PRaTDL : un protocole fondé sur le test et la revue de code pour l'apprentissage de la programmation

Franck Silvestre[1] and Jean-Baptiste Raclet[1]

Université Toulouse III, Laboratoire de Recherche en Informatique de Toulouse,
F-31062 Toulouse Cedex 9, France

Résumé Cet article présente une nouvelle approche d'enseignement de la programmation intitulée "apprentissage de la programmation dirigée par les tests et la revue de code" (PRaTDL). Cette proposition s'inscrit dans les travaux sur l'apprentissage par les tests (TDL). Nous fournissons aux étudiants un ensemble exhaustif de tests unitaires automatisés servant de spécifications détaillées du logiciel qu'ils doivent développer. De plus, afin de favoriser les interactions entre pairs, une ou plusieurs phases de revue de code sont introduites dans chaque séquence. Ces phases de revue sont orchestrées avec la plate-forme Tsaap-Notes.

Le PRaTDL a été expérimenté dans le contexte d'un enseignement de Master Informatique avec un groupe de 24 étudiants. Une analyse des résultats est discutée et des pistes d'amélioration sont proposées.

Keywords: Apprentissage de la programmation, développement dirigé par les tests, revue de code par les pairs, évaluation par les pairs

1 Introduction

En ingénierie logicielle, les tests représentent un dispositif incontournable pour vérifier et valider un système logiciel [1]. L'adoption massive des méthodes agiles dans l'industrie [2] a mis de facto sur le devant de la scène le développement dirigé par les tests (TDD pour *Test Driven Development*). Cette pratique d'ingénierie, formalisée initialement dans le cadre de la méthode XP [3], consiste à développer les fonctionnalités d'un logiciel dans une succession de cycles commençant par l'écriture d'un test, suivi de l'écriture du code faisant passer le test et terminant par l'amélioration du code principal (refactoring). Cette approche redéfinit complètement le rôle des tests en ingénierie logicielle : le TDD est une méthode d'analyse, de conception et de développement reposant sur l'écriture de tests automatisés tout au long du cycle de développement [4]. Bien que l'approche bénéficie d'un réel succès d'estime, le TDD ne suit pas la courbe d'adoption des méthodes agiles dans l'industrie [5]. Dans leur étude, les auteurs énoncent sept facteurs de limitation de l'adoption du TDD parmi lesquelles deux sont directement liés au manque de connaissances et de compétences sur le TDD et l'écriture des tests.

Plusieurs initiatives ont été prises pour introduire le TDD dans le champ académique [6]. L'apprentissage dirigé par les tests (TDL pour *Test Driven Learning*) a été introduit dans [7] pour relever le défi d'apprendre à programmer en utilisant les tests afin d'expliciter l'usage et le comportement de portions de codes. Les résultats des expérimentations montrent que les étudiants gagnent en compréhension lorsqu'ils sont en situation d'apprentissage dirigée par les tests [8] et qu'ils améliorent leurs performances aux évaluations.

Notre avons initié notre expérimentation dans le cadre d'apprentissages de notions de programmation avancées sur un groupe de 24 étudiants inscrits en Master spécialisé dans l'ingénierie logiciel. L'équipe pédagogique de l'unité d'enseignement cible de l'expérimentation a choisi de promouvoir le TDD. Cependant, l'écriture de tests est coûteuse en temps [9]. D'autre part, certains objectifs d'apprentissage requièrent des interactions sociales comme cet attendu décrit dans [10] : "Être capable de discuter de comment un problème peut être résolu par différents algorithmes, chacun avec différentes propriétés". Cet article présente un nouveau protocol inspiré par le TDL et baptisée PRaTDL pour *Peer Review and Test-driven Development Learning* relevant les deux défis suivants :

1. Fournir aux étudiants des activités dans lesquelles l'écriture de code de tests est évitée tout en maintenant une forte culture TDD ;
2. Fournir aux étudiants des activités couvrant un périmètre d'objectifs d'apprentissage incluant ceux nécessitant des interactions sociales.

Le papier est structuré en 4 parties : la section 2 introduit le protocol PRaTDL ; la section 3 détaille la première expérimentation menée utilisant l'approche PRaTDL ; la section 4 présente notre analyse des résultats obtenus ; enfin, la conclusion présente les idées clés guidant nos travaux futurs.

