHub Classroom

Dans cette section, nous expliquerons d'abord brièvement ce que sont Git, GitHub et GitHub Classroom avec l'intention que la suite du manuscrit, qui repose en partie sur ces éléments, soit intelligible pour le plus grand nombre.

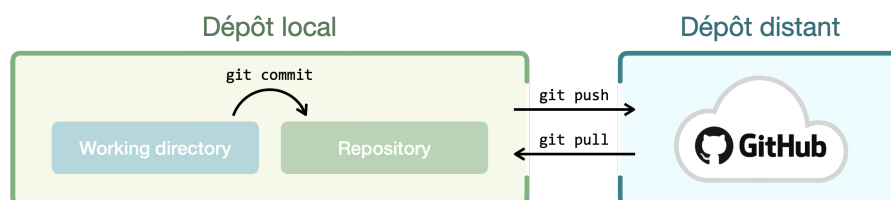


FIGURE 3.1 – Schéma illustrant les interactions entre Git local et GitHub

3.1.1.1 *Git*

Git est un logiciel de gestion de versions de code source décentralisé. C'est-à-dire un logiciel qui permet, entre autres choses, de sauvegarder des modifications de fichiers de façon optimisée, de supporter le travail en collaboration grâce à différents mécanismes, et de garder un historique de toutes les modifications, permettant par la même occasion de revenir sur n'importe quelle de ces modifications. Cet outil est très couramment utilisé dans l'industrie du développement logiciel. En effet, l'utilisation d'un logiciel de gestion de versions est mentionné dans le SWEBOK [120], référentiel produit par l'IEEE et qui regroupe les connaissances, pratiques et techniques bien établies dans le champ de l'ingénierie du logiciel. De plus, l'utilisation de Git est communément présente dans les programmes d'enseignement supérieur.

L'idée principale est que chaque étudiant-e travaille dans son dépôt local, et traite les questions du TP unes à unes tout en les marquant comme résolues une fois finies avec la commande *commit* (voir la figure 3.1 qui illustre l'utilisation des commandes Git évoquées ici). Elle permet de sauvegarder les modifications des fichiers, formant un ensemble que l'on peut appeler *diff*, et d'autres métadonnées, associées à la résolution d'une question dans son dépôt. Parmi les métadonnées, on peut retrouver la date, l'auteur ou l'autrice, ainsi qu'un message entré librement par l'auteur ou l'autrice. Puis, l'étudiant-e synchronise le dépôt distant (par souci de simplicité, nous ne rentrerons pas ici dans le détail des mécanismes de branches). Pour cela, la synchronisation du dépôt local à partir du dépôt distant se fait avec la commande *pull* et, dans le sens inverse, avec la commande *push*.

3.1.1.2 *GitHub*

GitHub est, entre autres, une forge, ou plus simplement un hébergeur de dépôts Git offrant également d'autres services comme le suivi des bugs, la gestion de la documentation ou la gestion de projet plus largement. Il existe d'autres forges Git comme Gitlab ou Bitbucket mais nous nous sommes concentré sur GitHub qui est la forge utilisée dans la plupart des TP des établissements d'enseignement supérieur dans lesquelles nous avons fait les expérimentations. Les enseignant-es ont accès à tous les dépôts des étudiant-es créés pour les TP dans lesquels elles ou ils interviennent.

3.1.1.3 *GitHub Classroom*

GitHub Classroom est un site web qui permet aux enseignant-es de créer des *assignments*, qui représentent des travaux individuels ou à réaliser en groupe, proposés par un-e enseignant-e à une cohorte d'étudiant-es, à partir d'un squelette de dépôt Git sur GitHub. Plus généralement, nous considérons qu'un *assignment* correspond à une

séance de TP ou une série de séances de TP dont le contenu pédagogique se présente et doit être traité par les étudiant-es de manière séquentielle. Souvent, un *assignment* est lié à une feuille d'exercices, mais il peut arriver qu'il englobe plusieurs feuilles d'exercices dans le cadre d'un projet par exemple. Pour chaque *assignment*, l'enseignant-e obtient un lien qu'elle ou il peut distribuer aux étudiant-es. En cliquant sur ce lien, chaque étudiant-e crée automatiquement son propre dépôt GitHub initialisé avec le contenu du squelette de l'*assignment*. Les étudiant-es peuvent ensuite créer leur dépôt local issu de ce dépôt GitHub. Chaque dépôt GitHub des étudiant-es est ensuite accessible aux enseignant-es habilité-es pour suivre leur progression sur l'*assignment*.

Pour les enseignant-es, utiliser cet outil représente un triple intérêt. Cela permet d'abord de distribuer une base de travail à toute la cohorte d'étudiant-es via la création du dépôt. De plus, l'enseignant-e a alors immédiatement accès au résultat du travail des étudiant-es au fur et à mesure qu'il évolue. Enfin, en cliquant sur le lien, avant de créer le dépôt, les étudiant-es peuvent créer un groupe ou en rejoindre un existant et tous les membres d'un même groupe auront alors accès au dépôt créé.

Ainsi, l'outil permet via un simple lien de résoudre trois problématiques : la distribution en masse d'une base de travail, la formation des groupes et la mise en place de leur environnement de travail collaboratif, et la collecte du travail des étudiant-es.

Dans la mesure où notre objectif est d'accompagner la prise de décision de l'enseignant-e à partir de l'activité Git des étudiant-es, il est indispensable de s'appuyer sur sa conception de situation d'apprentissage, et plus particulièrement sur la séquence pédagogique effectivement mise en œuvre en classe. À cette fin, nous présentons à présent le protocole PRaTDL, introduit dans [91, 107], qui permet de structurer les séquences pédagogiques lors de TP en génie logiciel. Ce protocole servira de cadre dans plusieurs contextes expérimentaux présentés dans le chapitre suivant.

3.1.2 Le protocole PRaTDL



FIGURE 3.2 – Illustration du protocole PRaTDL

PRaTDL [91, 107] est une approche d'apprentissage de la programmation supportée par le protocole que l'on décrit ensuite et illustré dans la figure 3.2. Ce protocole intégrant séquences de programmation en autonomie dirigée par les tests et séquences de revue de code par les pairs.

Itérations à implémenter, dans l'ordre :**ITER_1 : Hello, Bob**

La fonction prend en paramètre un nom (une chaîne de caractère) et retourne la chaîne "Hello, <nom>".

Par exemple, si on lui passe "bob", elle retourne "Hello, Bob" (à noter, la majuscule sur le nom).

```
fix ITER_1
```

ITER_2 : Gestion des cris

Si on donne une chaîne de caractères en majuscule, alors répondre en majuscule, et terminer par un point d'exclamation.

Par exemple, si on donne "JERRY", alors retourner "HELLO, JERRY !"

indice : la méthode "toUpperCase()" de la classe String peut vous simplifier la vie. On rappelle que deux chaînes de caractères peuvent être comparées en utilisant *equals*.

```
fix ITER_2
```

ITER_3 : Gestion des null et des chaînes vides

Si on passe en paramètre une chaîne de caractère vide, nulle ou blanche (que des espaces), alors retourner "Hello, my friend".

Par exemple, si on lui passe " ", elle retourne "Hello, my friend".

Indice : la méthode "trim()" de la classe String peut vous simplifier la vie.

```
fix ITER_3
```

FIGURE 3.3 – Extrait d'un sujet de TP typique pouvant être utilisé dans le cadre du protocole PRaTDL

L'objectif est de traiter les exercices d'une feuille d'exercices (dont un extrait est présenté dans la figure 3.3) au cours d'une ou plusieurs séances de TP encadrées par un-e enseignant-e. Les étudiant-es sont libres de travailler en dehors des séances également. Pour chaque exercice, ou question, une série de tests automatisés est donnée. Ces tests permettent de vérifier que la spécification de la fonctionnalité à développer dans la question est bien suivie. Une fois ajoutés et après que l'étudiant-e ait vérifié que les tests échouent car la fonctionnalité n'est pas encore développée, l'étudiant-e se lance dans la phase d'implantation de celle-ci. Peu de guidage est fourni à ce moment-là, et l'étudiant-e est donc libre des moyens à mobiliser pour répondre à la question, le test assurant que le résultat y répond fonctionnellement. Ainsi, la question peut être traitée de plusieurs façons différentes au sein du groupe d'étudiant-es.

Pour s'assurer que chacun-e est confronté-e à la problématique que la question soulève, une phase de revue par les pairs peut être déclenchée grâce à un outil dédié tel qu'Elaastic. Lors de cette phase, chaque étudiant-e est invité-e à répondre à des questions sur une notion précise et à partager des extraits de code en lien avec cette notion. Ensuite, chacun-e peut lire les réponses anonymisées d'une partie aléatoire du groupe et se prononcer sur leur pertinence. Enfin, l'enseignant-e peut faire un point de restitution sur les réponses afin

que tout le monde puisse comprendre la notion peu importe la façon choisie de répondre à la question. L'intérêt de cette liberté donnée aux étudiant-es est d'abord de favoriser un apprentissage actif mais aussi de générer le plus de cas possibles de résolution de la question. L'intention est de mettre en perspective plus largement la notion abordée dans la question et de détecter plus rapidement et efficacement les erreurs et les mauvaises conceptions. Il apparaît dès lors que cette phase est pertinente si un nombre important d'étudiant-es a pu traiter les questions liées à la revue par les pairs. C'est pourquoi la question de l'orchestration de la séquence et en particulier du moment du déclenchement de cette phase de revue par les pairs est cruciale.

Après celle-ci, la phase de travail en solo peut continuer si besoin, et d'autres phases de revue par les pairs peuvent éventuellement avoir lieu dans la séance.

Pour suivre la progression des étudiant-es, il leur est demandé, une fois que les tests passent avec un verdict de succès signifiant que la question est traitée, de marquer que la question est résolue. Pour ce faire, Git est utilisé, ainsi que GitHub afin de rendre le travail des étudiant-es accessible à l'enseignant-e. Ainsi, les étudiant-es vont devoir jalonner leur progression en créant un *commit* pour chaque question résolue, avec un message dédié identifiant celle-ci. Afin de permettre à l'enseignant-e de suivre efficacement la progression des étudiant-es, un message est fourni dans le sujet pour chaque question. Les étudiant-es n'ont qu'à copier-coller ce message lors de la création de leur *commit*.

Enfin, l'enseignant-e doit pouvoir suivre la progression des étudiant-es entre les séances car elle ou il distribue des corrections d'une partie des questions de la feuille quand assez d'étudiant-es les ont traitées.

Prenons un exemple à partir de l'extrait de feuille d'exercices montré en figure 3.3 pour illustrer le déroulement du protocole. La séance de TP commence par une phase de travail en solo. Dans cet exemple, les tests ne sont pas fournis directement, seulement la spécification, le comportement attendu, et les étudiant-es doivent écrire les tests et le code les faisant passer avec succès. Dans d'autres contextes, les tests peuvent être fournis, mais dans tous les cas, cette phase de travail en solo est guidée par des tests à satisfaire.

Une fois que l'enseignant-e estime qu'une partie suffisante du groupe a résolu les questions "ITER_1" et "ITER_2", elle ou il lance une phase de revue par les pairs sur ces deux questions.

Quand cette phase est terminée, la phase de travail en solo sur les exercices reprend. Dans notre exemple, les étudiant-es traitent la question "ITER_3" et ainsi de suite jusqu'à la fin de la séance.

Plus tard, l’enseignant-e sonde le travail du groupe sur GitHub et estime la proportion d’étudiant-es ayant répondu aux trois premières questions suffisante pour en publier la correction.

Ultérieurement, lors d’une séance suivante, les étudiant-es continueront la feuille avec les questions restantes, d’autres revues seront planifiées et d’autres corrections seront données selon le même principe que précédemment.

Nous présentons à présent Git4School, un tableau de bord conçu pour aider les enseignant-es à suivre la progression des étudiant-es au sein de leurs dépôts Git. L’outil décrit dans la prochaine section a été développé initialement pour répondre au besoin précis de faciliter l’orchestration d’interventions pédagogiques en TP dans le cadre de l’approche PRaTDL.

3.2 GIT4SCHOOL

Git4School affiche des indicateurs sur la progression des étudiant-es à travers différents graphes dans le cadre du protocole PRaTDL mais peut être utilisé dans n’importe quel autre cadre qui nécessite le suivi des *commits* sur des dépôts Git. Je l’ai développé en 2019 à l’occasion d’un stage de Master 1 supervisé par Jean-Baptiste Raclet et Franck Silvestre. J’ai poursuivi le développement de la plateforme lors d’une alternance avec mon équipe encadrante actuelle, au cours de laquelle nous avons développé toute la partie concernant la configuration ainsi que la partie *DevOps* et gestion de projet. Nous ne développerons pas les détails techniques de l’application entière dans le manuscrit puisque nous avons déjà écrit une documentation technique et le code source du projet est entièrement et librement accessible sur le dépôt public GitHub ¹.

Dans les sections suivantes, nous présenterons les fonctionnalités, les choix de conception importants ainsi que quelques points techniques. La section 3.2.1 donne des informations sur l’architecture générale du projet ainsi que des éléments techniques divers. La section 3.2.2 débute la description fonctionnelle de l’application avec la configuration des *assignments*. La section 3.2.3 détaille les fonctionnalités de Git4School.

3.2.1 Éléments d’architecture

L’architecture de Git4School a été conçue avec comme intention d’ajouter le moins que possible de traitement et stockage côté serveur de données personnelles autres que celles concédées à GitHub dans le cadre de ses conditions générales d’utilisation. Git4School est structuré autour de 3 composants, présentés dans la figure 3.4, qui interagissent entre eux de sorte à rendre opérationnel le site tout en minimisant

Le DevOps connaît un essor important depuis quelques années dans l’industrie. Il s’agit d’une pratique technique visant à l’unification du développement logiciel et de l’administration des infrastructures informatiques nécessaires à l’exploitation du logiciel en contexte opérationnel. Concrètement, il s’agit d’automatiser des processus comme le déploiement, la livraison et l’intégration.

1. <https://github.com/git4school/git4school-visu>

l'accès de chacun des composants aux seules données utiles à leur bon fonctionnement. Contrairement à l'architecture courante des applications web, nous n'avons pas de serveur *backend* créé pour Git4School, ce qui permet de réduire la propagation des données personnelles des utilisateurs et utilisatrices. Pour héberger le site Git4School, nous utilisons le service d'hébergement web de Firebase, distribuant le code Javascript de l'application aux utilisateurs et utilisatrices. Firebase nous facilite aussi l'authentification à Git4School via GitHub. Pour récupérer les données des dépôts étudiants, nous utilisons l'API REST de GitHub. Le troisième composant est le navigateur de l'utilisateur ou de l'utilisatrice qui exécute côté client le code Javascript et reçoit directement les données de GitHub. Finalement, dans l'état actuel du projet et pour l'expérimentation, les données sont d'abord stockées sur GitHub dans le cadre de la sauvegarde du travail étudiant, puis envoyées côté client où elles sont traitées localement à l'exécution. Pour nos travaux, nous avons utilisé GitHub, mais dans l'éventualité où une institution entreprend d'héberger sa propre forge Git, comme une version hébergée de Gitlab, l'architecture et la conception de Git4School permettrait alors une faible volatilité des données personnelles en dehors des structures habilitées (serveur Gitlab de l'institution et machines des enseignant-es) tout en minimisant la difficulté de sa mise en place.

Pour assurer la continuité entre les utilisations de Git4School et permettre à un utilisateur ou une utilisatrice de ne pas avoir à recréer ses *assignments* à chaque fois qu'elle ou il utilise la plateforme, nous sauvegardons dans le navigateur les informations concernant la configuration des *assignments*, en dehors des *commits*. Pour cela, nous utilisons la librairie Dexie.js qui permet de manipuler *IndexedDB*. Il s'agit d'une base de données non relationnelle orientée objet basée sur Javascript et intégrée au navigateur dans laquelle on peut stocker des données sous différents formats : chaînes de caractères, nombres, objets, tableaux et dates. Les objets créés dans cette base de données ne sont accessibles que par le code exécuté dans les sites associés au nom de domaine lors de leur création. Ainsi, on s'assure que d'autres applications n'auront pas accès aux configurations.

Concernant l'authentification, nous utiliserons deux mots pour marquer la différence sémantique entre, d'une part, la connexion à Git4School par le biais du service d'authentification de GitHub Classroom, et d'autre part, l'authentification auprès de GitHub, caractérisée par la possession en mémoire du token GitHub permettant l'utilisation de l'API GitHub. Dans le premier cas, on parlera de connexion et d'être connecté-e, et l'utilisateur ou l'utilisatrice est alors associée à une session de connexion gérée par Firebase. Cette session perdure de façon indépendante aux sessions d'exécution de Git4School. Dans le deuxième cas, on dira qu'un utilisateur ou une utilisatrice est authentifiée, le client possède le token GitHub, forcément délivré après

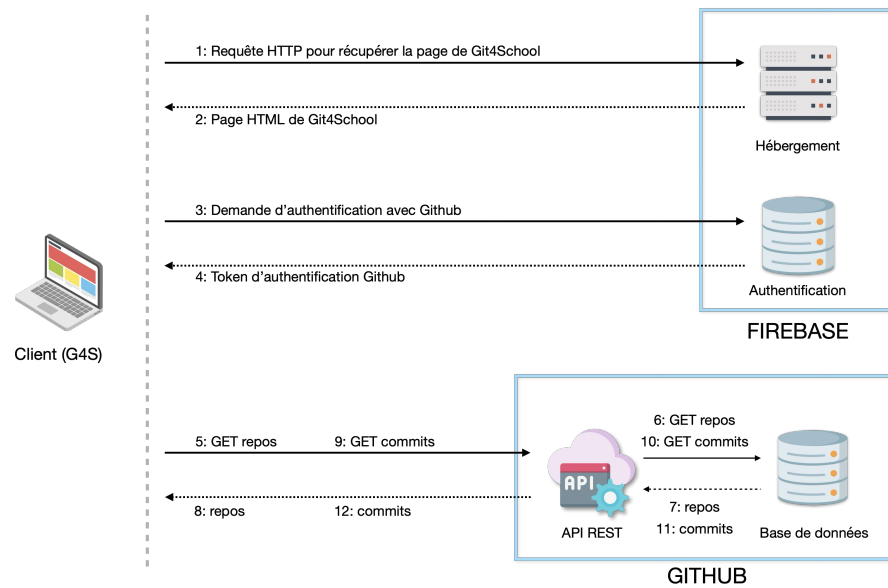


FIGURE 3.4 – Diagramme d'architecture et de communication de Git4School

la connexion. Par mesure de sécurité, ce token n'est stocké qu'en mémoire, dans une variable Javascript, le temps du contexte d'exécution. Comme illustré dans la figure 3.5, dès que l'utilisateur ou l'utilisatrice recharge Git4School, le token est perdu et il est alors nécessaire de se connecter à nouveau pour être authentifiée. Toutefois, grâce à la gestion des sessions de connexion de Firebase, il est possible d'éviter à l'utilisateur ou l'utilisatrice d'avoir à se reconnecter manuellement. Le diagramme UML d'états-transitions de la figure 3.5 montre plus en détails ce mécanisme : lorsque l'utilisateur ou l'utilisatrice arrive sur la page d'accueil de Git4School, le système vérifie si elle ou il est connecté via une session de connexion (qui perdure jusqu'à la déconnexion manuelle de l'utilisateur ou l'utilisatrice). Si c'est le cas, on lance la procédure d'authentification sans intervention humaine. De cette façon, nous avons mis en place un mécanisme de type "*keep me logged in*" sans stocker de token côté client et sans *backend*. Ainsi les données restent sécurisées tout en étant uniquement stockées côté client respectant ainsi leur propriété sans autorisation supplémentaire.

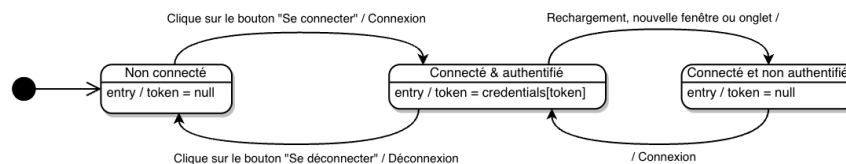


FIGURE 3.5 – Machine à états représentant le processus d'authentification de Git4School

En conclusion, Git4School a été conçu selon une approche respectueuse de la protection des données, en ne stockant aucune donnée

personnelle sur un serveur et en s'appuyant exclusivement sur le mécanisme d'authentification de GitHub. Cette architecture légère garantit, par conception, la conformité de l'outil aux exigences légales en matière de protection des données personnelles.

3.2.2 Configuration

Après s'être authentifié avec GitHub, l'utilisateur ou l'utilisatrice se retrouve face à la liste des *assignments* qu'elle a créés. Un *assignment* est le premier élément intervenant dans la configuration de Git4School. Lors de la création dans Git4School d'un *assignment*, dont l'interface est présentée dans la figure 3.6, les seuls champs obligatoires sont un titre et une durée de séance par défaut (initialement réglée à 1h30). D'autres champs sont utiles à titre purement indicatif à l'attention de l'utilisateur ou l'utilisatrice : le cours, le programme dans lequel il s'inscrit et l'année. Les champs "date de début" et "date de fin" sont les dates entre lesquelles les étudiant-es travaillent en séance ou hors séance. Ils ne sont pas essentiels mais ils répondent à une problématique que l'on a rencontrée à l'utilisation. En effet, lorsque les dépôts des étudiant-es sont créés à partir du squelette de dépôt initial (avec GitHub Classroom), ils héritent des *commits* de ce dernier. Il est alors possible d'avoir un *commit* initial datant de plusieurs années avant la première séance de TP. Il peut aussi arriver qu'un-e enseignant-e regarde les dépôts après la fin des séances et ne veut pas visualiser les *commits* faits après. Ces dates conditionnent la récupération des *commits* depuis GitHub et affecte donc également l'export des données et sont importantes si l'on veut analyser les traces entre deux dates précises.

Enfin, le champ "questions" permet de renseigner les questions de la feuille d'exercices dans l'ordre dans lequel elles doivent être traitées. C'est cet ordre qui est utilisé par la suite pour calculer la progression des étudiant-es.

Pour profiter des fonctionnalités de Git4School, il faut encore renseigner les dépôts Git des étudiant-es dans l'*assignment*. Il est possible d'ajouter un à un chaque dépôt, l'accès est alors vérifié, mais il est bien plus pratique d'utiliser la fonction d'ajouts multiples car on peut avoir jusqu'à plusieurs centaines de dépôts pour un cours dans les cas extrêmes. Comme le montre la figure 3.7, il suffit d'entrer le début de l'identifiant du dépôt. Grâce à GitHub Classroom, tous les dépôts créés depuis le lien fourni pour l'*assignment* ont le même préfixe défini au préalable ("2023-24-semp-projet-ilu1-" dans notre cas). Les nouveaux dépôts créés appartiennent à une organisation GitHub elle aussi définie au préalable (ici "UPS-ILU"). Ainsi, en tapant "UPS-ILU/2023-24-semp-projet-ilu1-", on obtient tous les dépôts de l'*assignment* créés par les étudiant-es.

Configurations

Métadonnées Dépôts

Editer les métadonnées

Titre *

Cours Programme Année

Date de début Date de fin

Questions Durée de séance par défaut *

Pour ajouter une question, tapez la puis appuyez sur entrer

Aucune question

* Ces champs doivent être remplis pour valider le formulaire.

FIGURE 3.6 – Capture d’écran de la configuration d’un *assignment* de Git4School

Ajouter des dépôts

UPS-ILU/2023-24-semp-projet-ilu1-

✓	Nom :	URL :
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...
✓	2023-24-semp-projet-ilu1-i	https://github.com/UPS-ILU/2023-24-semp-projet-ilu1-...

32 selected / 32 total

Indiquez le groupe de TP pour tous les dépôts sélectionnés

Ajouter

FIGURE 3.7 – Capture d’écran de l’ajout des dépôts Git des étudiant-es dans Git4School

Pour chaque dépôt, un nom et un groupe de TP sont associés. S’il est possible de les renseigner manuellement lors d’un ajout individuel ou de les modifier après cela, ces champs peuvent être récupérés automatiquement grâce à un format attendu du README présent dans chaque dépôt.

La figure 3.8 montre un exemple du README que l’on demande aux étudiant-es de remplir pour retrouver leur nom et leur groupe de TP. Git4School lit ce fichier et retrouve ces informations grâce à des expressions régulières que l’on détaille dans la documentation utilisateur². Pour respecter le droit à l’anonymat des étudiant-es, si

2. <https://git4school.github.io/docs/conventions.html>

```
# First IT practical

### LAST NAME : DOE
### First name : John
### TP group :
- [ ] 11
- [x] 12
- [ ] 21
- [ ] 22
```

FIGURE 3.8 – Capture d’écran du format standardisé du README dans Git4School

elles ou ils ne veulent pas faire figurer leur nom dans le README du dépôt GitHub, bien qu’il soit privé généralement, il est possible de ne pas le remplir avec ces informations. L’enseignant-e doit alors prendre connaissance du pseudonyme GitHub des étudiant-es désireuses et désireux de maintenir leur anonymat sur GitHub, puis d’entrer manuellement leur nom et prénom lors de la configuration de l’*assignment* sur Git4School.

Enfin, pour faciliter l’utilisation de l’outil pour tou·tes les enseignant-es du TP, il est possible d’exporter la configuration d’un *assignment* pour l’importer sur un autre ordinateur. Ainsi, l’enseignant-e en charge du cours peut préparer l’*assignment* et diffuser la configuration à tou·tes les intervenant-es du TP.

3.2.3 Fonctionnalités

Le tableau de bord met à disposition de l’enseignant-e trois graphes. Ils mettent en perspective les *commits* récupérés depuis GitHub et les données sur le contexte d’orchestration des séances fournies par l’enseignant-e. Ces dernières données sont les horaires des séances et les dates des revues de code par les pairs et des corrections (revues et corrections auxquelles on pourra faire référence en parlant de *jalons*) et permettent de contextualiser les *commits*. Grâce à cela, Git4School apporte des indicateurs pour l’aide à la prise de décisions. Un point commun subsiste entre ces graphes, ils permettent de visualiser des indicateurs sur les *commits* des étudiant-es, typés en associant un marqueur temporel via leur couleur selon leur antériorité ou leur postériorité avec les jalons fournis par l’enseignant-e. Le principe est simple :

- Le *commit* est noir s'il n'est pas un marqueur de résolution de question ; on appelle alors cela un *commit* intermédiaire.
- Le *commit* est vert s'il marque la résolution d'une question et qu'il est antérieur à la revue par les pairs associée.
- Le *commit* est orange s'il marque la résolution d'une question et qu'il est postérieur à la revue associée mais antérieur à la correction associée.
- Le *commit* est rouge s'il marque la résolution d'une question et qu'il est postérieur à la correction associée.

3.2.3.1 Visualisation des commits dans le temps



FIGURE 3.9 – Vue d'ensemble du tableau de bord de Git4School

Le premier graphe "Vue d'ensemble" (cf. la figure 3.9) permet de visualiser les *commits* de chaque étudiant-e. L'axe des abscisses représente les dates, tandis que l'axe des ordonnées liste les étudiant-es. Les *commits* sont représentés par des points sur le graphe, placés selon leur date de création, permettant ainsi de voir à quel moment précis chaque *commit* a été généré.

On retrouve également plusieurs éléments contextuels importants sur ce graphe :

- Les plages horaires des séances de TP, représentées par des rectangles bleus
- Les jalons, sous forme de lignes verticales :
 - Bleu pour les revues.
 - Rouge pour les corrections.
 - Noir pour les autres jalons divers.

Ces jalons et séances peuvent être ajoutées par l'enseignant-e en cliquant directement sur le graphe à la date souhaitée, offrant un moyen simple et intuitif d'ajouter ces données contextuelles (cf. les

The figure consists of two side-by-side screenshots of web forms in the Git4School application.

(a) Ajout d'une séance (Create a session): This form has a title bar 'Créer une séance' with a close button. It includes a 'Date' field with a calendar icon (set to 2022-10-02), 'Heure de début' (start time) and 'Heure de fin' (end time) fields with up/down arrows (set to 15:35 and 17:05 respectively), a 'Groupe de TP' dropdown menu (set to 'Tout'), and a 'Notes personnelles' text area. A red asterisk indicates that these fields are required for validation. A blue 'Confirmer' button is at the bottom.

(b) Ajout d'un jalon (Create a milestone): This form has a title bar 'Créer un jalon' with a close button. It includes a 'Date' field with a calendar icon (set to 2022-12-09), an 'Heure' field with a clock icon (set to 22:45), a 'Label' text field, a 'Groupe de TP' dropdown menu (set to 'Tout'), a 'Questions' section with a text input and a 'Ajouter une question' button, a 'Type de jalon' section with radio buttons for 'Correction', 'Revue', and 'Autre' (with a red error message 'Un type de jalon doit être choisi !'), and a 'Notes personnelles' text area. A red asterisk indicates that these fields are required for validation. A blue 'Confirmer' button is at the bottom.

FIGURE 3.10 – Captures d’écran des fenêtres d’ajout de données contextuelles dans Git4School

interfaces d’ajout des données contextuelles dans la figure 3.10).

De plus, l’interface propose plusieurs options de filtrage destinées à faciliter la lecture du graphe. L’utilisateur peut ainsi restreindre l’affichage à un groupe de TP spécifique grâce à une liste déroulante située au-dessus du graphe, ou encore se focaliser sur les *commits* associés à une question résolue, afin d’analyser plus finement des moments clés de l’activité étudiante.

Ces fonctionnalités de visualisation et de filtrage permettent à l’enseignant·e d’avoir une vue d’ensemble détaillée et personnalisable de l’activité des étudiant·es au fil du temps.

Deux niveaux d’interaction sont possibles avec les *commits*, correspondant à deux grains d’inspection distincts. Un survol à la souris fait apparaître une infobulle contenant l’horodatage, le message ainsi que l’auteur ou l’autrice du *commit*. Un clic sur le *commit* ouvre quant à lui, dans un nouvel onglet, le *diff* correspondant sur GitHub, comme l’illustre la figure 3.11.

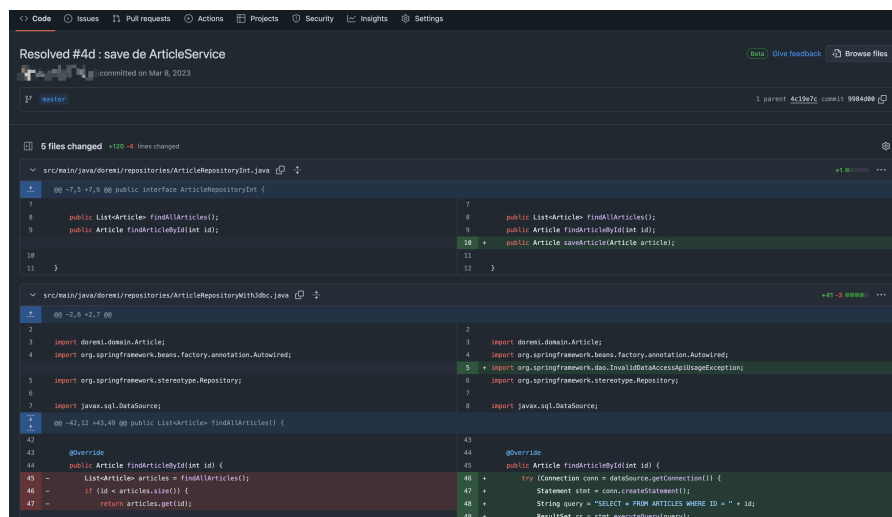


FIGURE 3.11 – Capture d'écran du *diff* ouvert sur GitHub en cliquant sur un *commit* d'étudiant-e dans Git4School

3.2.3.2 Visualisation des commits et de la progression de chaque étudiant-e

Le deuxième graphique "Commits des étudiants" (cf. figure 3.12) est un graphique en barre empilée. Chaque barre représente un dépôt d'un-e étudiant-e. En ordonnée, nous avons choisi d'utiliser la part des *commits* sur le total en pourcentage et non le nombre de *commits* car cela permet de se rendre compte plus facilement des proportions de chaque type de *commits* pour chaque étudiant-e et de comparer les profils. En passant le curseur sur une barre, le nombre de *commits* et la part précise des *commits* de ce type par rapport au nombre total de *commits* de l'étudiant-e sont affichés.

De plus, deux courbes sont affichées. La courbe bleu foncé indique la dernière question résolue (l'ordre des questions étant celui avec lequel les questions ont été définies lors de la configuration de l'assignment). La courbe bleu clair indique le nombre total de *commits* faits au total. Ces deux courbes prennent comme référence de valeur maximale la plus grande valeur parmi tous les dépôts affichés. Encore une fois, on peut ici aussi filtrer selon le groupe de TP, filtre qui reste actif pour une même session pour faciliter l'utilisation de Git4School par un-e enseignant-e qui est en charge d'un groupe.

Enfin, pour ce graphe et le prochain, il est possible de recharger les données depuis GitHub grâce à un bouton. Il est aussi possible de visualiser l'état du graphe à un temps antérieur grâce au curseur horizontal au dessus du graphe.

3.2.3.3 Visualisation du taux d'achèvement des questions

Le troisième graphique "Avancement des questions" (cf. figure 3.13) est aussi un graphique à barre empilée mais cette fois-ci les barres représentent la répartition des type de *commits* pour chaque question.

FIGURE 3.12 – Graphe des *commits* de chaque étudiant-e dans Git4School

Il y a aussi le filtre de groupe de TP et le curseur pour voir le graphe à un temps antérieur. Un nouveau curseur est présent et permet de régler un seuil représenté par une barre rouge horizontale. Il s'agit d'un élément graphique qui permet à l'enseignant-e de repérer très rapidement les questions pour lesquelles elle ou il peut planifier une revue ou une correction. Lorsque l'on passe la souris sur une barre, on peut voir les étudiant-es qui ont résolu la question avec le type de *commit* associé ou si elles ou ils n'ont pas encore résolu la question.

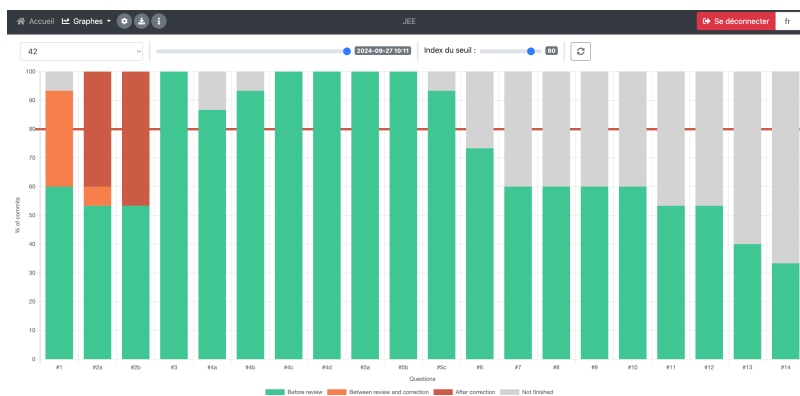


FIGURE 3.13 – Graphe du taux d'achèvement des questions dans Git4School

Nous venons de présenter des éléments de conception et les fonctionnalités de Git4School, concluant ainsi la phase de conception de cette première itération du cycle DBR. Les deux chapitres suivants abordent la *phase d'évaluation*, d'abord qualitative puis quantitative, à travers des expérimentations de Git4School conduites dans différents contextes.

ÉVALUATION QUALITATIVE : RETOURS D'EXPÉRIENCE DES ENSEIGNANT·ES

4.1	Contexte des expérimentations	48
4.2	Étude de deux cas représentatifs	48
4.2.1	Étude de cas 1	49
4.2.2	Étude de cas 2	50
4.3	Résultats et discussion	50
4.3.1	Déclenchement des interventions planifiées . .	50
4.3.2	Obtenir un feedback sur l'efficacité d'une inter- vention planifiée	51
4.3.3	Identification des étudiant·es en difficulté . . .	51
4.3.4	Information sur l'activité des étudiant·es en de- hors des cours	52
4.3.5	Qualification de la difficulté d'un exercice . . .	52
4.3.6	Comparaison de niveaux entre groupes d'étu- diant·es	53
4.3.7	Synthèse des observations et limites	53
4.3.8	Analyse au regard de la conscience de la situation	54
4.3.9	Limites de l'étude	55

Ce chapitre ouvre la *phase d'évaluation* de la première itération du cycle DBR et constitue la première des deux évaluations de Git4School. L'enjeu est d'évaluer dans quelle mesure l'outil soutient la prise de décision enseignante en séance, c'est-à-dire s'il permet de percevoir l'activité des étudiant·es (niveau 1 de la conscience de la situation) et de comprendre la situation pédagogique (niveau 2). Pour cela, nous adoptons une approche qualitative fondée sur les retours d'expérience des enseignant·es ayant utilisé Git4School au cours d'une année scolaire dans différents enseignements, selon des modalités d'utilisation légèrement variables.

La section 4.1 décrit le contexte dans lequel les expérimentations ont été conduites. La section 4.2 détaille deux études de cas représentatifs des observations faites sur l'ensemble de l'expérimentation. Enfin, la section 4.3 propose une synthèse des constats relatifs à l'utilisation de Git4School comme outil d'aide à la prise de décision en cours de séance, ainsi qu'une analyse au regard du cadre de la conscience de la situation et les limites de cette étude qui motivent l'évaluation quantitative présentée dans le chapitre suivant.

4.1 CONTEXTE DES EXPÉRIMENTATIONS

Durant l'année universitaire 2019-2020, Git4School a été expérimenté dans le cadre de 5 enseignements différents portant sur le développement logiciel, au sein de 2 établissements d'enseignement supérieur français, couvrant les niveaux CITE 2011¹ 5, 6 et 7 :

- Institut Universitaire de Technologie de Rodez, dans le cadre de son DUT et de sa Licence Professionnelle en informatique ;
- Université Toulouse 3 Paul Sabatier, dans le cadre de son Master en génie logiciel.

Le tableau 4.1 présente quelques données sur ces enseignements. Nous avons conduit une étude qualitative, à partir des expérimentations dans les 5 premiers cours listés dans le tableau 1 en annexe, qui ont eu lieu dans l'année universitaire 2019-2020. Les protocoles concernant l'orchestration des cours sont précisés dans les sections suivantes mais, de manière générale, les étudiant·es devaient résoudre des séries de questions à partir d'une feuille d'exercices et déclarer qu'une question avait été résolue à l'aide d'un *commit* au sein d'un dépôt Git individuel. Pour chaque cours, les séances de TP ont été supervisées et l'ordre des questions était fixe.

Nous avons analysé les données de 180 dépôts Git créés par 156 étudiant·es, contenant 3082 *commits*.

Niveau CITE	5 (DUT)	6 (Licence Pro)	7 (Master)	
Année d'étude	2 ^{ème}	3 ^{ème}	1 ^{ère}	2 ^{ème}
Nombre de dépôts	73	16	55	36
Nombre d'étudiant·es	49	16	55	36

TABLE 4.1 – Nombre de dépôts en fonction du niveau d'études

4.2 ÉTUDE DE DEUX CAS REPRÉSENTATIFS DES OBSERVATIONS FAITES SUR L'ENSEMBLE DE L'EXPÉRIMENTATION

Pour présenter nos résultats, nous nous concentrons sur 2 des 5 expérimentations faites entre 2019 et 2020, car elles représentent deux cas avec des contextes différents et elles permettent d'illustrer toutes les observations que nous présentons par la suite dans la section 4.3. En effet, ces études de cas ont eu lieu dans le cadre de cours d'informatique dispensés à différents niveaux de l'enseignement supérieur et suivant différents modèles d'apprentissage. Ces cas permettent de s'intéresser à l'utilisation de Git4School en Master génie logiciel et en DUT informatique.

1. https://fr.wikipedia.org/wiki/Classification_internationale_type_de_l%27%C3%A9ducation

Dans le premier cas, le protocole PRaTDL a été suivi et des tests étaient fournis pour valider les questions, alors que dans le deuxième cas, le protocole d'enseignement suivi était plus basique, sans orchestration de la phase d'instruction par les pairs grâce au protocole PRaTDL et sans fournir de tests de validation des exercices. Les deux études de cas visaient à fournir des éléments de réponse à la QR1, en explorant dans quelles mesures les visualisations fournies par Git4School peuvent concrètement aider les enseignant-es à prendre des décisions au cours des différentes séquences composant leur cours.

4.2.1 Étude de cas 1

La première étude a eu lieu en deuxième année de Master en génie logiciel dans le cadre d'un cours consacré à l'API de persistance de Java (JPA). Un groupe de 36 étudiant-es était inscrit à ce cours. Le groupe a suivi 10 séances de 2 heures chacune, au cours desquelles les étudiant-es ont mis en pratique les connaissances théoriques qu'elles ou ils avaient reçues précédemment sur JPA. Ces séances ont suivi le protocole PRaTDL présenté dans la section 3.1.2 pour couvrir d'une part l'acquisition de compétences individuelles en programmation et d'autre part l'acquisition de compétences sociales liées au développement logiciel. Ainsi, les séances se sont déroulées comme une succession de cycles composés de 3 phases ordonnées comme suit :

- Phase de travail en autonomie : chaque étudiant-e travaille individuellement sur un même projet spécifié par un ensemble de tests unitaires et d'intégration. Les résultats et les messages fournis lors des tests automatisés donnent aux étudiant-es un *feedback* qui facilite leur progression vers chaque objectif d'apprentissage.
- Phase de revue par les pairs : les étudiant-es participent à une séquence de revue par les pairs durant laquelle elles et ils sont invités à présenter et à argumenter en faveur de leur solution à un exercice donné ou à présenter leur compréhension de l'utilisation de l'API. Cette phase de revue par les pairs est orchestrée à l'aide de la plateforme Elaastic² qui met en œuvre certains principes d'apprentissage actif présentés par SILVESTRE, VIDAL et BROISIN dans [108].
- Phase de restitution : l'enseignant présente les résultats et commente une ou plusieurs solutions ou explications. Il anime les discussions. À la fin de chaque contribution significative au projet, il a été demandé à chaque étudiant-e de réaliser un *commit* et un *push* de son travail sur son dépôt GitHub dès que tous les tests passaient avec succès sur la partie concernée du projet. L'indication de soumettre le travail faisait partie de l'énoncé du sujet.

2. <https://elaastic.irit.fr/elaastic-questions>

4.2.2 Étude de cas 2

La deuxième étude de cas a eu lieu en deuxième année d'un cours de DUT informatique visant à acquérir des connaissances et des compétences en matière de développement d'applications web avec le langage de programmation PHP. Deux groupes d'environ 25 étudiant·es ont participé au cours. La plupart d'entre elles et eux étaient novices sur le sujet enseigné. Pendant 5 semaines, l'enseignant a utilisé Git4School. Durant cette période, chaque semaine, les étudiant·es ont reçu un cours théorique d'une heure sur l'utilisation d'une bibliothèque pour connecter des applications PHP avec des bases de données relationnelles ou sur la pertinence du modèle de conception Model-View-Controller (MVC) dans une application web.

Chaque semaine, les étudiant·es ont aussi mis en pratique ce contenu théorique pendant une session d'une heure trente suivant un modèle d'apprentissage assez simple : elles et ils devaient essayer de compléter les exercices jusqu'à ce que l'enseignant décide d'interrompre le travail individuel pour donner des conseils ou présenter la solution d'un exercice ou de l'ensemble de l'exercice. Les étudiant·es étaient encouragé·es à s'entraider, mais aucun protocole d'enseignement précis n'a été établi pour parvenir à cette instruction par les pairs, contrairement à la première étude de cas où le protocole PRaTDL guidait cette phase.

Pour permettre l'utilisation de Git4School, il a été demandé aux étudiant·es de réaliser un *commit* et *push* de leur travail à la fin de chaque exercice. Les indications de *commit* et *push* du travail ont été explicitement énoncées dans les sujets des différents TP. Les exercices n'étant pas accompagnés de tests automatiques, les étudiant·es décidaient elles-mêmes et eux-mêmes si l'exercice était terminé. Une vérification par l'enseignant était possible sur demande.

4.3 RÉSULTATS ET DISCUSSION

Nous discutons maintenant des observations qui ont émergé de l'utilisation de Git4School au cours des études de cas décrites précédemment.

4.3.1 Déclenchement des interventions planifiées

La visualisation du taux d'achèvement par exercice a été très utile pour déclencher les interventions prévues dans les situations d'apprentissage (phases de revue par les pairs pendant les séquences relatives à l'étude de cas 1 et phases de restitution dans les deux études de cas). En effet, il est possible de déclencher une intervention à partir d'un seuil configurable de taux d'achèvement des étudiant·es. Comme le montre la figure 3.13, le seuil personnalisé apparaît sur la visualisation sous la forme d'une ligne horizontale rouge. Par exemple, pour un

exercice donné, l'enseignant de la première étude de cas déclenchait la phase de revue par les pairs lorsque plus ou moins 70% des étudiant-es avait terminé l'exercice.

