Vers une méthode pédagogique innovante pour former les étudiants aux approches agiles dans l'enseignement supérieur : A.L.P.E.S

Jannik Laval, Mathieu Vermeulen [1]

Introduction

En 2001, un groupe de développeurs en informatique de premier plan a conçu et rédigé le Manifeste agile [1]. Ce manifeste visait à proposer un nouveau paradigme pour le développement de logiciels autour de quatre principes fondateurs comme alternative aux méthodes classiques de gestion de projets informatiques :

— les individus et leurs interactions plus que les processus et les outils ;
— des logiciels opérationnels plus qu'une documentation exhaustive ;
— la collaboration avec les clients plus que la négociation contractuelle ;
— l'adaptation au changement plus que le suivi d'un plan.

Aujourd'hui, les approches et méthodes agiles (avec, à titre d'exemple, le célèbre *SCRUM* [3]) basées sur ce manifeste sont très répandues dans le monde du développement informatique mais aussi dans d'autres secteurs de l'entreprise.

Dès lors, il semble judicieux d'initier les futurs ingénieurs et les futurs cadres aux approches agiles. L'idée défendue n'est pas seulement d'enseigner la gestion de projet agile, mais d'intégrer cet enseignement dans une démarche pédagogique de manière transversale. Par ailleurs, la pédagogie par projet est largement utilisée dans

1. Université de Lyon, IUT, département QLIO.

DOI:10.48556/SIF.1024.21.93

l'enseignement supérieur et possède de nombreuses qualités intrinsèques. Cela dit, la question d'une pédagogie par projet intégrant les concepts des approches agiles se pose :

(i) comment les approches agiles pourraient-elles être intégrées dans l'apprentissage par projet ?

(ii) les approches agiles pourraient-elles être enseignées par la pédagogie par projet ?

Dans cet article, nous travaillons sur deux hypothèses : (1) un cours dans l'enseignement supérieur peut être transformé en un cours par projet ; (2) un modèle bien défini et la méthode associée peuvent aider les enseignants à remodeler leurs cours pour introduire des concepts agiles.

A cette fin, plusieurs travaux ont été menés intégrant des expérimentations de conception de cours basés sur des projets agiles. Ces derniers ont été proposés depuis septembre 2014 dans différentes formations et établissements [4, 5, 6].

Contexte

En passant en revue les expériences en matière d'apprentissage par projet [7, 8, 9] et de gestion de projet agile [3, 10], l'intégration des approches agiles dans l'éducation présente de nombreux avantages notamment d'adaptation de l'apprentissage au niveau de l'apprenant.

Apprentissage par projet

Dans le domaine de l'éducation, les projets sont définis comme des tâches complexes, basées sur des problèmes difficiles, qui impliquent les apprenants dans la conception, la résolution de problèmes, la prise de décision ou l'investigation. Dans un tel projet, un apprenant travaille seul ou forme un groupe de projet de manière autonome pendant une période déterminée. Un projet se termine par la remise d'un livrable.

Gestion de projet agile

Le Manifeste agile [1] énonce les bases d'un paradigme pour le développement des logiciels informatiques. Les approches agiles de gestion de projet sont nées d'un besoin en informatique d'adapter continuellement les projets aux besoins du client ou des utilisateurs, en mettant l'humain, et les interactions humaines, au centre du projet. La gestion de projet agile nécessite des concepts et des méthodes qui aident les différents participants (concepteurs, développeurs, utilisateurs finals, etc.) à communiquer pendant les phases du projet et à s'interroger sur les prochains développements à réaliser.

Une caractéristique clé de la gestion de projet agile est de permettre aux équipes de développer leur projet de manière itérative. Une difficulté centrale commune réside dans la définition des composants du projet à développer de manière à rendre possible un processus itératif.

Les outils issus des approches agiles peuvent être utilisés dans une grande variété de situations. Principalement, certains outils comme le *planning board*, permettent d'organiser et de suivre le projet.

Pour aider les développeurs et les chefs de projet à créer une expérience agile, *SCRUM* [3] fournit un cadre et des outils pour suivre l'avancement du projet.

