PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Réorganisation de l'enseignement dans un cours introductif en analyse et conception de logiciels
Auteurs: Authors:	Éric Germain, François Guibault, & Nikolay Radoev
Date:	2024
Type:	Communication de conférence / Conference or Workshop Item
Référence: Citation:	Germain, É., Guibault, F., & Radoev, N. (2024, June). Réorganisation de l'enseignement dans un cours introductif en analyse et conception de logiciels [Reorganization of teaching in an introductory course in software analysis and design]. [Paper]. Canadian Engineering Education Association (CEEA-ACEG 2024), Edmonton, Alberta, Canada (7 pages). Published in Proceedings of the Canadian Engineering Education Association (CEEA). https://doi.org/10.24908/pceea.2024.18581

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/62207/
Version:	Version officielle de l'éditeur / Published version Révisé par les pairs / Refereed
Conditions d'utilisation: Terms of Use:	Creative Commons Attribution-Utilisation non commerciale 4.0 International / Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC)

Document publié chez l'éditeur officiel
Document issued by the official publisher

Nom de la conférence: Conference Name:	Canadian Engineering Education Association (CEEA-ACEG 2024)
Date et lieu: Date and Location:	2024-06-15 - 2024-06-19, Edmonton, Alberta, Canada
Maison d'édition: Publisher:	Surveillance Studies Network
URL officiel: Official URL:	https://doi.org/10.24908/pceea.2024.18581
Mention légale: Legal notice:	This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (https://creativecommons.org/licenses/by-nc/4.0/).

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

RÉORGANISATION DE L'ENSEIGNEMENT DANS UN COURS INTRODUCTIF EN ANALYSE ET CONCEPTION DE LOGICIELS

Éric Germain, François Guibault et Nikolay Radoev
Polytechnique Montréal
{eric.germain, francois.guibault, nikolay.radoev} @ polymtl.ca

Résumé – *Le Département de génie informatique et génie logiciel de Polytechnique Montréal propose un cours introductif d'analyse et de conception de logiciels aux étudiants au début de leur cursus en génie informatique et logiciel. Leurs réactions montrent qu'ils aimeraient consacrer plus de temps à des exemples et exercices. En parallèle, les étudiants de 2e et 3e année semblent oublier certaines notions clés de modélisation logicielle apprises précédemment. Cet article présente un projet qui a été entrepris pour résoudre ces deux problèmes. Une séance de tutorat d'une heure par semaine a été insérée dans le cours introductif, en plus des heures en classe et de laboratoire. Nous avons également développé un site web accessible au public qui présente les notions importantes du Unified Modeling Language (UML), le langage de modélisation de logiciels le plus courant. Les résultats indiquent des pistes d'amélioration. Le projet a mis en évidence la nécessité de clarifier certaines questions fondamentales concernant l'enseignement de l'UML.*

Abstract – *The Department of Computer and Software Engineering at Polytechnique Montréal offers a basic software analysis and design course to students at the beginning of their coursework in computer and software engineering. Feedback shows that they would like to spend more time studying examples and completing exercises. In parallel, students in their 2nd and 3rd years seem to forget some key software modelling notions learned earlier. This paper presents a project that was undertaken to address both issues. First, a one-hour tutorial session per week was introduced in the basic software analysis and design course, in addition to the hours in the classroom and laboratory. Second, we developed a publicly available website that presents the important notions of the Unified Modeling Language (UML), the most common software modelling language. Results point to further possible improvements. The project highlighted the need to clarify some fundamental questions regarding the teaching of UML.*

Keywords: Software engineering, modelling, active learning.