4.3.2 *Obtenir un feedback sur l'efficacité d'une intervention planifiée*

Les trois visualisations proposées par Git4School affichent les *commits* des étudiant-es avec une couleur en fonction de la position dans le temps du *commit* par rapport aux interventions déclenchées par l'enseignant-e. En particulier, la visualisation du taux d'achèvement par exercice donne un aperçu précis de l'efficacité des interventions planifiées en montrant combien de *commits* sont réalisés par les étudiant-es après chaque type d'intervention. Un-e enseignant-e peut alors facilement prendre conscience de l'impact de ses interventions. Plus les interventions planifiées font partie de la conception de l'apprentissage, plus les informations obtenues sont riches. En effet, dans l'étude de cas 2 où les évaluations par les pairs n'étaient pas explicitement prévues, il n'a pas été possible de mesurer l'impact des revues par les pairs qui étaient improvisées car elles n'étaient pas prises en compte dans l'outil comme des jalons. D'autre part, les informations fournies par Git4School étant faciles à partager, Git4School peut aider à l'adoption de techniques, de ressources et/ou d'outils pédagogiques à fort impact à plus grande échelle dans une institution.

4.3.3 *Identification des étudiant-es en difficulté*

La visualisation du type de *commits* pour chaque étudiant-e permet d'identifier rapidement celles ou ceux ayant le plus de difficultés.

Par exemple, dans la figure 4.1, on peut voir que la quatrième barre est dominée par l'orange et le rouge, ce qui indique que l'étudiant-e correspondant fait rarement les exercices tout-e seul-e et, dans la plupart des cas, après la publication de la solution par l'enseignant.

Lors de séances de TP avec de petits groupes d'étudiant-es, l'enseignant-e peut ne pas avoir besoin d'un tel outil pour identifier les étudiant-es en difficulté. Cependant, Git4School donne rapidement des indicateurs pour identifier les étudiant-es en difficulté avec lesquelles et lesquels l'enseignant-e peut décider d'interagir sans attendre qu'elles ou ils demandent de l'aide. Git4School permet ainsi aux enseignant-es de mieux répartir le temps passé avec les étudiant-es sur la base d'indicateurs objectifs sur leurs besoins et a donc été utilisé comme aide pour réaliser des interventions non planifiées ciblant les étudiant-es en difficulté.

Dans le contexte de l'enseignement hybride ou à distance, induit par les confinements liés à l'épidémie COVID19 par exemple, Git4School a été très utile pour identifier les étudiant-es en difficulté en com-



FIGURE 4.1 – Capture d'écran de la visualisation des *commits* par étudiant dans Git4School dans le cadre d'un des cas d'étude

pensant en partie l'absence de signaux donnés par les étudiant·es traditionnellement lors des séances de TP en face à face.

4.3.4 Information sur l'activité des étudiant·es en dehors des cours

La vue d'ensemble (figure 3.9) affiche tous les *commits* effectués par les étudiant·es lors des sessions de TP (fond bleu clair) et en dehors de la classe (fond blanc). Git4School permet ainsi d'identifier les étudiant·es qui poursuivent leur travail en dehors des sessions de TP.

Pour l'étude de cas 2, il s'agissait en fait d'un moyen de découvrir que moins de 5% des étudiant·es avaient l'habitude de travailler en dehors des sessions de TP, même en cas de demande explicite de travail à réaliser à la maison. L'enseignant s'est questionné sur ce manque d'engagement des étudiant·es. Le paragraphe suivant décrit comment Git4School a permis d'identifier la cause la plus probable : un niveau de difficulté du travail demandé inapproprié au regard du niveau d'expertise des étudiant·es.

4.3.5 Qualification de la difficulté d'un exercice

La visualisation du taux d'achèvement par exercice (Figure 3.13) peut aider à identifier une situation anormale pour un exercice donné.

Dans l'étude de cas 1, le seuil de déclenchement de la prochaine intervention prévue n'était pas atteint quand les étudiant·es ne pouvaient pas effectuer l'exercice de façon autonome. Cette observation, pouvant être faite en temps réel par l'enseignant·e en consultant le tableau de bord, offre alors la possibilité de réaliser une remédiation sur les concepts qui sont visés dans l'exercice.

Si la visualisation du taux d'achèvement sur l'ensemble des exercices est largement dominée par la couleur rouge, cela signifie que les étudiant·es n'ont réalisé les exercices qu'après la publication de la solution fournie par l'enseignant·e. Cette situation peut remettre en cause la conception de la situation d'apprentissage dans son ensemble. En effet, dans l'étude de cas 2, comme les premiers exercices ont pris plus de temps que prévu, l'enseignant a anticipé la publication de la solution de ces exercices afin que les étudiant·es puissent consacrer plus de temps aux exercices suivants jugés plus importants par l'enseignant.

Dans cette situation, Git4School fournit des indicateurs objectifs pour aider à identifier les ressources pédagogiques inadaptées pour atteindre les objectifs d'apprentissage initiaux.

4.3.6 *Comparaison de niveaux entre groupes d'étudiant·es*

La deuxième étude de cas a impliqué deux groupes d'étudiant·es travaillant en parallèle sur les mêmes exercices. Assez rapidement, les visualisations sont apparues très différentes dans les deux groupes. Après les deux premières sessions, alors que le premier groupe montrait une distribution compacte des *commits* verts sur les quatre premiers exercices, le second groupe montrait une distribution plus large et inégale des *commits* verts jusqu'au septième exercice.

Git4School est ainsi apparu comme un outil inattendu pour identifier les différences dans les progrès des étudiant·es d'un même groupe ainsi que l'hétérogénéité entre deux groupes inscrits au même cours. En réaction, l'enseignant a pu donner du travail supplémentaire aux étudiant·es les plus avancé·es afin de maintenir un rythme plus cohérent entre les deux groupes. D'autres choix auraient pu être faits (par exemple, mobiliser les étudiant·es avancé·es pour aider les autres), mais l'important ici est de constater que Git4School a permis la mise en œuvre d'interventions correctives imprévues déclenchées sur la base d'informations exactes.

4.3.7 *Synthèse des observations et limites*

Les résultats de l'étude montrent que Git4School permet aux enseignant·es d'obtenir des informations utiles pour la prise de décision en séance, que ce soit pour des interventions planifiées ou non planifiées. Les visualisations proposées permettent d'identifier rapidement les étudiant·es en difficulté, de suivre leur activité en dehors des cours, d'évaluer la difficulté des exercices et de comparer les niveaux entre différents groupes.

L'outil semble particulièrement pertinent dans trois contextes, comme illustré dans les sous-sections précédentes :

- Pour les grands groupes où il est plus difficile pour l'enseignant·e d'avoir une vision globale de la progression de chaque étudiant·e (cf. section 4.3.3).
- Dans le cadre de l'enseignement à distance où les signaux habituels donnés par les étudiant·es en présentiel sont absents (cf. section 4.3.4).
- Pour les enseignements nécessitant un suivi régulier de la progression comme les TP (cf. section 4.3.5 et section 4.3.6).

Les retours des enseignant·es suggèrent que Git4School facilite la mise en place d'une pédagogie personnalisée en permettant d'adapter les interventions selon les besoins identifiés. L'outil aide également à détecter des problèmes dans la conception de la situation d'apprentissage, comme un niveau de difficulté inadapté des exercices, permettant ainsi d'améliorer itérativement les ressources pédagogiques.

4.3.8 *Analyse au regard de la conscience de la situation*

Les résultats de l'évaluation de Git4School peuvent être relus à travers le prisme du modèle de conscience de la situation [29] introduit dans l'état de l'art.

Au **niveau 1 (perception)**, Git4School remplit pleinement son rôle. Les visualisations temporelles des *commits*, les barres de progression et les graphiques de répartition rendent visible l'activité des étudiant·es qui serait autrement difficilement accessible à l'enseignant·e, en particulier dans les grands groupes (cf. section 4.3.3) ou en enseignement à distance (cf. section 4.3.4). Le tableau de bord agit comme un instrument de perception augmentée, signalant en temps réel l'avancement individuel et collectif.

Au **niveau 2 (compréhension)**, les résultats sont plus nuancés. Certaines fonctionnalités soutiennent effectivement la compréhension de la situation : l'identification des étudiant·es en difficulté (cf. section 4.3.3), la comparaison entre groupes (cf. section 4.3.6) et la qualification de la difficulté des exercices (cf. section 4.3.5) relèvent d'une interprétation de la situation qui dépasse la simple perception. Cependant, cette compréhension reste partielle. Les visualisations de Git4School ne permettent pas d'accéder aux processus d'apprentissage sous-jacents : l'enseignant·e peut constater qu'un·e étudiant·e est en retard, mais ne peut pas comprendre pourquoi elle ou il rencontre des difficultés, ni observer les étapes intermédiaires de la résolution.

Le **niveau 3 (projection)**, qui permettrait d'anticiper les évolutions futures de la situation (par exemple, prédire qu'un·e étudiant·e va décrocher), n'est pas soutenu explicitement par le tableau de bord dans sa forme actuelle. Cependant, les retours des enseignant·es montrent qu'elles et ils prennent effectivement des décisions d'intervention en s'appuyant sur le tableau de bord, ce qui implique une capacité de projection opérationnelle. Conformément au cadre d'Endsley [29, 30],

leurs modèles mentaux de la dynamique d’une séance de TP leur permettent d’inférer les évolutions probables à partir des données de perception et de compréhension fournies par l’outil. Dans le contexte spécifique des séances de TP de programmation, où les configurations de difficulté sont récurrentes et connues des enseignant-es expérimenté-es, les décisions d’intervention (aller aider un-e étudiant-e en retard, ajuster la difficulté d’un exercice) relèvent de décisions routinières au sens de Klein [64] : l’enseignant-e expérimenté-e reconnaît une configuration et active une réponse connue, sans nécessiter de projection explicitement calculée par un outil.

L’absence de soutien explicite du niveau 3 ne constitue donc pas une limite rédhibitoire dans notre contexte d’usage spécifique. Cette analyse met néanmoins en évidence un plafond de compréhension (niveau 2) imposé par la nature des traces disponibles, qui limite la compréhension fine des processus d’apprentissage sous-jacents. Comme nous le verrons dans le chapitre suivant, l’analyse par PM tentera d’approfondir ce niveau en explorant les enchaînements de comportements, mais se heurtera elle aussi aux limites intrinsèques des traces manuelles.

4.3.9 Limites de l’étude

L’étude présentée comporte plusieurs limites que nous présentons maintenant.

Premièrement, elle repose sur les observations et retours des enseignant-es impliqués dans l’équipe de recherche. Bien que leurs observations semblent pertinentes, cela introduit un biais potentiel de subjectivité puisqu’elles et ils sont à la fois concepteurs et conceptrices de l’outil et utilisateurs et utilisatrices.

De plus, l’étude ne permet pas de mesurer quantitativement l’impact réel des interventions pédagogiques déclenchées grâce aux visualisations de Git4School sur la progression des étudiant-es. Elle montre que l’outil permet d’identifier des situations nécessitant une intervention, mais pas l’efficacité de ces interventions sur la qualité des apprentissages.

Les résultats présentés dans ce chapitre, bien que prometteurs, reposent principalement sur des observations subjectives des enseignant-es impliqués dans la conception de l’outil. Pour consolider ces résultats, une approche complémentaire est nécessaire : analyser objectivement les données collectées par Git4School pour y déceler des schémas de comportement qui ne sont pas directement visibles à travers les visualisations du tableau de bord.

C’est l’objet du chapitre suivant, qui présente une étude quantitative utilisant les techniques de PM pour analyser les données augmentées issues de Git4School (traces Git combinées aux informations contextuelles saisies par l’enseignant-e via les jalons). L’objectif est double :

d'une part, vérifier si ces données permettent de découvrir des comportements d'apprentissage complexes qui seraient difficilement identifiables par simple observation ; d'autre part, évaluer si la détection automatisée de ces comportements pourrait, à terme, enrichir le tableau de bord d'indicateurs pertinents pour guider les interventions pédagogiques.

ÉVALUATION QUANTITATIVE : ÉTUDE DES COMPORTEMENTS DES ÉTUDIANT·ES GRÂCE AU PROCESS MINING (PM)

5.1	Le Process Mining	58
5.1.1	Concepts fondamentaux	58
5.1.2	Mise en œuvre du Process Mining avec BupaR	59
5.1.3	Types de visualisations offertes par BupaR	60
5.1.4	Application à l'apprentissage de la programmation	63
5.2	Description des jeux de données	64
5.2.1	Prétraitement des données	65
5.2.2	Analyse avec BupaR	67
5.3	Résultats	68
5.3.1	Activité : question résolue	69
5.3.2	Activité : type de <i>commit</i>	72
5.4	Discussion	72
5.4.1	Identification de comportements des étudiant·es	72
5.4.2	Facteurs influençant le suivi du processus de résolution de question	74
5.4.3	Nuances dans l'interprétation des données	75
5.5	Bilan de l'itération et nouvelle problématique	75
5.5.1	Synthèse des apports	75
5.5.2	Limites convergentes	76
5.5.3	Problématique induite	77

Ce chapitre poursuit la *phase d'évaluation* de cette première itération du cycle DBR et constitue la seconde évaluation de Git4School, complémentaire à l'évaluation qualitative du chapitre 4. Celle-ci a montré que Git4School soutient efficacement la perception de l'activité des étudiant·es (niveau 1 de la conscience de la situation) et partiellement la compréhension de la situation pédagogique (niveau 2), mais les résultats reposent principalement sur des observations subjectives des enseignant·es. Pour consolider et approfondir ces constats, nous adoptons ici une approche quantitative.

L'enjeu est de déterminer si les données collectées par Git4School, c'est-à-dire les traces Git associées aux informations contextuelles saisies via les jalons, recèlent des schémas de comportement des étudiant·es qui ne sont pas directement visibles dans les visualisations du tableau de bord.

L'objectif est double. Il s'agit d'une part d'apporter de nouveaux éléments objectifs de réponse à la QR1 en identifiant ces schémas.

D'autre part, il convient d'évaluer si leur détection automatisée pourrait, à terme, enrichir l'outil et ainsi approfondir le soutien au niveau 2 de la conscience de la situation.

Pour cela, nous nous appuierons sur des données similaires à celles générées avec Git4School lors des 5 expérimentations présentées dans la section 4.1 ainsi que sur la technique du PM.

5.1 LE PROCESS MINING

Cette section définit ce qu'est le Process Mining (PM) et introduit les différentes notions nécessaires pour comprendre la suite de la description de l'étude quantitative.

Le PM vise à découvrir, surveiller et améliorer les processus réels en extrayant des connaissances à partir des journaux d'événements disponibles dans les systèmes d'information [1]. Ces connaissances peuvent prendre différentes formes : par exemple, l'identification de goulots d'étranglement afin d'optimiser l'efficacité du processus, la mise en évidence de schémas comportementaux récurrents susceptibles d'orienter des modifications structurelles plus profondes, ou encore la vérification de la conformité entre le déroulement réel d'un processus et le modèle cible attendu.

5.1.1 Concepts fondamentaux

Un processus est défini comme une suite d'activités réalisées dans un certain ordre afin d'atteindre un objectif donné. Chaque exécution d'une activité correspond à un événement, et la succession chronologique des événements associés à une instance particulière du processus constitue une trace. Une activité représente une étape spécifique du processus, par exemple, dans notre contexte, la résolution d'un exercice particulier. Une même activité peut apparaître plusieurs fois au sein d'une trace ou dans plusieurs traces différentes. Les traces peuvent être de longueur variable : certaines ne comportent qu'un seul événement tandis que d'autres en comprennent un grand nombre, avec ou sans boucles. Enfin, les traces sont regroupées en cas, chaque cas correspondant à une instance complète du processus.

Dans le schéma de la figure 5.1, on visualise un exemple avec trois cas, identifiés simplement par les individus qui réalisent les activités, formant trois traces distinctes. On peut aussi remarquer que la même activité (identifiée par la couleur et le remplissage du cercle) peut être réalisée plusieurs fois dans une même trace, ou pas du tout.

L'ensemble de toutes ces traces constitue un journal d'événements, qui est la matière première du PM. À partir de ce journal, des techniques d'analyse permettent de construire un modèle de processus, qui représente visuellement les différents chemins empruntés par les cas à travers les activités. Ce modèle permet de comprendre les

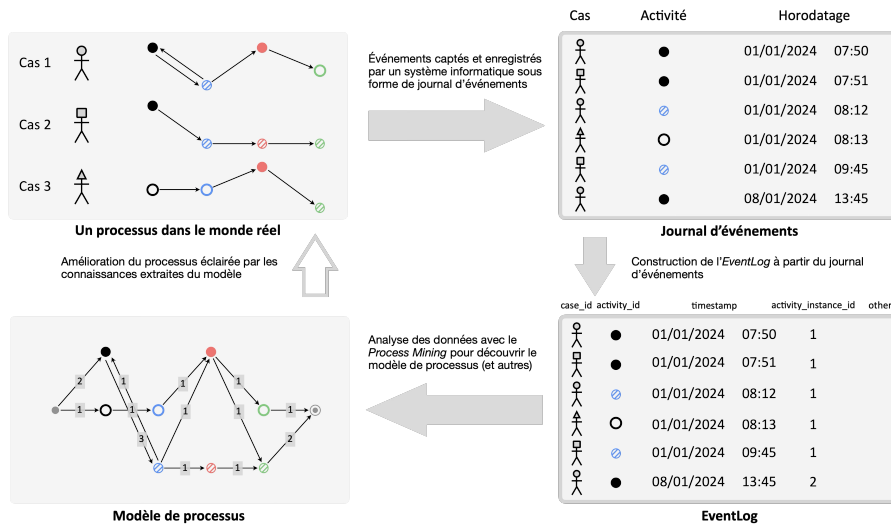


FIGURE 5.1 – Schéma illustrant les concepts du PM

schémas comportementaux typiques, d'identifier les variations et les anomalies, et potentiellement d'améliorer le processus.

5.1.2 Mise en œuvre du Process Mining avec BupaR

Pour appliquer les techniques de PM, nous avons utilisé la bibliothèque bupaR. Cette dernière est une collection d'outils dédiée à l'analyse de processus à partir de journaux d'événements. Chaque activité est associée à des métadonnées telles qu'un identifiant unique, une période d'activité, et d'autres attributs contextuels selon la librairie utilisée pour analyser ces journaux. Cette structure est nécessaire pour l'analyse des processus afin de retracer les chemins empruntés par chaque processus. Ainsi, il est en général nécessaire de transformer la structure des données de journalisation pour obtenir des journaux d'événements utilisables. Dans le cadre de l'utilisation de bupaR, il est nécessaire de transformer les données de journalisation brutes en EventLog qui est la spécification d'un journal d'événements pour la librairie. Un EventLog a pour éléments principaux :

- Un cas : chaque cas est identifié par l'attribut `case_id`.
- Des événements : les événements sont représentés par les attributs `activity_id` et `activity_instance_id`.
- Des attributs temporels : chaque événement est associé à un horodatage qui indique quand il a commencé et terminé et est donné dans l'attribut `timestamp`.
- Des attributs contextuels : des informations supplémentaires sont incluses pour enrichir l'analyse via l'attribut `ressource_id`.

Par exemple, dans la figure 5.2, si on s'intéresse au cas d'une étudiante nommée Julie qui traite une feuille d'exercice donnée. Elle marque la question 1 comme résolue, puis la question 2, puis dé-

cide d'améliorer la solution à la question 1 et la marque à nouveau comme résolue. On aura un processus composé de 3 événements. Le `case_id` sera "Julie" car on identifie le processus unique lié à sa progression dans les exercices (on suppose pour l'exemple que le nom de l'étudiante est unique dans la cohorte des étudiant-es). L'attribut `activity_id` correspond à l'identifiant de l'activité, soit "#1", "#2" puis "#1" à nouveau quand elle revient sur la résolution de la question 1. Cependant, à ce moment-là, l'attribut `activity_instance_id` doit être différent de celui à la première occurrence car il s'agit d'une nouvelle instance de résolution de la question par la même étudiante (donc le même processus, le même `case_id`).

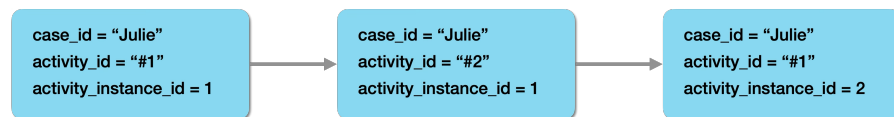


FIGURE 5.2 – Exemple d'un processus de résolution d'exercices par une étudiante

5.1.3 Types de visualisations offertes par BupaR

BupaR propose plusieurs types de visualisations pour analyser les processus, chacune apportant un éclairage différent sur les données.

Carte de processus (*process map*, exemple en figure 5.3) : le flux des activités est représenté sous la forme d'un graphe orienté, où les sommets représentent les activités du processus et les arcs représentent les transitions entre ces activités. Cette structure permet de visualiser clairement l'enchaînement des activités et leurs relations. On distingue plusieurs variantes selon ce que représentent les arcs :

- Une **carte de fréquence** (*frequency map*) permet de visualiser les chemins les plus empruntés dans le processus ; l'épaisseur des arcs indique la fréquence des transitions entre activités. Cette représentation est particulièrement utile pour identifier les parcours dominants et les schémas de comportement récurrents.
- Une **carte de performance** (*performance map*) analyse les durées entre activités en colorant et étiquetant les arcs et les sommets pour représenter des métriques temporelles comme la moyenne, le maximum, le minimum ou la médiane. Deux types de durées peuvent être analysées : l'*idle_time* qui mesure le temps écoulé entre la fin d'une activité source et le début d'une activité cible, et l'*inter_start_time* qui mesure la durée entre les débuts respectifs des activités source et cible, prenant donc en compte la durée de l'activité source. Cette visualisation met en évidence les goulots d'étranglement potentiels et les zones nécessitant une optimisation.

- Une **carte animée** (*animate map*) offre une visualisation dynamique montrant l'évolution du processus dans le temps, où des points représentent chaque cas se déplaçant à travers la carte du processus. Cette animation permet d'observer le parcours individuel de chaque cas et d'identifier des comportements atypiques ou déviants par rapport au flux normal des activités.

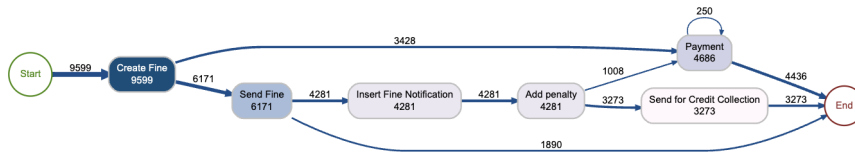


FIGURE 5.3 – Exemple d'une carte de processus

Matrice de processus (exemple en figure 5.4) : les relations entre activités sont représentées sous une forme matricielle dans laquelle chaque cellule indique la fréquence ou la probabilité de transition entre deux activités. Cette visualisation est particulièrement efficace pour identifier les schémas de transition récurrents et les chemins inhabituels. La matrice peut également révéler des motifs cachés qui ne seraient pas immédiatement visibles dans une représentation graphique traditionnelle.

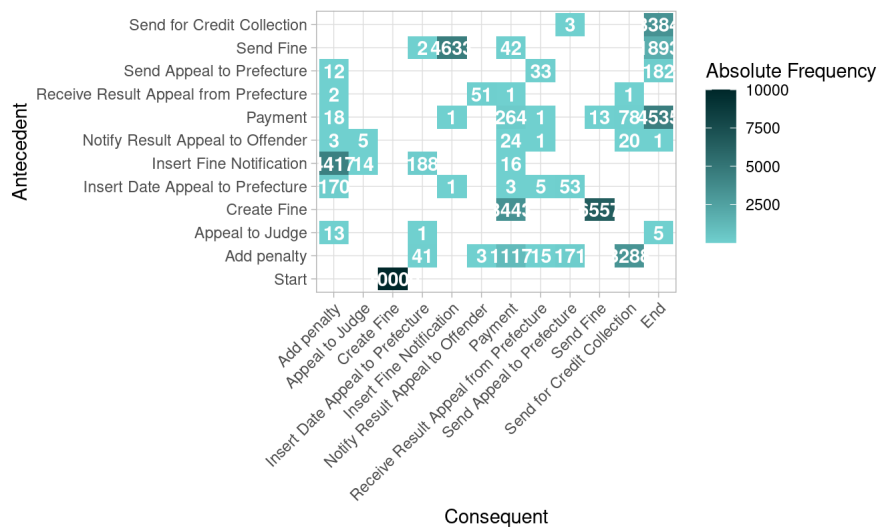


FIGURE 5.4 – Exemple d'une matrice de processus

Diagramme en points (exemple en figure 5.5) : dans cette représentation chronologique, chaque point représente un événement sur une ligne horizontale temporelle correspondant à un cas. Cette visualisation permet d'observer la distribution temporelle peu précise des

activités, notamment les périodes d'intense activité, et les motifs répétitifs dans le processus. Elle est particulièrement utile pour détecter les anomalies temporelles et les schémas récurrents dans le temps.

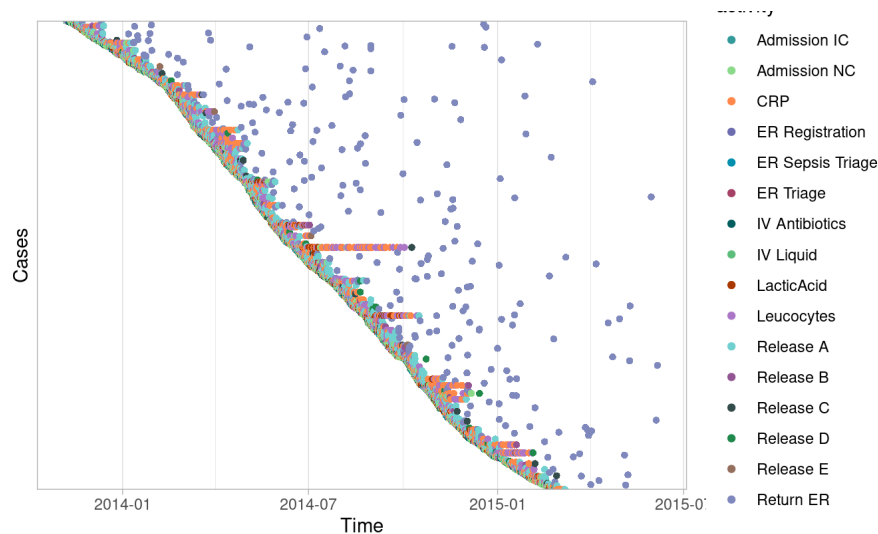


FIGURE 5.5 – Exemple d'un diagramme en points

Explorateur de traces (exemple en figure 5.6) : les différentes traces ou séquences d'activités présentes dans le processus peuvent être visualisées. Il permet notamment de mettre en évidence les séquences d'activités les plus fréquentes ou les moins courantes. Cette visualisation est complémentaire à la carte de processus car elle permet de suivre le cheminement précis de chaque cas, notamment lorsqu'une même activité est répétée plusieurs fois, ce qui n'est pas directement visible dans une carte de processus.

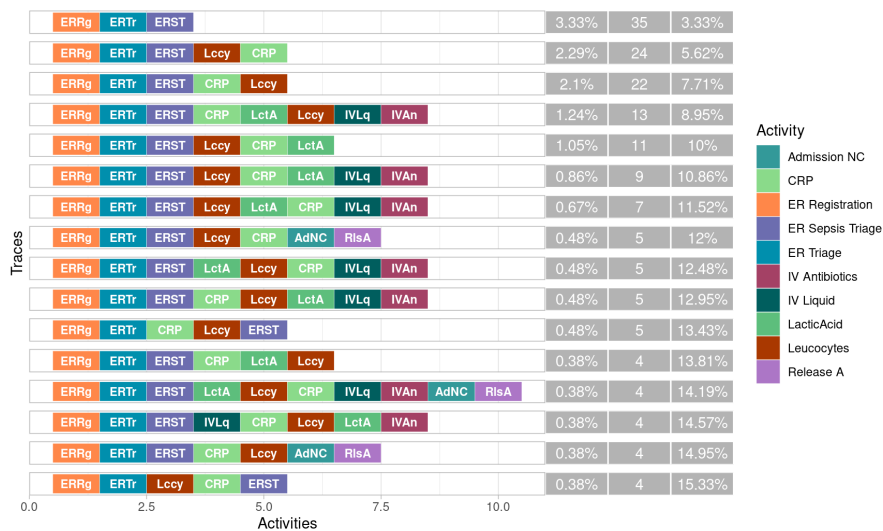


FIGURE 5.6 – Exemple d'un explorateur de traces

Spectre de performance (exemple en figure 5.7) : dans cette représentation chronologique de tous les cas, on peut examiner en détail les durées des activités au sein d'un processus. Elle affiche les durées des activités pour chaque cas sous forme de segments colorés, où chaque segment représente un cas. Les couleurs indiquent généralement les quartiles de temps d'exécution, avec des nuances allant du bleu (plus rapide) au rouge (plus lent). Cette visualisation permet l'identification des variations de performance et des anomalies potentielles.

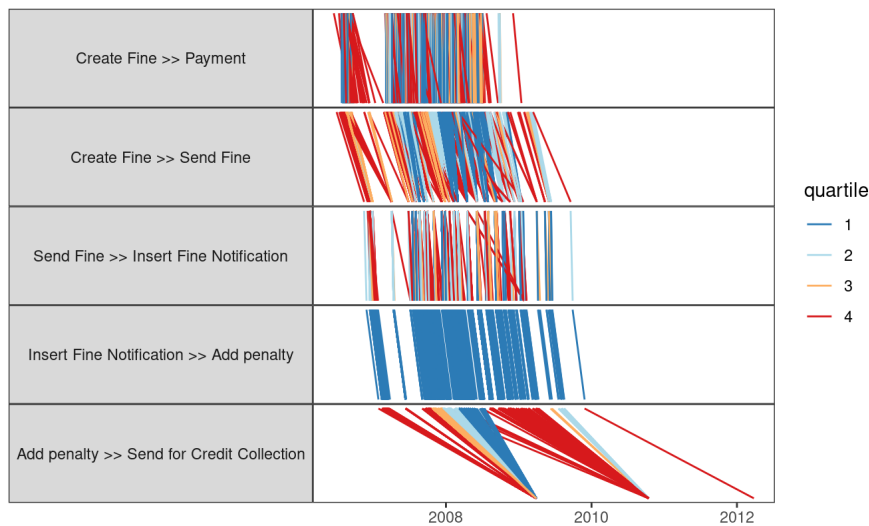


FIGURE 5.7 – Exemple d'un spectre de performance

Ces différentes visualisations se complètent mutuellement et permettent une analyse approfondie des processus. Par exemple, alors qu'une carte de processus peut révéler la structure globale du flux d'activités, la matrice de processus peut mettre en évidence des relations spécifiques entre activités qui seraient moins visibles dans la représentation graphique. De même, le diagramme en points et le spectre de performance offrent des perspectives temporelles complémentaires qui enrichissent la compréhension globale du processus.

5.1.4 Application à l'apprentissage de la programmation

Dans le contexte de l'apprentissage de la programmation, le PM a déjà montré son utilité pour analyser les comportements des étudiant-es à travers leurs interactions avec les systèmes de gestion de versions comme Git [31, 78, 105]. Cette approche permet notamment d'identifier des schémas de progression, des difficultés récurrentes, et d'adapter les interventions pédagogiques en conséquence.

Dans le cadre du suivi de la progression des étudiant-es, chaque étudiant-e constitue un cas distinct, avec sa propre séquence d'événements (ou trace). En effet, on peut imaginer qu'un-e étudiant-e

réalise plusieurs fois l'activité de résolution de la question 1 : une première fois, puis une deuxième fois pour la corriger. Cela soulève un point important concernant la définition de la portée exacte des activités. En effet, il existe deux façons de considérer une activité de résolution de question : soit l'activité est terminée dès qu'un·e étudiant·e marque la question comme résolue, soit elle n'est terminée qu'au dernier marquage de résolution. Dans le premier cas, si un·e étudiant·e marque une question comme résolue puis la corrige plus tard en la marquant à nouveau comme résolue, on considère deux activités distinctes. Dans le second cas, on considère une seule activité qui est mise à jour jusqu'au dernier marquage où elle est finalement terminée. Dans notre étude, nous avons choisi la première approche, traitant chaque marquage comme une activité distincte. Ce choix est motivé par notre volonté d'observer le processus d'apprentissage dans sa granularité la plus fine. En effet, le fait qu'un·e étudiant·e revienne sur une question précédemment validée témoigne d'une démarche itérative (auto-correction, raffinement suite à un feedback) qui constitue un signal pédagogique fort. Masquer ces répétitions en ne conservant que l'état final nous priverait de l'observation de ces dynamiques de correction.

5.2 DESCRIPTION DES JEUX DE DONNÉES

Une deuxième étude, de nature quantitative et exploratoire, a été menée afin d'analyser les données issues des expérimentations exportées depuis Git4School. L'objectif était d'identifier des schémas comportementaux pertinents pouvant éclairer la prise de décision des enseignant·es grâce aux indicateurs intégrés à l'outil. Cette étude s'appuie sur le format de données décrit dans la section 3.2.3, qui combinent les traces Git extraites de GitHub avec des informations contextuelles saisies par l'enseignant·e sous la forme de jalons.

L'analyse repose sur des données collectées entre 2020 et 2022, comprenant une nouvelle instance des cinq premiers cours ainsi qu'un cours supplémentaire, dont le jeu de données est nommé « *OurBusiness* ». Parmi les six expérimentations initialement disponibles, une a été écartée car elle ne comportait aucune intervention pédagogique planifiée (revue ou correction) pendant les séances. Plus précisément, dans ce cours, aucune revue n'a été effectuée en séance et les corrections ont eu lieu après l'ensemble des travaux pratiques, mais avant l'examen final. La correction n'ayant donc pas pu influencer le comportement des étudiant·es pendant les séances, les données correspondantes ont été exclues de l'analyse.

Au total, cinq jeux de données ont été retenus, représentant environ 200 étudiant·es et 3 100 *commits* répartis sur deux années universitaires (2020-2022). La répartition selon le niveau d'études est présentée dans le tableau 5.1.

Niveau CITE	5 (DUT)	6 (Licence pro)	7 (Master)	
Année d'étude	2 ^{ème}	3 ^{ème}	1 ^{ère}	2 ^{ème}
Nombre de dépôts	160	11	49	32
Nombre d'étudiant-es	160	11	49	32

TABLE 5.1 – Nombre de dépôts en fonction du niveau d'études lors de l'analyse quantitative

Les données que nous récupérons à l'export de Git4School contiennent des informations additionnelles aux données Git tirées de GitHub, comme expliqué dans la section 3.2.3, avec les données sur le contexte d'orchestration des séances fournies par l'enseignant-e.

On dit que chaque trace, *commit*, est typée selon son antériorité ou sa postériorité par rapport à une intervention pédagogique planifiée de l'enseignant-e. Ces interventions sont soit des revues par les pairs orchestrées avec Elastic, soit des corrections fournies aux étudiant-es, et concernent un ensemble de questions. Dans la suite de l'analyse, nous nous référerons à ces *commits* typés comme suit :

- **Intc** intermédiaire : le *commit* ne résout pas une question ;
- **Bfrr** avant la revue : le *commit* résout une question avant la revue associée ;
- **Brac** entre la revue et la correction : le *commit* résout une question entre la revue et la correction associées ;
- **Aftc** après la correction : le *commit* résout une question après la correction associée.

Nous devons ensuite transformer ces traces en journaux d'événements afin de pouvoir utiliser les techniques de PM.

5.2.1 Prétraitement des données

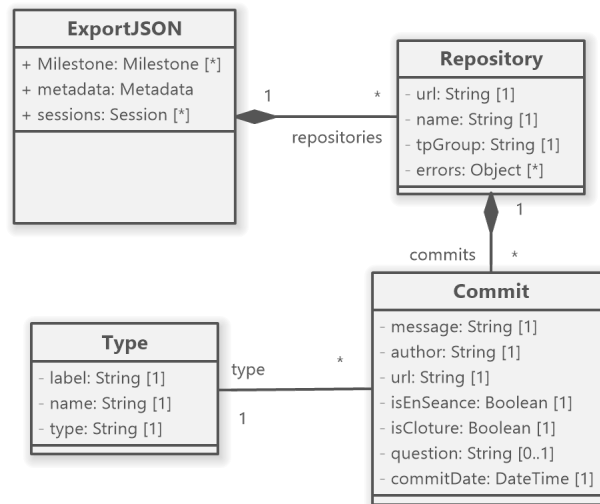
Nos jeux de données comprennent, entre autres¹, la liste des dépôts GitHub de chaque étudiant-e avec la liste des *commits* effectués pendant les travaux pratiques (voir figure 5.8a).

Avant d'analyser les données, une phase de prétraitement a été nécessaire pour adapter la structure des données à la bibliothèque bupaR. Pour cela, un script Python¹ a été écrit avec deux objectifs :

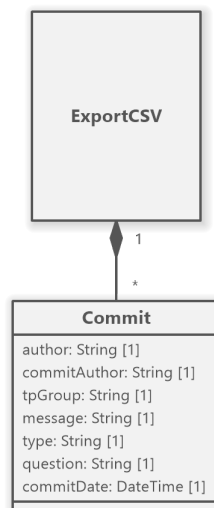
- transformer la structure du modèle de données d'une liste de dépôts GitHub à une liste de *commits* ;
- convertir le format de fichier en CSV.

Les données fournies par Git4School sont au format JSON. Ce format étant difficile à manipuler avec le langage R, nous utilisons une bibliothèque Python pour transformer au format CSV. Nous en profitons

1. Pour plus de détails, visitez le dépôt public : <https://gitlab.irit.fr/talent/TALENT/around-g4s/G4S-to-PM-scripts-and-data-2022>



(a) Modèle de l'export JSON de Git4School



(b) Modèle du CSV utilisé pour l'analyse

FIGURE 5.8 – Modèles de la structure des jeux de données

pour filtrer les données souhaitées pour la suite du traitement : le nom de l'étudiant-e, l'identifiant de l'auteur ou l'autrice du *commit* (il sera nécessaire pour le filtrage décrit dans la section suivante), le groupe de l'étudiant-e, le message brut du *commit*, le type du *commit* et la question résolue s'il y en a une (ou une chaîne vide dans le cas contraire).

Finalement, des fichiers CSV sont produits avec une structure simple (voir figure 5.8b), requise pour la partie analyse présentée dans la section 5.2.2. Ces données seront suffisantes pour construire les journaux d'événements pour deux types d'activités : (1) la résolution d'une

question et (2) le travail dans un contexte pédagogique identifié par le type de *commits*.

5.2.2 Analyse avec BupaR

Après le prétraitement, les jeux de données sont presque prêts à être utilisés avec bupaR. Deux types de journaux d'événements ont été construits, correspondant aux deux types d'activités mentionnés précédemment. L'utilisation de bupaR requiert d'ajouter deux informations, le statut de l'activité et l'identifiant d'une instance d'une activité pour un cas.

Deux colonnes ont donc été ajoutées. Tout d'abord, la colonne *status* est nécessaire pour indiquer les différents statuts par lesquels passe une activité au cours d'un processus. Dans notre cas, on ne considère que des activités ponctuelles (puisque nous n'avons dans les traces que le marqueur de la fin de la réalisation d'une question), un événement marque donc le début et la fin d'une activité. Nous attribuons alors la valeur *complete* pour la colonne *status* à toutes les lignes.

Ensuite, nous ajoutons la colonne *activity_instance* avec une valeur incrémentée sur chaque ligne pour différencier chaque activité pour le même cas. Pour construire les journaux d'événements avec la résolution d'une question en tant qu'activité, nous avons exclu les *commits* **Intc** car nous ne voulons analyser que les séquences des questions résolues.

Enfin, nous filtrons les lignes pour exclure les *commits* automatiques de GitHub Classroom. En effet, lorsque les étudiants créent leur dépôt, un premier *commit* est effectué automatiquement par un robot (dont l'auteur est `github-classroom[bot]`) pour avoir le code de démarrage du cours.

Comme le montre le tableau 5.2, nous avons construit deux journaux d'événements distincts par jeu de données en fonction des propriétés susmentionnées. Ce tableau formalise la correspondance entre les concepts abstraits pour construire un journal d'événements (activité, ressource) et nos données concrètes.

	Basé sur la résolution de questions	Basé sur le type de <i>commit</i>
<code>case_id</code>	<code>author</code>	<code>author</code>
<code>activity_id</code>	<code>question</code>	<code>type</code>
<code>resource_id</code>	<code>type</code>	<code>question</code>
<code>timestamp</code>	<code>commitDate</code>	<code>commitDate</code>

TABLE 5.2 – Construction des journaux d'événements selon l'activité

À l'aide du script, nous avons généré deux rapports R markdown². Ils présentent les mêmes 13 types de graphes, mais l'un utilise les

2. <https://rmarkdown.rstudio.com/>

journaux d'événements basés sur la résolution d'une question, tandis que l'autre se base sur le travail dans un contexte pédagogique identifié par le type de *commits*. BupaR utilise le paquet heuristiquemineR. Ce package utilise les fréquences des transitions entre les activités pour construire les matrices de précédence (ou réseau causal) et les cartes de processus. La section suivante présente les graphiques les plus pertinents pour la suite de la discussion.

5.3 RÉSULTATS

Dans cette section, nous exposons les résultats des analyses avec BupaR en deux temps, d'abord en prenant comme activité la résolution des questions caractérisée par l'identifiant de la question résolue, puis en considérant comme activité la résolution des questions caractérisée par le type des *commits*.

La section 5.3.1 s'intéresse à l'analyse de l'enchaînement des événements afin d'évaluer le respect du processus de résolution des questions par les étudiant-es et d'identifier d'éventuels comportements s'écartant d'une résolution strictement linéaire des questions. Cette analyse repose principalement sur l'étude des cartes de processus, qui permettent de visualiser les transitions les plus fréquentes, ainsi que sur l'utilisation des explorateurs de traces pour examiner en détail les séquences d'événements les plus courantes.

Bien que les cartes animées offrent également la possibilité d'observer l'enchaînement des événements pour un cas spécifique, leur exploitation s'est avérée difficile dans notre contexte. En effet, la distribution temporelle particulière des *commits*, concentrés sur des créneaux hebdomadaires de deux heures mais étalés sur une période beaucoup plus longue, produit une visualisation peu exploitable. L'impossibilité d'ajuster la vitesse de l'animation conduit à une représentation où les points, symbolisant les cas, alternent entre de longues périodes de mouvements très lents et de brefs moments d'activité intense lors des *commits*.

Notre analyse s'est également portée sur l'étude des durées moyennes entre les événements, aspect peu développé dans Git4School. Pour ce faire, nous avons utilisé les cartes de performance, complétées par les diagrammes en points et les spectres de performance qui offrent une représentation pertinente de la dimension temporelle des données.

La section 5.3.2 se concentre sur l'identification des schémas comportementaux récurrents des étudiant-es, notamment en ce qui concerne leur retard ou non dans la résolution des questions. Cette analyse s'appuie principalement sur l'étude des matrices de processus, qui permettent de visualiser de manière synthétique les probabilités des transitions entre les différents événements du processus.

5.3.1 Activité : question résolue

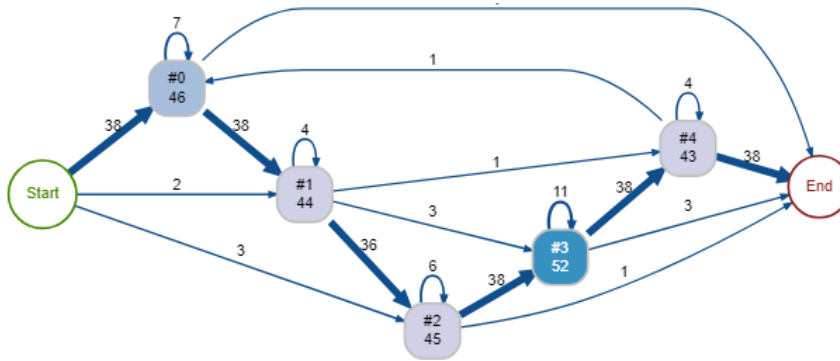


FIGURE 5.9 – *Process map* de résolution des questions pendant une séance de travaux pratiques

La figure 5.9 montre les séquences des réponses des étudiant-es à une question. Notons que chaque nœud dans ce graphique et tous les autres graphiques de type *process map* correspond à un *commit* et donc à une étape particulière de la feuille de travail. Idéalement, le processus de résolution devrait être séquentiel, de la première à la dernière question car les sujets que nous avons mis en place ne permettent pas de sauter une question. Cependant, le graphique montre que la réalité est tout autre. Il existe plusieurs types de transitions entre des questions non consécutives :

- des boucles ;
- des transitions de Q_n (question $\#n$) directement vers l'état End, c'est-à-dire la fin du processus (n n'étant pas la dernière question de la feuille d'exercices) ;
- des transitions de Q_n à Q_{n+m} ($m > 1$) ;
- des transitions de Q_n à Q_{n-m} ($m \geq 1$).