Approches agiles dans l'enseignement supérieur

SCRUM a été la base de certaines expériences comme Lego4SCRUM[11] ou les cours d'enseignement du génie logiciel [12]. Dans ces approches, le but est d'enseigner directement la gestion de projet agile : c'est l'objectif pédagogique principal de ces cours. Il semble difficile d'adapter ces exemples à des disciplines autres que l'informatique.

A.L.P.E.S. [4, 6] est une adaptation de ces approches et de leurs outils associés pour une construction pédagogique. Tirant parti à la fois du paradigme de la pédagogie par projet et des principes de base de la programmation en binôme (méthode de programmation à deux sur une même machine : un *driver* qui écrit le code, et un observateur qui assiste et corrige le *driver*) [13], A.L.P.E.S. définit les séances de travail sous forme de travaux pratiques et met l'accent sur le travail collaboratif en groupe. La véritable différence d'A.L.P.E.S. par rapport aux approches existantes est que le cours est lui-même construit sous forme de projet. Alors que les approches par projet mettent les apprenants en situation de projet, A.L.P.E.S. met le cours entier en projet. Cela inclut donc l'enseignant, et permet au cours d'être évolutif et réflexif. Le reste de cet article se concentre sur les notions de développement itératif et de *user stories* utilisés dans A.L.P.E.S.

A.L.P.E.S. à travers les notions et outils agiles

La plupart des outils présentés ici sont issus de projets agiles et plus particulièrement de la méthode *SCRUM* [14]. Plusieurs solutions logicielles et applications web sont disponibles permettant de les mettre en pratique. Cependant, cette section porte davantage sur la description de la philosophie de l'outil que sur sa mise en œuvre à travers des dispositifs, des applications ou des services web.

User story

Dans l'approche *SCRUM*, une *user story* représente une fonctionnalité d'une application logicielle. Elle est représentée par une phrase simple qui décrit l'attente du composant à développer. Une *user story* illustre le besoin d'un utilisateur. Chaque *user story* est noté sur un Post-it. La *user story* est un élément central de l'approche

A.L.P.E.S., car elle permet à l'enseignant de réorganiser un cours en fonction de son application réelle, d'ajouter ou d'enlever des compétences en fonction de la vélocité de l'apprenant.

Pour concevoir une *user story*, l'enseignant doit identifier le « besoin pédagogique » et écrire une phrase avec cette structure : « En tant que *[rôle]*, je veux *[faire une tâche]*, afin que *[un but de l'action]* ». De cette façon, la *user story* est proche d'un scénario et d'une concrétisation d'un objectif pédagogique.

Pour écrire correctement une *user story*, nous recommandons qu'elle soit écrite en suivant le principe INVEST :

— indépendant : chaque *user story* doit être aussi indépendant que possible des autres afin de pouvoir les réaliser dans n'importe quel ordre ;
— négociable : tant qu'elle n'a pas été commencée, il doit être possible de modifier une *user story* ;
— valeur : la *user story* doit apporter une valeur pédagogique à l'apprenant ;
— estimable : une *user story* doit être estimable en termes de complexité ;
— suffisament petite (*small*) : plus la *user story* est courte, plus elle sera simple et claire ; l'apprenant gagne en confiance ;
— testable : pour chaque *user story*, des critères de tests objectifs doivent être mis en place pour vérifier que les connaissances assimilées sont correctes.

Dans le cadre d'un cours, une *user story* représentera la mise en œuvre d'un ou plusieurs objectifs pédagogiques. Cet objectif pourra être clairement établi pour les apprenants sur le « but de l'action ».

Décomposition temporelle : sprint

Dans la terminologie *SCRUM*, un *sprint* identifie une itération dans le développement logiciel. Les *sprints* sont traités entre deux versions du logiciel planifiées à des échéances régulières. Un *sprint* est principalement composé de sessions de développement entre une session de *sprint planning* et une session de *sprint review*. Le *sprint planning* consiste à définir quelles fonctionnalités (*user stories*) seront développées pendant le temps alloué. Le *sprint review* consiste à présenter les éléments développés, idéalement avec des démonstrations fonctionnelles. Ces deux éléments qui délimitent un *sprint* incluent l'enseignant dans un but pédagogique. Un sprint, dans A.L.P.E.S., est considéré comme une séance de cours. Il dure généralement deux ou quatre heures.