2 Le protocole PRaTDL

Cette section décrit le protocole PRaTDL introduit dans les enseignements de notre expérimentation : les étudiants ont à développer un projet dont les fonctionnalités peuvent être conçues à l'aide des notions de programmation visées. Les consignes générales sont indiquées sur un format standard d'énoncé rédigé en langage naturel. Les questions exigeant en réponse l'écriture de code sont systématiquement accompagnées du jeu de tests automatisés devant passer avec succès pour considérer que l'exercice a été accompli avec succès. Enfin, chaque séance est structurée sous la forme d'une alternance de séquences de travail en autonomie sur le projet (phases TDD) et de séquences de partage et d'échanges entre pairs (phases PR).

Afin de conserver un alignement pédagogique tel que décrit dans [11], il convient de proposer des activités individuelles aux étudiants comparables aux activités proposées lors des évaluations sommatives puisque ces évaluations doivent être individuelles. Les phases TDD durant lesquelles les étudiants travaillent individuellement remplissent cette part du contrat pédagogique. De plus, un grand nombre d'objectifs d'apprentissage présentés dans le document [10] relatifs à la programmation sont couverts par les phases TDD. Néanmoins, certains

attendus ne peuvent pas être pris en compte lors d'activités menées individuellement. Par exemple, les phases TDD ne peuvent pas répondre aux objectifs requérant la fourniture d'explications ou la discussion avec autrui. Ce constat nous a conduit à introduire les activités de revues de code par les pairs (phases PR) orchestrées de telle sorte qu'elles répondent aux exigences d'interactions sociales non couvertes par les phases TDD. Le tableau 1 présente un extrait des résultats d'apprentissages attendus décrit dans [10] et couverts respectivement pas les phases TDD et les phases PR.

Table 1. Extrait des objectifs d'apprentissage couverts par le protocole PRaTDL

Id.	Objectif d'apprentissage	Phases
1	Analyser le comportement de programmes simples impliquant les éléments de base suivants : variables, expressions, affectations, E / S, structures de contrôle, fonctions, passage de paramètres et récursivité.	
2	Concevoir, implémenter, tester et déboguer un programme qui utilise chacune des structures de programmation suivantes : calcul de base, E / S simples, structures conditionnelles et itératives, définition de fonctions et passage de paramètres.	TDD
3	Écrire un programme utilisant des E / S de fichiers pour assurer la persistance entre plusieurs exécutions.	
4	Écrire des programmes qui utilisent chacune des structures de données suivantes : tableaux, enregistrements structures, chaînes, listes chaînées, piles, files d'attente, ensembles et cartes.	
5	Construire, exécuter et déboguer des programmes en utilisant les environnements de développement intégrés modernes	
6	Construire et déboguer des programmes en utilisant les bibliothèques standardisées disponibles pour un langage de programmation donné	
7	Discuter sur les moyens de résoudre un problème avec différents algorithmes ayant chacun leurs propres propriétés.	PR
8	Expliquer le comportement de programmes simples impliquant les éléments de base suivants : variables, expressions, affectations, E / S, structures de contrôle, fonctions, passage de paramètres et récursivité.	
9	Participer à une révision du code d'une petite équipe axée sur l'exactitude des composants.	
10	Analyser dans quelle mesure le code d'un autre programmeur respecte la documentation et les standards d'un style de programmation.	

3 Détail de l'expérimentation

L'expérimentation s'est déroulée dans un enseignement de deuxième année de Master en ingénierie du logiciel sur les notions de correspondance objet relationnel et leur mise en œuvre avec l'API Java pour la Persistance (JPA). Le protocole PRaTDL a été appliqué durant la totalité du cours soit 20h en séances avec les étudiants. Les 24 étudiants inscrits disposaient de connaissances sur les tests unitaires automatisés (cours dispensés en licence). Les étudiants suivent leur cursus en alternance : 3 jours par semaine en entreprise et 2 jours à l'Université. Les journées à l'Université se composent de 6h à 8h de cours par jour. En conséquence, l'équipe enseignante tente de minimiser les travaux à réaliser en dehors du temps universitaire. Dans un tel contexte, le temps passé en présentiel avec les étudiants est une ressource critique et l'efficacité de l'approche pédagogique déployée en séance en est d'autant plus cruciale.

3.1 Détail des phases TDD

Le tableau 2 liste les principales caractéristiques des tests fournis aux étudiants pendant les phases TDD. On remarque que les tests peuvent être utilisés pour spécifier ce que le code principal doit faire (caractéristique 2) mais aussi comment il doit le faire (caractéristiques 1 et 3).