On aperçoit que certain-es étudiant-es bouclent sur la même question. Bien que cela ne représente pas la majorité des transitions, on les retrouve dans tous les jeux de données que nous avons étudiés. Les transitions entre un sommet Q_n et End indiquent que la dernière question résolue par l'étudiant-e était la $\#n$. On observe plusieurs transitions entre Q_n et Q_{n+m} , m étant supérieur à 1. Cela indique donc une transition entre une question et une question qui n'est pas directement la suivante. Or un-e étudiant-e ne peut pas sauter une question, de par la construction du sujet qui contraint de résoudre les questions les unes à la suite des autres.

On observe également des transitions moins fréquentes de Q_n à Q_{n-m} , ce qui représente un-e étudiant-e qui marque une question antérieure comme résolue.

Tous les travaux pratiques que nous avons analysés présentent des *transitions chaotiques* à l'exception de la figure 5.10. Dans celle-ci, seules deux transitions sur 204 (0.98%) sautent une question.

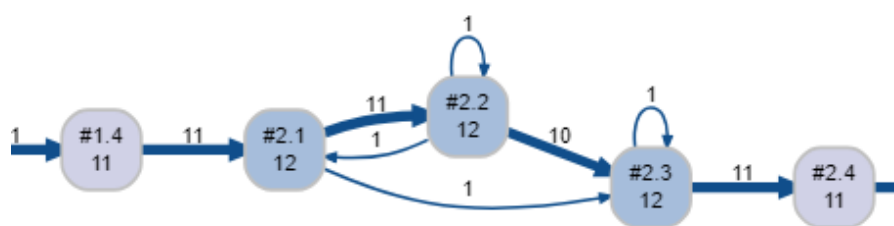
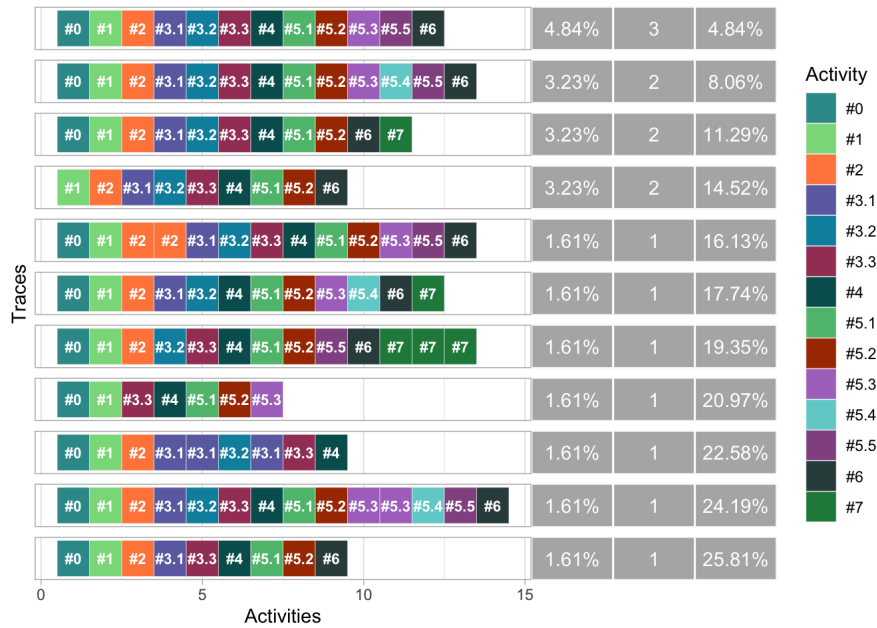


FIGURE 5.10 – Un extrait de la *process map* montrant un processus de résolution de questions bien suivi

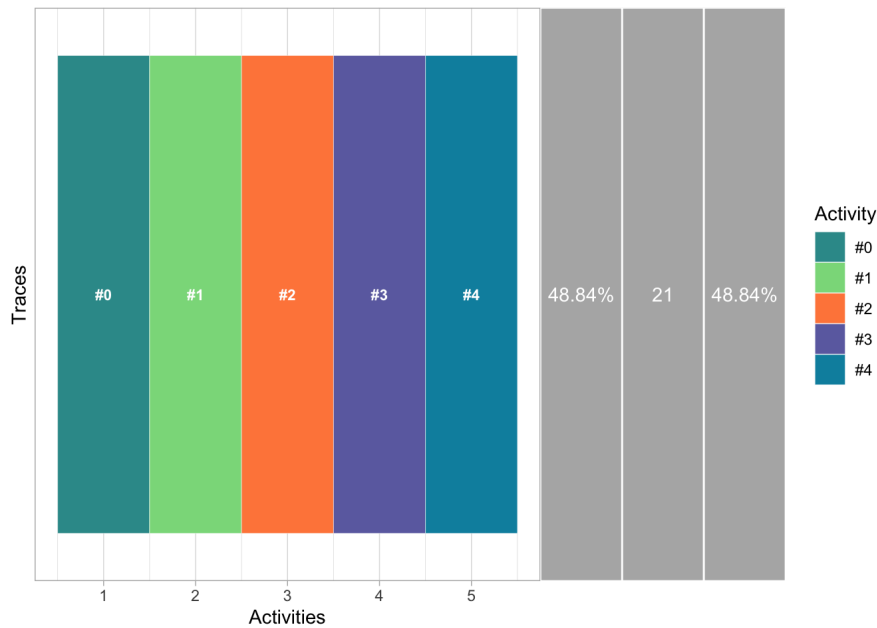
La figure 5.11 présente deux exemples d'explorateurs de traces, chacun représentant 25% des traces totales de leur jeu de données respectif. La figure 5.11a illustre le cas le plus fréquent dans nos analyses : une grande diversité de traces qui diffèrent légèrement les unes des autres, avec des questions qui peuvent être répétées ou sautées, sans qu'un motif clair n'émerge. En revanche, la figure 5.11b, issue du jeu de données FilmProvider, représente un cas particulier où près de 50% des traces suivent un même motif. Cette observation corrobore les résultats précédents observés dans la carte de processus, où ce même jeu de données présentait très peu de transitions chaotiques.

Parmi les analyses disponibles en PM, le profil de performance permet d'étudier la dimension temporelle des processus en mesurant des durées entre activités. Dans notre cas, nous nous sommes intéressés au temps moyen écoulé entre la résolution de deux questions successives. La figure 5.12, issue de l'analyse de performance d'un de nos jeux de données, illustre ce type d'analyse sous forme de graphe, où chaque arête est étiquetée avec la durée moyenne séparant les activités représentées par ses nœuds de départ et d'arrivée. Cependant, notre approche est limitée car Git4School ne capture pas les heures de début d'activité. En effet, nous ne disposons que de l'horodatage des *commits* de résolution pour chaque question, donc de la fin de l'activité de résolution de la question. Bien qu'il soit théoriquement possible d'estimer la durée de résolution en calculant l'intervalle entre deux activités consécutives, cette méthode s'avère peu fiable en pratique. La figure 5.12 illustre parfaitement cette problématique : l'arête reliant les questions #2 et #3 indique une durée moyenne de 187,31 heures. Cette valeur, manifestement aberrante, inclut le temps écoulé entre deux séances de travaux pratiques et ne reflète donc pas le temps réel consacré à la résolution de la question.

La figure 5.13 présente deux diagrammes en points illustrant la distribution temporelle des événements, soit les *commits*. La figure 5.13a montre un cas où les points sont très alignés verticalement, indiquant que les étudiant-es résolvent leurs questions principalement pendant



(a) Exemple typique d'explorateur de traces que l'on obtient



(b) Cas particulier de FilmProvider présentant peu de transitions chaotiques

FIGURE 5.11 – Explorateurs de traces des deux jeux de données PDO_MVC et FilmProvider

les séances de travaux pratiques. À l'inverse, la figure 5.13b présente une distribution plus éparse des points, suggérant que les étudiant-es travaillent également en dehors des créneaux dédiés. Cette visualisation est similaire à celle proposée dans la vue "Vue d'ensemble" de Git4School.

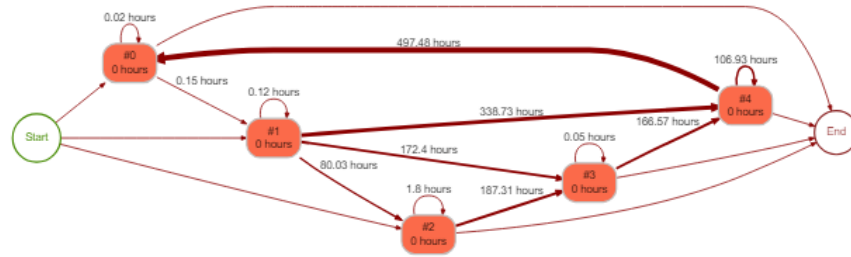


FIGURE 5.12 – *Process map* du profil de performance (utilisant la fonction moyenne)

5.3.2 Activité : type de commit

La matrice de précédence basée sur les antécédents a été générée (voir figure 5.14). Ce graphique montre la probabilité que le prochain *commit* soit d'un type particulier en fonction du type du dernier *commit*. De tous les antécédents, celui qui a le plus de chances d'aboutir à un *commit* **Aftc** est le *commit* **Aftc** lui-même. Il en va de même pour les *commits* **Bfrr**.

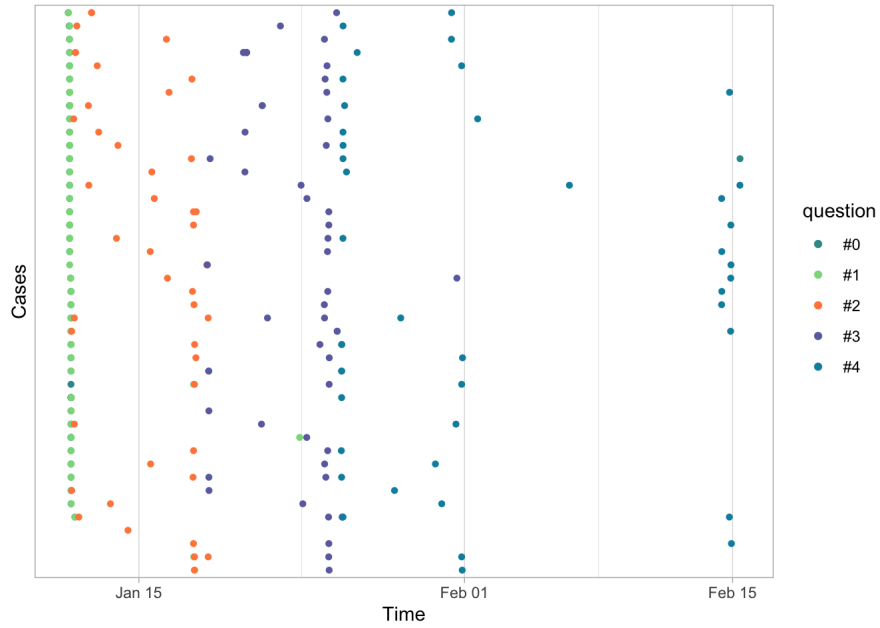
Enfin, la matrice montre que la probabilité la plus faible d'aboutir à un *commit* **Aftc** est un *commit* **Intc**. Cette probabilité est de 4,56%, alors qu'elle est de 10,93% pour l'état **Bfrr**. En d'autres termes, un-e étudiant-e est moins susceptible de produire un *commit* **Aftc** s'il a fait un *commit* **Intc** au lieu d'un *commit* **Bfrr**.

5.4 DISCUSSION

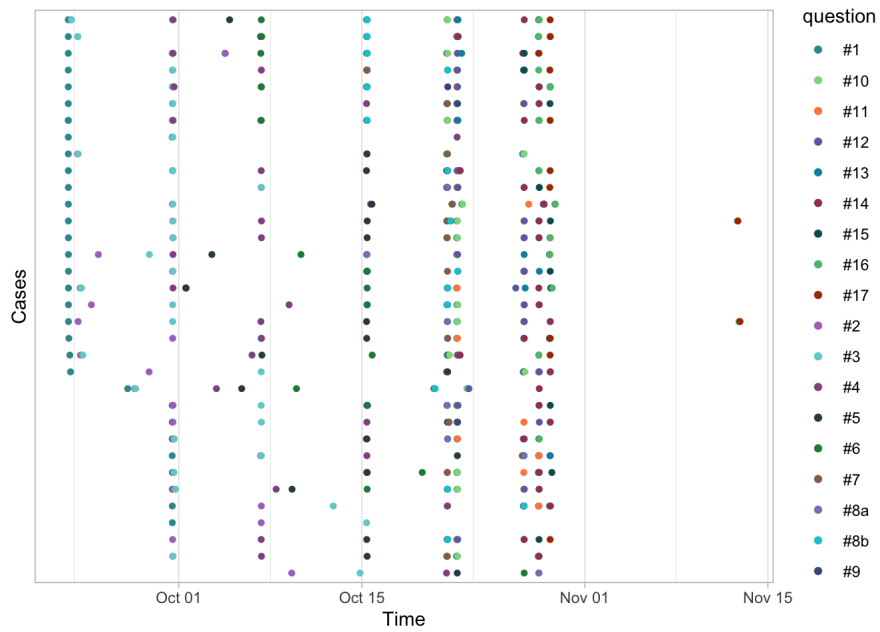
Dans cette section, nous discutons des résultats présentés précédemment en trois parties. D'abord, nous discutons des comportements spécifiques des étudiant-es qui émergent de l'analyse des données. Ensuite, nous identifions des facteurs qui influencent le suivi du processus de résolution des questions. Enfin, nous apportons des nuances importantes dans l'interprétation des données et présentons les apports méthodologiques de notre approche.

5.4.1 Identification de comportements des étudiant-es

Les arêtes $Q_n \rightarrow Q_{n-m}$ montrent que les étudiant-es reviennent aux questions précédentes, ce qui est un comportement non-perceptible dans les visualisations offertes par Git4School. Il y a deux raisons pour lesquelles un-e étudiant-e revient à une question précédente : (1) elle ou il a simplement oublié quelque chose, (2) elle ou il veut corriger ou améliorer sa solution (en raison d'une seconde réflexion ou d'un conseil ou d'une réponse de l'enseignant-e, par exemple). La



(a)



(b)

FIGURE 5.13 – Diagrammes en points des deux jeux de données *FilmProvider* et *M2DL*

dernière raison montre que l'étudiant-e est engagé-e dans un processus d'apprentissage, mais, en l'état, nous ne pouvons pas affirmer qu'il s'agit de la raison principale de ce schéma.

Dans la figure 5.14, nous avons vu que le *commit* **Intc** est le moins susceptible de conduire à un *commit* **Aftc**. Le fait d'effectuer des *commit* **Intc** a une influence positive sur la résolution des questions.

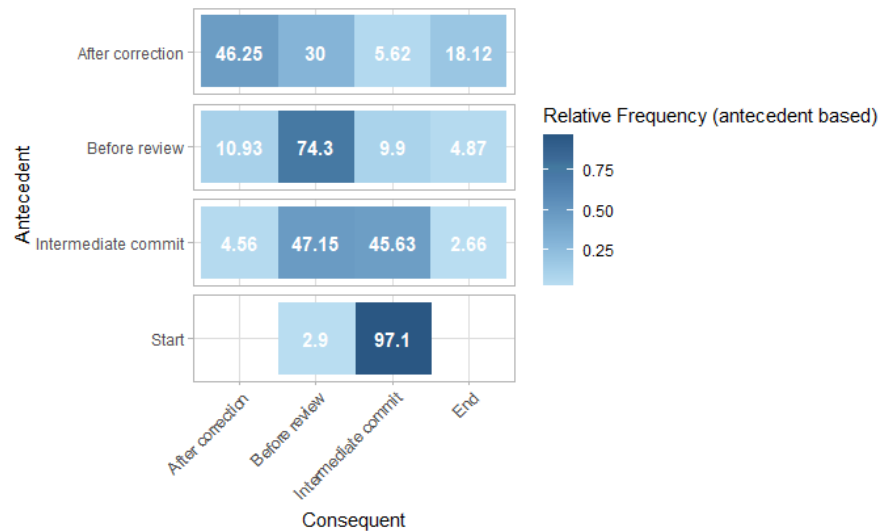


FIGURE 5.14 – Matrice de précedence basée sur le type de *commit* comme activité

Avoir une grande proportion de *commit* **Intc** par rapport au reste du groupe est un indicateur intéressant car il montre que l'étudiant-e s'engage dans le travail à faire en suivant une démarche courante et encouragée dans l'industrie du génie logiciel, à savoir faire des *commits* de son travail régulièrement.

Pour finir avec cette figure, nous pouvons constater la forte probabilité de transition entre deux *commits* **Bfrr** ou deux *commits* **Aftc** que nous pourrions respectivement appeler *boucle vertueuse* et *boucle vicieuse*. Bien que ces deux comportements soient relativement courants (et pas seulement dans le contexte de l'apprentissage), ils peuvent constituer des indicateurs pertinents si on les observe au niveau du groupe. Par exemple, dans un groupe, plus la probabilité d'une *boucle vicieuse* est faible, plus la capacité du groupe à rattraper son retard est grande.

5.4.2 Facteurs influençant le suivi du processus de résolution de question

Concernant le processus décrit dans la figure 5.10 qui contient très peu de transitions chaotiques, nous nous sommes demandés ce qui avait engendré ce résultat sachant que le processus de résolution des questions était semblable aux autres cours. Après avoir interrogé l'enseignant en charge de ce cours, les trois raisons de ce faible taux de *commits* chaotiques sont :

- une dépendance importante entre les questions ;
- un groupe de petite taille ;
- des rappels réguliers du processus de résolution des questions par *commit*.

Nous pouvons donc tirer trois facteurs influençant positivement le processus de résolution de questions :

- La dépendance entre les questions force les étudiant·es à suivre une progression séquentielle, limitant ainsi les possibilités d'erreur de marquage de question résolue.
- La petite taille du groupe permet un meilleur suivi, facilitant l'encadrement du processus de résolution de question.
- Les rappels fréquents sur le processus de résolution de question maintiennent les étudiant·es focalisé·es sur la méthode attendue.

5.4.3 Nuances dans l'interprétation des données

Parce que nos jeux de données présentent essentiellement des questions obligatoires, nous avons supposé que les *commit* **Aftc** étaient un signe négatif que l'étudiant·e est en retard. Cependant, une résolution tardive pour des questions optionnelles signifie que l'étudiant·e essaie de comprendre la solution. Cela montre qu'elle ou il est engagé·e dans un processus d'apprentissage approfondi. Donc, il est important de savoir quelles questions sont facultatives pour voir la différence et avoir une interprétation correcte.

Les jeux de données comportant très peu de *transitions chaotiques* montrent qu'analyser les données sans tenir compte du contexte de leur production n'est pas pertinente. Pour éviter cela, des entretiens avec les enseignant·es concerné·es sont nécessaires.

5.5 BILAN DE L'ITÉRATION ET NOUVELLE PROBLÉMATIQUE

Cette section conclut la première itération du cycle DBR en synthétisant les résultats des trois phases (analyse, conception, évaluation) et en formulant la problématique qui motivera la seconde itération. Les deux évaluations menées dans cette itération, qualitative (chapitre 4) et quantitative (chapitre 5), offrent un regard complémentaire sur les apports et les limites de Git4School.

5.5.1 Synthèse des apports

L'évaluation qualitative a montré que Git4School permet aux enseignant·es d'obtenir des informations utiles pour la prise de décision en séance. L'outil s'avère particulièrement pertinent pour les grands groupes, l'enseignement à distance et le suivi régulier de la progression en TP. Les enseignant·es rapportent pouvoir identifier les étudiant·es en difficulté, qualifier la difficulté des exercices et comparer les niveaux entre groupes, autant d'usages qui facilitent la mise en place d'une pédagogie personnalisée.

L'évaluation quantitative a permis de dépasser ces observations subjectives en analysant objectivement les données collectées par

Git4School grâce au PM. Cette analyse a mis en évidence des schémas de comportement difficilement perceptibles dans les visualisations du tableau de bord, tels que les retours aux questions précédentes, les boucles vertueuses et vicieuses de résolution, ou encore l'influence de facteurs pédagogiques (dépendance entre questions, taille du groupe, rappels du processus) sur le suivi du protocole de résolution.

En termes de conscience de la situation [29], les deux évaluations convergent : Git4School soutient efficacement le **niveau 1** (perception) en rendant visible l'activité des étudiant-es, et partiellement le **niveau 2** (compréhension) en permettant d'identifier certaines situations pédagogiques et certains schémas comportementaux. Bien que le **niveau 3** (projection) ne soit pas soutenu explicitement par l'outil, les enseignant-es mobilisent leurs schémas d'expertise [30, 64] pour construire cette projection à partir des données affichées, ce qui leur permet de prendre des décisions d'intervention adaptées.

5.5.2 Limites convergentes

Cependant, les deux évaluations révèlent aussi des limites convergentes quant à l'utilisation conjointe de Git4School et du protocole PRaTDL pour identifier de manière automatique des comportements d'apprentissage.

La première limite est le **manque de fiabilité des données**. En effet, comme le montre la figure 5.9, un certain nombre d'étudiant-es ne suivent pas strictement le processus de production d'un *commit* lorsqu'elles ou ils terminent une question. Certain-es étudiant-es peuvent travailler, résoudre plusieurs questions et faire un *commit*, ou faire les *commits* bien après, de sorte qu'ils apparaissent comme des *commits* **Aftc** alors que ce n'est pas le cas.

La deuxième limite est le **manque de précision des traces**. Comme nous l'avons vu dans la section 5.3.1, nous disposons seulement du *commit* de résolution de question comme témoin de l'activité de l'étudiant-e. Nous avons abordé le problème que cela engendre en commentant la figure 5.12 lorsque deux résolutions de question sont séparées par un temps de pause entre deux séances de TP. Des durées aberrantes entre deux résolutions apparaissent alors et cela impacte fortement la possibilité d'analyser ces données avec des méthodes comme le profil de performance. De plus, en dehors de ces cas, il est aussi difficile de quantifier précisément la durée d'activité réelle de l'étudiant-e dans une séance de TP. Avec pour seule information la résolution d'une question, et faisant l'hypothèse que l'étudiant-e enchaîne directement sur la résolution de la prochaine question, nous n'avons que le début et la fin de la résolution d'une question. Comment savoir si un-e étudiant-e est actif-ve entre deux *commits* de résolution ? Comment peut-on alors différencier un-e étudiant-e qui prend du temps pour résoudre un exercice car elle ou il a des difficultés, mais qui est actif

et active, d'un-e étudiant-e qui n'est pas engagé-e dans la résolution de l'exercice et qui résout sa question au dernier moment grâce à la solution d'un-e camarade ou à des outils comme *ChatGPT* ?

Pour remédier à chacune de ces deux situations, la méthode à employer est différente et il est important d'avoir des indicateurs qui peuvent compléter l'observation de l'enseignant-e de l'attitude des étudiant-es en séance, surtout lorsque celle-ci n'est pas possible (si l'enseignant-e est trop sollicité-e par les étudiant-es ou si l'enseignement se fait à distance).

Enfin, la troisième limite réside dans le **manque de généricité** de la méthode. Celle-ci repose exclusivement sur des travaux pratiques utilisant Git, ce qui soulève deux difficultés majeures. D'une part, elle est difficilement applicable auprès d'étudiant-es de première année, car elle suppose un niveau minimal de maîtrise de Git. D'autre part, elle restreint son champ d'application aux étudiant-es engagé-es dans des activités de développement logiciel.

5.5.3 *Problématique induite*

Ces trois limites (manque de fiabilité, manque de précision et manque de généricité) ont une cause commune : la dépendance envers une génération manuelle des traces par les étudiant-es, via des *commits* dont la granularité et la fréquence sont laissées à leur discrétion. Tant que la collecte de données reposera sur cette action explicite, les analyses qui en découlent seront contraintes par les biais inhérents à ce mode de génération. Pour dépasser ces blocages, il ne suffit pas d'affiner les méthodes d'analyse, mais il faut repenser fondamentalement la manière dont les traces sont produites.

En termes de conscience de la situation, la compréhension fine des processus d'apprentissage, c'est-à-dire comprendre *comment* les étudiant-es travaillent, et pas seulement *où* elles et ils en sont, reste hors de portée tant que les traces se limitent aux *commits* manuels de résolution de question.

Pour approfondir le niveau 2 de la conscience de la situation, il est nécessaire de disposer de traces plus riches, plus fines et plus fiables.

C'est ce constat qui nous amène à formuler la problématique suivante : **Comment améliorer la génération de traces pour raffiner le suivi des étudiant-es en rendant visibles leurs comportements d'apprentissage au cours des résolutions d'exercices de TP de programmation ?**

Cette problématique ouvre sur une nouvelle itération du cycle DBR, présentée dans la partie suivante, qui se concentrera sur la conception et l'évaluation d'un système de génération automatique de traces d'apprentissage.

Deuxième partie

ITÉRATION 2 : GÉNÉRATION DE TRACES D'APPRENTISSAGE

L'itération précédente a montré que le tableau de bord soutient efficacement la perception de l'activité des étudiant·es (niveau 1 de la conscience de la situation), mais que la compréhension fine des processus d'apprentissage (niveau 2) reste limitée par la nature des traces manuelles. Cette partie présente la seconde itération du cycle DBR, qui vise à approfondir ce niveau 2 en améliorant la génération de traces, afin de rendre observables les comportements d'apprentissage qui demeurent invisibles avec les seuls commits manuels. Comme pour l'itération précédente, elle se structure selon les trois phases du DBR : une *phase d'analyse* à travers un état de l'art sur la génération de traces et le Process Mining, une *phase de conception* du système LAWG, et une *phase d'évaluation* en trois étapes (vérification, validation et expérimentation en contexte écologique).

ÉTAT DE L'ART

6.1	Collecte et analyse de traces en programmation	82
6.1.1	Enjeux et état des lieux	82
6.1.2	Niveaux de granularité des traces	82
6.1.3	Cadre de référence pour l'instrumentation	83
6.2	Représentation et gestion des traces d'apprentissage	83
6.2.1	xAPI et les Learning Record Stores	83
6.2.2	Les Systèmes à Base de Traces (SBT) et kTBS	84
6.2.3	ProgSnap2 : un format standardisé pour les données de processus de programmation	85
6.2.4	Git comme générateur et système à base de traces d'apprentissage	85
6.3	Génération de traces pour le suivi fin de l'activité	87
6.3.1	Approches fondées sur les journaux Git	87
6.3.2	Instrumentation d'environnements de développement	88
6.3.3	Synthèse et positionnement	89
6.4	Qualité des données : cadres théoriques	90

L'itération précédente a permis de concevoir et d'évaluer Git4School, un TBA exploitant les traces issues de Git pour soutenir la prise de décision enseignante en temps réel. Au regard du modèle de conscience de la situation [29], les évaluations qualitative et quantitative ont convergé vers un constat clair : Git4School soutient efficacement le **niveau 1** (perception) en rendant visible l'activité des étudiant-es, et partiellement le **niveau 2** (compréhension) en permettant d'identifier certaines situations pédagogiques. Si le **niveau 3** (projection) n'est pas soutenu explicitement par l'outil, les enseignant-es expérimenté-es parviennent néanmoins à prendre des décisions d'intervention en mobilisant leurs schémas d'expertise [30, 64]. Cependant, la compréhension fine des processus d'apprentissage, c'est-à-dire comprendre comment les étudiant-es travaillent et pas seulement où elles et ils en sont, reste hors de portée tant que les traces se limitent aux *commits* manuels de résolution de question. Ces limites (manque de fiabilité, de précision et de généralité) ont une cause commune : la dépendance envers une génération manuelle des traces par les étudiant-es.

Cette seconde itération du cycle DBR vise donc à dépasser ce plafond en repensant fondamentalement la manière dont les traces sont produites. L'objectif est de disposer de traces plus riches, plus fines et plus fiables, afin d'approfondir le niveau 2 de la conscience de la

situation et de rendre observables les comportements d'apprentissage au cours de la résolution d'exercices. Pour ce faire, nous explorons la conception d'un système de génération automatique de traces qui soit indépendant de l'IDE utilisé par l'étudiant-e, et qui intègre le contexte pédagogique fourni par l'enseignant-e.

Ce chapitre dresse d'abord un état de l'art des approches de collecte, de représentation et de génération de traces fines d'apprentissage en programmation, avant de formuler les questions de recherche qui guideront la conception et l'évaluation de notre proposition.

6.1 COLLECTE ET ANALYSE DE TRACES EN PROGRAMMATION

Avant d'examiner les formats de représentation et les techniques de génération de traces, il convient de situer le champ de recherche dans lequel s'inscrit cette itération : la collecte et l'analyse de données de processus de programmation à des fins éducatives.

6.1.1 Enjeux et état des lieux

Dans leur rapport de groupe de travail ITiCSE, IHANTOLA et al. [51] dressent un panorama des travaux en *educational data mining* et LA appliqués à la programmation. Leur revue de la littérature (2005–2015) met en évidence que la grande majorité des études reposent sur des métriques simples (nombre de soumissions, taux de réussite aux tests, temps passé) et portent sur un seul cours dans une seule institution. Les auteurs et autrices identifient un besoin de données de processus plus riches, permettant de dépasser la simple mesure de résultats pour comprendre comment les étudiant-es résolvent les problèmes. Ce constat rejoint directement les limites que nous avons identifiées à l'issue de l'itération 1 : les *commits* manuels de résolution de question ne capturent que le résultat, pas le processus.

6.1.2 Niveaux de granularité des traces

La richesse des traces collectées dépend fondamentalement de leur niveau de granularité. VIHAVAINEN, LUUKKAINEN et IHANTOLA [117] distinguent trois niveaux dans le contexte de la programmation :

1. Les **soumissions** (*submissions*) : le code est enregistré au moment de sa remise finale ou de sa validation par un système de tests. C'est le niveau le plus grossier.
2. Les **instantanés** (*snapshots*) : le code est capturé à chaque événement significatif (sauvegarde, compilation, exécution de tests). Ce niveau intermédiaire permet d'observer le processus de construction du code.

3. Les **frappes** (*keystrokes*) : chaque modification du code source, même d'un seul caractère, est enregistrée. Ce niveau le plus fin permet d'étudier les micro-comportements d'édition.

Plus la granularité est fine, plus les données sont riches, mais plus les volumes sont importants et les défis d'interprétation croissent. Dans notre contexte, les *commits* manuels de Git4School se situent en deçà du niveau des soumissions (un *commit* par question résolue), ce qui justifie la recherche d'un niveau intermédiaire plus informatif.

6.1.3 Cadre de référence pour l'instrumentation

HUNDHAUSEN, OLIVARES et CARTER [50] proposent un modèle de processus en quatre phases pour les LA fondées sur l'instrumentation d'environnements de développement : (1) collecte des données, (2) analyse des données, (3) conception d'interventions, et (4) mise en œuvre des interventions. Leur revue critique des systèmes existants met en évidence deux lacunes récurrentes : d'une part, le lien entre les événements techniques capturés (compilations, exécutions, sauvegardes) et les dimensions pédagogiques significatives (progression dans une tâche, stratégie de résolution) reste rarement explicité. D'autre part, il n'existe pas de format de données standardisé partagé entre les différentes plateformes.

Ces constats structurent les enjeux que nous allons explorer dans les sections suivantes : comment représenter les traces (section 6.2), comment les générer de manière fine et contextualisée (section 6.3), et comment en évaluer la qualité (section 6.4).

6.2 REPRÉSENTATION ET GESTION DES TRACES D'APPRENTISSAGE

Dans le domaine des EIAH et des LA, la standardisation de la représentation des traces est un enjeu majeur pour permettre l'interopérabilité entre les systèmes. Parmi les approches les plus étudiées concernant les formats et les systèmes de gestion de traces d'apprentissage figurent, d'une part, les spécifications d'interopérabilité de type xAPI, et d'autre part, les Systèmes à Base de Traces (SBT) – dont le Kernel for Trace-Based Systems (KTBS) [20] – qui se concentrent sur la modélisation et la transformation de traces. Ces deux familles de solutions peuvent être combinées, par exemple en important des traces xAPI dans un SBT pour les restructurer et les analyser.

6.2.1 xAPI et les Learning Record Stores

La spécification *xAPI* (*Experience API*) est aujourd'hui l'une des principales normes d'interopérabilité des traces d'apprentissage. Elle

repose sur la génération d'assertions (*statements*) décrivant une expérience d'apprentissage selon une syntaxe atomique universelle : **Acteur + Verbe + Objet + [Contexte/Résultat]**. Ces traces sont envoyées et stockées dans un *Learning Record Store* (LRS), qui sert d'entrepôt centralisé recevant des données d'apprentissage hétérogènes depuis divers systèmes. Le LRS expose ensuite ces enregistrements via une API pour des requêtes ou des exports. Bien que la spécification permette techniquement d'inclure des données riches via son objet *result* (extensions JSON arbitraires, pièces jointes binaires), en pratique le format est majoritairement utilisé pour consigner des événements discrets (visionner une vidéo, répondre à un quiz), et les productions complexes de l'étudiant·e (comme du code source) sont généralement stockées dans un système annexe et simplement référencées par la trace.

À côté de xAPI, la spécification *IMS Caliper Analytics* constitue l'autre standard majeur d'interopérabilité des traces d'apprentissage. Caliper repose sur une structure similaire (Acteur + Action + Activité) mais utilise JSON-LD et définit des profils métriques prédéfinis pour différents types d'activités (évaluation, session, média, etc.), offrant un cadre plus structuré au prix d'une moindre flexibilité. Ces deux spécifications sont complémentaires et visent principalement l'interopérabilité à l'échelle institutionnelle.

6.2.2 Les Systèmes à Base de Traces (SBT) et kTBS

Issues principalement de la communauté de recherche EIAH francophone, les approches par Systèmes à Base de Traces (SBT) proposent une vision différente, centrée non pas sur l'interopérabilité globale mais sur la modélisation fine, le stockage et la transformation de traces au sein d'un environnement donné. Le cadre théorique des traces modélisées, formalisé par CHAMPIN, MILLE et PRIÉ [20] et développé dans les travaux de SETTOUTI [102], repose sur l'idée de traiter les traces numériques comme des objets informatiques de premier niveau, dotés d'un modèle explicite. Dans cette terminologie, l'unité élémentaire est l'*obsel* (élément observé), défini par un type, des horodatages, un sujet, des attributs et des relations avec d'autres obsels. Une trace est un ensemble organisé d'obsels conforme à un modèle de trace (*trace model*). Le noyau logiciel kTBS (*Kernel for Trace-Based Systems*) est l'implémentation de référence de ce cadre. L'avantage des SBT réside dans leurs mécanismes avancés pour calculer des « traces dérivées », c'est-à-dire abstraire des événements de bas niveau pour produire des indicateurs de plus haut niveau porteurs de sens pédagogique.

6.2.3 *ProgSnap2 : un format standardisé pour les données de processus de programmation*

Plus récemment, la communauté de recherche en enseignement de l'informatique a proposé un format spécifiquement conçu pour les données de processus de programmation. PRICE et al. [89] introduisent *ProgSnap2*, un format tabulaire flexible qui structure les événements d'activité de programmation (éditions de code, compilations, exécutions, soumissions) dans un tableau d'événements principal, chaque événement étant relié à un instantané de l'état du code source via un identifiant. Le format définit des colonnes obligatoires (type d'événement, identifiant, horodatage, sujet, état du code) et des colonnes optionnelles permettant d'intégrer des métadonnées sur le cours, l'exercice ou l'outil. Développé dans le cadre des communautés CSEDM (CS Education Data Mining) et SPLICE, *ProgSnap2* a permis de convertir avec succès des jeux de données issus de plusieurs institutions, démontrant sa capacité d'interopérabilité dans le domaine spécifique de la programmation. Toutefois, *ProgSnap2* reste un format de stockage à plat (journal d'événements chronologique) et ne modélise pas nativement la structure topologique des modifications ni le contexte pédagogique des tâches.

6.2.4 *Git comme générateur et système à base de traces d'apprentissage*

Notre approche s'appuie sur le système de gestion de versions Git et la plateforme GitHub pour la collecte des traces lors de séances de programmation. Bien que Git soit un outil industriel de génie logiciel, son architecture présente des analogies structurelles avec les systèmes à base de traces. Si l'on dresse un parallèle conceptuel, l'utilisation de Git s'apparente à la fois à un LRS décentralisé et à un SBT.

En effet, un *commit* Git peut être mis en correspondance avec la structure grammaticale de xAPI :

- **Acteur** : L'identité de l'apprenant·e est inscrite dans les métadonnées du commit (nom, e-mail de l'auteur ou l'autrice).
- **Verbe** : L'intention sémantique est portée par le message du commit. À la différence des verbes xAPI, qui s'appuient sur un vocabulaire contrôlé identifié par des IRI, le message de *commit* est en texte libre. Cependant, en imposant des conventions (ex. un message préfixé par *Fix* pour la résolution d'une question, ou des balises automatiques d'édition de code), le message peut jouer le rôle d'un verbe d'apprentissage explicite.
- **Objet** : L'objet est double. Au niveau macroscopique, c'est le dépôt (le projet) ; au niveau microscopique, ce sont les fichiers modifiés et suivis dans le *snapshot* du commit.
- **Contexte / Résultat** : Le système d'exploitation fournit un horodatage inaltérable, et le commit peut être enrichi d'autres

données contextuelles. Ce mapping reste toutefois plus limité que l'objet *context* de xAPI, qui permet de rattacher chaque trace à une activité parente, un instructeur ou un groupe.

Ce parallèle, bien qu'approximatif, illustre que les données Git portent nativement une partie substantielle de l'information nécessaire à la construction de traces d'apprentissage.

Cependant, utiliser Git présente des avantages décisifs par rapport à l'instrumentation classique d'un LRS avec xAPI dans le cadre spécifique de la programmation :

1. **La richesse de la "charge utile" (le diff)** : Contrairement à une trace classique qui indique simplement qu'un-e étudiant-e a agi sur une ressource, le commit Git contient l'intégralité de la production via le *diff*. Cette différence de code documente non seulement l'action, mais aussi la nature exacte de la transformation intellectuelle (les lignes de code ajoutées, modifiées, supprimées).
2. **Un contexte topologique (le DAG)** : Les traces stockées dans un LRS ou dans un format tabulaire comme ProgSnap2 forment généralement un journal plat et chronologique. Git structure les données sous la forme d'un graphe orienté acyclique (DAG). Chaque commit connaissant son parent, la trace peut modéliser la nature non-linéaire de la résolution de problèmes : les essais, les erreurs, les retours en arrière ou l'exploration de pistes parallèles. Notons cependant que les étudiant-es novices utilisent rarement les branches Git de manière spontanée, de sorte que cet avantage topologique se concrétise surtout lorsque l'outil de génération de traces exploite lui-même les branches pour organiser les données (ce que nous ferons avec LAWG).
3. **Une validité écologique maximale** : Git est l'environnement naturel de travail en développement logiciel. La trace est le sous-produit naturel de l'activité, générée localement puis synchronisée vers un serveur (GitHub), sans nécessiter de connexion constante, offrant une robustesse que de nombreux traceurs xAPI n'ont pas.

Dans ce contexte, nous utilisons Git et GitHub comme une base de traces premières, et notre plateforme Git4School agit comme un moteur de transformation, s'apparentant aux mécanismes d'un SBT, pour modéliser ces données brutes en indicateurs visuels interprétables par l'enseignant-e.

Toutefois, comme souligné à la fin de l'itération 1, s'en remettre uniquement à la création manuelle de *commits* par les étudiant-es limite la granularité et la fiabilité de ces traces. C'est pourquoi il est nécessaire d'examiner les méthodes de génération automatique.

6.3 GÉNÉRATION DE TRACES POUR LE SUIVI FIN DE L'ACTIVITÉ DES ÉTUDIANT·ES DURANT LES SÉANCES DE TP

Afin de motiver la conception de notre système LAWG, présenté dans les chapitres 7 et suivants, il est nécessaire d'examiner au préalable les méthodes existantes permettant de générer des traces plus fines sur l'activité de programmation des étudiant·es durant un TP. Cette section dresse un état de l'art de ces approches, organisées en deux familles : l'exploitation des journaux Git d'une part, et l'instrumentation d'environnements de développement d'autre part. Pour chaque famille, nous soulignons les apports et les limites en termes de granularité, de fiabilité et d'intégration du contexte pédagogique.

6.3.1 Approches fondées sur les journaux Git

Pour tenter de reconstituer une chronologie d'activité plus fine à partir de traces ponctuelles comme les *commits*, une approche courante dans la littérature consiste à définir des sessions de travail basées sur des seuils temporels d'inactivité [42, 77]. L'hypothèse sous-jacente est qu'une série d'actions rapprochées constitue une période d'activité continue. Toutefois, cette approximation est risquée, particulièrement dans un contexte de travail à distance ou asynchrone où les étudiant·es peuvent alterner fréquemment entre le travail sur le projet et d'autres activités personnelles, rendant l'estimation des temps de travail effectif peu fiable.

En 2021, MACAK et al. [73] proposent une méthode pour transformer systématiquement les journaux Git de projets d'étudiant·es en journaux d'événements exploitables par des outils de PM. Plutôt que de se contenter des métadonnées brutes de Git, ils dérivent pour chaque *commit* une série d'« activités » décrivant sa nature : taille du *commit* (petit/gros), orientation des changements (principalement des ajouts ou des suppressions), opérations liées à la gestion de branches (création de branche, *merge*, *pull request*) ou encore *commits* dédiés à la création ou la modification de fichiers de tests. Cette étape de transformation explicite permet d'obtenir un journal d'événements aligné sur des catégories pédagogiquement interprétables, qui peut ensuite être analysé par les techniques de PM pour identifier des motifs de comportement d'apprentissage dans les projets d'étudiant·es. Toutefois, dans ce travail, les traces restent limitées aux informations disponibles dans les *commits* eux-mêmes et ne sont pas enrichies par des informations contextuelles sur la progression des étudiant·es dans des activités pédagogiques plus fines (par exemple la résolution d'une question).

Une étude de 2022 [76] explore un moyen de générer automatiquement et de visualiser des traces d'activité des étudiant·es au fil des séances de TP. La plateforme EnCourse produit d'une part des indica-

teurs bruts directement dérivés des *commits* (temps passé sur le projet, fréquence des *commits*, nombre de lignes ajoutées ou supprimées) et, d'autre part, des indicateurs interprétés tels que des alertes de « malhonnêteté académique » potentielle ou des signaux lorsque les étudiant-es sont en retard. Pour automatiser la production de ces traces, MERKLE et al. intègrent les commandes de *commit* dans un fichier Makefile, de sorte qu'un *commit* soit déclenché à chaque compilation, ce qui garantit une granularité temporelle plus fine que dans un usage manuel de Git mais continue de caler les événements sur des actions techniques plutôt que sur la structure pédagogique des tâches.

Ces approches fondées sur Git partagent une limite commune : les traces sont déterminées par la fréquence et le contenu des *commits*, qui dépendent eux-mêmes des pratiques de l'étudiant-e ou de déclencheurs techniques (compilation), sans lien explicite avec le scénario pédagogique.

6.3.2 Instrumentation d'environnements de développement

Dans la même perspective de suivi fin de l'activité pendant les séances de programmation, plusieurs travaux se sont appuyés sur l'instrumentation d'environnements de développement intégrés. Le projet Blackbox [18] associé à l'IDE BlueJ collecte à grande échelle (plusieurs millions d'événements issus de plus de 100 000 utilisateurs et utilisatrices) les interactions des étudiant-es avec l'IDE (compilations, exécutions, modifications de code, utilisation de fonctionnalités) et les stocke sur un serveur centralisé pour analyse ultérieure. Les extensions de BlueJ utilisées par JADUD [52] ou FENWICK JR et al. [33], par exemple, tracent systématiquement les exécutions du compilateur et les messages d'erreur associés, ce qui permet d'étudier les erreurs de novices et les trajectoires de débogage à un niveau de granularité intermédiaire entre le *commit* et la simple soumission finale. Ces approches produisent donc des journaux d'événements détaillés mais sont généralement centrées sur un IDE spécifique et ne se connectent que peu aux scénarios pédagogiques ou aux séances de TP telles qu'elles sont orchestrées par les enseignant-es.