1. INTRODUCTION

Cet article présente deux innovations pédagogiques récentes touchant la formation à la modélisation en génie logiciel au sein des programmes de formation de l'ingénieur à Polytechnique Montréal, soit l'ajout de séances de travaux dirigés au cours de base en analyse et conception de logiciels, ainsi que le développement d'un guide en ligne visant à favoriser l'apprentissage actif du *Unified Modeling Language* [1], un langage de modélisation d'usage courant en génie logiciel. Ces innovations ont été mises en branle dans le cadre d'une démarche d'amélioration continue et font suite aux insatisfactions des étudiants et des enseignants quant à l'enseignement des aspects d'analyse et de conception en génie logiciel. Les sondages menés auprès des étudiants ayant été exposés aux deux innovations comportent un nombre de répondants trop faible pour permettre de tirer des conclusions significatives, mais offrent tout de même un portrait positif tout en pointant vers des pistes d'amélioration prometteuses à moyen terme.

2. LA FORMATION EN GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL À POLYTECHNIQUE MONTRÉAL

Les douze programmes de baccalauréat en ingénierie offerts à Polytechnique Montréal constituent la pierre d'assise de son offre de formation. Parmi ceux-ci figurent le baccalauréat en génie informatique et le baccalauréat en génie logiciel, tous deux chapeautés par le Département de génie informatique et génie logiciel. Ces deux programmes, d'une durée normale de quatre années d'études, ont beaucoup en commun; les deux premières années sont quasiment identiques, ce qui permet aux étudiants d'obtenir une base solide couvrant à la fois les enjeux matériels et logiciels du secteur des technologies de l'information. L'une des conséquences de cette organisation est qu'elle nécessite une réflexion quant à l'ordonnancement des notions à couvrir tout au long des deux programmes. Par exemple, on s'attendra à ce que certaines notions en génie logiciel (mais pas d'autres)

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

soient couvertes suffisamment en profondeur en début de programme, de manière que le diplômé en génie informatique développe un minimum de compétences sur le sujet.

Le cours obligatoire d'analyse et de conception de logiciels a justement comme vocation d'initier les étudiants des deux programmes à ces aspects centraux du génie logiciel. Ce cours est normalement suivi à la suite de deux cours de programmation, dont un cours de programmation avancée en langage C++. Il couvre plusieurs sujets, notamment : introduction au génie logiciel, principes fondamentaux en analyse et conception de logiciels, conception architecturale, patrons de conception. C'est dans ce cours, notamment, qu'on initie les étudiants à la modélisation orientée-objet et à l'utilisation d'une notation à cette fin. Il en résulte un cours assez chargé, généralement apprécié des étudiants, mais parfois perçu comme trop théorique. C'est que, comme c'est souvent le cas au baccalauréat en génie à Polytechnique, le cours était, jusqu'à récemment, donné en deux segments : un bloc théorique de trois heures chaque semaine sous la forme d'un cours magistral donné par le professeur, et une séance de travaux pratiques de trois heures à toutes les deux semaines sous la supervision d'un auxiliaire d'enseignement (généralement un étudiant plus avancé). Le cours n'a ainsi pas été conçu autour d'approches actives pour sa portion théorique, et la grande quantité de matériel à couvrir limite les ajustements pouvant être apportés en ce sens. Les commentaires recueillis auprès des étudiants montrent qu'ils souhaiteraient passer davantage de temps à étudier des exemples et à réaliser des exercices.

Par ailleurs, les programmes en génie informatique et génie logiciel, comme tous les programmes de baccalauréat à Polytechnique, comportent quatre projets intégrateurs qui visent à permettre aux étudiants d'intégrer les connaissances acquises au cours de chacune des quatre années d'études. Les projets sont réalisés en équipe et leur complexité et le niveau d'autonomie demandé aux étudiants augmentent avec chaque projet jusqu'à aller à un projet avec l'industrie à la 4e année.

Le *projet de logiciel d'application Web* constitue le projet intégrateur de deuxième année, et est normalement suivi au trimestre immédiatement après le cours d'analyse et conception. Ce projet présente la première occasion de mettre en pratique les notions théoriques vues durant les deux premières années. L'objectif du cours est de livrer un produit (site web, serveur dynamique et base de données) à partir d'un projet de départ et une description sommaire du résultat final attendu. Le projet de départ comprend une architecture de base servant de référence, et les étudiants sont amenés à appliquer les principes de conception pour faire évoluer ce projet vers le système final requis. Bien que le cours ne prévoie pas de livrable concret en termes de conception ou de modélisation formelle, il est fortement recommandé aux étudiants d'utiliser la première phase du projet pour établir une conception préliminaire avant de passer à la phase de développement.