3.2 Détail des phases de revues de code

Les phases PR ont été mises en œuvre en s'inspirant directement de l'approche d'Instruction par les Pairs (IP) introduite dans [12]. Les études expérimentales [13] montrent que l'IP est une approche d'évaluation formative tirant parti des interactions sociales ayant un impact positif sur l'engagement des étudiants et les résultats d'apprentissage. Différentes plateformes [14,15] fournissent un support technologique à l'IP permettant aux étudiants de confronter leur point de vue en s'appuyant sur les contributions qu'ils soumettent à la plateforme. Ces technologies permettent de réduire la distance entre les activités proposées en cours et les épreuves individuelles d'évaluation auxquelles sont sujets les étudiants, tout en maintenant des interactions sociales dans le processus d'apprentissage.

Les phase PR ont été orchestrées avec la plateforme web Tsaap-Notes proposant un protocole d'IP étendu [15]. Tsaap-Notes permet à l'enseignant de poser des questions aux étudiants et de gérer un processus en 3 étapes :

1. Pour chaque question, chaque étudiant, à l'aide d'un dispositif connecté (tablette, smartphone ou ordinateur portable), fournit une réponse (choix d'un item, argumentation au format texte libre) au système. L'enseignant dispose d'une interface l'informant en temps réel du nombre de réponses collectées.
2. Sur la base de cette information, l'enseignant peut démarrer la deuxième étape du processus : le système demande à chaque apprenant d'étudier et d'évaluer jusqu'à cinq contributions apportées par les autres étudiants. Durant cette étape, les étudiants peuvent éventuellement modifier leur réponse initiale et la soumettre de nouveau à la plateforme.

Table 2. Caractéristiques des tests

Id.	Caractéristique	Exemple

1	Un test ne compile que si les types de données manipulés dans le test sont créés correctement par l'apprenant dans le programme principal (idem pour les méthodes).	`// given: an enterprise with all`
		`// properties correctly set`
		`enterprise = new Enterprise();`
		`enterprise.setName("Company \&Co");`
		`enterprise.setDescription("Comp desc");`

2	Les tests dits "fonctionnels" (qui testent le comportement attendu en mode boite noire) ne passent que si l'apprenant a codé les algorithmes qui permettent d'atteindre les états des objets spécifiés par les tests.	`// when we switch between, enterprise`
		`// 1 and 2 on project 1`
		`project.switchEnterprise(enterprise2);`
		`// and we save the project`
		`Project savedProject =`
		`  enterpriseProjectService.save(project);`
		`// then the saved project is attached`
		`// to enterprise 2`
		`assertThat(savedProject.getEnterprise(),`
		`  is(enterprise2));`
		`// and enterprise 1 has only one`
		`// associated project`
		`assertThat(enterprise1.getProjects().`
		`  size(), is(1));`

3	Les tests dits "système" ou d'interactions permettent de forcer certains aspects de la solution : utilisation d'un composant précis ; collaboration entre objets, ...	`// when: trying to find the project by id`
		`enterpriseProjectService.`
		`  findProjectById(anId);`
		`// then: the find operation is`
		`// triggered`
		`// on the entity manager`
		`verify(entityManager).`
		`  find(Project.class, anId);`

3. L'enseignant peut décider de la publication du résultat en fonction du nombre d'étudiants ayant participé à la deuxième étape. Après publication, l'enseignant et les étudiants ont accès à la liste de toutes les contributions ordonnées de la mieux notée à la moins bien notée. L'enseignant et les étudiants ont alors l'opportunité d'échanger sur les résultats observés.

Nous avons utilisé Tsaap-Notes pour demander aux étudiants de défendre leurs solutions. Ils devaient fournir des portions de code et d'expliquer leurs choix d'implantation. Ainsi les étudiants partagent donc leurs codes mais aussi leurs motivations argumentées par écrit pour choisir une solution plutôt qu'une autre et disposent de la possibilité d'améliorer leurs solutions à la lumière des confrontations de points de vue.

3.3 L'enquête auprès des étudiants et évaluations sommatives

À la fin du cours, nous avons mené une enquête auprès des étudiants pour mesurer le bénéfice qu'ils ont perçu par rapport au protocole PRaTDL. Ils ont évalué, via une échelle linéaire de 1 ("pas du tout d'accord") à 10 ("tout à fait d'accord"), les trois affirmations suivantes :

Q1 : *J'ai appris beaucoup au cours de cette UE.*

Q2 : *Je tire complètement partie de la phase TDD.*
Q3 : *Je tire complètement partie de la phase PR.*
Une justification était requise dans une zone de texte libre.