Suivi du projet

Les outils de suivi du projet permettent à l'équipe de visualiser la direction ou les objectifs du projet, l'état actuel de son développement et son efficacité. Ces outils peuvent simplement prendre la forme de documents dans le répertoire du projet.

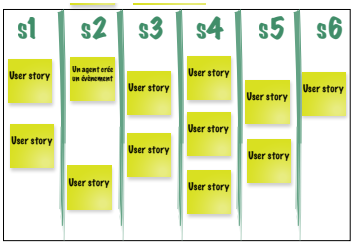

FIGURE 1. A.L.P.E.S. *planning board.*

Le planning board Il est composé de colonnes représentant chaque *sprint* (cf. figure 1). Chaque colonne contient les *user stories*, représentées sous forme de Post-it. Ce tableau permet à tout moment d'avoir une vue d'ensemble de l'avancement de l'apprenant : ce qui a déjà été fait (éléments des sprints précédents), ce qui est en cours de traitement (éléments du *sprint* actuel) et ce qui devra être réalisé (éléments des *sprints* futurs). Il permet de suivre la progression globale et donne aux apprenants une visibilité sur les objectifs pédagogiques. Le *planning board* est mis à jour uniquement pendant la session *sprint planning* avec la possibilité de modifier les colonnes cartographiant les *sprints* actuels et futurs.

Le task board Il contient trois colonnes (cf. figure 2) : *To Do*, *Doing* et *Done*. Chaque colonne comprend les *user stories* et les tâches associées en fonction de leur statut : pas encore réalisé, en cours de traitement ou terminé. La colonne *Doing* est également composée d'une zone *Help* qui est utile pour l'interaction avec l'enseignant. La principale raison pour laquelle les apprenants mettent des éléments en attente est qu'ils ont besoin de l'aide d'un enseignant ou d'une validation. Au cours d'un *sprint*, les Post-it modélisant les *user stories* et les tâches navigueront d'une zone à l'autre, de gauche à droite. Le *task board* permet de visualiser le statut des apprenants dans une session.

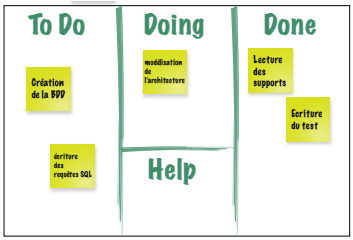

FIGURE 2. A.L.P.E.S. *task board.*

Bonnes pratiques

Le développement logiciel agile fournit un environnement favorable à l'expérimentation de « bonnes pratiques ». Au-delà d'une présentation de ces pratiques aux apprenants afin de les préparer à leur future carrière, ces pratiques apportent une valeur ajoutée, d'un point de vue pédagogique.

Programmation en binôme La programmation en binôme est une technique de développement logiciel agile dans laquelle deux développeurs travaillent ensemble sur un unique poste de travail. L'un, le pilote, écrit le code tandis que l'autre, l'observateur ou le navigateur, révise chaque ligne de code au fur et à mesure qu'elle est écrite. Les deux personnes changent fréquemment de rôle [13]. Il est important de présenter aux apprenants travaillant ensemble sur un ordinateur unique les bénéfices de cette pratique.

Développement piloté par les tests L'idée principale définissant le développement piloté par les tests est que des tests doivent être mis en place avant de commencer tout développement. Cela permet aux apprenants de se poser les bonnes questions concernant la *user story* qu'ils vont réaliser. De plus, en effectuant fréquemment des tests (au moins un test par *user story*) les apprenants sont capables d'identifier les problèmes avant d'accumuler trop d'erreurs dans l'avancement de leur projet.

Versioning Il consiste à fixer les étapes de l'avancement du projet. Il permet aux apprenants de revenir en arrière en toute sécurité. Grâce à ce mécanisme, il est également intéressant de visualiser l'évolution des projets. En second lieu, le *versioning* aide les apprenants à travailler sur des ressources partagées.

Présentation des processus d'A.L.P.E.S.