Plus récemment, LYULINA et al. [72] proposent TaskTracker-tool, un ensemble d'outils d'instrumentation des IDE basés sur IntelliJ (PyCharm, IntelliJ IDEA, CLion) pour collecter des traces très fines de l'activité de programmation pendant la résolution de tâches. Le plugin TaskTracker suit tous les changements de code dans un fichier de solution dédié et enregistre chaque modification, même d'un seul caractère, sous forme d'instantanés successifs, tandis qu'un greffon complémentaire trace les actions réalisées dans l'IDE (copier-coller, exécutions, débogage, etc.), l'ensemble étant regroupé par tâche choisie dans une interface configurable à distance et envoyé à un serveur pour post-traitement (fusion code/événements, notation automatique

des instantanés via des tests, ajustement de la granularité). Dans la continuité de cette lignée, des outils plus récents comme KOALA [61] étendent l'idée en offrant une configuration plus flexible de ce qui est tracé dans les IDE JetBrains (instantanés de code, exécutions, usages de raccourcis, interactions avec des systèmes d'aide personnalisée) et en produisant des jeux de données de grande taille qui permettent de suivre l'évolution des pratiques d'édition et de débogage au fil des tâches.

6.3.3 Synthèse et positionnement

Ces travaux montrent qu'il est possible de générer automatiquement des traces d'apprentissage à partir d'outils de développement (Git, IDE) en s'appuyant soit sur les *commits* comme événements, soit sur des instantanés de code et des interactions dans l'IDE, parfois à très grande échelle. Le tableau 6.1 résume les caractéristiques des principaux systèmes présentés.

TABLE 6.1 – Synthèse des approches de génération de traces pour le suivi de l'activité de programmation.

Système	Source	Granularité	Indép. IDE	Contexte pédag.
Seuils temporels [42, 77]	Git	Commit	Oui	Non
MACAK et al. [73]	Git	Commit enrichi	Oui	Non
EnCourse [76]	Git + Make-file	Compilation	Oui	Non
Blackbox [18]	BlueJ	Interaction IDE	Non	Non
JADUD [52]	BlueJ	Compilation	Non	Non
TaskTracker [72]	JetBrains	Frappe clavier	Non	Partiel (tâche)
KOALA [61]	JetBrains	Configurable	Non	Partiel (tâche)
LAWG (notre prop.)	Git + FS	Modif. fichier	Oui	Oui

Cependant, dans tous les cas, les traces restent majoritairement calibrées sur l'outil (moment du *commit*, compilation, sauvegarde dans l'IDE, exécution) et n'intègrent que partiellement, voire pas du tout, les éléments du contexte pédagogique tels que la structure précise des activités, les intentions de l'enseignant-e ou les interventions effectuées pendant les TP. Par ailleurs, les approches par instrumentation d'IDE, bien qu'elles offrent la granularité la plus fine, créent une dépendance forte envers un environnement de développement spécifique, ce qui pose un problème de généralité dans des contextes où les étudiant-es sont libres de choisir leur outil.

À notre connaissance, aucun système existant ne combine les trois propriétés suivantes : (1) une génération de traces intégrant le contexte pédagogique (structure des exercices, progression dans les questions), (2) une indépendance vis-à-vis de l'IDE utilisé, et (3) une amélioration de la qualité des données par rapport aux *commits* manuels.

6.4 QUALITÉ DES DONNÉES : CADRES THÉORIQUES

Les limites identifiées à l'issue de l'itération précédente (manque de fiabilité, de précision et de généralité des traces) pointent vers un enjeu commun : la qualité des données collectées. Ces termes, bien qu'intuitifs, restent imprécis et ne s'appuient pas sur un cadre théorique établi. Afin de formuler des objectifs d'amélioration évaluables, il est nécessaire de recourir à des travaux qui formalisent la notion de qualité des données.

Le cadre théorique le plus cité dans ce domaine est celui proposé par WANG et STRONG [119]. Il repose sur l'idée que la qualité des données n'est pas une propriété absolue mais doit être évaluée selon plusieurs dimensions, en fonction de l'usage qui en est fait. Les auteurs distinguent quatre grandes catégories : la qualité intrinsèque (exactitude, crédibilité), la qualité contextuelle (pertinence, actualité, complétude), la qualité de représentation (compréhensibilité, format) et la qualité d'accessibilité (facilité d'accès, sécurité).

Plus récemment, l'Organisation Internationale de Normalisation (ISO) a développé la norme *ISO/IEC 25012 :2008*¹, qui propose un modèle de qualité des données structuré en dimensions similaires. Cette norme distingue les caractéristiques inhérentes aux données (exactitude, complétude, cohérence, actualité, crédibilité) et les caractéristiques dépendantes du système (disponibilité, portabilité, recouvrabilité).

Ces deux cadres convergent sur un ensemble de dimensions fondamentales que nous retenons pour évaluer l'apport d'un système de génération automatique de traces :

- **Exactitude (*accuracy*)** : le degré de conformité des données à la réalité qu'elles décrivent. Cette dimension recouvre ce que nous avons qualifié informellement de *fiabilité* des traces à l'issue de l'itération précédente.
- **Complétude (*completeness*)** : la mesure dans laquelle toutes les informations nécessaires sont présentes. Elle correspond à ce que nous avons désigné par la *richesse* des traces.
- **Cohérence (*consistency*)** : l'absence de contradictions internes et le respect des règles du processus sous-jacent.
- **Actualité (*timeliness*)** : le degré auquel les données sont enregistrées au moment opportun et sont suffisamment à jour pour être utiles.

1. <https://www.iso.org/fr/standard/35736.html>

Dans la suite de ce manuscrit, nous utiliserons systématiquement ces quatre dimensions pour caractériser la qualité des traces et évaluer les améliorations apportées par notre système.

Comme nous l'avons vu à la fin de l'itération 1, l'étude quantitative utilisant le PM a révélé des limites importantes dans la qualité des traces générées manuellement par les étudiant-es via leurs *commits*. Ces limites ont une cause commune : la dépendance envers une génération manuelle des traces, via des *commits* dont la granularité et la fréquence sont laissées à la discrétion des étudiant-es. Le cadre de qualité des données présenté ci-dessus permet de reformuler ces limites en termes précis : les traces manuelles souffrent d'un défaut d'*exactitude* (les messages de *commit* ne reflètent pas fidèlement l'activité réelle), de *complétude* (un seul *commit* par question ne capture pas le processus de résolution), d'*actualité* (les *commits* sont réalisés tardivement par rapport à l'activité) et de *cohérence* (le respect du protocole de *commit* est laissé à la discrétion de l'étudiant-e).

C'est ce constat qui nous amène à formuler la problématique suivante : **Comment améliorer la génération de traces pour raffiner le suivi des étudiant-es en rendant visibles leurs comportements d'apprentissage au cours des résolutions d'exercices de TP de programmation ?**

Pour répondre à cette problématique, nous devons lever les verrous identifiés précédemment en opérant deux changements majeurs.

Premièrement, pour améliorer l'exactitude, la complétude et l'actualité des traces, il est nécessaire de changer de paradigme en passant d'une logique de capture de résultats (les *commits* de résolution de question) à une logique de capture de processus (la résolution d'une question en cours). Cela implique d'identifier des processus beaucoup plus fins au sein d'une résolution d'exercice, ce qui techniquement requiert d'aller au-delà de l'analyse des seuls *commits* pour mettre en place une écoute des événements de modification de fichiers. Par ailleurs, pour répondre au manque de généricité, cette capture doit s'opérer de manière agnostique vis-à-vis de l'outil de développement utilisé par l'étudiant-e. Nous privilégions donc une approche par un script observateur externe plutôt qu'un plugin spécifique à un IDE particulier. Ces choix de conception nous conduisent à la première sous-question de recherche :

Question de recherche 2.1

Comment améliorer l'exactitude, la complétude, la cohérence et l'actualité des traces générées, sur la base des événements de modifications de fichiers, de sorte à pouvoir identifier des comportements d'apprentissage complexes ?

Deuxièmement, l'introduction d'un tel outil d'observation continue dans l'environnement de travail de l'étudiant-e soulève de nouveaux

défis. Contrairement à des *commits* manuels qui sont contrôlés par l'étudiant-e, une génération automatique et fréquente de traces risque de créer du "bruit" dans l'espace de travail, voire de perturber les habitudes de développement. Nous posons donc la seconde sous-question :

Question de recherche 2.2

Comment garantir que la génération de traces améliorée ne perturbe ni les activités de l'étudiant-e ni celles de l'enseignant-e ?

Il est donc impératif de garantir une séparation suffisante entre les traces d'apprentissage générées pour les besoins de l'analyse, et celles directement liées au travail de l'étudiant-e. Cette séparation est une condition nécessaire pour que le dispositif reste transparent, acceptable pour les utilisateurs, et qu'il ne biaise pas l'activité d'apprentissage observable.

Cette problématique de non-perturbation et de préservation de l'intégrité du travail étudiant impose une rigueur scientifique particulière. Il ne s'agit pas seulement de concevoir un outil fonctionnel, mais de valider qu'il est sans risque pour le processus d'apprentissage. C'est pourquoi nous avons mis en place un protocole d'évaluation structuré en trois étapes (Vérification, Validation, Expérimentation). Cette démarche, détaillée au chapitre 8, est nécessaire pour s'assurer que le système répond à ce critère fondamental de robustesse et de transparence avant d'être déployé en situation réelle d'enseignement pour répondre à nos questions de recherche.

Pour ce faire, nous avons cherché un compromis entre G4S et En-Course. Nous avons donc conçu un système de génération automatique de traces qui prend en compte le besoin d'avoir des informations contextuelles sur la progression des étudiants en temps réel, tout en maximisant la fiabilité et la généricité. Ces travaux sont présentés dans les chapitres suivants.

CONCEPTION DE LAWG, SYSTÈME INTERACTIF POUR LA GÉNÉRATION AUTOMATIQUE DE TRACES GIT D'ACTIVITÉ

7.1	Configuration	94
7.2	Encadrement des sessions de travail	95
7.3	Génération automatique des traces d'activité	96
7.4	Interface de résolution d'un exercice	97
7.5	Historisation des <i>commits</i> automatiques sur une branche dédiée	98

La phase d'analyse ayant permis de formuler les questions de recherche, ce chapitre aborde la *phase de conception et de développement* de la seconde itération du cycle DBR.

Pour répondre aux questions de recherche posées en fin de chapitre précédent, nous avons conçu le système interactif LAWG (pour *Learning Analytics With Git*) écrit en Python, open-source¹ et multiplateforme (Windows, MacOS et Linux). En agissant comme une interface entre l'étudiant·e et son dépôt Git, LAWG génère des traces enrichies et fiables tout en abstrayant Git pour les étudiant·es.

Le système interactif LAWG présuppose un contexte d'enseignement assez générique, fondé sur la résolution de feuilles d'exercices proposées lors de séances de TP. Ces exercices peuvent être constitués de plusieurs *questions*. Nous définissons la notion de *session de travail* comme étant une période d'activité d'un·e étudiant·e, durant laquelle elle ou il manipule des ressources (fichiers de programmation, textes, images, . . .), qui constituent son *espace de travail*, pour résoudre des questions. Un·e étudiant·e peut travailler en dehors des séances de TP encadrées, une session de travail ne se limite donc pas aux seules séances de TP.

L'espace de travail d'un·e étudiant·e existe localement sur sa machine, et à distance. L'espace à distance permet à l'enseignant·e d'accéder au travail de l'étudiant·e. L'étudiant·e peut ensuite synchroniser son espace local si elle ou il doit travailler sur plusieurs machines ou avec d'autres étudiant·es. Lorsqu'un·e étudiant·e a fini de travailler sur une question, elle ou il valide ses changements, puis passe à la question suivante. Dans notre cas, l'espace de travail est géré avec le logiciel de gestion de versions Git. Pour valider ses changements, elle ou il effectue un *commit*, ce qui enregistre localement toutes les modifications faites aux fichiers depuis son dernier *commit*. L'enregistrement

1. Disponible sur Github à cette adresse : <https://github.com/git4school/LAWG>

est toujours associé à un message entré par l'étudiant-e au moment du *commit* (par exemple, Exercice 1 résolu). Pour synchroniser l'espace de travail distant (le dépôt distant) avec son espace local (le dépôt local), elle ou il exécute la commande Git *push*, et pour synchroniser son espace local avec celui à distance, elle ou il exécute la commande Git *pull*. Enfin, l'enchaînement des *commits* sur l'espace de travail constitue une branche. Il existe une branche sur laquelle l'étudiant-e évolue par défaut, mais elle ou il peut faire le choix d'en créer une nouvelle à partir de celle-ci. Les branches évoluent de façon distincte, de sorte que l'étudiant-e peut changer de branche pour isoler certains types de modifications (travail sur une fonctionnalité, correction de bug, amélioration du code, etc.).

7.1 CONFIGURATION

Avant toute utilisation de LAWG, une configuration minimale est requise, à la fois du côté de l'enseignant-e lors de la conception du sujet de TP, et du côté des étudiant-es lors de son exécution.

Pour préparer le sujet, l'enseignant-e met en place un espace de travail servant de modèle, composé d'un ensemble requis d'éléments permettant aux étudiant-es de travailler. Dans cet espace de travail, l'enseignant-e doit intégrer et configurer LAWG, en y ajoutant les fichiers exécutables correspondant aux systèmes d'exploitation susceptibles d'être utilisés par les étudiant-es. L'enseignant-e doit créer le fichier de configuration `settings.yml`, dont les champs requis sont listés dans l'extrait 7.1.

Listing 7.1 – Structure du fichier `settings.yml`

```

1  repo_path: .
2  ssh_path: /home/user/.ssh/id_rsa
3  questions: []
4  groups: []

```

Nous souhaitons que l'utilisation du système soit aussi simple et fiable que possible. Pour cela, il est nécessaire de fournir la liste des questions auxquelles l'étudiant-e devra répondre, à travers le champ `questions`. C'est également à cette étape que les champs `groups` et `repo_path` peuvent être renseignés. Une fois cela fait, l'enseignant-e peut publier le sujet et le dépôt modèle.

C'est au tour de l'étudiant-e de créer son espace de travail à partir du dépôt modèle et de terminer la configuration. Il ne reste que le champ `ssh_path` dans lequel il faut indiquer le chemin vers la clé SSH enregistrée dans la plateforme qui héberge le dépôt Git distant.

Une des principales problématiques avec la conception d'un tableau de bord, comme Git4School, est l'identification des étudiant-es. Pour ce faire, dans la première itération, il était demandé aux étudiant-es

de remplir le fichier `README.md` de leur projet avec leur nom et leur groupe. Le `README.md` pouvait contenir d'autres informations comme des instructions, et le tableau de bord devait donc extraire l'identité à l'aide d'un *parser*. De fait, le succès de l'identification de l'étudiant-e était fortement dépendant de celle-ci ou celui-ci, qui devait (1) ne pas oublier de remplir le `README.md`, (2) remplir le `README.md` correctement, et (3) ne pas modifier le `README.md` ultérieurement de sorte que le *parsing* ne peut plus se faire correctement. Afin de fiabiliser ce processus, au premier démarrage, LAWG demande à l'étudiant-e d'entrer son nom et son groupe et génère le fichier `IDENTITY.json` (pour un exemple, voir l'extrait 7.1). Ce fichier sera ensuite lu par le tableau de bord Git4School pour identifier chaque étudiant-e.

Listing 7.2 – Structure du fichier `IDENTITY.json`

```

1 {
2   "first_name": "John",
3   "last_name": "Doe",
4   "group": "TPA12"
5 }
```

Notre système interactif offre trois grandes fonctionnalités, explicitées dans les sous-sections suivantes :

- encadrer les sessions de travail, en sauvegardant toutes les ressources à la fin de chaque session ;
- observer toutes les modifications apportées aux fichiers de l'espace de travail pour les sauvegarder en temps réel ;
- offrir une interface en ligne de commande aux étudiant-es pour leur permettre de marquer une question comme résolue.

L'observation des modifications en temps réel permet de capturer le processus d'élaboration de la réponse, produisant ainsi des traces plus riches et nombreuses pour identifier des comportements complexes (QR2.1). En parallèle, l'encadrement des sessions et l'interface dédiée visent à fiabiliser la collecte des données (QR2.1).

7.2 ENCADREMENT DES SESSIONS DE TRAVAIL

Le début et la fin d'une session de travail d'un-e étudiant-e sont jalonnés par des *commits* nommés "Resume" et "Pause" dans la branche courante. Comme illustré dans la figure 7.1, nous avons fait le choix d'utiliser une autre branche, `auto`, pour isoler l'historique des *commits* automatiques déclenchés par LAWG. Cette branche est créée à partir de la branche courante au lancement de LAWG si elle n'existe pas. Cela permet à l'étudiant-e d'avoir une branche courante simple avec peu de *commits*, et à l'enseignant-e d'avoir une branche distincte qui retrace l'activité complète de l'étudiant-e. Certains *commits* sont présents dans les deux branches.

L'implantation de ce mécanisme sera détaillé plus après dans la section 7.5.

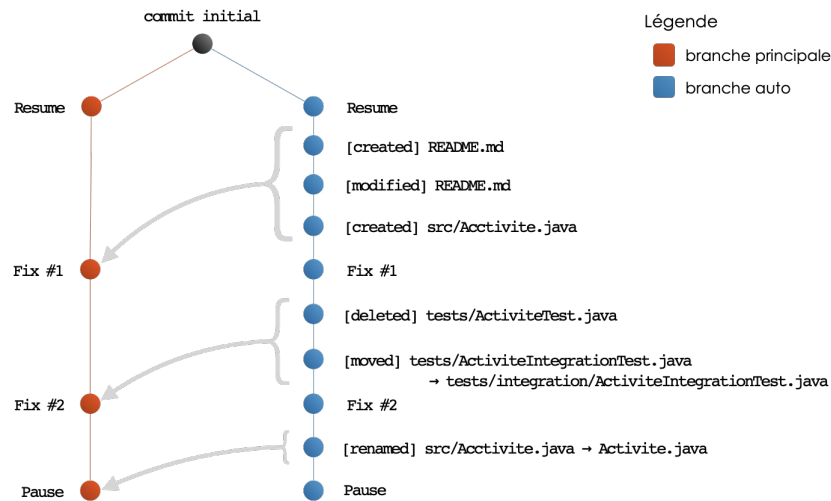


FIGURE 7.1 – Exemple de l'utilisation du système interactif LAWG lors de la réalisation d'exercices de travaux pratiques

Lors de la fin d'une session de travail, marquée par l'utilisation de la commande `exit` de LAWG par l'étudiant-e, tous les changements en cours sont sauvegardés sous un commit "Pause" et envoyés dans l'espace de travail distant. Ensuite, tous les fichiers de l'espace de travail local de l'étudiant-e sont supprimés. De cette façon, aux yeux de l'étudiant-e, son espace de travail est vide, il ne reste plus que l'exécutable du système et le fichier de configuration. Lorsqu'elle ou il reprend son travail, l'étudiant-e doit lancer le système interactif LAWG pour restaurer son espace de travail. Le système peut ainsi tracer automatiquement le démarrage d'une nouvelle session de travail, représenté par un commit Resume dans la branche auto.

7.3 GÉNÉRATION AUTOMATIQUE DES TRACES D'ACTIVITÉ

Dès le lancement de LAWG, les modifications des fichiers de l'espace de travail sont observées et donnent lieu à des événements système. Chaque événement entraîne la production d'un *commit* dans la branche auto. Les événements observés sont : la *modification*, la *création*, la *suppression*, le *déplacement* et le *renommage* d'un fichier.

Pour ces *commits* automatiques, nous avons défini une forme de message minimaliste pour faciliter le traitement par la suite :

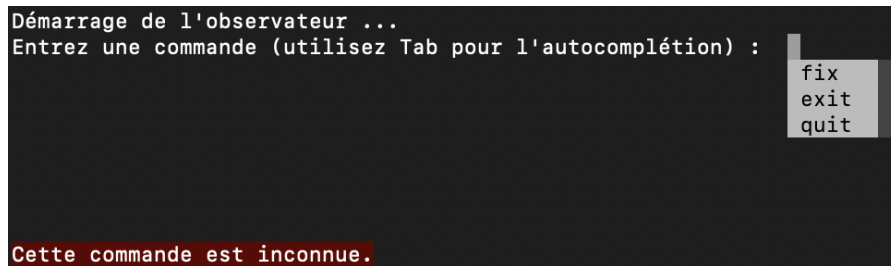
- "[moved] <chemin_fichier> -> <chemin_nouveau_dossier>"
- "[renamed] <chemin_fichier> -> <nouveau_nom_fichier>"
- "[<evenement>] <chemin_fichier>"

Ainsi, dans l'exemple de la figure 7.1, on peut voir que l'étudiant-e démarre sa session de travail avec la création, puis la modification d'un fichier README.md, et crée un fichier Acctivite.java avant de marquer la question #1 comme résolue. Elle ou il fait ensuite des opérations sur des fichiers de tests, résout la question #2, puis se rend

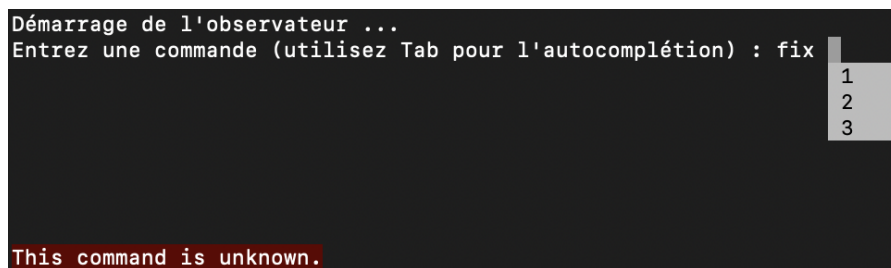
compte qu'elle ou il a fait une faute sur le nom d'un fichier et corrige l'erreur. L'étudiant-e ferme ensuite sa session, ce qui a pour effet de produire le commit "Pause".

7.4 INTERFACE DE RÉOLUTION D'UN EXERCICE

Notre système interactif LAWG propose également une interface en ligne de commande, qui met notamment à disposition la commande `fix`. Elle permet à l'étudiant-e de marquer une question comme résolue, de sauvegarder les changements associés et de les envoyer dans l'espace de travail distant. Plus précisément, tous les changements depuis le dernier appel à cette même commande, ou depuis le lancement d'une session de travail, sont validés sous forme d'un *commit* dans la branche courante et dans la branche auto (dont le message est de la forme "Fix <question>") puis synchronisés dans l'espace de travail distant. Dans l'exemple donné en figure 7.1, l'étudiant-e a utilisé la commande `fix` pour les questions #1 et #2, dont les *commits* associés sont visibles dans les deux branches, puis la commande `exit`, qui a produit le *commit* "Pause".



(a) Suggestion des commandes



(b) Suggestion des questions

FIGURE 7.2 – Captures d'écran des suggestions de LAWG

Afin de répondre à la question de la fiabilité des traces, le système offre une suggestion des commandes disponibles (cf. figure 7.2a) en fonction des caractères déjà entrés et l'étudiant-e ne peut pas lancer une commande qui n'existe pas ou qui est mal formée. Lorsqu'il reconnaît la commande `fix`, le système interactif affiche la liste des questions auxquelles l'étudiant-e doit répondre (cf. figure 7.2b), renseignées auparavant lors de la configuration.

7.5 HISTORISATION DES *commits* AUTOMATIQUES SUR UNE BRANCHE DÉDIÉE

Comme exposé dans la section 5.5, nous voulons garantir une séparation claire entre les traces d'activité générées automatiquement par le système et la production propre de l'étudiant-e, conformément à la QR2.2.

Pour répondre à cette exigence, nous avons opté pour la création d'une branche dédiée, nommée *auto*, spécifiquement réservée à l'enregistrement des *commits* automatiques produits par le système. Cette organisation permet de dissocier explicitement les traces d'activité, utilisées à des fins d'analyse, du travail réalisé par l'étudiant-e sur la branche principale. Ainsi, l'utilisation de LAWG n'altère que très peu l'historique de la branche principale, qui demeure très semblable à un usage classique de Git sans l'outil, contribuant à préserver l'intégrité du dépôt de l'étudiant-e et à éviter toute confusion entre production et génération de traces.

Pour aborder la suite de cette section, il est nécessaire de rappeler la distinction que Git établit entre trois espaces : le répertoire de travail, qui contient les fichiers en cours de modification ; l'index, qui correspond à une zone de préparation des changements sous la forme d'un instantané ; et enfin le dépôt, où sont conservées les versions validées dans l'historique des *commits*. Git détermine ce qui doit être ajouté à l'historique (le dépôt) au prochain *commit* en comparant l'index avec le dernier *commit* effectué, identifié par le pointeur HEAD : seules les différences seront prises en compte. L'algorithme d'historisation que l'on propose exploite cette architecture pour enregistrer les *commits* automatiques sur la branche *auto* sans perturber le répertoire de travail de l'étudiant-e.

Le résultat attendu de l'algorithme ressemble au graphe de la figure 7.1. Plus précisément, l'étudiant-e travaille sur la branche courante, en général *main*. Le pointeur HEAD référence cette branche, de sorte que chaque *commit* manuel s'y rattache directement. Les modifications à tracer doivent être transférées sur la branche *auto* pour y être enregistrées par un *commit*. Puis, l'environnement de travail doit revenir à l'état initial, sans que le répertoire de travail ne soit modifié, c'est-à-dire que les modifications doivent être toujours présentes pour être sauvegardées plus tard par l'étudiant-e dans un *commit*.

Une première solution à laquelle nous avons pensée consistait à utiliser la commande Git checkout pour basculer temporairement sur la branche *auto*, effectuer le *commit*, puis revenir sur la branche principale. Cependant, cette approche s'est révélée problématique d'un point de vue utilisabilité. Bien que le contenu des fichiers reste identique entre les deux branches, l'exécution du checkout déclenche des alertes dans la plupart des environnements de développement (IDE), signalant un changement de fichiers ouverts. Cette interruption,

Voir le code source de
LAWG pour plus de
détails.

Listing 7.3 – Pseudo-code de la synchronisation des *commits* automatiques sur une branche dédiée

```

1 reset_branch("auto")
2 add_files_to_index(file_paths)
3 commit_changes(message, amend=False, allow_empty=True)
4 force_branch("auto")
5 reset_to('HEAD@{2}')

```

fréquente et intrusive, nuit à l'expérience utilisateur et peut perturber le déroulement du travail de l'étudiant-e.

Pour pallier cette difficulté, nous avons conçu un algorithme alternatif exploitant la commande Git `reset`, permettant de manipuler directement la branche pointée par le pointeur HEAD et l'index, sans jamais changer de branche ni modifier les fichiers ouverts dans l'éditeur (le répertoire de travail).

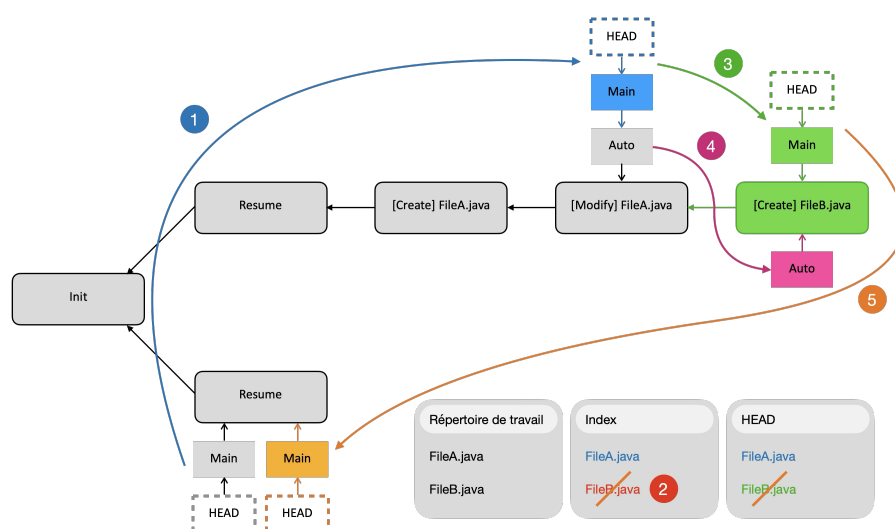


FIGURE 7.3 – Schéma illustrant le mécanisme d'historisation des *commits* automatiques sur une branche dédiée (auto) sans perturber la branche principale.

L'algorithme illustré en figure 7.3 procède en 5 étapes :

- ❶ Déplacer la branche main (et le pointeur HEAD associé) sur la branche auto et synchroniser l'index avec l'état de cette branche : le reset effectué sur la branche auto est dit "*mixed*", c'est-à-dire qu'il met à jour l'index mais pas le répertoire de travail. La modification du fichier est donc toujours présente pour être sauvegardée.
- ❷ Ajouter le fichier à enregistrer à l'index : la commande `add` ajoute la modification du fichier à l'index, préparant le prochain *commit*.
- ❸ Créer un nouveau *commit* à partir du *commit* pointé par HEAD, c'est-à-dire le dernier *commit* de la branche auto. La modification

du fichier est donc sauvegardée dans ce *commit* et le pointeur de la branche main est avancé dessus.

- 4 Mettre à jour le pointeur de la branche auto pour qu'il pointe vers ce nouveau *commit* : on force le pointeur de la branche auto à se déplacer sur ce nouveau *commit*, sans avoir changé notre branche courante (c'est-à-dire sans avoir effectué de checkout). À la fin de cette étape, les pointeurs des branches main et auto pointent sur le même dernier *commit*.

- 5 Annuler les modifications contextuelles faites sur la branche main : effectuer un second reset pour ramener HEAD, la branche main et l'index à leur état antérieur (les deux dernières modifications du pointeur HEAD), annulant ainsi toute modification contextuelle effectuée depuis le début de l'algorithme.

Cette séquence d'opérations présente l'avantage majeur de ne provoquer aucune modification visible dans le répertoire de travail comme on peut le vérifier dans le schéma explicatif en figure 7.3 : aucun fichier n'est modifié, déplacé ou supprimé, et l'étudiant-e ne subit aucune interruption dans son répertoire de travail. Ce mécanisme garantit ainsi une historisation fiable et transparente des traces d'activité, tout en préservant l'expérience utilisateur et l'intégrité du travail de l'étudiant-e.

ÉVALUATION DE LAWG EN TROIS ÉTAPES

8.1	Évaluation en situation simulée (Vérification)	102
8.2	Évaluation en situation réelle (Validation)	103
8.3	Expérimentation en contexte écologique	104
8.3.1	Méthode	104
8.3.2	Participants	105
8.3.3	Protocole	106
8.3.4	Préparation de l'expérimentation	106
8.3.5	Déroulement de l'expérimentation	107
8.3.6	Matériel	107
8.3.7	Méthodes d'analyse	108

La phase de conception achevée, ce chapitre et le suivant abordent la *phase d'évaluation* de la seconde itération du cycle DBR. Cette phase vise à évaluer le prototype LAWG selon un protocole structuré en trois étapes progressives.

La vérification et la validation sont deux étapes essentielles dans le développement d'un système logiciel. La vérification¹ vise à s'assurer que le système répond aux spécifications techniques définies, tandis que la validation² permet de confirmer que le système répond aux besoins des utilisateur·rices dans leur contexte d'utilisation réel.

Dans le cas de LAWG, ces deux phases prennent une importance particulière. En effet, l'outil est destiné à être utilisé par des étudiant·es dans un cadre pédagogique où la fiabilité et la robustesse sont essentielles. Or une défaillance du système pourrait non seulement perturber le déroulement des séances de TP, mais aussi impacter la progression et la motivation des apprenant·es. De plus, la diversité des environnements d'utilisation, qu'il s'agisse des systèmes d'exploitation, des niveaux de maîtrise de Git ou des contextes pédagogiques, exige une attention particulière lors des phases de vérification et de validation.

Concrètement, vérifier LAWG implique de contrôler, en conditions simulées, que les mécanismes de génération de traces fonctionnent conformément aux spécifications : génération automatique des *commits* lors des modifications de fichiers, bon fonctionnement des commandes de marquage de questions, gestion correcte des sessions de travail et synchronisation des branches Git. Valider LAWG implique ensuite de déployer l'outil auprès d'étudiant·es en conditions réelles pour

1. [https://sebokwiki.org/wiki/Verification_\(glossary\)](https://sebokwiki.org/wiki/Verification_(glossary))

2. [https://sebokwiki.org/wiki/Validation_\(glossary\)](https://sebokwiki.org/wiki/Validation_(glossary))

éprouver sa robustesse face à des usages variés et à des situations imprévues, et s'assurer que le dispositif est utilisable sans perturber le déroulement des séances.

Enfin, une fois ces deux conditions préalables satisfaites, une troisième étape d'expérimentation en contexte écologique permet d'évaluer la capacité des traces générées par LAWG à répondre aux questions de recherche, en particulier l'identification de comportements d'apprentissage complexes (QR2.1) et la non-perturbation de l'activité des étudiant·es et des enseignant·es (QR2.2). Ce protocole en trois étapes vise ainsi à garantir progressivement la fiabilité technique de l'outil avant d'en évaluer l'apport scientifique.

8.1 ÉVALUATION EN SITUATION SIMULÉE (VÉRIFICATION)

Après la conception et le développement de LAWG, nous avons procédé à une phase de vérification du logiciel en conditions simulées de TP. Cette phase avait pour objectif de s'assurer du bon fonctionnement de l'outil dans des conditions proches de son utilisation nominale par les étudiant·es, et de détecter d'éventuels problèmes avant son déploiement à plus grande échelle.

Pour cela, nous avons nous-mêmes utilisé LAWG en suivant le sujet d'un TP de développement logiciel de niveau Master, dans lequel il était demandé de marquer chaque question comme résolue à l'aide d'un *commit*. Nous avons traité une partie du sujet en incluant volontairement une pause dans l'utilisation de LAWG pour simuler une transition entre deux séances de TP. Cette approche visait à tester l'utilisation standard de l'outil plutôt que des cas limites ou extrêmes. Nous avons notamment vérifié :

- la génération automatique des traces lors des modifications de fichiers ;
- le bon fonctionnement de la commande `fix` pour marquer les questions comme résolues ;
- la gestion des sessions de travail (commandes `exit` et `restauration`) ;
- la synchronisation entre les branches `master` et `auto` ;
- le comportement de l'auto-complétion des commandes et des questions.

Les résultats de cette phase de vérification ont été concluants : les traces intermédiaires (les *commits* générés automatiquement lors de la modification d'un fichier) ont été correctement générées, le mécanisme de marquage des questions comme résolues fonctionnait conformément aux attentes, et la gestion des sessions (fermeture et restauration) s'est déroulée comme prévu, avec les fichiers de l'espace de travail rendus inaccessibles jusqu'au prochain lancement de LAWG. Cette vérification nous a ainsi permis de confirmer le bon fonctionnement

de l'outil dans le cadre d'une utilisation simulant le déroulement d'un TP pour un étudiant.

8.2 ÉVALUATION EN SITUATION RÉELLE (VALIDATION)

Suite à cette phase de vérification, nous avons procédé à une phase de validation en conditions réelles en déployant l'outil auprès d'étudiant-es lors d'une véritable séance de TP, préalablement à l'expérimentation à plus grande échelle.

Puisque cette phase était destinée à déceler d'éventuels dysfonctionnements, ce test a été fait auprès d'étudiant-es en Master de développement logiciel, très habitués à l'utilisation de Git. Pour ce cours, le travail des étudiant-es n'était pas évalué. Grâce à cela, si un dysfonctionnement venait à affecter le travail des étudiant-es, il leur était facile de le corriger avec les commandes Git qui permettent de revenir sur des opérations Git faites précédemment.

Cette phase de validation en conditions réelles nous a permis de :

- vérifier la robustesse de l'outil face à des utilisations variées ;
- recueillir les retours informels des étudiant-es sur l'ergonomie de l'interface ;
- identifier des cas d'utilisation non anticipés ;
- valider le processus d'installation et de configuration ;
- confirmer la capacité de l'outil à générer des traces exploitables dans un contexte proche de celui de l'expérimentation.

La phase de validation a permis de confirmer que l'interface de LAWG était perçue comme intuitive et aisément prise en main par les étudiant-es, selon leurs retours informels en séance. Dès la présentation succincte de l'outil en début de séance, la majorité des utilisateur-rices ont pu l'exploiter de manière autonome.

Dans un contexte d'utilisation nominal, l'outil s'est comporté conformément aux attentes, reproduisant les résultats obtenus lors de la phase de vérification. Les traces étaient correctement générées, les questions marquées comme résolues de façon pertinente, et la gestion des sessions respectait les spécifications prévues.

Toutefois, quelques étudiant-es ont été confronté-es à une erreur critique : en cas d'arrêt inopiné de l'application, leur espace de travail se retrouvait dans un état incohérent, empêchant toute reprise de l'activité sans intervention manuelle. Cela pouvait arriver, par exemple, lorsqu'un-e étudiant-e lançait LAWG dans le terminal intégré d'un IDE et que ce dernier s'arrêtait inopinément, stoppant aussi LAWG.

En effet, si un événement coupait l'algorithme d'historisation (cf. figure 7.3) en plein milieu de son exécution, entre les étapes ❶ et ❺, l'historique des *commits* de la branche *main* ne peut pas retourner à son état initial, et contiendra l'historique des *commits* de la branche *auto*. Étant donné que nous utilisons la version "*mixed*" de la commande *reset*, qui ne modifie jamais les fichiers du répertoire de travail, aucun

travail de l'étudiant-e ne sera perdu. Cependant, l'état de l'espace de travail sera incohérent, et l'étudiant-e ne pourra pas reprendre son travail sans intervention manuelle de la part de l'enseignant-e. Elle ou il doit alors identifier à quelle étape le programme a été interrompu pour revenir en arrière d'autant d'étapes que nécessaire.

En conclusion de cette deuxième phase d'évaluation, le mécanisme d'historisation à deux branches tel que nous l'avions initialement implanté ne s'est pas révélé assez robuste aux erreurs. Afin d'éviter celles-ci lors de l'expérimentation en contexte écologique avec des étudiant-es moins expérimenté-es, nous avons fait le choix de n'utiliser qu'une unique branche pour à la fois la génération automatique des traces et la production de l'étudiant-e.

8.3 EXPÉRIMENTATION EN CONTEXTE ÉCOLOGIQUE

Alors que les étapes précédentes de vérification et de validation s'attachaient à éprouver la fiabilité technique et fonctionnelle de LAWG, l'expérimentation en contexte écologique présentée dans cette section vise à apporter des éléments de réponse aux questions de recherche. Il s'agit plus précisément d'évaluer la capacité des traces générées à rendre compte de comportements d'apprentissage complexes (QR2.1), tout en vérifiant que le dispositif ne perturbe pas l'activité de l'étudiant-e ni celle de l'enseignant-e (QR2.2).

Afin de répondre à ces questions, nous nous sommes fixés plusieurs objectifs spécifiques pour cette expérimentation. Premièrement, nous souhaitons mesurer l'impact de l'outil sur la richesse et l'exactitude des données collectées, ainsi que sur sa capacité à tracer finement et précisément le déroulement des activités des étudiant-es. Deuxièmement, l'objectif est de reconstituer et d'examiner les processus précis suivis par les étudiant-es, dans notre cas le respect du cycle TDD, ce qui serait impossible sans les traces enrichies. Enfin, nous visons à évaluer l'utilisabilité de LAWG et son acceptation par les différents acteurs et actrices impliquées, étudiant-es et enseignant-es, pour nous assurer de sa non-intrusivité.

8.3.1 *Méthode*

Pour évaluer LAWG, nous avons mené une expérimentation en Licence Informatique pendant laquelle on a demandé aux étudiant-es d'utiliser le système interactif tout en résolvant des questions (cf. figure 8.1).

Pour familiariser les étudiant-es avec l'outil, celui-ci a été utilisé lors de la séance précédant l'expérimentation. Durant l'expérimentation, les étudiant-es devaient d'abord lancer LAWG avant de commencer à travailler sur le sujet. Une fois une question n (identifiée "ITER_ n : <nom de la question>") résolue, elles et ils utilisaient la commande

Itérations à implémenter, **dans l'ordre** :

ITER_1 : Hello, Bob

La fonction prend en paramètre un nom (une chaîne de caractère) et retourne la chaîne "Hello, <nom>".

Par exemple, si on lui passe "bob", elle retourne "Hello, Bob" (à noter, la majuscule sur le nom).

```
fix ITER_1
```

ITER_2 : Gestion des cris

Si on donne une chaîne de caractères en majuscule, alors répondre en majuscule, et terminer par un point d'exclamation.

Par exemple, si on donne "JERRY", alors retourner "HELLO, JERRY !"

indice : la méthode "toUpperCase()" de la classe String peut vous simplifier la vie. On rappelle que deux chaînes de caractères peuvent être comparées en utilisant **equals**.

```
fix ITER_2
```

ITER_3 : Gestion des null et des chaînes vides

Si on passe en paramètre une chaîne de caractère vide, nulle ou blanche (que des espaces), alors retourner "Hello, my friend".

Par exemple, si on lui passe " ", elle retourne "Hello, my friend".

Indice : la méthode "trim()" de la classe String peut vous simplifier la vie.

```
fix ITER_3
```

FIGURE 8.1 – Extrait du sujet posé aux étudiant-es pour l'expérimentation

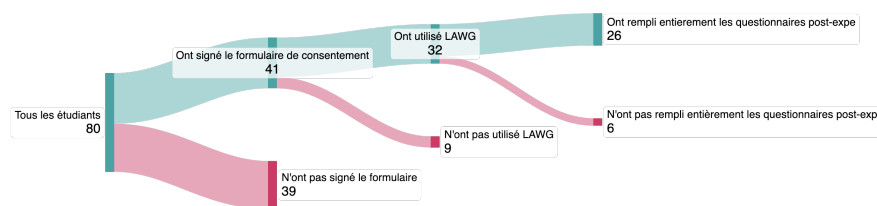


FIGURE 8.2 – Diagramme de Sankey montrant le nombre de participant-es à chaque étape de l'expérimentation

fix dans LAWG pour la marquer comme résolue. Ce protocole, expliqué en début de séance et régulièrement rappelé par l'enseignant, était également mentionné à la fin de chaque question.

8.3.2 Participants

L'expérimentation a été menée durant un cours portant sur l'écriture de tests et le développement piloté par les tests (*Test-Driven Development*, abrégé en TDD par la suite). Les participant-es à l'expérimentation sont des étudiant-es en deuxième année de Licence Informatique. L'utilisation basique de Git a été vue en première année. Le groupe de participant-es est composé d'environ 80 étudiant-es (cf. figure 8.2) et encadré par cinq enseignant-es, dont deux appartenaient à l'équipe de recherche en charge de l'expérimentation. Parmi ce groupe, 41 ont signé le formulaire de consentement, 32 ont utilisé LAWG, 26 ont aussi rempli entièrement les questionnaires post-expérimentation.

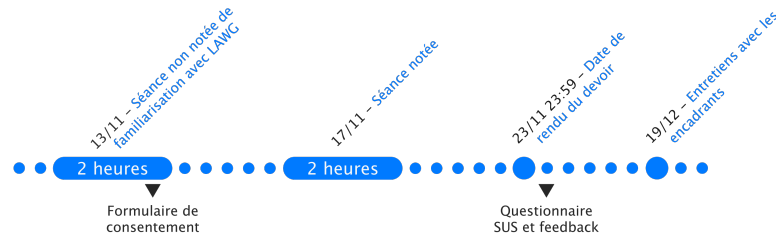


FIGURE 8.3 – Calendrier de l'expérimentation

8.3.3 Protocole

L'expérimentation est temporellement divisée en 3 phases (cf. figure 8.3). La première phase, qui est une étape de préparation pour laquelle les données générées ne font pas partie de l'expérimentation, détaillée dans la sous-section 8.3.4. La deuxième phase concerne le début concret de l'expérimentation, c'est la séance de TP noté. Et la troisième phase est la partie de travail en autonomie des étudiant-es qui finiront le travail hors séance. Les deux dernières phases sont détaillées en sous-section 8.3.5.

8.3.4 Préparation de l'expérimentation

Afin d'optimiser le temps consacré à la séance de TP notée, LAWG a été présenté et déployé lors de la séance précédente, non notée. Cette première utilisation a permis aux étudiant-es de se familiariser avec le système interactif et a servi de moment d'introduction à l'outil. C'est également au cours de cette séance que nous avons présenté en détail le fonctionnement de LAWG et fait signer le formulaire de consentement aux volontaires.

Cette séance préliminaire nous a permis d'identifier les principales interrogations des étudiant-es concernant l'utilisation de LAWG. Pour y répondre efficacement, nous avons intégré dans le fichier `README.md` du projet toutes les informations nécessaires à sa configuration et à son utilisation. Ce document présente les fonctionnalités essentielles de l'outil, les étapes d'installation et de démarrage, une brève description de l'expérimentation ainsi qu'une FAQ reprenant les questions fréquemment posées. Par souci de transparence, il contient également un lien vers le dépôt GitHub de LAWG.