Afin d'obtenir les crédits ECTS correspondants à ce cours, les étudiants ont eu à passer deux évaluations : une épreuve orientée pratique de 2 heures pendant laquelle les étudiants ont eu à résoudre des exercices présentés exactement sous le même format lors des phases TDD ; une épreuve orientée théorie de 2 heures durant laquelle les étudiants étaient interrogés sur les concepts clés de JPA et les problèmes récurrents résolu par l'API. Les questions de ce devoir étaient en lien direct avec les thèmes abordés durant les phases PR.

Nous avons veillé à l'alignement des objectifs d'apprentissage, avec les activités dispensées pendant les cours et les épreuves d'évaluation sommative tel que cela est recommandé dans [11].

4 Résultats et analyses

En tant qu'enseignant, pendant le cours, nous bénéficions alors d'un dispositif pédagogique favorisant d'une part un apprentissage où chaque étudiant avance à son rythme et, dans le meilleur des cas, un apprentissage auto-régulé. En effet, les étudiants travaillent durant les phases TDD sur des exercices en étant guidés par les feedbacks fournis par l'exécution des tests. Cette approche a permis à certains étudiants d'appréhender et de comprendre seuls les notions avancées abordées pendant le cours. En conséquence, l'enseignant dispose de davantage de temps à consacrer aux étudiants ayant des difficultés sur certains exercices.

Comme indiqué dans la section 3, les étudiants ont été sollicités pour répondre à une enquête. Sur 24 étudiants inscrits, nous avons obtenu 22 réponses. De ces réponses émerge un fort sentiment d'avoir appris. En effet, à la question Q1, la totalité des étudiants répond avec une note supérieure à 5 ; ils sont 64% à y répondre avec une note supérieure à 7 sur 10. Ce résultat traduit la perception par les étudiants d'un approfondissement dans les apprentissages sachant que cet enseignement est de niveau avancé et s'inscrit dans la continuité directe de plusieurs enseignements reçus en Master 1. De plus, il concerne une API que certains étudiants avaient mis en oeuvre au cours de leur stage de Master 1.

Concernant les modalités d'apprentissage, la phase de travail individuel s'appuyant sur le TDD est plébiscitée par les étudiants : ils sont 95.5% à répondre à la question Q2 avec une note supérieure à 5 et 72.7% avec une note supérieure à 7. Les termes les plus fréquents dans les justifications fournies font apparaître un sentiment d'émancipation ("autonomie", "découvrir", "seul", "recherche"), d'encapacitation ("pouvoir", "permet") et d'apprentissage ("savoir", "apprendre").

La question Q3 a permis d'évaluer la perception des étudiants sur la phase de confrontations et d'échanges. Les étudiants répondent à 95.5% avec une note supérieure à 5 et 86.4% avec une note supérieure à 7. Les mots les plus fréquents des commentaires des étudiants sur cette phase font apparaître un sentiment d'approfondissement d'une part ("mieux", "réflexion", "confronter","constructif") et de compréhension d'autre part ("comprendre", "connaissance", "effectuer").

Enfin, certains étudiants ont manifesté leur doute sur leur capacité à écrire un code plus conséquent et non guidé par les tests en utilisant l'API JPA. Nous réfléchissons à ajuster les objectifs d'apprentissage et l'organisation du cours afin de prendre en compte ces retours qui nous paraissent pertinents.

Les résultats obtenus pour l'épreuve orientée théorie sont représentées dans le tableau 3 dans lequel la fréquence des notes pour chaque intervalle de taille 2 est mentionnée. Les notes de cette épreuve écrite sont très bonnes. Ces résultats confortent les sentiments de compréhension et d'apprentissage exprimés par les étudiants durant l'enquête.

Table 3. Répartition des notes en fonction de l'épreuve

	0-2	2-4	4-6	6-8	8-10	10-12	12-14	14-16	16-18	18-20
Epreuve théorique	0	0	0	0	1	7	4	6	4	2
Epreuve pratique	2	1	0	4	3	1	2	1	2	8

Du côté de l'épreuve pratique, on observe des résultats plus hétérogènes : un tiers des étudiants obtiennent des résultats faibles ou très faibles (en dessous de 8) alors que la moitié d'entre eux obtiennent une très bonne note (supérieure à 15). Nous manquons actuellement d'éléments pour comprendre ce qui a généré des blocages chez les étudiants n'ayant pas réussi l'épreuve. Ceci est un indicateur majeur sur la nécessité d'améliorer le protocole PRaTDL ; nous proposons dans la dernière section les pistes d'amélioration en cours d'exploration.