A.L.P.E.S. se compose de deux processus : l'un décrivant le déroulement d'un cours, l'autre la création du cours. Nous vous présentons ici le processus décrivant le déroulement d'un cours. L'autre processus est décrit dans l'article [6].

Une session de cours est organisée en trois parties : (i) le début de la session où l'enseignant organise les objectifs et les apprenants organisent leur temps, (ii) pendant la séance où il s'agit d'organiser la session en cours, (iii) après la séance où l'enseignant prépare la prochaine session sur la base de l'avancement des apprenants. La figure 3 montre le processus tel qu'il devrait être exécuté. Chaque tâche a un objectif, des entrées et des sorties, et des participants. Dans chaque tâche, la lettre T (*Teacher*) ou S (*Student*) indique qui est le participant à la tâche.

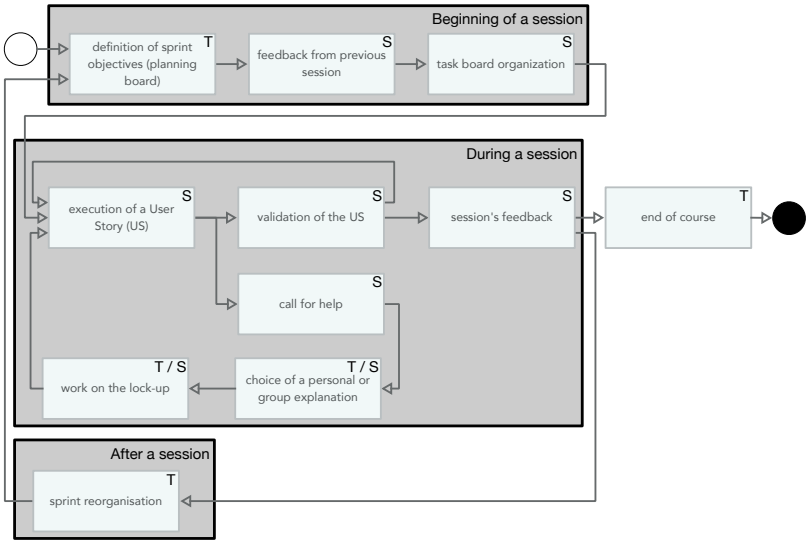

FIGURE 3. Déroulement d'une session A.L.P.E.S. (S pour *Student* et T pour *Teacher*).

Début d'une session. Ce groupe de tâches est l'organisation de la session. Il prend 5 à 10 minutes :

— définition des objectifs du *sprint* (*planning board*) : l'objectif de cette tâche est d'expliquer aux apprenants ce qu'ils vont réaliser au cours de la session ; cette tâche est soutenue par le *planning board* :

- l'entrée de la tâche est le *planning board* ;
- le résultat est un ensemble de *user stories* que les apprenants devront compléter ;
- l'acteur est le professeur ; les apprenants suivent les instructions.

— *feedback* des sessions précédentes : une fois que les apprenants ont une vision des *user stories* à réaliser, ils vont faire le lien avec les connaissances et compétences acquises lors des sessions précédentes. C'est à ce moment-là que le *feedback* de la session précédente intervient. Ce *feedback* est intéressant à ce moment car il permet de relier les sessions entre elles et de remobiliser les connaissances et compétences acquises précédemment :

- l'entrée de cette tâche est l'ensemble des *user stories* à réaliser pendant la session, dans les outils, cela représente une colonne du *planning board* ;
- la sortie est la même liste de choses à faire que l'entrée ;

— les acteurs sont les apprenants ; leur activité durant cette phase est indispensable à la bonne consolidation des connaissances. L'enseignant est présent pour répondre aux questions ;

— organisation du *task board* : tous les apprenants sélectionnent les *user stories* à faire dans la colonne correspondant à la session du jour, et placent cette liste dans le *task board*, dans la colonne *To Do*. Dans cette liste sont également ajoutées les *user stories* qui n'ont pas été réalisées lors de la session précédente (celles en retard) :

— l'entrée de cette tâche est l'ensemble des *user stories* dans une colonne du *planning board* ;
— la sortie est le *task board* prêt à être utilisé pour la session ;
— les acteurs sont les apprenants ;