Pour faciliter la mise en place du TP, nous avons fait le choix de distribuer les exécutables de LAWG plutôt que demander aux étudiant-es de les récupérer pendant la séance. Pour ces TP, les étudiant-es devaient créer un projet Eclipse localement puis le déposer sur Moodle. Pour faciliter la distribution des exécutables, nous avons créé un squelette de projet avec les fichiers requis pour commencer à travailler, dont les exécutables de LAWG pour toutes les distributions principale-

ment utilisées dans l'université (Windows 10 et 11, MacOS 11 et 12, Debian 11, Ubuntu 20 et 22). Un projet Eclipse minimal a été initialisé avec un fichier `.gitignore` correspondant à ce type de projet. Durant cette étape de création du squelette de projet, nous avons également configuré LAWG côté enseignant. Cela correspond à la création du fichier `.settings.yml` renseignant les groupes de TP et les identifiants des questions du sujet. Afin que les étudiant·es puissent créer leur propre projet en un clic, nous avons utilisé Github Classroom et le squelette de projet précédemment créé.

Enfin, la dernière étape est la modification du sujet. Nous avons ajouté le lien vers *l'assignment Github Classroom* et un court rappel du processus de marquage des questions résolues avec LAWG en préambule. À la fin de chaque question, nous avons aussi ajouté un rappel d'exécuter la commande `fix` de LAWG.

8.3.5 *Déroulement de l'expérimentation*

Dès le début de la séance, les étudiant·es disposaient d'environ deux heures pour commencer à travailler sur le sujet de TP dans un cadre encadré. Il était clair pour toutes et tous que ce travail était noté. Les étudiant·es volontaires utilisaient LAWG, tandis que celles et ceux qui ne participaient pas à l'expérimentation travaillaient directement avec Git. À la fin de la séance, elles et ils sauvegardaient leur travail afin de pouvoir le poursuivre en dehors du temps encadré.

Pour reprendre leur travail là où elles et ils l'avaient laissé, les volontaires relançaient LAWG. Une fois leur travail terminé, elles et ils apposaient le tag `v1.0.0` à l'aide de la commande `finish`. À partir de ce moment, celles et ceux qui le souhaitent pouvaient répondre au questionnaire SUS mis en ligne sur Moodle et formuler des retours sur l'utilisation de LAWG.

8.3.6 *Matériel*

Puisque nous manipulons des données personnelles, nous avons écrit un formulaire de consentement³ à destination des étudiant·es participant à l'expérimentation. Il donne des informations sur l'objectif et le déroulé de l'expérimentation, sur les données collectées, dont la nature et la durée de leur stockage.

Pour mesurer l'utilisabilité du système interactif LAWG, nous avons utilisé le questionnaire System Usability Scale (SUS) (*System Usability Scale*), proposé par BROOKE [17]. Le SUS est un outil standardisé d'évaluation de l'utilisabilité perçue, largement utilisé dans les domaines de l'interaction humain-machine et du génie logiciel. Il repose sur 10

3. Tous les formulaires, les données et autres ressources liées à cette expérimentation seront disponibles ici : https://osf.io/92ztw/?view_only=3a10cc1e76384fe590a0a22074313b7b

affirmations auxquelles les répondant-es indiquent leur degré d'accord sur une échelle de Likert à 5 items, produisant un score unique sur 100. Ce questionnaire présente plusieurs avantages qui ont motivé notre choix : sa brièveté, qui limite la charge imposée aux participant-es en fin de séance, sa fiabilité éprouvée sur un grand nombre d'études [6], et l'existence de seuils d'interprétation normés permettant de situer le score obtenu par rapport à un large corpus de systèmes évalués [7]. Ces propriétés en font un instrument particulièrement adapté à notre contexte, où l'objectif est d'obtenir une mesure rapide et comparable de l'utilisabilité perçue de LAWG par les étudiant-es. Nous avons utilisé une version française du questionnaire. Ce questionnaire est anonyme et est distribué avec une question supplémentaire pour d'éventuels retours sur l'utilisation de LAWG. Il a été distribué en utilisant Google Forms, via la plateforme Moodle car nous voulions que les étudiant-es puissent y répondre après avoir terminé leur travail, afin de maximiser le temps d'utilisation de LAWG. Nous avons eu 18 réponses, dont 6 retours d'expérience.

Enfin, pour obtenir les retours des enseignants et collecter les éventuels retours d'étudiant-es qu'ils ont pu avoir, nous avons organisé des entretiens guidés avec chacun d'entre eux. L'audio des entretiens a été enregistré pour permettre une retranscription précise, et nous nous sommes assurés de l'accord des enseignants en leur faisant signer un formulaire de consentement avant l'entretien. Tous les enseignants ont accepté et signé le formulaire.

8.3.7 Méthodes d'analyse

Dans cette section, nous détaillons les méthodes d'analyse mises en œuvre pour à partir des données recueillies lors de l'expérimentation. Notre démarche s'articule autour de plusieurs axes complémentaires.

Nous débutons par le calcul d'indicateurs portant sur les dimensions de qualité des traces identifiées dans la section 6.4 : exactitude, complétude, cohérence et actualité. Ces indicateurs permettent de comparer les traces produites avant l'introduction de LAWG et celles obtenues après sa mise en place.

L'analyse centrale de cette expérimentation concerne l'exploitation des traces à l'aide d'outils de PM afin d'identifier des comportements complexes, impossibles à capturer sans LAWG. L'objectif est de reconstituer et d'examiner les processus précis suivis par les étudiant-es, dans notre cas le respect du cycle TDD. En effet, grâce à la granularité et à la structuration des traces fournies par LAWG, il devient possible d'identifier la chronologie des modifications faites sur les fichiers au sein de la résolution d'une question et de vérifier l'alignement de certaines pratiques avec les recommandations pédagogiques. Le TDD est un cycle de 5 phases [9] : ① écrire le test ② lancer tous les tests et vérifier que le nouveau test ne passe pas ③ écrire le code minimal

faisant passer le test ④ lancer tous les tests et vérifier qu'ils passent tous ⑤ améliorer le code (*refactor*) si besoin. Nous voulons voir si ces phases sont identifiables grâce aux traces générées par LAWG. Cette identification, qui n'était pas envisageable avec les traces disponibles auparavant, constituerait un apport majeur de LAWG.

Enfin, nous menons plusieurs analyses relatives à l'utilisabilité de LAWG et à son acceptation par les différents acteurs impliqués, étudiant·es et enseignant·es. Nous analysons les résultats du questionnaire SUS, dont les scores ont été calculés selon la méthode présentée dans [17], afin d'obtenir une mesure quantitative de la satisfaction des utilisateurs et utilisatrices quant à l'utilisation de LAWG, et l'utilisabilité subjective de l'outil. Nous examinons également les retours écrits des étudiant·es, qui apportent un éclairage qualitatif sur leur expérience d'utilisation, ainsi que les entretiens réalisés avec les enseignants, permettant de recueillir leur point de vue sur l'intégration de LAWG dans le dispositif pédagogique et sur les éventuelles difficultés rencontrées.

L'ensemble de ces analyses vise à fournir une vision générale de l'impact de LAWG sur la qualité des données collectées, de l'acceptabilité de l'outil par les étudiant·es et les enseignant·es, ainsi que de la facilité de sa mise en place par les enseignant·es.

RÉSULTATS ET DISCUSSION

9.1	Résultats	111
9.1.1	Indicateurs sur la qualité des traces	111
9.1.2	Identification du Test Driven Development (TDD)	131
9.1.3	Formulaire SUS et retours des étudiant-es	138
9.1.4	Entretiens avec les enseignants	139
9.2	Discussion	144
9.2.1	Des nouvelles traces exploitables pour des nouveaux comportements	144
9.2.2	Analyse au regard de la conscience de la situation	146
9.2.3	Un besoin de séparer les traces générées avéré	147
9.2.4	Une adhésion modérée malgré des bénéfices reconnus	148
9.2.5	Recommandations de conception	148
9.2.6	Limites	149
9.3	Bilan de l'itération	151
9.3.1	Synthèse des apports	151
9.3.2	Bilan au regard de la conscience de la situation	151

Ce chapitre poursuit la *phase d'évaluation* de la seconde itération du cycle DBR en présentant les résultats de l'expérimentation en contexte écologique et en les discutant au regard des questions de recherche. Il se compose de deux parties. La première présente les résultats de manière descriptive, structurés par type d'analyse, des plus quantitatives vers les plus qualitatives. La seconde partie propose une interprétation transversale de ces résultats en les agrégeant pour répondre aux questions QR2.1 et QR2.2. De plus, conformément à notre démarche pragmatiste, nous consacrons une section à l'adhésion des étudiant-es et des enseignant-es à LAWG ainsi qu'à la perception par les enseignant-es de la facilité de mise en place de l'outil dans un cours existant.

9.1 RÉSULTATS

9.1.1 Indicateurs sur la qualité des traces

Comme présenté dans la section 6.4, nous évaluons la qualité des traces selon quatre dimensions : exactitude, complétude, cohérence et actualité. Nous posons l'hypothèse que c'est principalement sur ces dimensions que LAWG permet d'améliorer la qualité des données

collectées. En effet, l'outil permet de générer un nombre bien plus important de traces, ce qui rend possible une observation plus fine et plus détaillée des comportements des étudiant-es, ainsi qu'une meilleure actualisation des données collectées. Les *commits* sont toujours générés suite à des contributions faites par les étudiant-es, mais LAWG automatise leur enregistrement et la rédaction des messages associés, ce qui limite les erreurs de saisie et les oublis. Cela permet de réduire les risques d'informations manquantes ou imprécises, d'assurer une meilleure cohérence dans la structuration des traces, et de reconstituer plus fidèlement la chronologie des actions réalisées. Afin de vérifier cette hypothèse, nous avons calculé des indicateurs spécifiques pour chacune de ces dimensions, ce qui nous permettra d'évaluer de manière objective l'apport de LAWG sur la qualité des traces générées dans le contexte pédagogique étudié.

Ces indicateurs seront calculés sur deux groupes d'étudiant-es distincts : d'une part, celles et ceux ayant utilisé LAWG lors de l'expérimentation, et d'autre part, celles et ceux n'ayant pas utilisé LAWG mais ayant tout de même signé le formulaire de consentement, qui constituent le groupe contrôle. Cette comparaison permettra d'évaluer de manière plus rigoureuse l'impact de l'utilisation de LAWG sur la qualité des traces produites, en isolant l'effet de l'outil par rapport à une situation témoin.

Sur les 9 étudiant-es n'ayant pas utilisé LAWG, nous devons en exclure 4 qui ont soit utilisé LAWG au tout début puis arrêté, soit produit très peu de *commits*, nous laissant 5 étudiant-es dans notre groupe contrôle.

9.1.1.1 *Actualité*

L'actualité d'une donnée se définit comme le degré auquel celle-ci reflète l'état le plus récent de l'objet ou de l'événement qu'elle décrit. Elle se mesure notamment par l'écart temporel entre la survenue d'un événement et la production de la trace correspondante. Par ailleurs, la dimension d'actualité englobe également la notion de disponibilité temporelle, c'est-à-dire la capacité de la donnée à être accessible au moment opportun, lorsque la personne utilisatrice en a effectivement besoin.

L'hypothèse que l'on fait concernant l'apport de LAWG sur cette dimension de la qualité des données est la suivante : générer des traces (des *commits*) au moment le plus proche de ce que les systèmes d'exploitation nous le permettent et de manière indépendante à l'outil d'édition de texte utilisé, lorsque la modification d'un fichier est sauvegardée, permet d'approcher la meilleure actualité. De plus, multiplier la production de traces permet non seulement de réduire l'écart entre chaque trace, la durée d'activité réelle de l'étudiant-e entre chaque *commit*, mais aussi de rendre ses traces atomiques en ce qui concerne

les modifications associées et donc de rendre compte de la chronologie précise des événements de modification de fichier.

Pour évaluer l'actualité des traces produites, nous avons choisi de calculer, pour chaque étudiant-e, le temps moyen séparant deux traces successives (c'est-à-dire l'intervalle moyen entre deux *commits*). Cette mesure permet d'estimer à quel point les traces sont produites de manière rapprochée dans le temps, et donc de juger de leur fraîcheur et de leur capacité à refléter fidèlement le déroulement réel de l'activité. Nous avons ensuite agrégé ces moyennes individuelles pour obtenir, d'une part, la moyenne du groupe contrôle (étudiant-es n'ayant pas utilisé LAWG) et, d'autre part, celle du groupe expérimental (étudiant-es ayant utilisé LAWG). Cette comparaison entre les deux groupes nous permettra d'objectiver l'apport de LAWG sur la dimension d'actualité des traces, en identifiant si l'outil favorise effectivement une production plus régulière et plus proche du temps réel des événements pédagogiques.

Les temps moyens calculés selon les groupes auxquels on s'intéresse sont présentés dans le tableau de la figure 9.3.

Concrètement, afin d'évaluer les temps moyens entre deux traces générées par les étudiant-es, plusieurs choix méthodologiques ont été adoptés.

D'abord, comme on demandait aux étudiant-es de générer le rapport de tests à la fin de leur TP, cela générerait plusieurs dizaines de fichiers, et donc autant de *commits* automatiques. Ces séries de *commits* ont été nettoyées afin d'éliminer un biais introduit par les blocs massifs de *commits* de type "[created]", jugés non représentatifs de l'activité réelle des étudiant-es. Donc, toutes les séquences comprenant plus de cinq *commits* consécutifs de ce type ont été supprimées.

Ensuite vient l'étape de calcul des durées entre chaque *commit*. Or, il faut garder en tête que nous voulons connaître les durées entre les *commits* lors d'une session de travail. C'est-à-dire que l'on veut être capable d'identifier un minimum les périodes pendant lesquelles il n'est pas possible que l'étudiant-e travaille et donc pendant lesquelles il est normal de ne voir aucun *commit*. Un exemple très parlant et qui fausse grandement les temps moyens, c'est lorsqu'une nuit sépare deux *commits*. Si on compte cet intervalle de temps, on introduit des valeurs extrêmes et qui n'ont pas vraiment de sens dans nos analyses. Un moyen simple pour prévenir ce problème est de ne pas compter les durées entre deux *commits* qui ne datent pas du même jour. Certes, on peut observer quelques cas de sessions de travail nocturnes s'étalant légitimement sur deux jours. Mais, le gain de ne pas compter les valeurs extrêmes que peuvent représenter ces périodes rend négligeable de ne pas compter quelques durées légitimes mais qui ne compteraient de toute façon pas pour grand-chose dans le calcul du temps moyen.

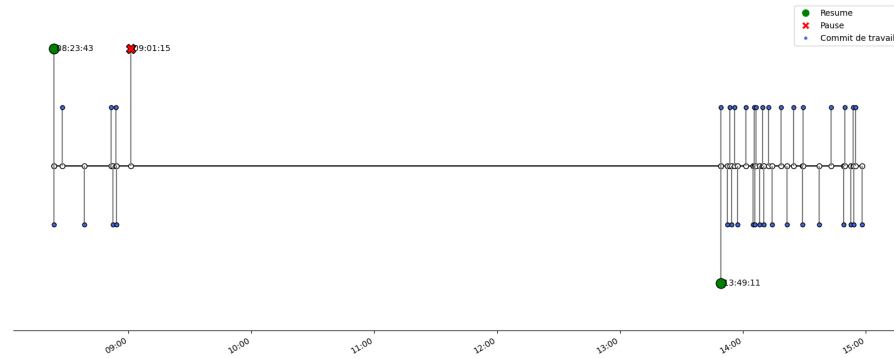


FIGURE 9.1 – Frise chronologique des *commits* d'un-e étudiant-e illustrant deux sessions de travail dans une même journée

La première approche de calcul des intervalles repose donc sur une agrégation journalière. Pour chaque étudiant-e, l'ensemble des écarts de temps entre *commits* réalisés au cours d'une même journée est rassemblé, puis les moyennes sont calculées, pour le groupe contrôle comme pour le groupe expérimental. Comme le montre la boîte à moustaches de la figure 9.2a, les temps moyens entre deux traces pour le groupe contrôle sont non seulement élevés (avec une moyenne de 1:14:17), mais aussi très dispersés avec un écart interquartile allant de 34:16 à 1:23:59. On retrouve aussi une valeur aberrante de 2:40:39. Dans la figure 9.2b, la première boîte à moustaches montre pour le groupe expérimental des valeurs bien plus basses en comparaison, avec une moyenne de 3:20, et un écart entre la valeur minimum et la valeur maximum bien plus petit (de 0:22 à 8:51). On peut aussi remarquer qu'aucune valeur aberrante n'est présente.

Cependant, cette mesure journalière présente une limite importante : elle ne permet pas de distinguer les cas où plusieurs sessions de travail distinctes ont lieu au cours d'une même journée. Dans de telles situations, des intervalles de temps très importants peuvent apparaître entre le dernier *commit* d'une session (par exemple le matin) et le premier *commit* de la session suivante (par exemple le soir). Ces valeurs extrêmes, qui ne reflètent pas une activité continue mais plutôt une interruption prolongée, ont pour effet d'augmenter artificiellement la moyenne des intervalles entre deux traces. Ce phénomène est illustré dans la figure 9.1, où la durée entre 09:01 et 13:49 sera comptée dans le calcul du temps moyen entre deux traces.

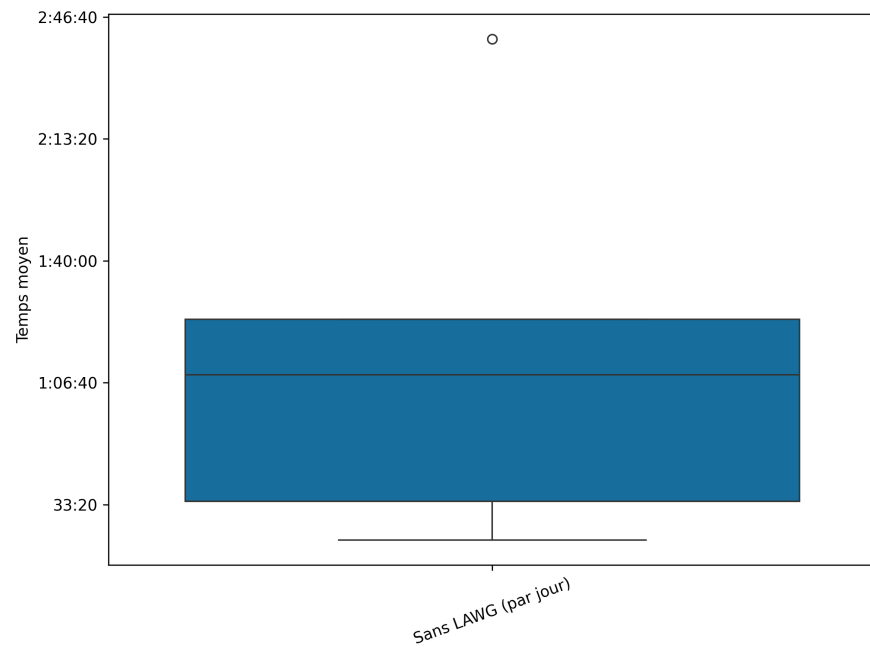
Une seconde approche que nous avons explorée pour le calcul des intervalles entre deux traces consiste à s'appuyer sur la notion de session de travail telle qu'elle est explicitement délimitée par l'outil LAWG. Dans ce cadre, une session est bornée par des *commits* particuliers, à savoir les *commits* de type "Resume" (reprise d'activité) et "Pause" (interruption d'activité). L'idée est alors de ne considérer, pour le calcul des durées entre deux *commits*, que les intervalles situés à l'intérieur d'une même session, c'est-à-dire entre deux *commits* consécutifs qui

ne sont ni des "Resume" ni des "Pause". Plus précisément, nous avons choisi d'exclure systématiquement les intervalles où l'un des deux *commits* est un "Resume", car il a été observé que certain-es étudiant-es fermaient LAWG de manière inopinée, sans générer le *commit* "Pause" attendu. Toutefois, le *commit* "Resume" généré lors du redémarrage de LAWG suffit à marquer le début d'une nouvelle session de travail, permettant ainsi de segmenter correctement l'activité. Cette approche, qui n'est possible qu'avec l'utilisation de LAWG, vise à affiner la mesure de l'actualité des traces en éliminant les longues périodes d'inactivité ou les interruptions non représentatives entre deux sessions de travail dans une même journée comme on peut le voir dans la figure 9.1.

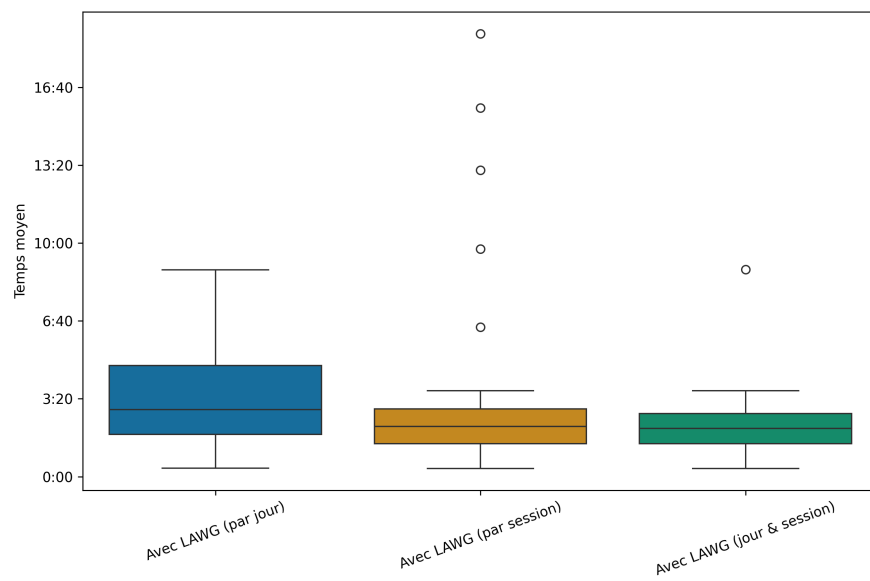
La deuxième boîte à moustaches de la figure 9.2b montre la répartition des durées moyennes avec cette deuxième approche. On constate que les valeurs obtenues avec cette seconde approche sont nettement moins dispersées que celles issues de l'agrégation journalière. En effet, l'écart interquartile s'établit de 1:25 à 2:54, ce qui traduit une concentration des temps moyens entre deux *commits* autour de la médiane. De plus, la moustache inférieure atteint 0:21 tandis que la moustache supérieure ne dépasse pas 3:41, indiquant que la majorité des intervalles se situent dans une plage temporelle restreinte. Cette moindre dispersion témoigne d'une meilleure homogénéité des durées calculées et d'une actualité accrue des traces produites grâce à la segmentation fine des sessions de travail permise par LAWG.

Cependant, bien que l'on observe sur le graphe un resserrement notable des valeurs autour de la médiane par rapport à la première approche, la moyenne obtenue avec cette méthode s'avère légèrement plus élevée, atteignant 3:36, soit un gain moyen par rapport à l'approche journalière de -3.74% (la troisième colonne du tableau de la figure 9.3 montre la moyenne des gains pour chaque étudiant-e de chaque couple groupe/approche).

La valeur étonnamment élevée de la moyenne s'explique par la présence de cinq valeurs aberrantes, correspondant à des intervalles exceptionnellement longs entre deux *commits*, qui viennent fausser la moyenne et en limiter la représentativité. Pour vérifier si, malgré tout, il existe une différence significative entre les temps moyens entre deux traces calculés selon l'approche journalière d'une part, et selon l'approche par sessions LAWG d'autre part, nous avons cherché à appliquer un test statistique approprié. Le test de Student (ou test T) pour échantillons appariés est souvent utilisé pour comparer deux jeux de données dépendants. Cependant, ce test suppose que la distribution des différences entre les paires de valeurs suit une loi normale. Dans notre cas, la distribution des différences s'écarte nettement de la normalité, avec la présence de valeurs aberrantes et une forte asymétrie. Si la littérature indique que le test T peut parfois rester robuste face à des écarts modérés à la normalité, il est généralement admis qu'il faut disposer de plus de 30 paires environ pour garantir la va-



(a) Groupe contrôle (sans LAWG)



(b) Groupe expérimental (avec LAWG)

FIGURE 9.2 – Comparaison des boîtes à moustaches des temps moyens entre deux *commits* pour les groupes contrôle et expérimental.

Groupe	Méthode de calcul des intervalles	Moyenne des gains par rapport à la méthode journalière	Temps Moyen du groupe
Étudiant-es sans LAWG	Journalière		1:14:17
Étudiant-es avec LAWG	Journalière		0:03:20
Étudiant-es avec LAWG	Sessions LAWG	-3,74 %	0:03:36
Étudiant-es avec LAWG et bonne utilisation	Sessions LAWG	23,31 %	0:02:03
Étudiant-es avec LAWG	Journalière & Sessions LAWG	24,76 %	0:02:14

FIGURE 9.3 – Tableau des statistiques des temps moyens entre deux *commits* pour les groupes contrôle et expérimental.

lidité du test dans ces conditions. Ici, non seulement le nombre de paires est limité, mais la distribution des différences est loin d'être normale. Les valeurs aberrantes présentes risquent de biaiser le calcul du test T, car celui-ci utilise des moyennes et est très sensible aux valeurs extrêmes. Pour ces raisons, nous avons choisi d'utiliser le test des rangs signés de Wilcoxon. Ce test non paramétrique permet de comparer deux échantillons appariés sans supposer la normalité des distributions. Il s'appuie sur la comparaison des médianes plutôt que des moyennes, ce qui le rend beaucoup moins sensible aux valeurs aberrantes. L'hypothèse nulle (H_0) postulait l'absence de différence entre les distributions ne pouvant pas être attribuée au hasard, tandis que l'hypothèse alternative (H_1) supposait une différence significative. Le test bilatéral a donné une valeur de $p = 0.00782$, nettement inférieure au seuil $\alpha = 0.05$ (niveau de confiance de 95%). Nous rejetons donc H_0 au profit de H_1 , ce qui indique que les différences observées entre les deux jeux de données sont statistiquement significatives et ne peuvent pas être attribuées au hasard.

De plus, l'analyse des résultats du test de Wilcoxon révèle que la somme des rangs positifs (406,5) est nettement supérieure à la somme des rangs négatifs (121,5). Cette observation indique que les durées moyennes calculées selon l'approche journalière sont significativement plus élevées que celles obtenues avec l'approche fondée sur les sessions de travail LAWG. Ainsi, malgré une moyenne globale des gains légèrement négative, le test statistique met en évidence un gain positif significatif en faveur de l'approche par sessions LAWG pour la mesure de l'actualité des traces.

En examinant les traces des cinq étudiant-es dont les durées moyennes entre deux traces apparaissent comme aberrantes, il s'avère qu'elles ou ils n'ont eu qu'une seule session LAWG couvrant plusieurs jours, du début à la fin du projet. Cette utilisation non conforme de l'outil a pour conséquence d'inclure dans le calcul des moyennes des intervalles de temps très longs, parfois entrecoupés de périodes d'inactivité telles

Nos jeux de données (les durées moyennes entre deux traces pour chaque étudiant-e) sont forcément appariés car ils proviennent des même traces, des mêmes étudiant-es.

que des nuits de sommeil. Ce phénomène, qui est précisément corrigé par l'approche journalière, fausse la représentativité des moyennes obtenues pour ces étudiant·es et explique la présence de valeurs extrêmes dans la distribution. Afin d'évaluer l'impact de cette utilisation non nominale de LAWG, nous avons recalculé les statistiques en excluant les étudiant·es présentant des moyennes aberrantes. Pour le sous-groupe d'étudiant·es ayant utilisé LAWG conformément, les résultats sont nettement plus favorables : le temps moyen entre deux traces descend à 02:03, et la moyenne des gains atteint 23,31%. Ce résultat confirme la pertinence de l'approche par sessions LAWG pour refléter plus finement l'activité réelle. Bien qu'il ne soit pas méthodologiquement acceptable d'écarter les données des participant·es ne respectant pas strictement le protocole, car cela introduirait un biais de sélection, cette analyse complémentaire vient confirmer la conclusion du test de Wilcoxon quant à l'apport de l'approche par sessions LAWG.

Dans la perspective d'affiner encore la mesure de l'actualité des traces et de mieux représenter l'activité réelle des étudiant·es, nous avons envisagé une troisième approche, combinant les avantages des deux méthodes précédentes, tout en intégrant toutes les étudiant·es, y compris les cas d'utilisation non conforme de LAWG. Le résultat est un temps moyen de 02:14 et une moyenne de gain légèrement supérieure à précédemment à 24,76%. On retrouve des valeurs très proches, voire meilleures, que celles obtenues avec l'approche par sessions LAWG appliquée uniquement aux étudiant·es ayant utilisé LAWG de manière conforme, mais cette fois-ci en intégrant l'ensemble des individus. Cette généralisation permet de conserver la représentativité de l'échantillon tout en bénéficiant d'une mesure affinée de l'actualité des traces.

Par ailleurs, si l'on observe la troisième boîte à moustaches de la figure 9.2b, qui illustre la distribution des durées selon cette troisième approche, on constate un profil de dispersion des valeurs quasiment identique à celui de la seconde approche. L'écart interquartile et l'étendue des moustaches témoignent d'une homogénéité similaire, confirmant que la prise en compte de l'ensemble des étudiant·es n'entraîne pas de dégradation notable de la qualité de la mesure. Par contre, on élimine presque toutes les valeurs aberrantes. Ainsi, cette approche hybride combine les avantages de la segmentation fine des sessions LAWG tout en assurant l'inclusion de tous les cas, ce qui la rend particulièrement pertinente pour une évaluation globale de l'actualité des traces produites.

9.1.1.2 Complétude

Selon le cadre de Wang et Strong [119], la complétude d'un jeu de données se réfère à la mesure dans laquelle l'ensemble des informations requises pour une utilisation spécifique est effectivement présent. Autrement dit, une donnée est considérée comme complète

si elle contient tous les éléments nécessaires à la compréhension et à l'exploitation du phénomène observé, sans lacunes ni omissions significatives, pour un consommateur ou une consommatrice de données. Cette dimension s'avère particulièrement cruciale dans le contexte de l'analyse de traces d'activité, où l'absence de certaines informations peut fausser l'interprétation des comportements ou limiter la portée des analyses.

Une information manquante pour identifier des comportements d'apprentissage complexes (comme vus précédemment) est la chronologie des modifications des fichiers, que l'on n'a pas avec un seul *commit* enregistrant une fois par fichier toutes les modifications. De plus, pour analyser l'activité réelle de l'étudiant-e, on a besoin de connaître les périodes d'activité au sein de la réalisation d'une question, ce que l'on ne peut pas faire avec un seul *commit*.

Au vu de cela, pour évaluer la complétude des traces produites avec et sans l'utilisation de LAWG, nous avons choisi de nous appuyer sur un indicateur quantitatif : la *densité de commits par question*. Cet indicateur correspond au nombre de *commits* réalisés par un-e étudiant-e ou générés par l'outil pour parvenir à la résolution d'une question donnée, c'est-à-dire jusqu'au *commit* marquant la résolution de cette question. L'idée sous-jacente est que plus le nombre de *commits* intermédiaires est élevé, plus la séquence de *commits* est susceptible de refléter fidèlement le cheminement réel de l'étudiant-e, en capturant les différentes étapes, essais, corrections et ajustements qui jalonnent la résolution du problème.

Ainsi, nous considérons qu'une densité de *commits* élevée traduit une granularité fine des traces, ce qui contribue à une meilleure complétude des données du point de vue du suivi du processus d'apprentissage. À l'inverse, une faible densité de *commits* peut indiquer que certaines étapes du raisonnement ou du travail n'ont pas été enregistrées, soit par omission, soit en raison d'une utilisation non systématique de l'outil de traçage. En comparant cet indicateur entre les groupes avec et sans LAWG, il devient possible d'évaluer dans quelle mesure l'usage de l'outil favorise la production de traces plus complètes, et donc plus riches pour l'analyse ultérieure des comportements d'apprentissage.

Pour calculer la densité de *commits* par question, nous avons procédé comme suit : pour chaque étudiant-e, les *commits* sont d'abord triés chronologiquement. Ils sont ensuite regroupés en blocs de travail délimités par des *commits* de type "ITER_x", qui marquent la fin d'une question donnée. À chaque bloc, on associe la question correspondant au jalon "ITER_x" terminal. Les *commits* jugés non pertinents (ceux liés à la pause/reprise, au *commit* initial ou à l'ajout d'une *deadline*) sont exclus afin de ne conserver que les contributions effectives. Le résultat est ainsi, pour chaque étudiant-e, un comptage du nombre de *commits* significatifs par question (Q1 à Q10). Cela fournit une mesure

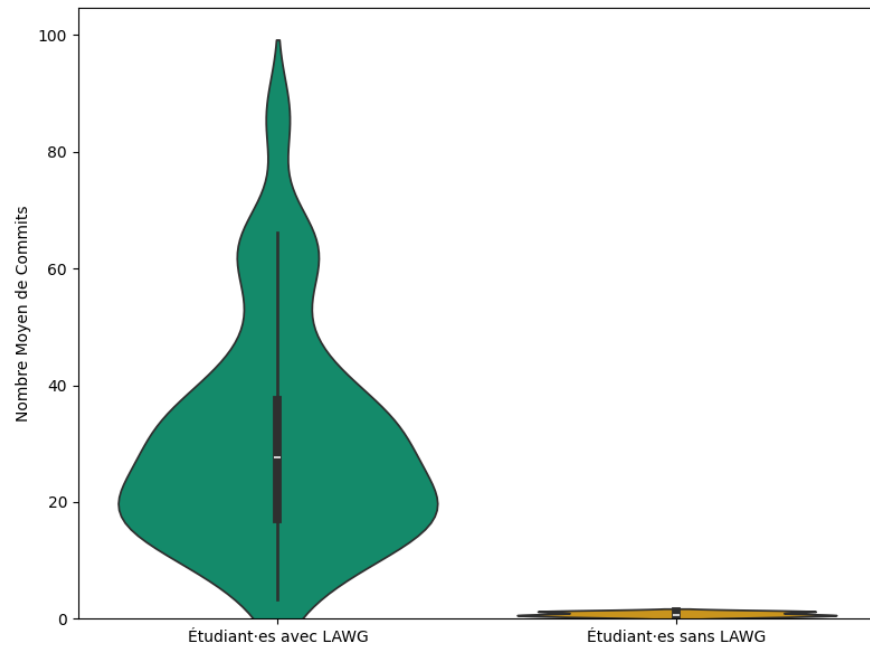


FIGURE 9.4 – Distribution globale de la densité de commits par question (diagramme en violon)

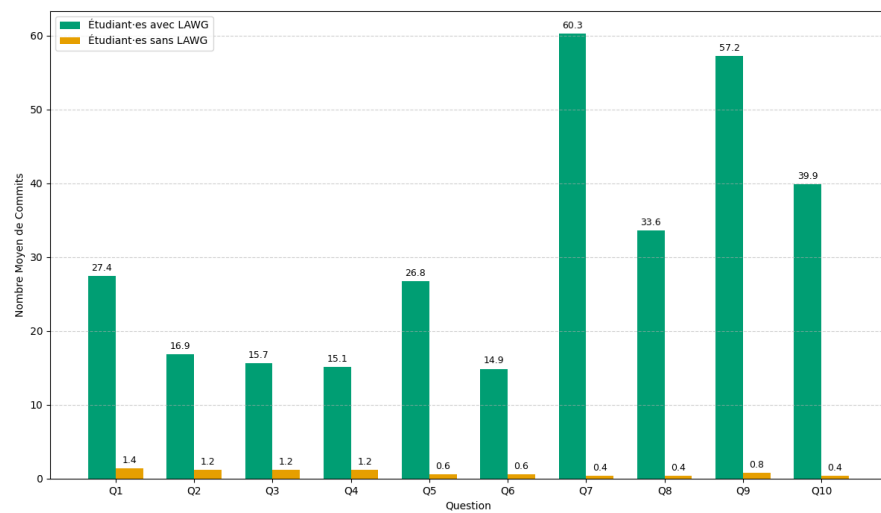


FIGURE 9.5 – Comparaison groupée de la densité de commits par question selon les groupes

de densité de *commits* reflétant l'intensité de travail allouée à chaque question.

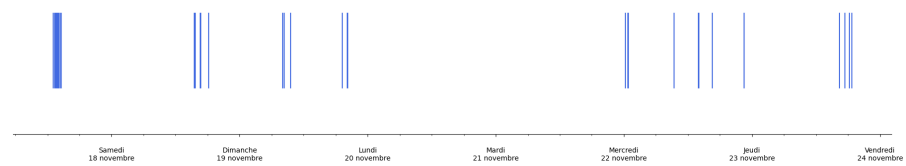
Les résultats sont présentés dans les figures 9.4 et 9.5. Sans grande surprise, la densité moyenne de *commits* par question se situe autour de 1 pour le groupe contrôle, sans LAWG, alors que le groupe expérimental, avec LAWG, a une densité moyenne de *commits* par question bien supérieure, environ 30 fois en moyenne. Ce résultat est naturellement dû aux *commits* générés automatiquement par LAWG. Bien que ce soit un résultat très attendu et assez trivial, il reste néanmoins un marqueur d'une meilleure complétude des traces produites avec LAWG. En multipliant les *commits* atomiques générés automatiquement à chaque action de l'étudiant·e, on trace mécaniquement une chronologie plus précise des modifications de chaque fichier. Comme on le montre dans la section 9.1.2, on peut profiter de cette chronologie pour catégoriser chacune de ces actions et identifier des comportements complexes.

Comme évoqué précédemment, l'un des manques majeurs des traces issues d'un unique *commit* par question réside dans l'absence d'information sur les périodes d'activité de l'étudiant·e. Il est donc impossible de reconstituer précisément le déroulement du travail, les pauses, ou les phases d'essais, de corrections ou de reprise du code. Pour illustrer concrètement cet enjeu, la figure 9.6 propose une comparaison visuelle grâce à des frises chronologiques des *commits* réalisés par les étudiant·es selon les différents groupes.

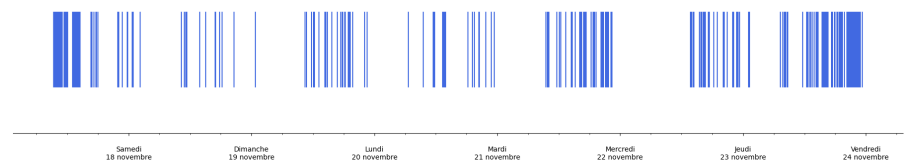
On y distingue trois situations : (a) les étudiant·es du groupe contrôle, sans LAWG, (b) les étudiant·es du groupe expérimental avec LAWG, mais en ne considérant que les *commits* de résolution de question, et (c) les étudiant·es du groupe expérimental avec LAWG, en affichant l'ensemble des *commits*. La frise (b) représente le même groupe que la frise (c), à savoir les étudiant·es ayant utilisé LAWG, dont les données générées automatiquement ont été retirées. Cela permet de reproduire des traces qui seraient issues d'un processus de résolution semblable à celui du groupe contrôle. L'objectif est de comparer l'apport des *commits* automatiques en matière de couverture temporelle du traçage en neutralisant le biais induit par la taille du groupe expérimental, près de six fois supérieure à celle du groupe contrôle.

Dans le groupe sans LAWG (a), la plupart des étudiant·es n'effectuent qu'un seul *commit* par question, souvent à la toute fin, ce qui laisse de larges zones blanches entre les *commits* de résolution. Cette absence de points intermédiaires illustre parfaitement le manque d'information sur la chronologie réelle du travail, et rend difficile, voire impossible, l'identification des périodes d'activité ou d'inactivité.

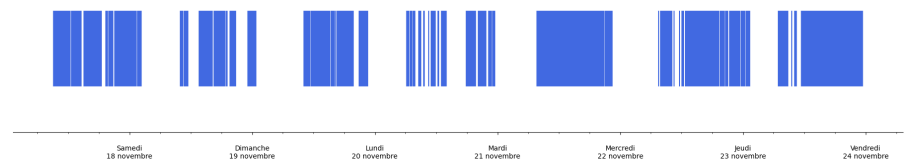
À l'inverse, la frise (c), qui inclut tous les *commits* générés automatiquement par LAWG, montre une densité beaucoup plus importante de points sur l'axe temporel. Les périodes d'activité sont alors bien mieux



(a) Frise chronologique des *commits* des étudiant-es sans LAWG



(b) Frise chronologique des *commits* des étudiant-es avec LAWG, dont seuls les *commits* de résolution de question sont affichés



(c) Frise chronologique des *commits* des étudiant-es avec LAWG

FIGURE 9.6 – Comparaison visuelle des frises chronologiques des *commits* des étudiant-es avec et sans LAWG.

documentées : il devient possible de suivre précisément la progression des étudiant-es, de repérer les phases d'essais, de corrections ou de retours en arrière, et d'obtenir ainsi une vision beaucoup plus fidèle du processus d'apprentissage.

Cette visualisation semble indiquer que l'utilisation de LAWG permet d'obtenir une couverture plus fine et plus continue de l'activité des étudiant-es, en comparaison avec l'approche manuelle qui aboutit principalement à des points de repère ponctuels. Cela suggère que l'outil pourrait pallier en partie le manque d'information sur les périodes d'activité tout au long du processus. Toutefois, pour aller au-delà de cette analyse graphique et limiter les biais dans l'interprétation, nous nous appuyons, dans les paragraphes suivants, sur une comparaison des métriques de couverture individuelle grâce à un test non paramétrique Mann-Whitney U, plus standardisées pour réduire le biais de taille d'échantillon, qui permettront une comparaison plus rigoureuse entre les groupes.

La couverture temporelle a été calculée individuellement pour chaque étudiant-e en reprenant la même méthode que précédemment pour générer les frises chronologiques. Les traces d'activité de chaque étudiant-e ont été isolées et filtrées pour ne conserver que les traces pertinentes, puis, une frise chronologique a été générée pour chaque étudiant-e, représentant ses moments d'activité par des barres verticales bleues. Enfin, cette image a été analysée pour calculer la couverture : le score final est le pourcentage de pixels bleus le long de l'axe temporel, représentant la proportion du temps total couverte par l'activité de l'étudiant-e.

La figure 9.7 présente une boîte à moustaches comparant la distribution des taux de couverture temporelle des commits entre le groupe contrôle et le groupe expérimental. On constate que la médiane de la couverture temporelle est considérablement plus élevée dans le groupe expérimental (6,43%) que dans le groupe contrôle (0,89%), et que la dispersion des valeurs y est également plus importante, avec certaines étudiant-es du groupe expérimental atteignant des taux proches de 15%. À l'inverse, la couverture observée dans le groupe contrôle reste faible et peu variable.

Pour évaluer rigoureusement l'effet de l'utilisation de LAWG, nous avons formulé l'hypothèse nulle (H_0) selon laquelle il n'existerait pas de différence significative entre les distributions des taux de couverture temporelle des deux groupes, et l'hypothèse alternative (H_1) selon laquelle l'usage de LAWG entraînerait une augmentation significative de cette couverture.

La taille réduite de l'échantillon contrôle ($n = 5$), la non-normalité des distributions et l'indépendance des groupes ont conduit à privilégier le test de Mann-Whitney U. Le résultat obtenu ($U = 159,5$, $p < .001$) montre que dans la quasi-totalité des paires possibles entre les valeurs du groupe contrôle et celles du groupe expérimental, la

Nous avons pris une ligne d'une hauteur de 1 pixel au centre de la frise, en partant horizontalement du début de la frise jusqu'à la fin de celle-ci.

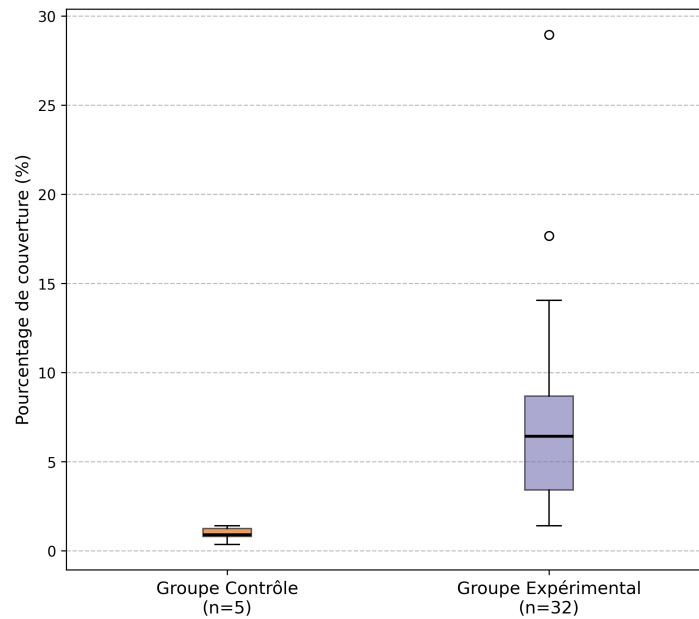


FIGURE 9.7 – Distribution de la couverture temporelle des commits selon le groupe contrôle et le groupe expérimental

couverture temporelle expérimentale est supérieure à celle du contrôle (159 fois sur 160), reflétant ainsi une différence très significative entre les deux groupes.