5 Conclusion et perspectives

Le protocole PRaTDL a été conçu pour l'enseignement de notions complexes de programmation. Cette nouvelle approche, d'une part, propose aux étudiants des activités maintenant une culture TDD forte tout en limitant le temps passé par les étudiants à écrire des tests et, d'autre part, permet de couvrir un large rang d'objectifs d'apprentissage incluant ceux nécessitant des interactions sociales. Nous avons procédé à une première expérimentation du protocole PRaTDL dans le cadre d'un cours de Master en ingénierie logiciel. L'analyse de l'enquête menée auprès des étudiants ayant participé montre qu'ils ont perçus d'importants bénéfices dans la mise en application du PRaTDL. Les résultats obtenus aux évaluations sommatives confortent l'idée que le protocole conduit les étudiants à un apprentissage en profondeur. Ces résultats seront prochainement consolidés dans le cadre d'une expérimentation plus rigoureuse incluant un groupe témoin.

Les résultats obtenus à l'épreuve pratique révèle des comportements de blocage qu'il nous est difficile d'interpréter à partir des informations collectées dans le contexte de cette première expérimentation. L'introduction d'outils tels que Git nous aidera à tracer plus finement l'activité des étudiants durant les

phases TDD. En effet, l'enregistrement des contributions dans un dépôt Git permettra la mesure quantitative du rythme de progression de chaque étudiant, du temps passé sur chaque exercice et probablement des problématiques bloquantes récurrentes. Nous pourrons apprendre davantage sur la manière dont les étudiants sont engagés dans les activités individuelles afin d'identifier d'éventuels patrons d'apprentissage. Nous souhaitons utiliser ces nouveaux indicateurs pour améliorer le protocole PRaTDL ainsi que sa mise en œuvre.

Références

1. Bourque, P., Fairley, R.E., et al. : Guide to the software engineering body of knowledge (SWEBOK (R)) : Version 3.0. IEEE Computer Society Press (2014)
2. Stavru, S. : A critical examination of recent industrial surveys on agile method usage. Journal of Systems and Software **94** (2014) 87–97
3. Beck, K. : Embracing change with extreme programming. Computer **32**(10) (1999) 70–77
4. Beck, K. : Test-driven development : by example. Addison-Wesley Professional (2003)
5. Causevic, A., Sundmark, D., Punnekkat, S. : Factors limiting industrial adoption of test driven development : A systematic review. In : Software Testing, Verification and Validation (ICST), IEEE (2011) 337–346
6. Desai, C., Janzen, D., Savage, K. : A survey of evidence for test-driven development in academia. ACM SIGCSE Bulletin **40**(2) (2008) 97–101
7. Janzen, D.S., Saiedian, H. : Test-driven learning : intrinsic integration of testing into the cs/se curriculum. In : ACM SIGCSE Bulletin. Volume 38., ACM (2006) 254–258
8. Janzen, D., Saiedian, H. : Test-driven learning in early programming courses. In : ACM SIGCSE Bulletin. Volume 40., ACM (2008) 532–536
9. George, B., Williams, L. : A structured experiment of test-driven development. Information and software Technology **46**(5) (2004) 337–342
10. ACM/IEEE-CS Joint Task Force on Computing Curricula : Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press (2013)
11. Biggs, J. : Enhancing teaching through constructive alignment. Higher education **32**(3) (1996) 347–364
12. Mazur, E. : Peer instruction : getting students to think in class. In : AIP Conference Proceedings, IOP INSTITUTE OF PHYSICS PUBLISHING LTD (1997) 981–988
13. Crouch, C.H., Mazur, E. : Peer instruction : Ten years of experience and results. American journal of physics **69**(9) (2001) 970–977
14. Bhatnagar, S., Lasry, N., Desmarais, M., Dugdale, M., Whittaker, C., Charles, E.S. : An analysis of peer-submitted and peer-reviewed answer rationales, in an asynchronous peer instruction based learning environment. International Educational Data Mining Society (2015)
15. Silvestre, F., Vidal, P., Broisin, J. : 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. Springer (2015) 339–351