Pendant une session

— exécution d'une *user story* : l'apprenant choisit une tâche dans la colonne *To Do* du *task board*, et la place dans la colonne *Doing*. Comme les tâches sont indépendantes, il peut choisir celle qu'il veut, sachant qu'elles devront toutes être finalisées. Quand il a terminé, il peut valider la *user story*. S'il a un problème, il demande de l'aide en plaçant la *user story* dans la colonne *Help* :

— l'entrée est le *task board* dans la configuration suivante : la colonne *To Do* contient les *user story* ; la colonne *Doing* est vide ; la colonne *Done* peut contenir des tâches qui ont déjà été réalisées ;
— le résultat est le *task board* avec une *user story* placée soit dans la colonne *Doing* soit dans la colonne *Help* ;
— les acteurs sont les apprenants ;

— validation de la *user story* : l'apprenant valide la *user story* ; en d'autres termes, selon le sujet du cours, la validation peut se faire par l'exécution d'une procédure de test, manuellement par le contrôle effectif de l'enseignant ou par l'exécution d'un jeu de tests automatisé :

— l'entrée est la *user story* réalisée précédemment ;
— la sortie est le *task board* mis à jour : soit la *user story* est validée et va dans la colonne *Done*, soit elle n'est pas validée et doit être retravaillée ;
— les acteurs sont les apprenants.

— *feedback* de la session : lorsque toutes les *user stories* de la colonne *To Do* sont déplacées vers la colonne *Done*, ou lorsque la fin de la session est arrivée, les apprenants écrivent le *feedback* de ce qu'ils ont appris de la session ; le feedback est organisé de la manière suivante : « ce que j'ai appris », « ce que je ne comprends pas », « ce que j'ai résolu » ; en outre, les apprenants en profitent pour mettre à jour leur *task board* et leur *planning board* actualisé : les *user stories* de la colonne *Done* du *task board* sont déplacées vers la colonne de la session en cours du *planning board* ; les *user stories* des autres colonnes passent dans la colonne de la session suivante du *planning board* ; en parallèle, les apprenants mettent à jour les tableaux relatifs au projet :

— l'entrée est le *task board*, mis à jour à la fin de la session ;
— la sortie est un *task board* vide, un *planning board* mis à jour, un *feedback* sous forme textuelle ;
— les acteurs sont les apprenants.

— appel à l'aide : lorsqu'un apprenant est bloqué dans une *user story* à réaliser, il peut demander de l'aide au professeur via le *task board*. Il peut alors commencer une autre *user story* en attendant les réponses de l'enseignant :
— l'entrée est une *user story* sur laquelle l'apprenant a des difficultés ;
— la sortie est un *task board* mis à jour, avec la *user story* positionnée à l'endroit approprié ;
— l'acteur est l'apprenant.

Après une session L'enseignant adapte les différentes *user stories*. Il doit ensuite déplacer les *user stories* sur le planning, en les faisant avancer ou reculer en fonction de la progression des apprenants. Il peut également ajouter des *user stories* qui lui semblent nécessaires pour améliorer la pédagogie et la validation des compétences :
— l'entrée est le *feedback* de la session. C'est donc une évaluation qualitative via ce *feedback* et une évaluation quantitative via l'analyse des tableaux de tâches que l'enseignant doit réaliser ;
— la sortie est un planning actualisé. Il sera utilisé pour la prochaine séance ;
— l'acteur est l'enseignant.

Fin du cours La réorganisation des *sprints* concerne l'adaptation par l'enseignant des différentes *user stories* en fonction d'une évaluation qualitative et quantitative. Elle conduit à mettre à jour chaque colonne du *planning board* pour les cours suivants. Ainsi, d'année en année, l'enseignant capitalise et peut affiner ses cours en étant au plus près des besoins des apprenants :
— l'entrée est le *feedback* de la séance et le *planning board* modifié tout au long du cours ;
— la sortie est un tableau de planning mis à jour ;
— l'acteur est l'enseignant.