Ainsi, (H_0) est rejetée au seuil $\alpha = 0,05$, au profit de (H_1) : l'utilisation de LAWG s'accompagne bien d'un gain significatif en termes de couverture temporelle des traces d'activité des étudiant-es.

Afin de valider la robustesse de ce résultat et de l'illustrer de manière plus concrète, nous avons mené une analyse complémentaire par simulation. L'objectif était de répondre à une question simple : la performance de notre groupe contrôle est-elle simplement un peu plus faible, ou est-elle exceptionnellement basse par rapport à ce que l'on pourrait attendre de n'importe quel groupe de 5 étudiant-es utilisant l'outil ?

Afin de comparer rigoureusement la couverture temporelle observée dans le groupe expérimental (LAWG) à celle du groupe contrôle, nous avons utilisé une méthode de ré-échantillonnage basée sur la simulation de Monte Carlo pour simuler 10 000 groupes expérimentaux (à partir de sous-échantillons du groupe expérimental) comportant autant d'individus que le groupe contrôle.

Plus précisément, la démarche consiste à réaliser 10 000 tirages aléatoires indépendants : à chaque itération, cinq étudiant-es sont sélectionné-es au hasard, sans remise, parmi l'ensemble du groupe expérimental. Pour chaque groupe simulé ainsi constitué, nous calculons la couverture temporelle moyenne des *commits* selon la méthode précédemment définie. Ce procédé permet de produire une distribution empirique de la métrique, correspondant aux valeurs que l'on pour-

rait raisonnablement attendre pour des groupes de taille identique au groupe contrôle, issus du groupe expérimental.

Cette approche, fondée sur la simulation répétée de groupes aléatoires, permet d'estimer la variabilité intrinsèque de la couverture temporelle dans l'expérience, en tenant compte des fluctuations dues à la taille réduite du groupe contrôle. Il devient alors possible de comparer le score effectivement observé dans le groupe contrôle à la distribution simulée, afin d'en évaluer la représentativité et la significativité.

Le résultat, présenté dans la figure 9.8, met en évidence la proportion d'échantillons simulés (les barres bleues) obtenant une couverture égale ou inférieure à celle du groupe contrôle (le trait pointillé vertical rouge), fournissant ainsi une estimation empirique de la significativité de l'écart observé entre les deux groupes.

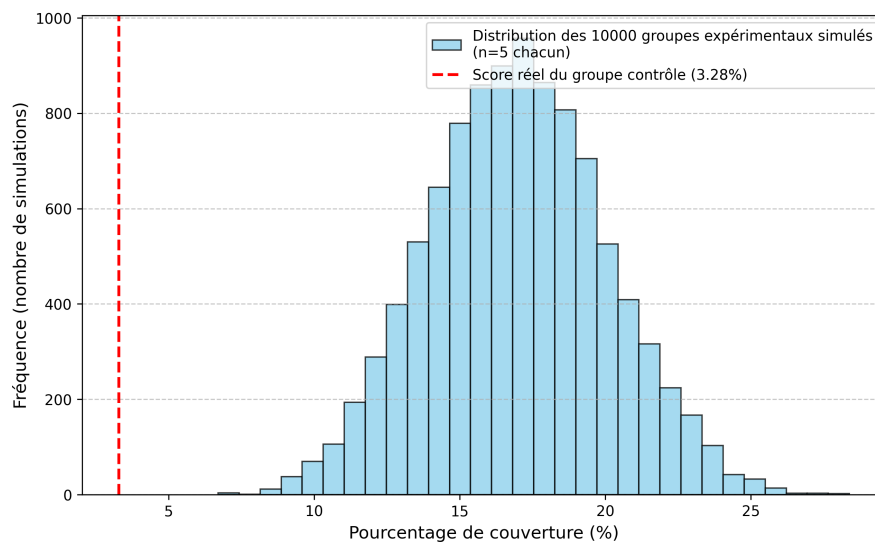


FIGURE 9.8 – Distribution de la couverture temporelle pour 10 000 groupes expérimentaux simulés (n=5) en comparaison du score du groupe contrôle

On constate ainsi qu'aucun des groupes simulés n'atteint le score obtenu par le groupe contrôle en matière de couverture temporelle. Ce constat corrobore les résultats du test de Mann-Whitney U présentés précédemment, et renforce l'idée que la faible couverture observée dans le groupe contrôle n'est pas un simple effet aléatoire ni une légère sous-performance, mais bien un résultat exceptionnellement bas au regard de la variabilité attendue parmi les utilisateurs et utilisatrices de LAWG. Cela souligne, une fois encore, l'apport significatif de l'outil pour obtenir des traces qui portent collectivement plus d'information sur la chronologie des modifications, ainsi que des traces plus continues des activités étudiantes.

9.1.1.3 *Exactitude*

La dimension d'*exactitude* d'une donnée, telle que définie par le cadre de Wang et Strong [119], renvoie au degré de conformité de la donnée à la réalité ou à une valeur de référence considérée comme correcte. Selon ces auteurs, l'exactitude est une propriété intrinsèque de la qualité des données, qui se mesure par la capacité de la donnée à représenter fidèlement l'objet, l'événement ou le concept auquel elle se rapporte. Une donnée exacte est donc exempte d'erreurs et reflète précisément l'information attendue, ce qui est essentiel pour garantir la fiabilité des analyses et des décisions qui en découlent. Dans le contexte de la traçabilité des activités étudiantes, l'exactitude des traces collectées conditionne la validité des interprétations sur les stratégies d'apprentissage et l'efficacité des outils mis en place.

Dans notre cas, la donnée est composée d'un message la caractérisant, ainsi que d'un ensemble de modifications de fichiers. L'intégrité de ce dernier est garantie par la technologie de gestion de version, ici Git. Le message du *commit*, en revanche, est défini par l'auteur ou l'autrice du *commit* et est donc sujet à des erreurs de frappe, de mauvais choix de mots, etc (comme nous avons pu le montrer dans le chapitre 5).

Pour l'analyse des erreurs contenues dans les messages de *commit* pour les deux groupes, nous avons considéré l'ensemble des *commits* réalisés, puis nous avons exclu tous les messages dits "attendus". Ces messages attendus comprenaient d'une part les messages strictement formatés du type "ITER_{i}" pour *i* allant de 1 à 10, et d'autre part des formulations routinières telles que "add deadline", "Initial commit", "Finish", "Resume", "Pause", ou encore des messages systématiques de type "Created", "Modified", "Deleted", "Moved".

Après cette étape de filtrage, aussi bien pour le groupe contrôle que pour le groupe expérimental utilisant LAWG, il restait environ une dizaine de messages de *commit* uniques par groupe à analyser, qui sont les suivants :

Groupe contrôle	Groupe expérimental
Commit fix mal formaté ITER_4-5	Commits fix mal formatés ITER 10 Iter 8 (la bonne version)
Commits intermédiaires ITER1.2 REFRACTOR create whoami.txt FINAL-test4 Merge branch 'master' ... ajout fichier	Commits intermédiaires Create Nom_étudiant.txt Update Welcome.java Update WelcomeTest.java test, test1 Merge branch 'master' ...

Dans le groupe expérimental, on trouve très peu de messages de *commit* mal formatés, à l'exception d'un cas particulier. Il s'agit d'un

étudiant qui n'a plus utilisé LAWG pour la résolution des dernières questions et dont deux *commits* sont mal formatés ("ITER 10" et "Iter 8 (la bonne version)"). C'est un cas intéressant qui illustre typiquement les atteintes à l'exactitude des données quand on laisse la liberté à l'étudiant-e de définir le contenu d'une donnée, ici le message de *commit*. Tous ses *commits fix* précédents sont correctement formatés, de même que tous les *commits fix* de ses camarades du groupe expérimental. On observe également d'autres *commits* intermédiaires qui n'ont pas été créés par LAWG mais par les étudiant-es elles-mêmes et eux-mêmes, qui peuvent donc avoir un format imprévisible.

Le même constat peut être fait pour le groupe contrôle, concernant les *commits* intermédiaires. Proportionnellement au nombre de *commits* du groupe, on observe une quantité plus grande de *commits* intermédiaires non formatés dans le groupe contrôle (à peu près autant en nombre absolu). Toutefois, il convient de noter la présence de quelques *commits* avec des messages difficilement interprétables automatiquement. Ces messages résultent d'une volonté des étudiant-es d'indiquer des étapes intermédiaires dans leur progression ("ITER1.2" qui est une étape intermédiaire entre deux questions résolues dont le message est semblable à un message de *commit fix*, et "ITER_4-5" qui est un message de *commit fix* mal formaté doublé d'une reprise de la question précédente).

En conclusion, on n'observe pas vraiment d'amélioration significative du point de vue de l'exactitude des traces générées par l'étudiant-e dans le groupe utilisant LAWG. Toutefois, on a beaucoup plus de données générées automatiquement et donc formatées, et donc la proportion des traces bien formatées et sans erreurs est plus grande grâce à LAWG.

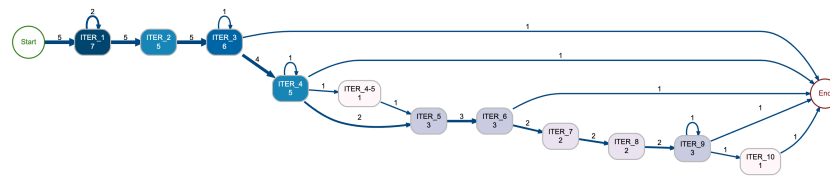
9.1.1.4 Cohérence

La cohérence, telle que définie par WANG et STRONG, se réfère à l'absence de contradictions internes entre les valeurs qui composent les données et au respect des règles formalisées, notamment celles qui traduisent la logique des processus métier sous-jacents. Dès lors, la cohérence ne concerne pas uniquement la qualité intrinsèque du contenu, mais implique également la conformité aux procédures et standards établis au sein du système. Dans le cadre de notre étude, qui porte sur le suivi des *commits* réalisés par des étudiant-es dans la réalisation linéaire d'une feuille d'exercices, cette dimension se traduit par la nécessité de respecter la règle consistant à consigner chaque résolution d'exercice par un *commit* identifié. L'omission de cette action, notamment lorsqu'un-e étudiant-e passe directement d'une question 1 à une question 2 sans enregistrer le progrès de la question 1, génère une incohérence structurelle qui compromet la fiabilité de la traçabilité et l'efficacité du suivi individuel et de groupe de la part de l'enseignant-e.

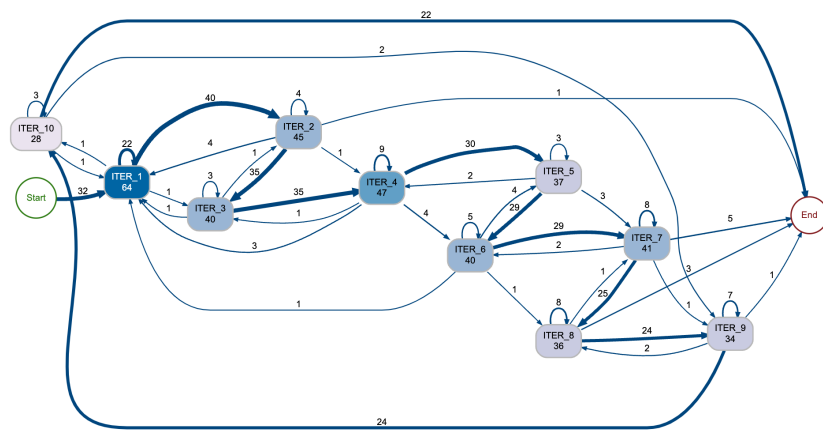
Pour évaluer la présence d'incohérences ou de transitions chaotiques dans la résolution des questions par les étudiant-es (comme les sauts de questions ou l'ordre non linéaire), nous avons appliqué la même démarche de PM que dans les analyses du chapitre 5.

Pour cela, nous avons filtré, dans chaque groupe, uniquement les *commits* correspondant à la résolution effective d'une question, c'est-à-dire ceux dont le message suit le format `ITER_...`. Cette approche permet de ne conserver que les traces pertinentes pour observer la progression question par question, en excluant les *commits* intermédiaires et générés automatiquement.

L'analyse de ces sous-ensembles permet alors de générer les cartes de processus et les explorateurs de traces illustrés à la figure 9.9.



(a) Carte de processus construite avec le groupe contrôle (sans LAWG)



(b) Carte de processus construite avec le groupe expérimental (LAWG)

FIGURE 9.9 – Cartes de processus issues de l'analyse des traces selon l'utilisation ou non de LAWG

Le groupe contrôle présente une carte de processus (cf. figure 9.9a) très séquentielle et linéaire. Le "chemin heureux" (la séquence la plus fréquente) est une progression directe sans retour en arrière. Les seules déviations notables sont des boucles (par ex., `ITER_1` sur elle-même 2 fois, `ITER_3` 1 fois), et on retrouve le *commit* que l'on a vu dans la section précédente dont le message n'est pas bien formaté `ITER_4-5`. L'absence totale de transitions "en arrière" (par exemple, de `ITER_4` à `ITER_2`) suggère que les étudiant-es complètent chaque itération

comme une tâche complète avant de passer à la suivante, sans revenir dessus.

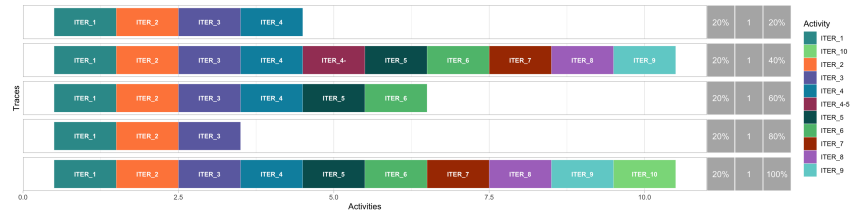
En revanche, le groupe expérimental présente un processus significativement plus complexe, long et pas tout à fait linéaire. La carte de processus (cf. figure 9.9b) illustre cette complexité. Elle est dominée par trois phénomènes absents du groupe témoin :

- **Augmentation importante des boucles** : On observe un grand nombre de répétitions de *commits* consécutifs sur la même question, comme par exemple ITER_1 répétée 22 fois, ITER_4 répétée 9 fois, etc. Cela reflète des allers-retours fréquents sur une même étape pour corriger d'une façon ou d'une autre le travail qui vient d'être fait.
- **Apparition plus fréquente des rétro-transitions** : Les étudiant-es reviennent régulièrement sur des questions précédentes, à travers des transitions comme de ITER_2 vers ITER_1 (4 fois), de ITER_4 vers ITER_1 (3 fois), ou encore de ITER_6 vers ITER_5 (4 fois). Ce comportement illustre la tendance à réviser (*refactoring*) ou corriger des réponses antérieures.
- **Présence marquée de transitions "sautantes"** : On remarque également des sauts d'une question à une autre non consécutive, par exemple le passage direct de ITER_4 à ITER_6 (4 fois). Ces transitions signalent des parcours non linéaires et peuvent représenter des oublis de consigner des étapes intermédiaires.

Si nous regardons une version de cette carte de processus avec non plus des valeurs absolues mais des valeurs relatives au nombre de cas différents passant par chaque transition, on s'aperçoit que peu d'étudiant-es sont concerné-es par ces trois phénomènes (environ 15.6% pour les boucles, 5% pour les rétro-transitions, 4% pour les transitions sautantes). Par contre, ces phénomènes sont présents sur quasiment toutes les questions contrairement au groupe contrôle.

Pour analyser plus finement la dynamique des séquences d'activités à l'échelle individuelle, nous avons généré des *explorateurs de traces* (*trace explorer*), où chaque séquence correspond au chemin type suivi par un-e étudiant-e. Afin de limiter la complexité du graphe et de faciliter l'interprétation, nous avons regroupé les événements consécutifs identiques. Autrement dit, lorsqu'une même question est résolue plusieurs fois de suite (par exemple, ITER_1, ITER_1, ITER_1), ces occurrences sont fusionnées en une seule étape. Cette agrégation permet de réduire le nombre de traces distinctes et ainsi d'obtenir une représentation plus lisible des parcours effectués.

La figure 9.10 présente le résultat de cette démarche pour chacun des deux groupes, représentant l'intégralité des traces observées dans les données.



(a) Explorateur de traces avec le groupe contrôle (sans LAWG)



(b) Explorateur de traces avec le groupe expérimental (LAWG)

FIGURE 9.10 – Explorateur de traces pour les deux groupes

Sans surprise, l'explorateur de traces pour le groupe contrôle (cf. figure 9.10a) montre cinq traces distinctes qui suivent une progression linéaire, commençant par ITER_1 et avançant dans l'ordre attendu (ITER_2, ITER_3, etc.).

Pour le groupe expérimental, comme on peut s'y attendre également, c'est un peu plus compliqué. L'explorateur de traces (cf. figure 9.10b) affiche bien plus de traces type possibles. Cela est dû à la grande divergence de cas possibles causée par les trois phénomènes expliqués précédemment. On remarque tout de même qu'exactement 50% des étudiant-es ont suivi un processus parfaitement linéaire et ont résolu toutes les questions, de ITER_1 à ITER_10. Cinq étudiant-es supplémentaires ont suivi ce processus linéaire mais que jusqu'à la question 7, ce qui nous amène à 65.62%.

Enfin, on s'aperçoit que le reste des traces présentent des transitions relativement chaotiques, dont certaines franchement divergentes, avec plus de 30 instances d'activités, donc de résolution de question. On remarque que près de 15.6% des étudiant-es ont résolu des questions plus de 10 fois.

En résumé, l'analyse des cartes de processus et des explorateurs de traces met en évidence des différences notables de structure et de diversité des traces collectées entre les deux groupes. Le groupe contrôle présente des traces exclusivement linéaires et homogènes, tandis que celles du groupe expérimental sont plus variées et présentent davantage de boucles et de transitions non linéaires. Ces résultats ne nous permettent pas de conclure que LAWG améliore les données générées du point de vue de la cohérence, et indiquent même une tendance

inverse, avec une plus grande divergence des traces observées par rapport aux traces attendues dans le groupe expérimental.

Au regard de l'ensemble des quatre dimensions évaluées, LAWG apporte une amélioration sur l'actualité et la complétude des traces, maintient un niveau comparable d'exactitude, et produit des résultats plus nuancés sur la cohérence. Ces indicateurs confirment que l'automatisation de la génération de traces enrichit les données disponibles pour l'analyse, même si elle fait émerger une complexité qui reflète davantage la réalité des processus de travail des étudiant-es.

9.1.2 *Identification du Test Driven Development (TDD)*

9.1.2.1 *Présentation du Test Driven Development (TDD)*

La définition précise du TDD ne fait pas complètement consensus dans la littérature ou dans l'industrie, notamment concernant l'ordre exact de certaines étapes ou la rigueur de leur enchaînement. Les variantes du TDD existent, en particulier autour du degré d'alternance entre développement du test et du code (avec des cycles de granularité variable) ou encore de l'obligation de toujours commencer par un test [9, 53].

Dans la suite, et afin de clarifier notre démarche, nous considérons la version stricte du TDD, à savoir celle où chaque itération commence systématiquement par la rédaction/modification d'un test (qui échoue), suivie de l'écriture ou la modification du code fonctionnel pour faire passer ce test, puis d'une étape de refactoring éventuel. C'est cette version du TDD qui a été enseignée aux étudiant-es qui ont participé à l'expérimentation. Lorsque nous parlerons de "suivi du TDD" ou "TDD" dans la suite, nous ferons donc référence à ce schéma strict, couramment qualifié d'approche "*test-first*".

Grâce aux traces générées avec LAWG, il est possible d'identifier certaines de ces phases : la phase d'écriture du test, la phase d'écriture du code pour faire passer le test, et le marquage de la question comme résolue.

9.1.2.2 *Description du prétraitement et de l'analyse*

Afin de préparer le jeu de données brutes pour l'analyse de PM, une phase de prétraitement a été nécessaire. Les traces de *commit* Git, dans leur forme originelle, ne sont pas directement exploitables en tant que journaux d'événements et requièrent une transformation pour définir les attributs clés du processus, notamment l'identifiant de cas, l'activité et l'horodatage.

Dans un premier temps, il a été nécessaire de convertir les messages de *commit*, qualitatifs (bien que générés par LAWG, ils comportent des parties variables avec le nom du fichier qui est modifié), en un

attribut catégoriel qui servira d'activité, que nous appellerons le type du *commit*.

Pour ce faire, une fonction de classification applique un ensemble de règles pour assigner à chaque *commit* un type d'activité. La logique d'extraction pour capturer les intentions de développement des étudiant·es est la suivante :

- **Identification des phases des sessions de travail** : Les événements de cycle de vie des sessions de travail ("Resume", "Pause", et "Finish") sont identifiés par correspondance exacte.
- **Identification des phases de modification de fichiers** : Une expression régulière est appliquée pour capturer les activités de développement via la modification de fichiers. Cette expression permet de classer les messages débutant par "Fix" comme une activité "fix". Simultanément, elle extrait le contenu textuel des balises de modification de fichier (par exemple, [deleted], [modified], [created]) pour l'utiliser directement comme nom d'activité.
- **Qualification TDD** : Point crucial pour la détection du TDD dit "test-first". Si le message de *commit*, quel que soit son type, se termine par le suffixe "Test.java", le qualificatif " test" est ajouté à l'activité identifiée (par exemple, "deleted test", "modified test"). Cette étape permet de différencier le travail sur le code de test de celui sur le code fonctionnel.

Les messages ne correspondant à aucune de ces règles (par exemple, "Initial commit", "WIP") ne se voient assigner aucun type et sont considérés comme du bruit.

Nous avons ensuite relié chaque *commit* à la question qu'il contribue à résoudre, en renseignant cette information dans une colonne. Pour cela, nous avons propagé la question associée à chaque événement "Fix" aux événements précédents, après un tri des données par auteur et autrice et par date. De plus, l'activité "Finish" a été explicitement assignée à sa propre catégorie de question avant la propagation, afin d'éviter une assignation incorrecte au dernier cas de résolution.

La définition de l'identifiant de cas (*case_id*) est particulièrement importante car utiliser uniquement l'auteur ou l'autrice du *commit* aurait été une erreur, notre objectif étant d'analyser le micro-processus de résolution de chaque question pour voir si le TDD est suivi dans chaque cas, et non pas le processus global tout au long du projet. Un "cas" est donc défini comme la séquence d'activités réalisée par un·e étudiant·e pour résoudre une question donnée. Pour ce faire, nous avons concaténé les identifiants des auteurs et autrices avec l'identifiant de la question traitée. Par exemple "o_ITER_1" pour l'étudiant·e o qui est en train de résoudre la question 1.

Dans un second temps, les événements non pertinents pour l'analyse ont été filtrés.

Nous avons ainsi pu analyser les comportements des étudiant-es sous différents angles, tant au niveau global qu'individuel, pour observer un comportement complexe, le suivi du TDD. Cette approche analytique a permis de faire émerger les tendances principales, discutées dans les sections qui suivent.

9.1.2.3 Analyse globale des comportements étudiants par Process Mining (PM)

La figure 9.11 présente la carte de processus (*relative-case*) du processus de résolution de question pour le groupe expérimental. Les fréquences associées aux transitions entre les différentes activités représentent la fréquence des cas (donc des couples étudiant-e-question) qui ont suivi ces transitions.

On observe que la majorité des cas passent par une boucle importante : les transitions entre "modified" et "modified test" affichent des fréquences élevées (72.64% et 83.02% dans un sens puis dans l'autre), indiquant une alternance fréquente entre la modification du code et la modification des tests. Cependant, on constate que seulement 50.94% des résolutions de questions commencent par la modification d'un fichier de test, contre 40.57% par la modification du code fonctionnel.

Autrement dit, le comportement de suivi strict du TDD, consistant à débiter chaque résolution par un test, n'est donc pas généralisé à l'ensemble des étudiant-es. Toutefois, l'alternance notable entre travail sur les tests et sur le code suggère que l'esprit du TDD consistant à écrire systématiquement les tests associés est présent, même si la démarche "test-first" n'est pas suivie dans tous les cas avec rigueur.

Les transitions allant vers "Finish" sont des cas à part après la résolution de la dernière question, il est donc normal de voir une part résiduelle de transitions vers "Finish".

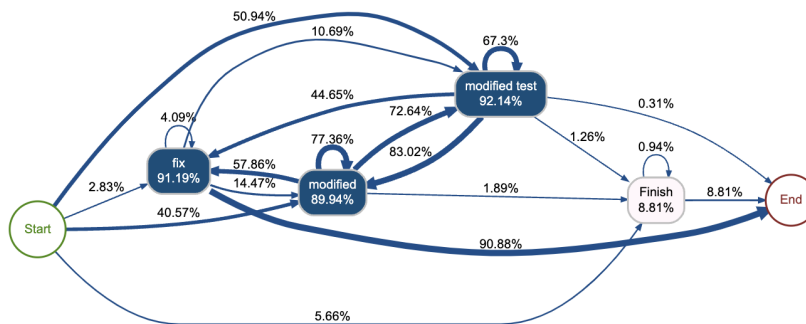


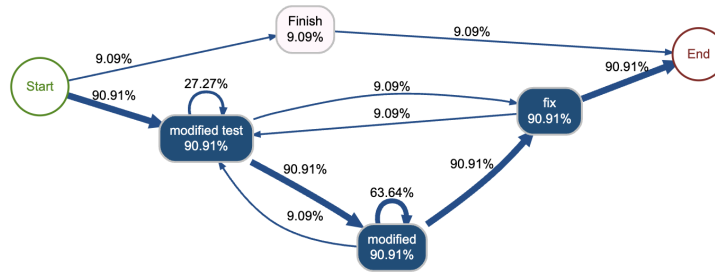
FIGURE 9.11 – Carte de processus (*relative-case*) représentant le processus de résolution de question pour le groupe expérimental avec LAWG

Dans la suite de notre analyse, nous nous sommes intéressés à des cas particuliers illustrant la diversité des comportements, du suivi strict du TDD à des approches plus flexibles. Cette démarche vise à illustrer concrètement la pluralité des pratiques repérées par LAWG.

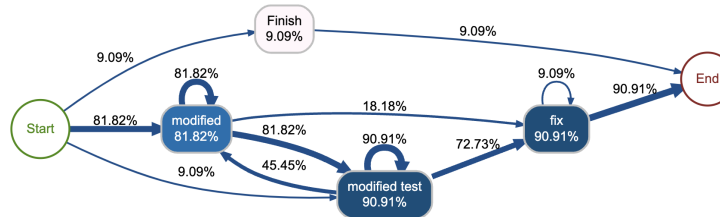
9.1.2.4 *Résultats individuels typiques concernant le suivi du Test Driven Development (TDD)*

Les cartes de processus de la figure 9.12 montrent deux cas typiques de suivi du TDD : un·e étudiant·e qui suit typiquement bien le TDD (figure 9.12a) et un·e étudiant·e ne suivant pas strictement le TDD en commençant par la modification d'un fichier de code fonctionnel (figure 9.12b). Dans le premier cas, toutes les résolutions de question débutent par la modification d'un fichier de test (un cas commençant par "Finish" ne correspondant pas à une résolution normale). Cela correspond à la démarche stricte du TDD : démarrer systématiquement chaque itération par l'écriture ou la modification d'un test. On note également que, dans la majorité des cas, l'étudiant·e enchaîne sur le code fonctionnel sans revenir sur le test, puis marque la question comme résolue.

Le second cas montre un·e étudiant·e qui ne suit pas l'ordre attendu du TDD strict : dans 81.82% des cas, la résolution commence par la modification du code fonctionnel, contournant la phase "test-first". L'alternance entre modification du code fonctionnel et modification de test est plus prononcée (dans 45.45% des cas, du code est à nouveau modifié après intervention sur le test), ce qui peut s'expliquer par une vérification a posteriori des tests, qui ne passent pas, nécessitant une adaptation du code face aux tests. Fait notable, dans 72.73% des cas, la résolution de la question se termine par une modification de fichier de test, ce qui s'éloigne de la démarche "test-first".



(a) Carte de processus (relative-case) pour un-e étudiant-e adoptant strictement le TDD.



(b) Carte de processus (relative-case) pour un-e étudiant-e ne suivant pas le TDD strict.

FIGURE 9.12 – Comparaison des cartes de processus illustrant l'application (haut) ou non (bas) du TDD dans les traces du groupe expérimental.

En examinant les données plus finement, nous avons trouvé un comportement intermédiaire. La figure 9.13 illustre cette situation. La majorité des résolutions de question commencent bien par un test, mais l'étudiant-e effectue ensuite des allers-retours fréquents entre test et code fonctionnel, ce qui traduit un ajustement, un tatonnement vers la solution qu'on ne retrouve pas dans le cas de la figure 9.12a, qui procédait de façon très linéaire, sans boucler sur les tests.

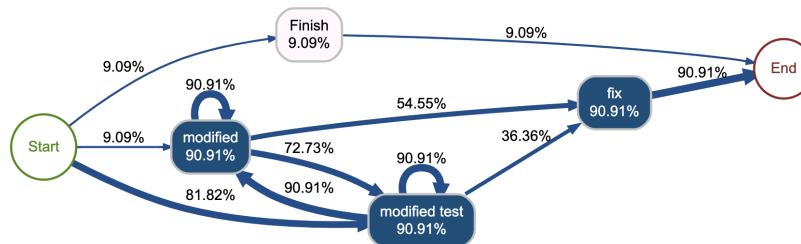


FIGURE 9.13 – Carte de processus (relative-case) illustrant un-e étudiant-e suivant le TDD avec un comportement typiquement itératif.

Pour étudier la chronologie exacte des activités dans ce type de cas, l'explorateur de traces de la figure 9.14 permet d'observer les séquences réelles. Afin de limiter la taille des séquences, longues à cause des nombreuses auto-boucles (produites par les *commits* automatiques de LAWG), nous avons agrégé ces occurrences répétées. Autrement dit,

une instance d'une activité peut en cacher une ou plusieurs autres dans cette visualisation.

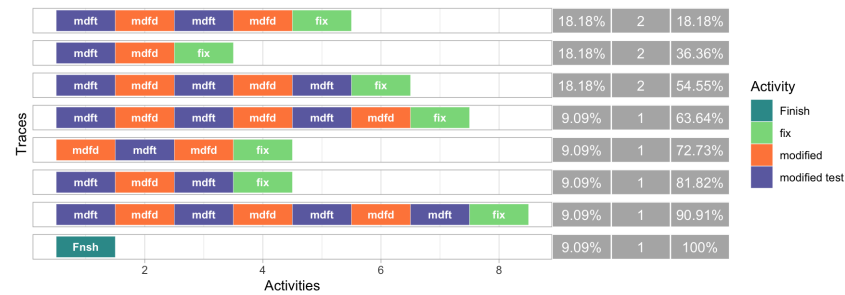


FIGURE 9.14 – Explorateur de traces illustrant un-e étudiant-e suivant le TDD mais avec une progression plus incertaine.

On remarque ici que l'étudiant-e alterne plusieurs fois entre modification du test et du code fonctionnel, entre 2 et 4 fois. On suppose qu'elle ou il était beaucoup plus hésitant concernant la solution à développer et les tests à écrire, contrairement au cas de la figure 9.12a où l'étudiant-e semblait avoir une solution claire en tête et écrivait les tests, puis le code fonctionnel d'une seule traite.

9.1.2.5 Adéquation entre la fréquence de suivi du Test Driven Development (TDD) et la note obtenue

Enfin, nous nous sommes demandés s'il existait un lien entre suivre scrupuleusement le TDD avec une approche "test-first" et la note obtenue par l'étudiant-e.

Pour cela, nous avons calculé la fréquence de suivi du TDD "test-first" pour chaque étudiant-e.

Avant d'étudier la relation entre la fréquence d'application de la démarche TDD ("test-first") et la note obtenue par les étudiant-es, nous avons d'abord examiné la distribution de ces deux variables afin de choisir une méthode d'analyse statistique appropriée. Le test de Shapiro-Wilk révèle que la fréquence de suivi du TDD suit une distribution normale ($p = 0,31 > 0,05$), contrairement aux notes qui s'écartent significativement de la normalité ($p < 0,001$).

Afin de compléter cette première vérification, nous avons également examiné la distribution des résidus issus d'une régression linéaire entre les deux variables. Ici encore, le test de normalité ($p = 0,001016$) mène au rejet de l'hypothèse nulle, ce qui confirme que les résidus ne sont pas normalement distribués.

Au vu des résultats de ces tests, et de la petite taille d'échantillon, il est recommandé de recourir à une méthode non paramétrique pour évaluer la corrélation entre les deux variables [22]. Nous avons donc opté pour le test de Spearman, qui ne requiert pas l'hypothèse de normalité et s'avère particulièrement adapté à l'analyse de relations

Ce qui équivaut à la fréquence des transitions entre "Start" et l'activité "modified test" quand on regarde la carte de processus (relative-case) du point de vue d'un-e étudiant-e, comme sur la figure 9.13.

entre variables continues ne respectant pas les conditions d'application des méthodes paramétriques classiques.

Pour évaluer si l'adoption du TDD (mesurée par la proportion de résolutions de questions commençant par une modification de test) est associée à la performance académique (la note obtenue au TP par l'étudiant-e), nous avons testé l'hypothèse nulle (H_0) d'absence de corrélation monotone $H_0 : \rho = 0$ contre l'hypothèse alternative (H_1), à savoir l'existence d'une corrélation monotone positive attendue $H_1 : \rho > 0$.

La figure 9.15 montre la relation entre la fréquence de suivi du TDD en début de résolution de question et la note obtenue par les étudiant-es. Le coefficient de corrélation des rangs de Spearman obtenu est relativement faible et négatif ($\rho = -0,16$), tout comme le montre la pente de la droite de régression de la figure 9.15. La valeur associée du test de significativité indique que la corrélation n'est pas statistiquement significative ($p = 0,38$) (comme le montre la dispersion des points autour de la droite de régression en dehors de l'intervalle de confiance à 95%), ce qui suggère que l'hypothèse H_0 n'est pas invalidée et qu'aucune association monotone notable n'existe dans cet échantillon entre la proportion d'activités "test-first" en début de résolution de question et la note finale au TP. L'analyse révèle ainsi que, sur ce jeu de données, aucun lien robuste n'apparaît entre l'adoption du TDD en situation réelle et la performance académique.

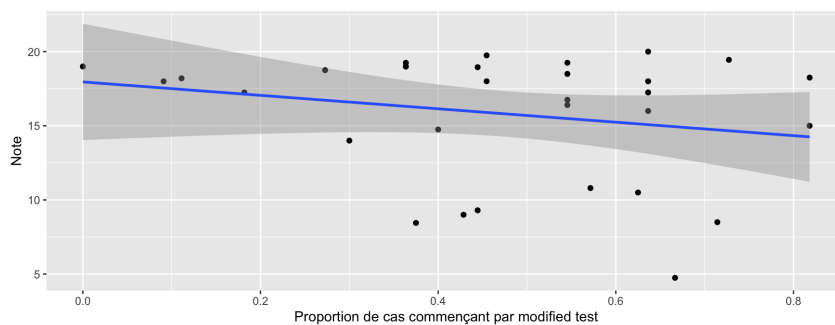


FIGURE 9.15 – Relation entre la fréquence de suivi du TDD et la note obtenue par les étudiant-es.

Il est à noter que les étudiant-es des trois cas étudiés précédemment ont tous et toutes obtenu une note au TP supérieure ou égale à 18.

9.1.3 Formulaire SUS et retours des étudiant·es

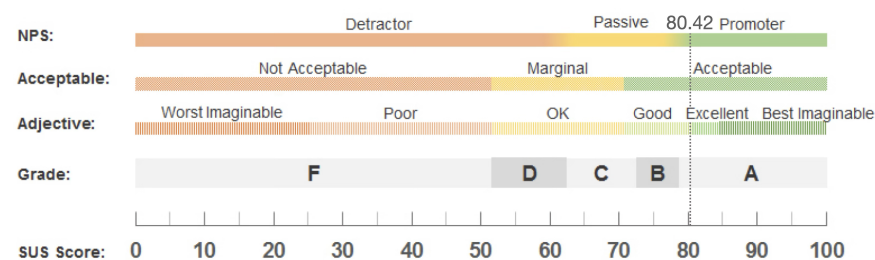


FIGURE 9.16 – Calcul du score moyen SUS (80.42) pour LAWG

Le formulaire SUS a été rempli par 18 étudiant·es. Le score moyen est de 80.42 et l'écart-type est de 8.80. D'après le score obtenu, le système est considéré comme "bon" selon [6], ou "acceptable" selon [7].

Nous avons eu 6 retours d'expériences de la part des étudiant·es. Ces retours sont plus mitigés que nos résultats sur l'utilisabilité. 2 étudiant·es font remonter que le fonctionnement de LAWG n'est pas très clair. D'abord, parce qu'on ne comprend pas bien ce que font les fonctionnalités de l'outil, et à quels dépôts il a accès. Ensuite, parce que l'interface ne permettrait pas clairement de voir si une action a bien été prise en compte. Sur ce point, l'étudiant·e fait référence au passage d'une question à une autre qui est rapide et sans *feedback*. D'autres retours font état d'une prise en main nécessaire de LAWG qui prendrait autant de temps qu'apprendre à utiliser Git. L'étudiant·e précise que le passage de Git à LAWG « peut être perturbant » et qu'elle ou il a plus confiance en Git qu'en LAWG.

Des retours positifs ont également été exprimés par les étudiant·es. L'un·e met en avant l'intérêt du concept, et un·e autre souligne que le logiciel est agréable à utiliser et qu'il facilite le marquage des questions, comparativement à l'utilisation de Git.

En dehors des retours écrits via le formulaire, nous avons aussi pu avoir des retours pendant et à la fin de la séance. Ces retours sont à prendre avec précaution car ils ne sont pas anonymes et sont donc sujets à des biais importants, même s'il s'agissait de retours spontanés. Plusieurs étudiant·es ont souligné que l'utilisation de LAWG leur paraissait plus simple que l'utilisation directe des commandes Git. Cette perception est notamment rapportée par un·e étudiant·e considéré·e comme bon·ne qui relève que la charge cognitive requise pour réaliser les actions avec LAWG est moindre que celle associée à l'utilisation de Git en ligne de commande. La simplification des interactions offertes par l'outil, en réduisant le nombre d'étapes, de choix ou de notions techniques à maîtriser, semble ainsi alléger la charge mentale lors de la réalisation des activités, rendant le processus

Plan d'interview post-expérimentation

Avez-vous eu des remarques sur LAWG et/ou la méthode de marquage de la progression émanant des étudiants ?

Avez-vous des remarques personnelles ?

SI PAS TROP DE RÉPONSE

Les étudiants participant à l'expérimentation ont-ils eu des difficultés à comprendre le fonctionnement / l'état courant / l'interface de LAWG ?

Les étudiants participant à l'expérimentation ont-ils eu des problèmes avec LAWG ?

- 1 appel ou moins : (1,2,... / une minorité / la moitié / une majorité)
- Plusieurs appels :
- Soucis entraînant l'abandon de LAWG :

Avez-vous eu des remarques (voire des refus d'utiliser LAWG) concernant des considérations éthiques (accès des données, tracking, peur liée à l'utilisation du système) ?

Avez-vous eu des remarques concernant "l'étrangeté" des questions posées dans LAWG ?

Si vous avez déjà fait ces TP avant, avez-vous ressenti que l'utilisation de LAWG a facilité la séance comparé à l'utilisation de Git ?

Pensez-vous que l'interface de LAWG est satisfaisante pour des étudiants de ce niveau d'étude ?

Avez-vous des idées d'amélioration de LAWG ?

La mise en place de LAWG vous a-t-elle paru compliqué et/ou complexe (sujet, configuration de LAWG et dépôt) ?

FIGURE 9.17 – Plan suivi pour les entretiens semi-directifs avec les enseignants

plus accessible et moins source de distraction pour l'ensemble des utilisateurs et utilisatrices.

9.1.4 Entretiens avec les enseignants

Pour avoir des retours sur l'utilisation de LAWG au sein de tous les groupes de TP, nous avons organisé des entretiens semi-directifs avec les enseignants. Comme il est possible de le voir dans la figure 9.17, l'objectif était de fournir à chaque enseignant un espace ouvert pour parler de son expérience, mais un ensemble de questions plus ciblées étaient prêtes au cas où l'enseignant ne savait pas par où commencer. Trois entretiens ont été menés, tous les enseignants ayant accepté d'être enregistrés.

Nous avons ensuite effectué une analyse de contenu thématique [8] des entretiens. Pour faire émerger des thèmes plus facilement, nous avons utilisé un codage inductif [12], en définissant des codes sur la base des retranscriptions des entretiens. Les thèmes que nous avons identifiés sont :

- Expérience utilisateur des étudiant·es
- Adhésion des étudiant·es à LAWG

	Enseignant 1 (E1)	Enseignant 2 (E2)	Enseignant 3 (E3)
Expérience Utilisateur des Étudiant-es	La majorité des étudiant-es n'ont pas eu de problème technique concernant LAWG. Pas de problème pour reprendre la session de travail à la maison malgré des appréhensions. Quelques étudiants surpris à la fermeture de session.	Quelques questions au début sur le processus de marquage des questions comme résolues. Pas de problème technique concernant LAWG.	Aucun problème technique concernant LAWG. Le processus de marquage des questions comme résolues à l'aide de la commande "fix" était très simple. Beaucoup de doutes et d'incertitudes sur ce que fait LAWG, sur les résultats des commandes ou son état courant. Manque de feedback.
Adhésion des étudiant-es à LAWG	Une grande part des étudiant-es ont choisi de ne pas utiliser LAWG. Retours positifs parmi les étudiant-es qui ont utilisé LAWG. Simplification des commandes Git et délégation de la responsabilité du commit bien vus par les étudiant-es.		Une grande part des étudiant-es ont choisi de ne pas utiliser LAWG.
Impact sur le travail de l'enseignant-e	Dimension anxiogène quant à l'utilisation de LAWG car les commits sont cruciaux pour la correction. Mise en place de LAWG dans le cours existant perçue comme très simple et légère. Le volume des commits dans la branche principale devient problématique pour la correction. Dégradation qui est acceptable si elle est contrebalancée par des outils de monitoring. Vrai apport perçu concrètement via l'identification du cheminement intellectuel de l'étudiant-e impossible sans LAWG.	La mise en place de LAWG n'est pas trop compliquée. Le volume des commits dans la branche principale ainsi que l'atomicité des commits (en termes de modification) rend la correction plus difficile. Apport perçu de LAWG pour identifier les phases du TDD suivies par les étudiant-es.	Le volume des commits dans la branche principale devient problématique pour la correction. Suggestion de séparer les commits automatiques des commits de résolution de question produits par les étudiant-es. L'export des dépôts des étudiant-es devient assez lourd à cause de fichiers exécutables de LAWG. Mise en place de LAWG dans le cours existant perçue comme simple et naturel pour un-e enseignant-e en informatique.
Accessibilité de LAWG pour des étudiant-es de niveau licence informatique	LAWG serait plus apprécié des étudiant-es avec une interface graphique plus riche, qui regroupe entièrement Git. En l'état l'outil a l'avantage d'être très simple, avec peu de commandes.	Les notions de Personal Access Token ou de clé SSH pour l'authentification à GitHub peuvent susciter des questions de la part d'étudiant-es de ce niveau sans être insurmontable.	Les notions de Personal Access Token ou de clé SSH peuvent être un peu compliquées mais c'est intéressant de s'y confronter. La commande "fix" rend vraiment simple l'acte de faire "add, commit, push", autant qu'un Control-S.
Besoin d'apprendre à manipuler Git	Il y a un choix à faire autour de la conception de LAWG selon si l'apprentissage de Git est un objectif pédagogique ou non. Je voulais que les étudiant-es aient un premier contact avec Git et ses notions et l'état courant de LAWG réussi à aller dans ce sens car toute l'expérience Git n'est pas cachée.		Apprendre à utiliser Git est un objectif pédagogique et LAWG va un peu à l'encontre de cet objectif. Suggestion de quand même leur laisser faire "git add ; git commit ; git push".

FIGURE 9.18 – Matrice à condensés présentant la synthèse des propos de chaque enseignant pour chaque thème identifié lors de l'analyse de contenu des entretiens

- Impact sur le travail de l'enseignant-e
- Accessibilité de LAWG
- Besoin d'apprendre à manipuler Git

Dans la matrice à condensés présentée dans la figure 9.18, nous avons synthétisé les propos de chaque enseignant pour chaque thème identifié en prenant en compte chaque code regroupé dans un même thème.

Dans les prochaines sections, nous allons présenter les propos des enseignants concernant chaque thème.

9.1.4.1 L'expérience utilisateur des étudiant-es

L'expérience des étudiant-es avec LAWG est marquée par une contradiction entre la simplicité technique et l'incertitude fonctionnelle. Les trois enseignants s'accordent à dire qu'aucun problème technique majeur n'a été rencontré par les étudiant-es qui ont utilisé l'outil. Le processus de marquage via la commande `fix` est même jugé « très simple, très clair » (E3).