Conclusions

Dans cet article, nous avons présenté le concept d'approche agile dans l'enseignement, le déroulement d'un cours en A.L.P.E.S. et la définition des processus permis par l'utilisation complète de l'approche dans un cadre pédagogique. La démarche vient du terrain, comme toutes les démarches agiles. Nous avons construit l'approche A.L.P.E.S. en explorant les différents principes et les différents outils des approches agiles. Nous avons pu identifier trois classes principales, à savoir (1) les principes de temps, (2) les principes de surveillance et (3) le principe de rôle, à partir des objets agiles que nous avons étudiés. Par conséquent, un cours dans le sens de l'approche

A.L.P.E.S. est composé d'un ensemble d'éléments provenant des différentes classes de principes identifiés.

Dans les articles précédents [4, 5, 6], nous avons abordé la création d'un cours en A.L.P.E.S. et les retours des apprenants. Les définitions présentées nous fournissent un cadre pour établir les conditions minimales à la réalisation d'un cours en format A.L.P.E.S.

Références

[1] Beck, K. ; Beedle, M. ; Van Bennekum, A. ; Cockburn, A. ; Cunningham, W. ; Fowler, M. ; Grenning, J. ; Highsmith, J. ; Hunt, A. ; Jeffries, R. ; et al. Manifesto for Agile Software Development. Technical Report, 2001. Available online : https://agilemanifesto.org/ (accessed on 30 June 2022).

[3] Schwaber, K. ; Beedle, M. *Agile Software Development with Scrum* ; Prentice Hall : Upper Saddle River, NJ, USA, 2002 ; Volume 1.

[4] Vermeulen, M. ; Fleury, A. ; Fronton, K. ; Laval, J. LES ALPES : Approches agiles pour l'enseignement supérieur. In Proceedings of the Colloque Questions de Pédagogies dans l'Enseignement Supérieur (QPES 2015), Brest, France, June 2015 ; pp. 243–248.

[5] Vermeulen, M. ; Laval, J. ; Serpaggi, X. ; Pinot, R. Soyez agiles dans les ALPES ! Une pédagogie en mode agile. In Proceedings of the Colloque Questions de Pédagogie dans l'Enseignement Supérieur (QPES 2017), Grenoble, France, June 2017.

[6] Laval, J. ; Fleury, A. ; Karami, A.B. ; Lebis, A. ; Lozenguez, G. ; Pinot, R. ; Vermeulen, M. Toward an Innovative Educational Method to Train Students to Agile Approaches in Higher Education : The A.L.P.E.S. Educ. Sci. 2021, 11, 267. https://doi.org/10.3390/educsci11060267

[7] John, W. ; Thomas, W. *A Review of Research on Project-Based Learning* ; The Autodesk Foundation : San Rafael, California , USA, March 2000.

[8] Karabulut-Ilgu, A. ; Jaramillo Cherrez, N. ; Jahren, C.T. A systematic review of research on the flipped learning method in engineering education. *Br. J. Educ. Technol.* 2018, *49*, 398–411.

[9] Kingston, S. *Project Based Learning & Student Achievement : What Does the Research Tell Us* ? PBL Evidence Matters ; Buck Institute for Education, Novato, CA : 2018 ; Volume 1, No. 1.

[10] Highsmith, J. ; Cockburn, A. Agile software development : The business of innovation. *Computer* 2001, *34*, 120–127.

[11] Ouitre, F. ; Lambert, J.L. Le lego4scrum, un dispositif agile pour enseigner le management de projet-Innovation Pédagogique. In Proceedings of the Colloque Questions de Pédagogie pour l'Enseignement Supérieur (QPES 2015), Brest, France, 21 June 2015.

[12] Mahnič, V. Scrum in software engineering courses : an outline of the literature. *Glob. J. Eng. Educ.* 2015, *17*, 77–83.

[13] McDowell, C. ; Werner, L. ; Bullock, H. ; Fernald, J. The effects of pair-programming on performance in an introductory programming course. In Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, New York, NY, USA, February 2002 ; pp. 38–42.

[14] Sutherland, J. ; Schwaber, K. The scrum guide. In*The Definitive Guide to Scrum : The Rules of the Game* ; https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf accessed 27 Mai 2021, November 2017.