Cependant, cette simplicité entraîne peut-être un manque de *feedback* qui génère du doute et de l'anxiété. L'enseignant 3 rapporte que les étudiant-es, et même lui, avaient « un petit doute sur ce que [l'outil] fait, et notamment au moment du "fix", savoir est-ce qu'il a bien fait le commit, est-ce qu'il a bien fait le push ». Cette incertitude est une source d'inquiétude, particulièrement pour les étudiant-es travaillant sur des machines qui ne sont pas les leurs, craignant de perdre leur

travail car « s'ils perdaient tout le travail de la séance, ça pouvait être problématique » (E3).

Ce manque de retour d'information se retrouve lors de la fermeture de l'outil. L'enseignant 1 note une « stupéfaction » chez un étudiant lorsque les fichiers disparaissent du répertoire de travail, un événement qu'il qualifie lui-même de « violence symbolique qui est... qui m'a surprise, en fait » (E1). Il explique : « Le fait de voir le truc disparaître, c'est étrange comme sentiment [...] ton premier sentiment, [...] c'est que tu viens de perdre les 2h30 précédentes » (E1).

9.1.4.2 Adhésion des étudiant·es à LAWG

L'adoption de l'outil par les étudiant·es est modérée et semble guidée par un calcul bénéfice-risque. Les enseignants 1 et 3 rapportent qu'une partie des étudiant·es a fait le choix de ne pas participer. L'enseignant 1 évoque « ceux qui ont considéré que le bénéfice-risque était négatif », une décision potentiellement liée à une méfiance vis-à-vis de la stabilité du système ou à des craintes sur le suivi de leur activité, bien que l'enseignant 3 précise avoir « bien précisé ce que ça implique, ce que l'outil allait traquer ».

Pour celles et ceux qui ont participé, les retours sont majoritairement positifs. L'enseignant 1 souligne que la simplification des commandes Git a été appréciée, citant un retour d'étudiant pour qui « c'était plus facile de ne pas avoir à taper les "git commit", les "git push" ». Plus encore, le transfert de la responsabilité de la gestion de version de l'étudiant·e vers l'outil a été un facteur d'adhésion : « ça n'était plus leur responsabilité. Pour le coup. Ça devenait celle de l'outil. Et ça, c'était bien accueilli » (E1). L'outil est alors perçu comme un avantage qui allège la charge cognitive et permet de se concentrer sur le cœur de l'exercice.

9.1.4.3 Impact sur le travail de l'enseignant·e

Du point de vue des enseignants, l'impact de l'intégration de LAWG est ambivalent. D'un côté, la mise en place est perçue comme « vraiment simple » (E1) et « assez naturelle [...] pour un enseignant en informatique » (E3). Surtout, l'outil est reconnu comme un apport majeur pour l'analyse du travail des étudiant·es. Il permet de « voir le cheminement intellectuel qu'ils ont eu pour produire, pour répondre aux questions, ce qui est vachement intéressant [...] à laquelle tu n'as pas accès sinon » (E1).

D'un autre côté, LAWG complexifie significativement le processus de correction. Les trois enseignants s'accordent sur le fait que le volume de *commits* générés est un obstacle. Pour l'enseignant 3, les *commits* importants se retrouvent « vraiment noyés dans la masse des autres commits ». L'enseignant 2 précise que le problème n'est pas tant le nombre de commits que leur granularité : « les changements pour

chaque commit c'est trop petit. Si tu veux voir les changements de chaque question c'est un peu difficile ». Ce flux constant de micro-sauvegardes sur la branche principale rend l'historique difficile à lire et laisse le *commit* de résolution de la question pauvre en information. Aussi, selon l'enseignant 3, les exécutable contenus dans les dépôts alourdissent l'export des dépôts.

Enfin, l'enseignant 1 ajoute une dimension anxiogène personnelle, liée à la confiance nécessaire pour intégrer un outil externe sur une « ressource critique » comme les *commits*, nécessaires à l'évaluation : « l'introduction d'un outil sur un truc qui est aussi critique, c'est pas neutre comme décision ».

9.1.4.4 Accessibilité de LAWG

L'outil est unanimement jugé accessible pour des étudiant-es de niveau licence en informatique. Son fonctionnement en lignes de commande et ses rares commandes sont considérés comme simples. Pour l'enseignant 1, « l'interface, ce qu'elle réussit bien, c'est qu'il y a très peu de commandes, en fait. "fix"... "fix" et "quit" ».

Les seules difficultés potentielles identifiées concernent la configuration initiale, notamment « ces histoires de "personal access token", connexion par clé, SSH » (E3). L'enseignant 2 confirme avoir eu des questions à ce sujet en début de séance. Cependant, l'enseignant 3 nuance ce point en affirmant que « si jamais ils s'y confrontent, c'est un bon exercice aussi », transformant une difficulté technique en opportunité d'apprentissage.

Bien que l'interface actuelle soit fonctionnelle, l'enseignant 1 suggère qu'une interface graphique plus riche serait appréciée des étudiant-es : « Je pense qu'ils auraient préféré, qu'ils auraient été plus à l'aise avec un truc qui ressemble au client, au client Git fenêtré ».

9.1.4.5 Besoin d'apprendre à manipuler Git

Ce thème révèle un conflit pédagogique important. Les enseignants 1 et 3 insistent sur le fait que l'apprentissage de Git est un objectif pédagogique important. Or, en masquant la complexité des commandes (`git add`, `git commit`, `git push`), LAWG va à l'encontre de cet objectif.

L'enseignant 3 est très clair sur ce point : « Je trouve que c'est important pendant les séances qu'ils puissent expérimenter l'outil Git, apprendre, se confronter aux problèmes qu'il peut y avoir avec Git. Et du coup, l'outil empêchait un peu ça ».

L'enseignant 1 formule ce dilemme comme un choix de conception fondamental : « Si tu as envie qu'ils bossent la partie technique Git, par contre, plus LAWG fait des trucs sans te le dire, moins nous on progresse sur l'utilisation de Git ». L'enseignant 3 vit lui-même cette contradiction, reconnaissant que la commande `fix` est un confort appréciable (« c'est agréable parce que c'est hyper facile [...] c'est

presque aussi simple que de faire Contrôle-S ») tout en admettant que « ça va un peu à l'encontre de ce que j'ai dit sur l'utilisation de Git ». La simplicité offerte par l'outil, bien qu'appréciée, est perçue comme un frein à un apprentissage technique jugé essentiel.

Cette critique est toutefois nuancée par l'enseignant 1 lui-même. Il reconnaît que, puisque LAWG n'encapsule pas entièrement l'expérience Git (les étudiant·es devant toujours réaliser le `git clone` initial et gérer leur authentification), une première sensibilisation à l'écosystème Git a bien lieu. Cette approche partielle lui semble être un compromis pertinent pour le niveau des étudiant·es. Il estime que même si les commandes quotidiennes sont masquées, l'essentiel du concept est transmis : « Même s'ils ne font pas les "git add ; git commit ; git push", ils savent qu'il y a un dépôt, que c'est ça qui fait foi [...]. Même s'ils sont un peu petits [...] pour être capables de faire des gestes compliqués en Git, ils ont quand même fait un pas dans la direction ». Ainsi, sans former à la manipulation technique, l'outil introduirait tout de même la notion fondamentale de dépôt de code.

En synthèse, les entretiens font apparaître un tableau contrasté. Les enseignants reconnaissent la valeur analytique de LAWG et la simplicité de son intégration dans les dispositifs existants, tout en identifiant deux tensions principales : d'une part, le manque de *feedback* de l'outil génère de l'incertitude chez les étudiant·es ; d'autre part, l'abstraction des commandes Git, si elle allège la charge cognitive, entre en conflit avec l'objectif pédagogique de maîtrise du logiciel de gestion de version. Le volume élevé de traces produites est à la fois perçu comme une richesse pour l'analyse et comme un obstacle pour la correction, soulignant le besoin d'outils dédiés à l'exploitation de ces données.

L'ensemble des résultats présentés dans cette section dessine un bilan globalement positif pour LAWG. Sur le plan quantitatif, l'outil améliore la qualité des traces sur les dimensions d'actualité et de complétude, sans dégradation notable de l'exactitude, tandis que la cohérence présente un tableau plus nuancé. L'analyse par PM a permis d'identifier des comportements d'apprentissage complexes comme le suivi du TDD, révélant une diversité de profils parmi les étudiant·es. Sur le plan de l'utilisabilité, le score SUS situe LAWG dans la catégorie des systèmes jugés acceptables, bien que les retours qualitatifs des étudiant·es et des enseignant·es fassent émerger des points de vigilance concernant le *feedback*, la confiance et le compromis avec l'apprentissage de Git. La section suivante discute ces résultats au regard des questions de recherche.

9.2 DISCUSSION

9.2.1 Des nouvelles traces exploitables pour des nouveaux comportements

Les entretiens avec les enseignant-es confirment que LAWG a permis de collecter des traces d'une plus grande richesse et d'une valeur inaccessible sans l'outil. Si on regarde du point de vue de la **complétude** (*completeness*) et de la **valeur ajoutée** (*value-added*) des données, l'enseignant 1 affirme avoir pu observer des comportements qu'il n'aurait pu voir sans LAWG : « Ça permet de voir le cheminement intellectuel qu'ils ont eu pour produire, pour répondre aux questions, ce qui est vachement intéressant. [...] à laquelle tu n'as pas accès sinon ». Cette observation de l'enseignant est d'ailleurs corroborée par nos analyses dans les sections portant sur la **complétude** des traces et sur l'**identification des comportements TDD** (voir sections 9.1.1.2 et 9.1.2). Nous montrons en effet que l'utilisation de LAWG rend possibles de nouveaux types d'observations plus fines sur la progression réelle des étudiant-es, leur démarche pas à pas, ainsi que l'ordre précis dans lequel émergent les différentes phases (tests, code, corrections). Ce niveau de détail et cette richesse de données étaient inaccessibles dans le dispositif sans LAWG, confirmant bien le ressenti exprimé par l'enseignant.

Du point de vue de l'**exactitude** (*accuracy*) et de la **cohérence** (*consistency*) des données, l'enseignant 1 met en avant le transfert de responsabilité de l'étudiant-e vers l'outil : « Le fait de partir sur LAWG, ça n'était plus leur responsabilité. Pour le coup. Ça devenait celle de l'outil. Et ça, c'était bien accueilli ». Cela signifie que pour cet enseignant, l'outil apporte un avantage du point de vue de ces deux dimensions, que l'on peut rapprocher du concept de fiabilité des données.

Pour autant, l'analyse de la cohérence des traces a révélé un résultat plus mitigé : les traces sont parfaitement fiables quand issues du groupe contrôle, et elles le sont en majorité quand issues du groupe expérimental. On peut toutefois relever qu'il existe une faible part des étudiant-es qui ont suivi un processus un peu plus chaotique comme expliqué dans la section 9.1.1.4. Tout en rappelant que le groupe contrôle est probablement trop petit pour statuer sur un effet négatif de LAWG sur la dimension de cohérence des données, on émet l'hypothèse que l'utilisation de LAWG apporte une dimension plus stricte au processus de résolution des questions. De ce fait, les étudiant-es pourraient avoir plus tendance à systématiquement marquer à nouveau une question comme résolue après être revenu-es dessus, contrairement aux étudiant-es n'utilisant pas LAWG. Autrement dit, on suppose que là où les traces du groupe contrôle montrent une progression linéaire artificielle, celles du groupe expérimental, issues de LAWG, révèlent un peu mieux la réalité du travail de développement : les hésitations, les retours en arrière et les corrections. L'outil

ne rendrait pas les étudiant-es plus cohérent-es, il rendrait visible la complexité réelle de leur processus d'apprentissage. Malgré tout, la subsistance de "transitions sautantes" restent un problème du point de vue du suivi en temps réel de la progression des étudiant-es, mais l'utilisation de LAWG permet même dans ces cas-là de suivre finement le travail des étudiant-es sur les fichiers, et permettrait de retrouver le fil réel des événements, malgré une défaillance éventuelle de la part déclarative du processus de résolution de question.

Dans le cas étudié du suivi du TDD, l'analyse n'a pas mis en évidence de corrélation significative entre l'adoption du TDD dans sa forme stricte "test-first" et la performance académique mesurée par la note obtenue au TP. Ce résultat peut être interprété de manière cohérente au regard des critères d'évaluation : l'antériorité des tests par rapport au code ne fait pas partie des critères pris en compte dans la notation. Les critères d'évaluation portent principalement sur la qualité du code livré par l'étudiant-e. Dans cette perspective, l'analyse des données issues de LAWG met en lumière un décalage entre les attendus pédagogiques d'une formation (comme le respect du cycle TDD) et les critères effectivement mobilisés pour évaluer les apprentissages. Elle souligne ainsi la capacité de l'outil à révéler des non-alignements structurels entre objectifs, pratiques et dispositifs d'évaluation, dès lors qu'ils se manifestent dans les traces de modifications de fichiers.

Si l'on examine plus en détail cette absence de corrélation, une question demeure : pourquoi le suivi du TDD n'est-il pas associé à une meilleure note, alors que plusieurs travaux ont montré un lien entre TDD et amélioration de la qualité du code ? La qualité du code peut en effet être décomposée en deux dimensions : une qualité interne, centrée sur la structure et la maintenabilité du code, et une qualité externe, relative au comportement fonctionnel du logiciel. La littérature fait généralement état d'un effet positif du TDD sur la qualité externe du code, mais elle reste moins concluante quant à ses bénéfices sur la qualité interne [38, 92]. Ce constat peut s'expliquer par le fait que les développeurs et développeuses tendent souvent à négliger la phase de *refactoring*, pourtant essentielle à la qualité interne. Donc, l'ordre de l'écriture des tests ne semble pas être prépondérant dans l'amélioration de la qualité globale du code, au contraire de la phase de *refactoring*. Or, dans l'expérimentation nous n'avons pas pu identifier ces phases grâce à LAWG, et cet indicateur pourrait peut-être être plus corrélé à la note au TP.

Au-delà de la notation, l'outil s'avère être un instrument de diagnostic pédagogique intéressant. La capacité à distinguer des profils d'apprenant-es très différents (le "strict", le "non-strict", l'"hésitant") offre à l'enseignant-e une granularité d'analyse inédite. Cela permettrait d'envisager des interventions pédagogiques ciblées, par exemple pour rassurer les étudiant-es au profil "hésitant" ou corriger les pratiques des profils "non-strict", indépendamment de la note finale.

Donc en guise d'éléments de réponses à la question QR2.1, les analyses menées ont clairement démontré la valeur ajoutée de LAWG pour l'amélioration de la qualité des données collectées, notamment en termes de complétude et de précision, permettant ainsi d'observer des processus d'apprentissage complexes et des cheminements intellectuels qui resteraient autrement inaccessibles. L'outil s'est révélé être un instrument précieux pour l'extraction de comportements complexes, tels que le suivi détaillé des phases de développement, les hésitations et les retours en arrière, offrant une vision plus fidèle de la réalité du travail des étudiant-es. Bien que certaines limites aient été identifiées, notamment concernant la corrélation directe avec la performance académique ou la gestion des transitions chaotiques, LAWG ouvre des perspectives significatives pour un diagnostic pédagogique fin et une compréhension approfondie des stratégies d'apprentissage.

9.2.2 *Analyse au regard de la conscience de la situation*

Les apports de LAWG peuvent eux aussi être analysés à travers le prisme du modèle de conscience de la situation [29].

Au **niveau 1 (perception)**, LAWG permet d'avoir des données qui amélioreront considérablement le support offert par rapport à la première itération. La densité de *commits* multipliée par 30 et la réduction de l'intervalle moyen entre deux traces (de 1 h 14 à 3 min) transforment la granularité de la perception : l'enseignant-e n'observe plus des instantanés ponctuels de l'avancement, mais un flux quasi-continu de l'activité. Cette continuité temporelle rend possible une perception plus fidèle et plus actuelle du travail des étudiant-es.

Au **niveau 2 (compréhension)**, l'apport potentiel de LAWG est important. Les traces fines permettent d'accéder à des dimensions du processus d'apprentissage qui étaient inaccessibles avec les traces manuelles : les profils TDD révèlent la stratégie de développement adoptée par chaque étudiant-e, les phases d'hésitation et de retour en arrière peuvent témoigner des difficultés rencontrées, et des cheminements intellectuels deviennent observables. L'enseignant-e dispose ainsi non seulement de l'information sur l'avancement (perception), mais aussi d'éléments pour comprendre la nature et la dynamique du travail de chaque étudiant-e. Cette capacité correspond précisément au niveau 2 de la conscience de la situation : intégrer et interpréter les éléments perçus pour en dégager une signification pédagogiquement utile.

Le **niveau 3 (projection)** n'est pas directement adressé par nos travaux actuels. Comme établi en itération 1, cette absence n'est pas rédhibitoire : les enseignant-es expérimenté-es construisent cette projection à partir de leurs schémas d'expertise [30, 64]. Toutefois, la richesse des traces générées par LAWG ouvre des perspectives pour un soutien explicite de l'anticipation : la détection de motifs récurrents (par exemple, une phase prolongée d'hésitation suivie d'un abandon)

pourrait, à terme, alimenter des indicateurs prédictifs permettant à l'enseignant-e d'anticiper les difficultés avant qu'elles ne se cristallisent. Cette perspective constitue une piste de travaux futurs.

9.2.3 Un besoin de séparer les traces générées avéré

L'analyse des résultats expérimentaux établit la nécessité de dissocier les traces d'activité générées automatiquement des productions étudiantes. Cette séparation, opérée par l'usage de deux branches distinctes, répond à un impératif de lisibilité. En effet, les enseignants rapportent que les *commits* importants se retrouvent « vraiment noyés dans la masse des autres commits » (E3). L'enregistrement continu de micro-modifications sur la branche principale sature l'historique, et comme le souligne l'enseignant 2, « les changements pour chaque commit c'est trop petit », rendant difficile la lecture des changements par question. Ce bruit informationnel entrave le suivi et l'évaluation, le volume de *commits* étant perçu comme un « obstacle » qui « complexifie significativement le processus de correction ».

Cette granularité fine des données, si elle est isolée, n'en demeure pas moins précieuse. Elle implique toutefois le développement d'outils d'analyse dédiés, capables d'interpréter cette masse d'informations pour la rendre intelligible. L'adaptation de plateformes telles que Git4School devient dès lors indispensable pour transformer ces traces brutes en indicateurs pédagogiques exploitables et actionnables, réconciliant ainsi richesse de la collecte et caractère actionnable des indicateurs pour l'équipe enseignante.

Pour apporter des éléments de réponse à la QR2.2, sur le plan technique, la mise en œuvre de cette séparation au sein d'un même dépôt s'est heurtée aux limites intrinsèques des commandes standard de Git. L'évaluation en situation réelle (section 8.2) a démontré la fragilité d'une manipulation concurrente de deux branches via des commandes de bas niveau (telles que *reset* sans *checkout*). En cas d'interruption inopinée du processus, l'espace de travail se retrouve dans un état incohérent, nécessitant une intervention manuelle complexe pour être restauré, ce qui n'est pas acceptable dans un contexte d'apprentissage.

Face à cette impasse technique, une solution alternative émerge en repoussant les limites de l'utilisation conventionnelle du logiciel de gestion de version. Elle consiste à dupliquer le dossier ".git" afin de créer deux espaces virtuels distincts. Cette architecture permettrait d'opérer simultanément sur deux branches du même dépôt, contournant ainsi la contrainte de l'unicité de la branche active (HEAD) et permettrait de garantir l'intégrité du travail de l'étudiant-e tout en assurant une collecte de traces robuste et transparente.

9.2.4 *Une adhésion modérée malgré des bénéfices reconnus*

Si les avantages pédagogiques et analytiques de LAWG sont bien reconnus par les enseignant-es, l'adhésion des étudiant-es apparaît plus mitigée. Cette réticence peut s'expliquer par une perception des bénéfices de l'outil jugée insuffisante au regard des risques perçus, notamment l'incertitude quant à la sauvegarde de leurs fichiers de travail et la charge cognitive additionnelle liée à l'appropriation d'un nouvel outil. Plusieurs étudiant-es expriment une certaine méfiance vis-à-vis de la collecte automatisée et sont sensibles au manque de *feedback* clair sur les actions de l'outil, ce qui peut générer de l'anxiété. La simplicité technique de l'outil est appréciée, notamment par le rendu de commandes Git plus accessibles. D'ailleurs, deux étudiant-es ont d'abord commencé à travailler sans LAWG, puis ont demandé à l'utiliser pour la suite du TP car elles ou ils considéraient que le processus était plus simple. Mais cette même simplification fait débat car elle masque une partie essentielle de l'apprentissage technique lié à Git, pourtant valorisé dans le cursus.

Du point de vue des enseignant-es, la perception d'un avantage pédagogique majeur coexiste avec des défis pratiques tels que la surcharge informationnelle induite par le nombre élevé de commits atomiques, qui complexifie le travail de correction et d'analyse des données. Il est également à noter que l'intégration de LAWG dans les dispositifs pédagogiques existants n'a pas été perçue comme particulièrement lourde ou contraignante par les enseignants. La mise en place du système n'a nécessité que peu d'ajustements dans les pratiques (la mise en place de GitHub Classroom et l'adaptation du sujet), ce qui facilite son adoption par l'équipe enseignante. Cependant, les retours soulignent un besoin fort d'outillage pour accompagner les enseignant-es dans l'exploitation des traces fines produites : sans outils dédiés pour la visualisation, la synthèse ou l'extraction d'informations pertinentes, la lecture et l'analyse détaillée de ces données riches se révèlent rapidement fastidieuses, notamment lors de la correction ou du suivi individualisé. Le développement de tableaux de bord, de filtres adaptés ou d'outils de post-traitement automatisés apparaît encore une fois indispensable pour rendre pleinement opérationnelle la collecte de traces et concrétiser les bénéfices pédagogiques attendus.

9.2.5 *Recommandations de conception*

Dans cette section, nous donnons une liste de recommandations au sujet de la conception d'un système de génération automatique de traces fines, basée sur un logiciel de gestion de version. Ces recommandations sont issues de l'expérience que nous avons eue quant à la conception de LAWG, son expérimentation, et les retours que nous avons eu de la part des enseignants et des étudiant-es qui l'ont utilisé.

Premièrement, il est crucial d'aligner le niveau d'abstraction du logiciel de gestion de version avec les objectifs pédagogiques visés. La conception de l'outil doit varier selon que l'apprentissage du gestionnaire de version est un objectif prioritaire ou secondaire.

Si l'apprentissage de l'outil de versionnage n'est pas la priorité, il est recommandé de renforcer les contraintes de l'interface, notamment pour le marquage de la résolution d'une question. L'objectif est ici de minimiser la part déclarative afin de fiabiliser les données collectées, en automatisant autant que possible les interactions techniques qui pourraient distraire de l'objectif principal. De plus, à des niveaux d'étude précoces, l'utilisation de logiciels de gestion de version comme Git s'avère relativement complexe et parasite la dynamique d'apprentissage, auquel cas l'abstraction de ces outils est perçue comme bénéfique autant par les étudiant-es que par les enseignant-es.

À l'inverse, si la maîtrise du logiciel de gestion de version constitue un enjeu pédagogique central, l'utilisation du système de génération de traces devrait se rapprocher le plus possible de l'expérience standard de l'outil (par exemple, l'usage des commandes `add`, `commit`, `push` pour Git). Toutefois, il reste nécessaire de vérifier les informations saisies par l'étudiant-e concernant ce qui va constituer la charge utile des traces, comme le message de `commit` dans notre cas, pour garantir la qualité des traces. Une approche hybride, proposant par exemple des messages pré-remplis en fonction de la progression de l'étudiant-e, peut permettre d'adapter le niveau de guidage.

Par ailleurs, l'outil doit fournir des retours clairs et rassurants à chaque action effectuée sur le travail des étudiant-es. Cette transparence est essentielle pour réduire l'anxiété liée à l'automatisation et pour instaurer un climat de confiance. Il est également important d'être explicite sur la nature et l'usage des traces collectées, et d'envisager une adoption progressive ou facultative de l'outil pour favoriser son acceptation.

Enfin, la conception doit prendre en compte la réalité des environnements de travail des étudiant-es, souvent caractérisés par l'usage de multiples machines (souvent une ou plusieurs machines à l'université et une machine personnelle). Faciliter la synchronisation et le travail décentralisé est indispensable, car ces modes de fonctionnement représentent une part significative des cas d'usage réels, tout comme le travail en collaboration avec d'autres étudiant-es.

9.2.6 Limites

Malgré les apports significatifs de l'outil et la robustesse des analyses menées, plusieurs limites méthodologiques et pratiques doivent être soulignées afin de nuancer l'interprétation des résultats.

La taille réduite du groupe contrôle ($n=5$) représente une première limite. Ce faible effectif limite la puissance statistique des comparai-

sons, accroît la variabilité et rend les tests sensibles à la moindre valeur aberrante. De plus, la constitution des groupes, fondée en partie sur l'auto-sélection (les étudiant-es choisissant ou non d'adopter l'outil), peut introduire un biais de sélection. Les profils et attentes peuvent différer entre les groupes, et certains facteurs (motivation, compétences ou appétence technique, gestion du stress associé au suivi automatisé, etc.) n'ont pas pu être contrôlés.

L'expérimentation a porté sur une population restreinte et homogène d'étudiant-es en licence d'informatique, ce qui limite la portée des conclusions à d'autres contextes pédagogiques, niveaux ou disciplines. Par ailleurs, la possession d'une machine personnelle (qui conditionne en partie le bon usage de l'outil) n'est pas universelle à ce niveau de cursus, et pourrait s'avérer plus critique dans une cohorte plus diversifiée.

Les données d'expérience et les entretiens, quoique riches, présentent des risques de biais sociaux ou de désirabilité. Certains retours d'étudiants et d'enseignants ont été collectés sans anonymat, ce qui peut influencer la sincérité ou l'intensité des remarques (positives ou négatives). Les scores d'utilisabilité SUS reflètent une tendance générale mais ne rendent pas compte de toute la diversité individuelle des ressentis vis-à-vis du système.

L'automatisation introduite par l'outil tend à réduire l'effort de gestion de version, mais plusieurs enseignants soulignent que cela pourrait desservir l'apprentissage du "vrai" usage de Git, qui reste un objectif pédagogique. Cette abstraction, bien que bénéfique pour la collecte de traces, doit donc être mise en balance avec le risque d'appauvrissement de l'expérience formatrice sur l'outil de versioning.

Une autre limite identifiée concerne la gestion des sessions de travail lorsqu'un-e étudiant-e utilise sa machine personnelle et laisse l'outil actif sur de longues périodes, sans fermeture entre les séances. Cette pratique conduit à l'enregistrement d'une unique session couvrant l'ensemble du projet, ce qui engendre des valeurs aberrantes dans les mesures d'intervalles temporels entre les traces et nuit à la pertinence des analyses temporelles. Ce scénario, que nous n'avions pas anticipé lors de la conception, s'avère plus fréquent lorsque les étudiant-es disposent de leur propre équipement informatique, une situation moins généralisée au niveau Licence qu'en Master. Ainsi, on peut raisonnablement anticiper que la multiplication des usages sur machines personnelles accentuerait cette problématique dans des contextes différents (comme en Master dont la proportion d'étudiant-es utilisant leur machine personnelle est bien plus importante), rendant nécessaire la mise en place de mécanismes adaptés pour garantir la fiabilité et l'actualité des données collectées.

Enfin, la richesse des traces générées, si elle est précieuse pour l'analyse du cheminement intellectuel, s'accompagne d'une surcharge informationnelle réelle pour les enseignants. Le volume élevé de com-

mits rend difficile la lecture et l'exploitation directe des historiques, et requiert le développement d'outils ou de méthodes d'abstraction supplémentaires pour rester exploitable à grande échelle.

9.3 BILAN DE L'ITÉRATION

Cette section conclut la seconde et dernière itération du cycle DBR en dressant le bilan des trois phases (analyse, conception, évaluation) au regard des questions de recherche et du modèle de conscience de la situation. Cette itération avait pour objectif de dépasser le plafond de compréhension identifié à la fin de l'itération 1, en repensant la manière dont les traces d'apprentissage sont produites. Les travaux présentés dans cette partie permettent de dresser un bilan au regard des deux questions de recherche et du modèle de conscience de la situation [29] qui structure l'ensemble de ce manuscrit.

9.3.1 *Synthèse des apports*

Concernant la question QR2.1, LAWG a démontré sa capacité à améliorer la qualité des traces sur plusieurs dimensions : l'actualité (réduction du temps moyen entre deux traces), la complétude (densité de *commits* multipliée par 30) et la précision (accès aux étapes intermédiaires de résolution). Ces traces plus riches ont permis, grâce aux techniques de PM, d'identifier des comportements d'apprentissage complexes (respect ou non du cycle TDD, phases d'hésitation et de retour en arrière, cheminements intellectuels) qui restaient inaccessibles avec les traces manuelles de l'itération 1.

Concernant la question QR2.2, la séparation des traces automatiques et des productions étudiantes s'est avérée indispensable pour ne pas noyer l'enseignant-e sous un flux de micro-données. Toutefois, sa mise en œuvre technique reste délicate et constitue un défi d'ingénierie logicielle qui appelle des solutions plus robustes.

9.3.2 *Bilan au regard de la conscience de la situation*

L'itération 1 avait établi que Git4School soutenait efficacement le **niveau 1** (perception) et partiellement le **niveau 2** (compréhension) de la conscience de la situation, mais que la nature des traces manuelles imposait un plafond de compréhension empêchant d'accéder aux processus d'apprentissage sous-jacents.

Au **niveau 1 (perception)**, LAWG transforme la granularité de la perception. La densité de traces multipliée par 30 et la réduction de l'intervalle moyen entre deux traces (de 1 h 14 à 3 min) permettent de passer d'instantanés ponctuels de l'avancement à un flux quasi-continu de l'activité, offrant une perception plus fidèle et plus actuelle du travail des étudiant-es.

Au **niveau 2 (compréhension)**, l'apport de LAWG est significatif et constitue l'avancée principale de cette itération. Les traces fines rendent observables des dimensions du processus d'apprentissage qui étaient inaccessibles avec les traces manuelles : les profils TDD révèlent la stratégie de développement adoptée, les phases d'hésitation et de retour en arrière témoignent des difficultés rencontrées, et les cheminement intellectuels deviennent traçables. L'enseignant-e dispose ainsi non seulement de l'information sur l'avancement (perception), mais aussi d'éléments pour comprendre la nature et la dynamique du travail de chaque étudiant-e. Le plafond de compréhension identifié en itération 1 est donc partiellement levé : il est désormais possible de comprendre *comment* les étudiant-es travaillent, et pas seulement *où* elles et ils en sont.

Le **niveau 3 (projection)** n'est pas directement adressé par nos travaux, et comme montré en itération 1, les enseignant-es expérimenté-es compensent cette absence en construisant la projection à partir de leurs schémas d'expertise. Toutefois, la richesse des traces générées ouvre des perspectives pour un soutien explicite de l'anticipation : la détection de motifs récurrents (par exemple, une phase prolongée d'hésitation suivie d'un abandon) pourrait, à terme, alimenter des indicateurs prédictifs.

Cependant, cette avancée s'accompagne d'un paradoxe : en enrichissant la génération de données, on parasite le fonctionnement initial de Git4School sur le plan du niveau 1 du modèle de la conscience de la situation. La profusion de traces brutes risque en effet de saturer la perception de l'enseignant-e et de rendre difficile la lecture immédiate de l'activité, alors que le système était pensé pour faciliter celle-ci. Il apparaît donc nécessaire d'associer LAWG à un TBA adapté, comme une version enrichie de Git4School, afin que l'approfondissement du niveau 2 ne se fasse pas au détriment du niveau 1 et puisse réellement soutenir la prise de décision enseignante. Cette perspective de couplage constitue l'une des pistes de travaux futurs présentées dans la conclusion de ce manuscrit.

Troisième partie

CONCLUSION ET PISTES DE TRAVAUX FUTURS

CONCLUSION ET PERSPECTIVES

10.1	Rappel du contexte et de la problématique	155
10.2	Synthèse des contributions	156
10.2.1	Itération 1 : Visualiser pour agir avec Git4School	156
10.2.2	Itération 2 : Améliorer la qualité des données à la source avec LAWG	157
10.3	Discussion et mise en perspective	158
10.3.1	Le paradoxe de la richesse des données	158
10.3.2	Dilemme entre automatisation et apprentissage technique	158
10.3.3	L'alignement pédagogique révélé par les traces	159
10.4	Limites	159
10.5	Pistes de travaux futurs	159
10.5.1	Court terme : Travaux en cours et consolidation	160
10.5.2	Moyen terme : Robustesse et extension des capacités d'analyse	162
10.5.3	Long terme : Vers la généralisation et l'auto-régulation	163

10.1 RAPPEL DU CONTEXTE ET DE LA PROBLÉMATIQUE

L'enseignement supérieur fait face, au cours des dernières décennies, à une augmentation des effectifs rendant difficile le suivi individualisé des étudiant·es. Ce défi est particulièrement prégnant lors des séances de TP, où l'enseignant·e doit accompagner un grand nombre d'étudiant·es simultanément. Dans le contexte des TP de programmation, les outils de gestion de version tels que Git, incontournables dans l'industrie, ont naturellement intégré les cursus pour former les étudiant·es aux pratiques professionnelles. Cependant, les enseignant·es ne disposent pas des outils nécessaires pour exploiter les traces d'apprentissage brutes issues de ces outils, difficilement visualisables et interprétables pour la prise de décision dans leur forme actuelle et pour des effectifs étudiant·es importants.

Cette thèse, conduite selon une méthodologie de recherche orientée par la conception (DBR), a exploré deux problématiques.

La première a porté sur les moyens de soutenir le suivi étudiant et la prise de décision par les enseignant·es pendant les séances de TP d'informatique, compte tenu des contraintes d'orchestration et de l'existant de ces TP. Elle nous amène à notre première question de recherche :

- **QR1** : Dans quelle mesure un tableau de bord pour enseignant-es, construit à partir des traces générées par un outil de gestion de version et enrichies par des données contextuelles, permet-il de proposer des visualisations actionnables soutenant la prise de décision enseignante en temps réel ?

La seconde problématique, née des limites observées précédemment, visait à améliorer la génération de traces pour raffiner le suivi des étudiant-es en rendant visibles leurs comportements d'apprentissage au cours des résolutions d'exercices de TP de programmation. Elle s'est déclinée en deux questions de recherche :

- **QR2.1** : Comment améliorer l'exactitude, la complétude, la cohérence et l'actualité des traces générées, sur la base des événements de modifications de fichiers, de sorte à pouvoir identifier des comportements d'apprentissage complexes ?
- **QR2.2** : Comment garantir que la génération de traces améliorée ne perturbe ni les activités de l'étudiant-e ni celles de l'enseignant-e ?

10.2 SYNTHÈSE DES CONTRIBUTIONS

Nos travaux se sont déroulés selon une méthode de conduite de la recherche itérative DBR structurée en deux itérations.

10.2.1 *Itération 1 : Visualiser pour agir avec Git4School*

Dans un premier temps, nous avons conçu et développé Git4School, un TBA, un tableau de bord destiné aux enseignant-es et nourri des traces d'apprentissages issues du travail des étudiant-es via Git. L'objectif était de valoriser les données existantes (les dépôts Git des étudiant-es) pour fournir une vue synthétique de l'avancement de la classe. L'évaluation de cet outil a montré qu'il permettait effectivement d'identifier rapidement les étudiant-es en difficulté et de suivre leur activité sur le TP en dehors des cours. Il a également servi d'outil de diagnostic pour les enseignant-es, leur permettant de qualifier la difficulté en contexte réel des exercices et de comparer les niveaux entre différents groupes. Au regard du modèle de conscience de la situation [29], Git4School soutient efficacement le niveau 1 (perception de l'activité des étudiant-es) et partiellement le niveau 2 (compréhension de la situation pédagogique). Bien qu'il ne fournisse pas de support explicite au niveau 3 (projection), les enseignant-es expérimentés construisent cette projection à partir de leurs schémas d'expertise [30, 64], ce qui leur permet effectivement de prendre des décisions d'intervention adaptées. La limite principale reste l'accès aux processus d'apprentissage sous-jacents.

Cependant, cette première itération a mis en lumière des limites fondamentales inhérentes à la nature des données exploitées. En s'ap-

puyant sur les *commits* manuels des étudiant-es, nous étions confrontés à deux défis majeurs. D’une part, la nature déclarative des messages de *commit* compromettait la fiabilité des traces d’apprentissage, rendant difficile une interprétation objective et précise de l’activité réelle. D’autre part, le regroupement des modifications en un nombre restreint de *commits* entraînait une granularité temporelle insuffisante. Cette agrégation masquait des étapes intermédiaires cruciales du processus de résolution de question, empêchant ainsi des analyses fines et continues de l’activité des étudiant-es ainsi que la détection précoce des difficultés.

10.2.2 *Itération 2 : Améliorer la qualité des données à la source avec LAWG*

Face à ce constat, nous avons mené une seconde itération focalisée sur la production des données. Nous avons conçu LAWG, un système interactif qui encapsule l’environnement de travail de l’étudiant-e pour générer des traces de manière semi-automatique. L’expérimentation en contexte écologique a permis de valider l’efficacité de cette approche sur plusieurs plans :

- **Amélioration de la qualité des données** : Nous avons observé une amélioration significative de l’**actualité** des traces (réduction du temps moyen entre deux traces) et de leur **complétude** (densité de *commits* multipliée par 30 par rapport au groupe contrôle). La couverture temporelle de l’activité étudiante est ainsi devenue plus fine et continue.
- **Identification de comportements complexes** : Grâce à la complétude accrue des traces collectées, nous avons pu appliquer des techniques de PM pour identifier des comportements de haut niveau, invisibles auparavant. Nous avons notamment pu observer le respect (ou non) du cycle TDD et distinguer les phases d’hésitation ou de correction.

Ces avancées, relues à travers le modèle de conscience de la situation [29], correspondent à un approfondissement significatif du niveau 2 (compréhension) : les traces fines rendent observables certains processus d’apprentissage et stratégies des étudiant-es, permettant à l’enseignant-e de dépasser le simple constat d’avancement pour comprendre la dynamique du travail.

Au-delà de ces deux contributions logicielles, nos travaux ont également produit un ensemble de recommandations pour la conception d’outils de génération automatique de traces fines s’appuyant sur un logiciel de gestion de version, issues de l’analyse des difficultés rencontrées et des retours des utilisateurs et utilisatrices.

Concernant la question de l’amélioration des traces QR2.1, nos travaux ont démontré que l’approche d’encapsulation du logiciel de gestion de version Git avec LAWG a permis d’améliorer la qualité des traces du point de vue de l’actualité et de la complétude, rendant ainsi

possible l'identification de comportements d'apprentissage complexes via des techniques de PM. L'amélioration de l'exactitude quant à elle n'a pas pu être démontrée.

Concernant la question de la non-perturbation de l'activité des étudiant-es et de l'enseignant-e QR2.2, nos travaux ont montré que si la séparation conceptuelle est indispensable pour ne pas noyer l'enseignant-e sous un flux de micro-données, sa mise en œuvre technique (via deux branches distinctes manipulées par le même client Git) reste délicate et sujette à des instabilités en cas d'arrêt inopiné du système. Toutefois, la criticité de cette question a été soulignée lors des entretiens avec les enseignants et ce point reste important à traiter.

10.3 DISCUSSION ET MISE EN PERSPECTIVE

10.3.1 *Le paradoxe de la richesse des données*

L'introduction de LAWG a créé un paradoxe : en voulant combler le manque de données, nous avons généré une quantité massive d'informations (des centaines de micro-commits par étudiant-e). Si cette richesse est précieuse pour l'analyse automatisée et la recherche, elle devient un "bruit" pour l'enseignant-e qui tenterait de lire l'historique brut. Cela confirme la nécessité absolue de coupler des outils de collecte fine comme LAWG avec des outils de synthèse et d'abstraction comme Git4School. L'enseignant-e ne doit pas être confronté-e à la donnée brute, mais à des indicateurs de haut niveau construits à partir de cette donnée (par exemple : "L'étudiant-e semble bloqué-e sur la question 3 depuis 15 minutes", "L'étudiant-e ne respecte pas le cycle TDD"). Ce paradoxe peut être éclairé par le modèle de conscience de la situation [29] : la donnée brute massive correspond à une surcharge du niveau 1 (perception), qui, loin de favoriser la compréhension, peut la compromettre. Les indicateurs de haut niveau jouent précisément le rôle de facilitateurs du passage du niveau 1 au niveau 2, en transformant un flux de données difficilement appréhendable en informations interprétables.

10.3.2 *Dilemme entre automatisation et apprentissage technique*

Nos travaux ont également soulevé un dilemme pédagogique entre l'automatisation de la collecte de traces et l'apprentissage des outils techniques. En simplifiant l'usage de Git via des commandes abstraites (fix, resume) pour garantir la fiabilité des traces, LAWG prive potentiellement l'étudiant-e d'une opportunité d'apprentissage de l'outil Git lui-même. Ce dilemme n'appelle pas de réponse unique mais un positionnement pédagogique :

- Si l'objectif du cours est l'apprentissage de Git, un outil tel que LAWG doit être envisagé comme des "roues stabilisatrices" temporaires ou être désactivé.
- Si l'objectif est l'apprentissage de la programmation ou d'une méthodologie (comme le TDD), l'abstraction offerte par l'outil est bénéfique car elle réduit la charge cognitive extrinsèque liée à la gestion de version, permettant à l'étudiant-e de se concentrer sur la tâche principale.

10.3.3 L'alignement pédagogique révélé par les traces

Enfin, l'analyse des traces a révélé un décalage intéressant entre les prescriptions pédagogiques et l'évaluation. Nous n'avons pas trouvé de corrélation significative entre le suivi strict du TDD (tel qu'observé dans les traces) et la note finale au TP. Ce résultat suggère que l'évaluation était focalisée sur le "produit" fini (le code qui fonctionne et satisfait la spécification) plutôt que sur le "processus" (la méthode pour y arriver). Les outils de traçage fin comme LAWG offrent une opportunité de réaligner l'évaluation sur les compétences procédurales, en permettant de valoriser non seulement le résultat, mais aussi le processus pour y arriver.

10.4 LIMITES

Ces conclusions doivent être nuancées par certaines limites de nos travaux. La taille restreinte du groupe contrôle lors de la seconde expérimentation ($n = 5$) invite à la prudence sur la généralisation statistique absolue des résultats, bien que les tests non-paramétriques et les simulations aient confirmé la robustesse des tendances observées. De plus, nos expérimentations se sont déroulées dans un contexte spécifique (étudiant-es en informatique, cours sur les tests et le TDD), et la transférabilité de ces résultats à d'autres domaines ou d'autres types d'activités reste à démontrer. Enfin, les défis techniques rencontrés (gestion des sessions longues, robustesse de la double branche) rappellent que l'intrusion dans l'environnement de travail de l'étudiant-e est une opération délicate qui nécessite une ingénierie logicielle minutieuse et robuste pour ne pas perturber l'apprentissage.

10.5 PISTES DE TRAVAUX FUTURS

Au terme de ce travail, nous pensons que pour assurer une supervision efficace des séances de TP à grande échelle, il ne suffit pas d'optimiser la visualisation des données existantes. Il est nécessaire de perfectionner la production de ces données, d'extraire des informations riches sur les processus d'apprentissage, et de les rendre pleinement interprétables et actionnables pour l'enseignant-e. Cette

Typiquement, dans le cadre de cours de programmation, l'utilisation d'un logiciel de gestion de version comme Git est une tâche périphérique néanmoins nécessaire à maîtriser à partir d'un certain niveau d'étude. Il est donc très intéressant de garder la confrontation à cet outil dans le cadre de la plupart des cours, comme on peut le concevoir dans le cadre d'une tâche authentique avec une approche par compétence. Toutefois, dans d'autres domaines que la programmation, voire même que l'informatique, l'abstraction de l'outil Git apparaît bénéfique.

conviction rejoint les travaux de GAŠEVIĆ, DAWSON et SIEMENS [37] qui mettent en garde contre une approche purement centrée sur les données qui négligerait le contexte pédagogique et les processus d'apprentissage, ou encore ceux de LOCKYER, HEATHCOTE et DAWSON [70] sur l'importance d'aligner les traces collectées avec la conception de situation d'apprentissage pour qu'elles fassent sens pour l'enseignant-e. L'approche couplée "collecte semi-automatisée (LAWG) + visualisation synthétique (Git4School)" constitue une voie prometteuse en ce sens pour soutenir l'orchestration de classe [27]. Les travaux présentés dans cette thèse ouvrent ainsi de nombreuses perspectives de recherche, que nous avons structurées selon trois horizons temporels : le court terme, visant à continuer les travaux en cours sur les outils existants et leur consolidation ; le moyen terme, focalisé sur l'extension des capacités d'analyse des données ; et le long terme, explorant de nouveaux paradigmes pédagogiques.

10.5.1 Court terme : Travaux en cours et consolidation

La priorité immédiate est de finaliser les travaux initiés durant cette thèse pour consolider nos contributions logicielles et répondre aux limites identifiées lors de la thèse : le tableau de bord Git4School et le système de génération de traces LAWG, ainsi que leur interaction.

ADAPTATION DE GIT4SCHOOL À LA VOLUMÉTRIE DES TRACES FINES. L'intégration des traces produites par LAWG dans Git4School requiert une adaptation structurelle du tableau de bord face au changement d'échelle de la volumétrie des données. Actuellement, Git4School limite l'affichage à 100 commits par étudiant-e pour des raisons de performance. Cette contrainte, acceptable pour des dépôts standards, devient problématique avec LAWG qui génère un flux continu de *commits*, parfois plusieurs centaines sur la branche principale. Cette densité dépasse les capacités de rendu actuelles et nuit à la lisibilité des graphes.

Pour répondre à cette problématique, un prototype a été développé en collaboration avec Nathan Coquerel lors de son stage. La solution retenue repose sur un mécanisme d'agrégation dynamique des *commits* en fonction du niveau de zoom. Concrètement, les *commits* trop proches sont regroupés visuellement sous forme de rectangles (on peut voir la différence entre les deux figures 10.1a et 10.1b, où les *commits* sont agrégés en blocs dans la deuxième figure).

L'algorithme de regroupement est conçu pour préserver la structure logique de la résolution : un *commit* de résolution d'une question clôt systématiquement un groupe. Ainsi, une séquence de travail comportant deux résolutions distinctes formera nécessairement au moins deux groupes distincts. De plus, pour conserver l'information liée au type de *commit*, l'apparence du groupe reflète son contenu :

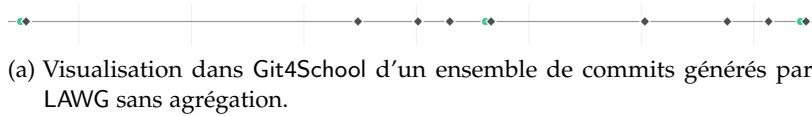


FIGURE 10.1 – Comparaison des visualisations de commits dans Git4School sans et avec agrégation.

Émotions perçues

Durant la réalisation de la tâche, j'ai ressenti des émotions positives. (1: Pas du tout d'accord, 7: Totalelement d'accord)

1 2 3 4 5 6 7

< Ok >

FIGURE 10.2 – Capture d'écran de la fenêtre affichée à l'étudiant-e lui demandant quel est son niveau d'émotions positives ressenti

si un groupe contient un *commit* de type *fix*, le rectangle adopte la couleur associée à son type (vert, orange ou rouge). Ce prototype, bien que fonctionnel, nécessite encore des travaux de finalisation pour être expérimenté.

COLLECTE *in situ* DE DONNÉES SUBJECTIVES. Une extension des capacités d'analyse que nous souhaiterions explorer est l'intégration de la collecte de données subjectives "au fil de l'eau". Les traces actuelles ne permettent d'inférer l'état cognitif et émotionnel de l'étudiant-e que de manière très indirecte, via l'identification de motifs de comportement dans les traces Git. Or, la littérature souligne l'importance de ces dimensions, comme en témoignent les travaux de MOLINARI et al. [80] sur le *reporting* émotionnel ou de PEKRUN et al. [85] sur les émotions d'accomplissement. Nous avons initié cette démarche exploratoire en intégrant un prototype de collecte de données subjectives au sein de LAWG. Concrètement, lors de chaque validation de question (commande *fix*), une micro-sollicitation invite l'étudiant-e à évaluer sur l'instant sa difficulté perçue et son niveau d'émotion positive (voir la Figure 10.2).

Cette méthode, inspirée de l'échantillonnage de l'expérience (*Experience Sampling Method*)[25] et des travaux de PARSONS et al. [83], permet de recueillir le ressenti des étudiant-es en temps quasi réel,

ce qui limite les biais de mémoire et de reconstruction *a posteriori* en réduisant le délai entre la réalisation de la tâche et son évaluation. En demandant une auto-évaluation après la tâche, on exploite l'effet de récence décrit par GLANZER et CUNITZ [40], la performance et l'expérience étant encore en mémoire à court terme et donc moins sujettes aux reconstructions ultérieures. Cependant, la littérature autour de l'échantillonnage de l'expérience reconnaît que des sondages trop fréquents ou mal placés peuvent fragmenter l'activité et perturber l'état de *flow* par exemple, qui requiert une attention soutenue, ce qui conduit à recommander des dispositifs d'échantillonnage de l'expérience plus parcimonieux et alignés sur la dynamique de l'activité [24]. Nous avons initié des expérimentations en contexte écologique de la fonctionnalité. Dans ce cadre, nos premiers travaux exploratoires ont visé à corrélérer ces données avec des mesures psychométriques standardisées collectées en fin de séance (telles que l'échelle de Flow *Eduflow* ou le sentiment d'efficacité personnelle), afin d'en valider la pertinence. Si la faisabilité technique est avérée, la conception de ces interactions requiert une vigilance méthodologique accrue, notamment pour s'assurer que l'instrumentation ne devienne pas intrusive et que les échelles utilisées (par exemple à base d'émojis) soient non-ambiguës [36].

10.5.2 *Moyen terme : Robustesse et extension des capacités d'analyse*

À moyen terme, des défis techniques et conceptuels doivent être relevés pour enrichir les capacités d'analyse et fiabiliser les outils.

FIABILISATION DE LA SÉPARATION DES TRACES. Comme nous l'avons discuté, la gestion de deux branches concurrentes dans un même dépôt est techniquement fragile. Une piste prometteuse serait d'explorer une architecture où le dossier `.git` est dupliqué temporairement, permettant d'avoir deux historiques totalement indépendants pointant sur deux branches différentes, sans risque d'interférence ou d'état incohérent en cas de crash. Cela garantirait une transparence totale pour l'étudiant-e tout en assurant la sécurité des données. Nous avons déjà vérifié que cette approche est réalisable en pratique en déroulant manuellement un algorithme utilisant ce mécanisme et il reste à automatiser ce processus et vérifier sa fiabilité.

INTÉGRATION DES INDICATEURS DE PROCESSUS. Il s'agirait également d'enrichir Git4School avec de nouvelles visualisations basées non plus seulement sur la progression, mais sur le processus lui-même. Par exemple, il serait possible de visualiser le respect du TDD par un indicateur graphique montrant pour chaque étudiant-e la proportion d'itérations respectant le cycle "test-first". De même, la fréquence élevée des *commits* automatiques permettrait de détecter très finement les

périodes d'inactivité ou de "piétinement" (beaucoup de modifications pour peu de progrès) pour alerter l'enseignant-e en temps réel.

DÉSAMBIGÜISATION DES COMPORTEMENTS ÉTUDIANTS. Grâce à l'enrichissement des données et à l'analyse temporelle fine des traces, il devient possible de désambiguïser certains comportements étudiants, notamment dans le cadre de l'évaluation. Au-delà de la simple détection de plagiat de code final, cette approche permettrait de repérer des anomalies de processus, telles que l'insertion soudaine de blocs de code importants ou l'absence de phases d'essais-erreurs, suggérant peut-être un copier-coller externe. L'ambiguïté de l'absence de phases d'essais-erreurs avec le cas d'un étudiant qui est simplement très confiant et avance d'un seul bloc comme dans la figure 9.12a pourrait être levée grâce à des analyses temporelles. Ces indicateurs, utilisés avec précaution, pourraient signaler des besoins d'accompagnement spécifiques.

ADAPTATION DU NIVEAU D'ABSTRACTION POUR L'APPRENTISSAGE DE GIT. Une perspective pédagogique importante concerne l'évolution du rôle de LAWG dans l'apprentissage des outils de gestion de version. Cette réflexion est notamment nourrie par les retours des enseignants lors des entretiens individuels. Actuellement, LAWG masque la complexité de Git pour réduire la charge cognitive extrinsèque des étudiant-es débutant-es. Cependant, la maîtrise de Git est une compétence professionnelle essentielle dans le domaine de la programmation et selon les objectifs pédagogiques, il serait important d'être confronté aux concepts sous-jacents. Une évolution future pourrait consister à implémenter un mécanisme d'estompage (*scaffolding and fading*), où l'outil exposerait progressivement les concepts sous-jacents (*commits*, *branches*) ou demanderait à l'étudiant-e de valider manuellement certaines actions, facilitant ainsi la transition vers l'usage standard de Git.

10.5.3 Long terme : Vers la généralisation et l'auto-régulation

Enfin, à plus long terme, nous envisageons d'étendre l'outil à d'autres domaines et de l'ouvrir directement aux étudiant-es.

EXTENSION À D'AUTRES DOMAINES. Si nos travaux se sont concentrés sur la programmation, la problématique du suivi de la progression individuelle d'étudiant-es en TP est universelle. L'approche de LAWG (encapsulation du logiciel de gestion de version pour générer des traces fines) pourrait être transposée à d'autres activités numériques, comme la rédaction de texte, ou plus largement tout domaine où l'activité est numérique et nécessite finalement de modifier des fichiers.

PERSONNALISATION DES DÉTECTIONS. Afin de rendre l'outil plus flexible et adaptable à différents contextes pédagogiques, il serait pertinent d'intégrer un langage de définition de règles de haut niveau. Cela permettrait aux enseignant·es de configurer les motifs comportementaux à détecter dans les traces (par exemple via des expressions régulières sur le code ou les métadonnées), sans nécessiter une modification du code source de LAWG. Cette fonctionnalité pourrait ouvrir la voie à une bibliothèque partagée de "comportements pédagogiques" définis par la communauté.

UN TABLEAU DE BORD ÉTUDIANT POUR FAVORISER L'AUTO-RÉGULATION VIA DES INDICATEURS FINS SUR LE PROCESSUS D'APPRENTISSAGE. Nous avons vu dans la première expérimentation de Git4School que montrer le tableau de bord aux étudiant·es était bien accueilli et leur permettait de mieux situer leur progression par rapport au groupe. Ainsi, fournir à l'apprenant·e un retour réflexif sur son processus d'apprentissage via Git4School pourrait favoriser l'auto-régulation, comme le suggèrent les revues de la littérature sur les tableaux de bord pour l'apprentissage [54]. En visualisant son propre processus (temps passé, respect de la méthode, comparaison anonymisée avec la classe), l'étudiant·e dispose d'indicateurs concrets pour comprendre ses stratégies et se réorienter, dépassant le simple retour sur la performance finale [101]. Nous pensons aussi à une difficulté peu adressée dans le domaine, dans l'enseignement supérieur et en situation de TP, mais qui est pourtant présente : la timidité et l'anxiété sociale, qui constituent des freins majeurs à la demande d'aide (*help-seeking*), particulièrement dans les grands groupes [60]. Cette réticence à solliciter l'enseignant·e, même en cas de difficulté avérée, peut creuser les inégalités et nuire à la réussite académique [32]. Le système, en détectant des comportements caractéristiques d'étudiant·es en difficulté, pourrait permettre de leur proposer d'alerter l'enseignant·e de façon discrète (via le tableau de bord), contournant ainsi la barrière sociale de la demande d'aide orale et favorisant une intervention pédagogique ciblée et bienveillante.

Quatrième partie

APPENDIX

Jeux de données	Nom du cours	Institution	Objectifs d'apprentissage	Niveau CITE
FilmProvider	XML et Services Web	IUT de Rodez	Savoir utiliser XPATH et créer un schéma XML	5
M1 SDL	Ingénierie des applications web dynamiques	Université Paul Sabatier	Conception (architecture multi-couches, rôle d'un framework) et technologie (J2EE : servlet, JSP, ORM, etc)	7 (pre-mière année M.Sc.)
M2 SDL	Optimisation des applications web dynamiques	Université Paul Sabatier	Être capable d'implémenter les notions avancées de JPA dans le contexte d'un projet Java accédant à une base de données relationnelle. Être capable d'utiliser Spring Boot pour développer une application backoffice exposant une API RESTFull	7
PDO_MVC	Programmation Web côté serveur	IUT de Rodez	Savoir utiliser SQL depuis un langage de programmation (PHP dans notre cas) et comment développer une application Web (en PHP dans notre cas) respectant le pattern MVC	5
db_my_activities	Programmation de bases de données	IUT de Rodez	Savoir développer des fonctions et procédures en SQL	5
OurBusiness	Persistence des données	IUT de Rodez	Comprendre les principes principaux des approches ORM et comment utiliser l'API JPA	6

TABLE 1 – Description des jeux de données

BIBLIOGRAPHIE

- [1] Wil van der AALST. « Big software on the run : in vivo software analytics based on process mining (keynote) ». In : *Proceedings of the 2015 international conference on software and system process*. 2015, p. 1-5.
- [2] Pasquale ARDIMENTO, Mario Luca BERNARDI, Marta CIMITILE et Fabrizio Maria MAGGI. « Evaluating coding behavior in software development processes : A process mining approach ». In : *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. IEEE. 2019, p. 84-93.
- [3] Kimberly E. ARNOLD et Matthew D. PISTILLI. « Course Signals at Purdue : Using Learning Analytics to Increase Student Success ». In : *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*. LAK '12. ACM, 2012, p. 267-270. DOI : [10.1145/2330601.2330666](https://doi.org/10.1145/2330601.2330666).
- [4] Poohridate ARPASAT, Nucharee PREMCHAIWADI, Parham POROUHAN et Wichian PREMCHAIWADI. « Applying Process Mining to Analyze the Behavior of Learners in Online Courses ». In : *Int. J. of Inf. and Education Technology*. T. 11. 10. 2021, p. 436-443.
- [5] Aneesha BAKHARIA, Linda CORRIN, Paula DE BARBA, Gregor KENNEDY, Dragan GAŠEVIĆ, Raoul MULDER, David WILLIAMS, Shane DAWSON et Lori LOCKYER. « A conceptual framework linking learning design with learning analytics ». In : *Proceedings of the sixth international conference on learning analytics & knowledge*. 2016, p. 329-338.
- [6] Aaron BANGOR, Philip KORTUM et James MILLER. « Determining what individual SUS scores mean : Adding an adjective rating scale ». In : *Journal of usability studies* 4.3 (2009), p. 114-123.
- [7] Aaron BANGOR, Philip T KORTUM et James T MILLER. « An empirical evaluation of the system usability scale ». In : *Intl. Journal of Human-Computer Interaction* 24.6 (2008), p. 574-594.
- [8] Laurence BARDIN. *L'analyse de contenu*. Presses universitaires de France, 2013.
- [9] Kent BECK. *Test driven development : By example*. Addison-Wesley Professional, 2022.
- [10] Philip BELL et Marcia C. LINN. « Scientific arguments as learning artifacts : designing for learning from the web with KIE ». In : *International Journal of Science Education* 22.8 (2000), p. 797-817. ISSN : 0950-0693. DOI : [10.1080/095006900412284](https://doi.org/10.1080/095006900412284).

- [11] Jens BENNEDSEN et Michael E. CASPERSEN. « Failure Rates in Introductory Programming ». In : *ACM SIGCSE Bulletin*. T. 39. 2. ACM, 2007, p. 32-36. DOI : [10.1145/1272848.1272879](https://doi.org/10.1145/1272848.1272879).
- [12] Mireille BLAIS et Stéphane MARTINEAU. « L'analyse inductive générale : description d'une démarche visant à donner un sens à des données brutes ». In : *Recherches qualitatives* 26.2 (2006), p. 1-18.
- [13] Robert BODILY et Katrien VERBERT. « Review of Research on Student-Facing Learning Analytics Dashboards and Educational Recommender Systems ». In : *IEEE Transactions on Learning Technologies* 10.4 (2017), p. 405-418. DOI : [10.1109/TLT.2017.2740172](https://doi.org/10.1109/TLT.2017.2740172).
- [14] Alejandro BOGARÍN, Rebeca CEREZO et Cristóbal ROMERO. « A survey on educational process mining ». In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 8.1 (2018), e1230.
- [15] Benedict du BOULAY. « Some Difficulties of Learning to Program ». In : *Journal of Educational Computing Research* 2.1 (1986), p. 57-73. DOI : [10.2190/3LFX-9RRF-67T8-UVK9](https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9).
- [16] Peter BRADFORD, Margaret PORCIELLO, Nancy BALKON et Debra BACKUS. « The Blackboard learning system : The be all and end all in educational instruction ? » In : *Journal of Educational Technology Systems* 35.3 (2007), p. 301-314.
- [17] John BROOKE. « Sus : a "quick and dirty" usability ». In : *Usability evaluation in industry* 189.3 (1996), p. 189-194.
- [18] Neil Christopher Charles BROWN, Michael KÖLLING, Davin MCCALL et Ian UTTING. « Blackbox : A large scale repository of novice programmers' activity ». In : *Proceedings of the 45th ACM technical symposium on Computer science education*. 2014, p. 223-228.
- [19] Rebeca CEREZO, Alejandro BOGARÍN, María ESTEBAN et Cristóbal ROMERO. « Process mining for self-regulated learning assessment in e-learning ». In : *Journal of Computing in Higher Education* 32.1 (2020), p. 74-88.
- [20] Pierre-Antoine CHAMPIN, Alain MILLE et Yannick PRIÉ. « Vers des traces numériques comme objets informatiques de premier niveau : une approche par les traces modélisées ». In : *Intellectica – La revue de l'Association pour la Recherche Cognitive* 59.1 (2013), p. 171-204. DOI : [10.3406/intel.2013.1090](https://doi.org/10.3406/intel.2013.1090).
- [21] Doug CLOW. « The Learning Analytics Cycle : Closing the Loop Effectively ». In : *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*. LAK '12. ACM, 2012, p. 134-138. DOI : [10.1145/2330601.2330636](https://doi.org/10.1145/2330601.2330636).

- [22] William Jay CONOVER. *Practical nonparametric statistics*. John Wiley & sons, 1999.
- [23] Adam COOPER. *What is Analytics ? Definition and Essential Characteristics*. CETIS Analytics Series 5. CETIS, 2012.
- [24] Mihaly CSIKSZENTMIHALYI et Mihaly CSIKSZENTMIHALY. *Flow : The psychology of optimal experience*. T. 1990. Harper & Row New York, 1990.
- [25] Mihaly CSIKSZENTMIHALYI et Reed LARSON. « Validity and reliability of the experience-sampling method ». In : *The Journal of nervous and mental disease* 175.9 (1987), p. 526-536.
- [26] Richard DE CHARMS. *Personal causation : The internal affective determinants of behavior*. Routledge, 2013.
- [27] Pierre DILLENBOURG et Patrick JERMANN. « Technology for classroom orchestration ». In : *New science of learning*. 2010.
- [28] Radim DOLAK. « Using process mining techniques to discover student's activities, navigation paths, and behavior in lms moodle ». In : *International Conference on Innovative Technologies and Learning*. Springer. 2019, p. 129-138.
- [29] Mica R. ENDSLEY. « Toward a Theory of Situation Awareness in Dynamic Systems ». In : *Human Factors : The Journal of the Human Factors and Ergonomics Society* 37.1 (1995), p. 32-64. DOI : 10.1518/001872095779049543. URL : https://webarchive.nationalarchives.gov.uk/ukgwa/20250319155650/https://assets.grenfelltowerinquiry.org.uk/CWJ00000027__Towards%20a%20Theory%20of%20Situation%20Awareness%20in%20Dynamic%20Systems_%2C%20by%20Mica%20R.%20Endsley%2C%20The%20Journal%20of%20the%20Human%20Factors%20and%20Ergonomics%20Society%2C%201995%2C%2037%281%29%2C%2032-64..pdf.
- [30] Mica R. ENDSLEY. « Theoretical Underpinnings of Situation Awareness : A Critical Review ». In : *Situation Awareness Analysis and Measurement*. Sous la dir. de Mica R. ENDSLEY et Daniel J. GARLAND. Lawrence Erlbaum Associates, 2000, p. 3-32.
- [31] Bjoern M ESKOFIER. « 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)*. 2020.
- [32] Nasser FALLAH. « Willingness to communicate in English, communication self-confidence, motivation, shyness and teacher immediacy among Iranian English-major undergraduates : A structural equation modeling approach ». In : *Learning and Individual Differences* 30 (2014), p. 140-147.

- [33] James B FENWICK JR, Cindy NORRIS, Josh ROUNTREE, Cole J SPICER et Scott D CHEEK. « Another look at the behaviors of novice programmers ». In : *ACM SIGCSE Bulletin* 41.1 (2009), p. 296-300.
- [34] Rebecca FERGUSON. « Learning Analytics : Drivers, Developments and Challenges ». In : *International Journal of Technology Enhanced Learning* 4.5/6 (2012), p. 304-317. DOI : [10.1504/IJTEL.2012.051816](https://doi.org/10.1504/IJTEL.2012.051816).
- [35] Stephen FEW. *Information Dashboard Design : The Effective Visual Communication of Data*. O'Reilly Media, 2006. ISBN : 978-0-596-10016-2.
- [36] Courtney L FRANCO et Jennifer MB FUGATE. « Emoji face renderings : Exploring the role emoji platform differences have on emotional interpretation ». In : *Journal of Nonverbal Behavior* 44.2 (2020), p. 301-328.
- [37] Dragan GAŠEVIĆ, Shane DAWSON et George SIEMENS. « Let's not forget : Learning analytics are about learning ». In : *TechTrends* 59.1 (2015), p. 64-71.
- [38] Mohammad GHAFARI, Timm GROSS, Davide FUCCI et Michael FELDERER. « Why research on test-driven development is inconclusive? » In : *Proceedings of the 14th ACM/IEEE International symposium on empirical software engineering and measurement (ESEM)*. 2020, p. 1-10.
- [39] Jean-Marie GILLIOT et Madjid SADALLAH. « A framework for co-designing effective LADs supporting sensemaking and decision making ». In : *International Journal of Learning Technology* 19.1 (2024), p. 109-130. DOI : [10.1504/ijlt.2024.137899](https://doi.org/10.1504/ijlt.2024.137899).
- [40] Murray GLANZER et Anita R CUNITZ. « Two storage mechanisms in free recall ». In : *Journal of verbal learning and verbal behavior* 5.4 (1966), p. 351-360.
- [41] Elena L. GLASSMAN, Jeremy SCOTT, Rishabh SINGH, Philip J. GUO et Robert C. MILLER. « OverCode : Visualizing Variation in Student Solutions to Programming Problems at Scale ». In : *ACM Transactions on Computer-Human Interaction*. T. 22. 2. ACM, 2015, p. 1-35. DOI : [10.1145/2699751](https://doi.org/10.1145/2699751).
- [42] Louis GLASSY. « Using version control to observe student software development processes ». In : *Journal of Computing Sciences in Colleges* 21.3 (2006), p. 99-106.
- [43] Davydd J GREENWOOD et Morten LEVIN. *Introduction to action research : Social research for social change*. SAGE publications, 2006.

- [44] Philip J. GUO. « CodeOpticon : Real-Time, One-To-Many Human Tutoring for Computer Programming ». In : *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST)*. ACM, 2015, p. 599-608. DOI : [10.1145/2807442.2807469](https://doi.org/10.1145/2807442.2807469).
- [45] Monika GUPTA, Ashish SUREKA et Srinivas PADMANABHUNI. « Process mining multiple repositories for software defect resolution from control and organizational perspective ». In : *Proceedings of the 11th Working Conference on Mining Software Repositories*. 2014, p. 122-131.
- [46] Lassi HAARANEN et Teemu LEHTINEN. « Teaching Git on the Side : Version Control System as a Course Platform ». In : *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. ACM, 2015, p. 87-92. DOI : [10.1145/2729094.2742608](https://doi.org/10.1145/2729094.2742608).
- [47] Kenneth HOLSTEIN, Grace HONG, Mera TEGENE, Bruce M. McLAREN et Vincent ALEVEN. « The Classroom as a Dashboard : Co-Designing Wearable Cognitive Augmentation for K-12 Teachers ». In : *Proceedings of the 8th International Conference on Learning Analytics and Knowledge. LAK '18*. ACM, 2018, p. 79-88. DOI : [10.1145/3170358.3170377](https://doi.org/10.1145/3170358.3170377).
- [48] Kenneth HOLSTEIN, Bruce M. McLAREN et Vincent ALEVEN. « Co-Designing a Real-Time Classroom Orchestration Tool to Support Teacher-AI Complementarity ». In : *Journal of Learning Analytics* 6.2 (2019), p. 27-52. DOI : [10.18608/jla.2019.62.3](https://doi.org/10.18608/jla.2019.62.3).
- [49] Christopher D. HUNDHAUSEN, Adam S. CARTER et Phillip CONRAD. « Gitcavator : A Tool for Excavating Software Development Processes from Git Repositories ». In : *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE)*. ACM, 2022, p. 785-791. DOI : [10.1145/3478431.3499295](https://doi.org/10.1145/3478431.3499295).
- [50] Christopher D. HUNDHAUSEN, Daniel M. OLIVARES et Adam S. CARTER. « IDE-Based Learning Analytics for Computing Education : A Process Model, Critical Review, and Research Agenda ». In : *ACM Transactions on Computing Education* 17.3 (2017), p. 1-26. DOI : [10.1145/3105759](https://doi.org/10.1145/3105759).
- [51] Petri IHANTOLA et al. « Educational Data Mining and Learning Analytics in Programming : Literature Review and Case Studies ». In : *Proceedings of the 2015 ITiCSE on Working Group Reports (ITiCSE-WGR '15)*. ACM, 2015, p. 41-63. DOI : [10.1145/2858796.2858798](https://doi.org/10.1145/2858796.2858798).
- [52] Matthew C JADUD. *An exploration of novice compilation behaviour in BlueJ*. University of Kent (United Kingdom), 2006.

- [53] David JANZEN et Hossein SAIEDIAN. « Test-driven development concepts, taxonomy, and future direction ». In : *Computer* 38.9 (2005), p. 43-50.
- [54] Ioana JIVET, Maren SCHEFFEL, Marcus SPECHT et Hendrik DRACHSLER. « License to evaluate : Preparing learning analytics dashboards for educational practice ». In : *Proceedings of the 8th international conference on learning analytics and knowledge*. 2018, p. 31-40.
- [55] Curt JONES. « Using subversion as an aid in evaluating individuals working on a group coding project ». In : *J. Comput. Sci. Coll* 25.3 (2010), p. 18-23.
- [56] Rasmus Leth JØRNØ et Karsten GYNTHNER. « What Constitutes an ‘Actionable Insight’ in Learning Analytics? » In : *Journal of Learning Analytics* 5.3 (2018), p. 198-221. DOI : [10.18608/jla.2018.53.13](https://doi.org/10.18608/jla.2018.53.13).
- [57] Libor JUHANÁK, Jiří ZOUNEK et Lucie ROHLÍKOVÁ. « Using process mining to analyze students’ quiz-taking behavior patterns in a learning management system ». In : *Computers in Human Behavior* 92 (2019), p. 496-506.
- [58] Rogers KALIISA, Ioana JIVET et Paul PRINSLOO. « A checklist to guide the planning, designing, implementation, and evaluation of learning analytics dashboards ». In : *International Journal of Educational Technology in Higher Education* 20.1 (2023), p. 28.
- [59] Rogers KALIISA, Kamila MISIEJUK, Sonsoles LÓPEZ-PERNAS, Mohammad KHALIL et Mohammed SAQR. « Have Learning Analytics Dashboards Lived Up to the Hype? A Systematic Review of Impact on Students’ Achievement, Motivation, Participation and Attitude ». In : *Proceedings of the 14th International Conference on Learning Analytics and Knowledge (LAK '24)*. ACM, 2024, p. 295-304. DOI : [10.1145/3636555.3636884](https://doi.org/10.1145/3636555.3636884).
- [60] Stuart A KARABENICK. « Seeking help in large college classes : A person-centered approach ». In : *Contemporary educational psychology* 28.1 (2003), p. 37-58.
- [61] Daniil KAROL, Elizaveta ARTSER, Ilya VLASOV, Yaroslav GOLUBEV, Hieke KEUNING et Anastasiia BIRILLO. « KOALA : a Configurable Tool for Collecting IDE Data When Solving Programming Tasks ». In : *Proceedings of the ACM Global on Computing Education Conference 2025 Vol 1*. 2025, p. 183-189.
- [62] Judy KAY, Nicolas MAISONNEUVE, Kalina YACEF et Osmar ZAÏANE. « Mining patterns of events in students’ teamwork data ». In : *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*. 2006, p. 45-52.

- [63] Csaba Zoltán KERTÉSZ. « Using GitHub in the Classroom — A Collaborative Learning Experience ». In : *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2015, p. 381-386. DOI : [10.1109/ISSPIT.2015.7394373](https://doi.org/10.1109/ISSPIT.2015.7394373).
- [64] Gary KLEIN. *Sources of Power : How People Make Decisions*. Cambridge, MA : MIT Press, 1998. ISBN : 978-0262611466.
- [65] Gary KLEIN, Jennifer K. PHILLIPS, Erica L. RALL et Deborah A. PELUSO. « A Data-Frame Theory of Sensemaking ». In : *Expertise Out of Context : Proceedings of the Sixth International Conference on Naturalistic Decision Making*. Sous la dir. de Robert R. HOFFMAN. Lawrence Erlbaum Associates, 2006, p. 113-155.
- [66] LAK '11 : *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*. 2011.
- [67] Kurt LEWIN. « Action research and minority problems ». In : *Journal of social issues* 2.4 (1946), p. 34-46.
- [68] Marcia C. LINN, Douglas CLARK et James D. SLOTTA. « WISE design for knowledge integration ». In : *Science Education* 87.4 (2003), p. 517-538. ISSN : 1098-237X. DOI : [10.1002/sce.10086](https://doi.org/10.1002/sce.10086).
- [69] Ying LIU, Eleni STROULIA, Kenny WONG et Daniel GERMAN. « CVS historical information to understand how students develop software ». In : *26th International Conference on Software Engineering-W17S Workshop" International Workshop on Mining Software Repositories (MSR 2004)"*. IET. 2004, p. 32-36.
- [70] Lori LOCKYER, Elizabeth HEATHCOTE et Shane DAWSON. « Informing pedagogical action : Aligning learning analytics with learning design ». In : *American Behavioral Scientist* 57.10 (2013), p. 1439-1459.
- [71] Ejwa LUNDBLAD et Adam EDHOLM. « Gitinspector — Statistical Analysis of a Git Repository ». Mém. de mast. Chalmers University of Technology, 2014.
- [72] Elena LYULINA, Anastasiia BIRILLO, Vladimir KOVALENKO et Timofey BRYKSIN. « TaskTracker-tool : A toolkit for tracking of code snapshots and activity data during solution of programming tasks ». In : *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, p. 495-501.
- [73] Martin MACAK, Daniela KRIZELOVA, Stanislav CHREN et Barbora BUHNOVA. « Using process mining for Git log analysis of projects in a software development course ». In : *Education and Information Technologies* (2021).
- [74] Wannisa MATCHA, Dragan GAŠEVIĆ, Abelardo PARDO et al. « A systematic review of empirical studies on learning analytics dashboards : A self-regulated learning perspective ». In : *IEEE transactions on learning technologies* 13.2 (2019), p. 226-245.

- [75] Riccardo MAZZA et Vania DIMITROVA. « CourseVis : A Graphical Student Monitoring Tool for Supporting Instructors in Web-Based Distance Courses ». In : *International Journal of Human-Computer Studies* 65.2 (2007), p. 125-139. DOI : [10.1016/j.ijhcs.2006.08.008](https://doi.org/10.1016/j.ijhcs.2006.08.008).
- [76] Larry MERKLE et al. « Tracking Large Class Projects in Real-Time Using Fine-Grained Source Control ». In : *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education* (2022), p. 565-570.
- [77] Keir MIERLE, Kevin LAVEN, Sam ROWEIS et Greg WILSON. « Mining student CVS repositories for performance indicators ». In : *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), p. 1-5.
- [78] Megha MITTAL et Ashish SUREKA. « 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, p. 344-353.
- [79] Inge MOLENAAR et Carolien AN KNOOP-VAN CAMPEN. « How teachers make dashboard information actionable ». In : *IEEE Transactions on Learning Technologies* 12.3 (2018), p. 347-355.
- [80] Gaëlle MOLINARI, Maxence TRANNOIS, Aurélien TABARD et Elise LAVOUÉ. « EMORE-L : un outil de reporting des émotions pour l'apprentissage à distance ». In : *Actes de la 28ième conférence francophone sur l'Interaction Homme-Machine*. 2016, p. 167-176.
- [81] Patrick MUKALA, Joos CAM BUIJS, Maikel LEEMANS et Wil MP van der AALST. « Learning analytics on coursera event data : A process mining approach ». In : *SIMPDA*. 2015, p. 18-32.
- [82] Abelardo PARDO et al. « OnTask : Delivering Data-Informed, Personalized Learning Support Actions ». In : *Journal of Learning Analytics* 5.3 (2018), p. 235-249. DOI : [10.18608/jla.2018.53.15](https://doi.org/10.18608/jla.2018.53.15).
- [83] Dale PARSONS, Krissi WOOD, Joy GASSON et Adon CM MOSKAL. « Development of a Self-Reporting Tool for Capturing Student Emotions during Programming Activities ». In : *Proceedings of the Twenty-First Australasian Computing Education Conference*. 2019, p. 64-68.
- [84] Lasse PAULSEN et Euan LINDSAY. « Learning analytics dashboards are increasingly becoming about learning and not just analytics – A systematic review ». In : *Education and Information Technologies* 29 (2024), p. 14279-14308. DOI : [10.1007/s10639-023-12401-4](https://doi.org/10.1007/s10639-023-12401-4).

- [85] Reinhard PEKRUN, Thomas GOETZ, Anne C FRENZEL, Petra BARCHFELD et Raymond P PERRY. « Measuring emotions in students' learning and performance : The Achievement Emotions Questionnaire (AEQ) ». In : *Contemporary educational psychology* 36.1 (2011), p. 36-48.
- [86] Wouter PONCIN, Alexander SEREBRENIK et Mark van den BRAND. « Mining student capstone projects with FRASR and ProM ». In : *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. 2011, p. 87-96.
- [87] Wouter PONCIN, Alexander SEREBRENIK et Mark VAN DEN BRAND. « Process mining software repositories ». In : *2011 15th European conference on software maintenance and reengineering*. IEEE. 2011, p. 5-14.
- [88] Bruno POSSAGHI et al. « Integrating multi-modal learning analytics dashboard in K-12 education : insights for enhancing orchestration and teacher decision-making ». In : *Smart Learning Environments* 12.1 (2025). DOI : [10.1186/s40561-025-00410-4](https://doi.org/10.1186/s40561-025-00410-4).
- [89] Thomas W. PRICE et al. « ProgSnap2 : A Flexible Format for Programming Process Data ». In : *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20)*. ACM, 2020, p. 356-362. DOI : [10.1145/3341525.3387373](https://doi.org/10.1145/3341525.3387373).
- [90] Luis P PRIETO, Kshitij SHARMA, Pierre DILLENBOURG et María JESÚS. « Teaching analytics : towards automatic extraction of orchestration graphs using wearable sensors ». In : *Proceedings of the sixth international conference on learning analytics & knowledge*. 2016.
- [91] Jean-Baptiste RACLET, Franck SILVESTRE et Mika PONS. « Mise en oeuvre d'approches pédagogiques fondées sur des pratiques de l'industrie du logiciel pour l'apprentissage de la programmation ». In : *2020 Colloque Didapro 8 : L'informatique, objets d'enseignements (DidaSTIC 2020)*. 2020, p. 1-12.
- [92] Yahya RAFIQUE et Vojislav B MIŠIĆ. « The effects of test-driven development on external quality and productivity : A meta-analysis ». In : *IEEE Transactions on Software Engineering* 39.6 (2012), p. 835-856.
- [93] Gregorio ROBLES et Jesús M GONZÁLEZ-BARAHONA. « Mining student repositories to gain learning analytics. An experience report ». In : *2013 IEEE Global Engineering Education Conference (EDUCON)*. 2013.

- [94] Vladimir RUBIN, Christian W GÜNTHER, Wil MP VAN DER AALST, Ekkart KINDLER, Boudewijn F van DONGEN et Wilhelm SCHÄFER. « Process mining framework for software processes ». In : *International conference on software process*. Springer. 2007, p. 169-181.
- [95] José A RUIPÉREZ-VALIENTE, Pedro J MUNOZ-MERINO, Jose A GASCON-PINEDO et C Delgado KLOOS. « Scaling to massiveness with analyse : A learning analytics tool for open edx ». In : *IEEE Transactions on Human-Machine Systems* 47.6 (2016), p. 909-914.
- [96] José A. RUIPÉREZ-VALIENTE et Yeong-Jin KIM. « Effects of GitHub Classroom on Software Development Education : A Case Study ». In : *Proceedings of the IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2020, p. 1098-1102. DOI : [10.1109/EDUCON45650.2020.9125262](https://doi.org/10.1109/EDUCON45650.2020.9125262).
- [97] Javier Santofimia RUIZ, Héctor J Pijeira DÍAZ, José A RUIPÉREZ-VALIENTE, Pedro J MUÑOZ-MERINO et Carlos Delgado KLOOS. « Towards the development of a learning analytics extension in open edX ». In : *Proceedings of the second international conference on technological ecosystems for enhancing multiculturality*. 2014, p. 299-306.
- [98] Madjid SADALLAH et Jean-Marie GILLIOT. « Generating LADs that Make Sense ». In : *Proceedings of the 15th International Conference on Computer Supported Education (CSEDU 2023)*. SciTePress, 2023, p. 35-46.
- [99] Madjid SADALLAH, Jean-Marie GILLIOT, Sébastien IKSAL, Katia QUELENNEC, Mathieu VERMEULEN, Laurent NEYSSENSAS, Olivier AUBERT et Rémi VENANT. « Designing LADs That Promote Sensemaking : A Participatory Tool ». In : *Educating for a New Future : Making Sense of Technology-Enhanced Learning Adoption (EC-TEL 2022)*. T. 13450. Lecture Notes in Computer Science. Springer, 2022, p. 587-593. DOI : [10.1007/978-3-031-16290-9_54](https://doi.org/10.1007/978-3-031-16290-9_54).
- [100] Beat A. SCHWENDIMANN, María Jesús RODRÍGUEZ-TRIANA, Andrii VOZNIUK, Luis P. PRIETO, Mina Shirvani BOROUJENI, Adrian HOLZER, Denis GILLET et Pierre DILLENBOURG. « Perceiving Learning at a Glance : A Systematic Literature Review of Learning Dashboard Research ». In : *IEEE Transactions on Learning Technologies* 10.1 (2017), p. 30-41. DOI : [10.1109/TLT.2016.2599522](https://doi.org/10.1109/TLT.2016.2599522).
- [101] Gayane SEDRAKYAN, Jochen DE WEERDT et Monique SNOECK. « Process-mining enabled feedback : “tell me what I did wrong” vs. “tell me how to do it right” ». In : *Computers in human behavior* 57 (2016), p. 352-376.

- [102] Lotfi Sofiane SETTOUTI. « Systèmes à base de traces modélisées : modèles et langages pour l'exploitation des traces d'interactions ». Thèse de doct. Université Claude Bernard Lyon 1, 2011.
- [103] Mike SHARPLES. « A challenge to enhance the system of education—A comment from a researcher perspective ». In : *Grand Challenge Problems in Technology-Enhanced Learning II : MOOCs and Beyond : Perspectives for Research, Practice, and Policy Making Developed at the Alpine Rendez-Vous in Villard-de-Lans*. Springer, 2015, p. 63-70.
- [104] Ben SHNEIDERMAN. « The Eyes Have It : A Task by Data Type Taxonomy for Information Visualizations ». In : *Proceedings of the IEEE Symposium on Visual Languages*. IEEE, 1996, p. 336-343. DOI : [10.1109/VL.1996.545307](https://doi.org/10.1109/VL.1996.545307).
- [105] Viktor SHYNKARENKO et Oleksandr ZHEVAHO. « Application of Constructive Modeling and Process Mining Approaches to the Study of Source Code Development in Software Engineering Courses ». In : *Journal of Communications Software and Systems* (2021).
- [106] George SIEMENS. « Learning Analytics : The Emergence of a Discipline ». In : *American Behavioral Scientist* 57.10 (2013), p. 1380-1400. DOI : [10.1177/0002764213498851](https://doi.org/10.1177/0002764213498851).
- [107] Franck SILVESTRE et Jean-Baptiste RACLET. « 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.
- [108] Franck SILVESTRE, Philippe VIDAL et Julien BROISIN. « Reflexive learning, socio-cognitive conflict and peer-assessment to improve the quality of feedbacks in online tests ». In : *Design for Teaching and Learning in a Networked World : 10th European Conference on Technology Enhanced Learning, EC-TEL 2015, Toledo, Spain, September 15-18, 2015, Proceedings 10*. Springer. 2015, p. 339-351.
- [109] Teo SUSNJAK, Gomathy Suganya RAMASWAMI et Anuradha MATHRANI. « Learning analytics dashboard : a tool for providing actionable insights to learners ». In : *International Journal of Educational Technology in Higher Education* 19.1 (2022), p. 12.
- [110] John SWELLER. « Cognitive Load During Problem Solving : Effects on Learning ». In : *Cognitive Science* 12.2 (1988), p. 257-285. DOI : [10.1207/s15516709cog1202_4](https://doi.org/10.1207/s15516709cog1202_4).
- [111] Pierre TCHOUNIKINE. « Clarifying design for orchestration : orchestration and orchestrable technology, scripting and conducting ». In : *Computers & Education* 69 (2013), p. 500-503.

- [112] Mike TISSENBAUM et Jim SLOTTA. « Supporting classroom orchestration with real-time feedback : A role for teacher dashboards and real-time agents ». In : *International Journal of Computer-Supported Collaborative Learning* 14.3 (2019), p. 325-351.
- [113] Mark VAN HARMELEN et David WORKMAN. « Analytics for learning and teaching ». In : *CETIS Analytics Series* 1.3 (2012), p. 1-40.
- [114] Anouschka VAN LEEUWEN et Nikol RUMMEL. « Orchestration Tools to Support the Teacher During Student Collaboration : A Review ». In : *Unterrichtswissenschaft* 47.2 (2019), p. 143-158. DOI : [10.1007/s42010-019-00052-9](https://doi.org/10.1007/s42010-019-00052-9).
- [115] Katrien VERBERT, Erik DUVAL, Joris KLERKX, Sten GOVAERTS et José Luis SANTOS. « Learning Analytics Dashboard Applications ». In : *American Behavioral Scientist* 57.10 (2013), p. 1500-1509. DOI : [10.1177/0002764213479363](https://doi.org/10.1177/0002764213479363).
- [116] Katrien VERBERT, Sten GOVAERTS, Erik DUVAL, José Luis SANTOS, Frans VAN ASSCHE, Gonzalo PARRA et Joris KLERKX. « Learning Dashboards : An Overview and Future Research Opportunities ». In : *Personal and Ubiquitous Computing* 18.6 (2014), p. 1499-1514. DOI : [10.1007/s00779-013-0751-2](https://doi.org/10.1007/s00779-013-0751-2).
- [117] Arto VIHAVAINEN, Matti LUUKKAINEN et Petri IHANTOLA. « Analysis of Source Code Snapshot Granularity Levels ». In : *Proceedings of the 15th Annual Conference on Information Technology Education (SIGITE '14)*. ACM, 2014, p. 21-26.
- [118] Feng WANG et Michael J. HANNAFIN. « Design-based research and technology-enhanced learning environments ». In : *Educational Technology Research and Development* 53.4 (2005), p. 5-23. ISSN : 1042-1629. DOI : [10.1007/bf02504682](https://doi.org/10.1007/bf02504682).
- [119] Richard Y WANG et Diane M STRONG. « Beyond accuracy : What data quality means to data consumers ». In : *Journal of management information systems* 12.4 (1996), p. 5-33.
- [120] Hironori WASHIZAKI. *Guide to the Software Engineering Body of Knowledge*. 2024.
- [121] Karl E. WEICK. *Sensemaking in Organizations*. Foundations for Organizational Science. Sage Publications, 1995. ISBN : 978-0-8039-7177-6.
- [122] Alyssa Friend WISE. « Designing pedagogical interventions to support student use of learning analytics ». In : *Proceedings of the fourth international conference on learning analytics and knowledge*. 2014, p. 203-211.

- [123] Alyssa Friend WISE et Yeonji JUNG. « Teaching with Analytics : Towards a Situated Model of Instructional Decision-Making ». In : *Journal of Learning Analytics* 6.2 (2019), p. 53-69. DOI : [10.18608/jla.2019.62.4](https://doi.org/10.18608/jla.2019.62.4).
- [124] Alyssa Friend WISE et Jovita VYTASEK. « Learning analytics implementation design ». In : *Handbook of learning analytics* 1 (2017), p. 151-160.