Le *projet d'évolution d'un logiciel* constitue le projet intégrateur de troisième année pour tous les étudiants en génie logiciel. Son objectif est d'intégrer les connaissances et compétences acquises en analyse, conception et tests du logiciel ainsi qu'en réseautique. À cette étape, on s'attend à ce que les étudiants possèdent tous les outils méthodologiques propres au domaine du génie logiciel. En outre, les étudiants doivent livrer un produit fonctionnel (généralement un jeu) qui réponde à la fois à des exigences « négociées » avec l'équipe académique et aux attentes implicites « normales » envers un système logiciel de ce type. Le cours-projet prévoit cependant la production d'un *document d'architecture logicielle* selon un modèle inspiré du *Unified Process for Education* (UPEDU) [2], lui-même dérivé du *Rational Unified Process* [3]. Cette activité vise spécifiquement l'intégration des compétences en modélisation et communication acquises dans le cours d'analyse et conception de logiciels. Soulignons au passage qu'il s'écoule normalement une année complète entre la complétion de ce dernier cours et l'inscription au cours-projet. Ce délai frôlait d'ailleurs deux ans jusqu'à un réaménagement récent des programmes de baccalauréat.

3. LE LANGAGE UML

Le *Unified Modeling Language* (UML) est un langage de modélisation de systèmes logiciels orientés-objets, populaire dans l'écosystème de la pratique et de la formation en génie logiciel. Il est notamment normalisé par l'Organisation internationale de normalisation (ISO) [4]-[5], et mentionné dans les cadres de référence SWEBOK 3.0 [6] et SE2014 [7]. L'UML est enseigné à Polytechnique depuis près de 25 ans, soit quasiment depuis l'avènement de la version 1.0. Budgen [8, p. 173] indique : « Essentially, the only widely documented source of object modelling notations is the UML. », et il serait difficile d'envisager des alternatives à l'UML aux fins d'enseignement de la modélisation en génie logiciel. Mais la popularité de l'UML ne le rend pas imperméable aux critiques. Considérons à titre d'exemple ce constat plutôt sévère:

> UML 2.0, for example, a major revision of the UML standard, didn't reflect the literature on empirical studies of software modelling or software design studies. Consequently, current approaches force developers and organisations to operate in a way that fits the approach instead of making the approach fit the people. [9, p. 85]

En somme, l'UML est un langage de modélisation à la fois incontournable et imparfait. Il importe ainsi de bien planifier son positionnement dans les programmes de formation en génie informatique et génie logiciel.

4. LA PROBLÉMATIQUE ET LES OBJECTIFS

En aval dans le programme, les cours-projets intégrateurs de 2e et 3e année constituent autant d'occasions de mettre en pratique les notions acquises dans les cours précédents. Ces cours-projets sont naturellement orientés vers l'apprentissage expérientiel. Toutefois, dans le cas de l'UML, l'absence d'une référence stable (et, de surcroit, en français) sur l'UML empêche les étudiants de réaliser une mise en pratique valable. C'est que la norme UML en vigueur [5] est très volumineuse et couvre bien davantage de notions que ce qu'il est souhaitable (voire possible) de voir durant les deux ou trois premières années d'études. Également, les références existantes sur la norme en vigueur, telles que celles prévues aux fins de l'obtention d'une certification professionnelle, sont souvent vagues ou encore se concentrent sur des aspects somme toute secondaires. Ce contexte tend à désorienter les étudiants, qui vont se rabattre sur un certain nombre de ressources informelles sur l'UML disponibles sur le web. Ces ressources ne sont pas toutes fiables ou à jour.

Plus en amont dans le programme, le cadre rigide du cours et la quantité de notions à couvrir rendaient très difficile la mise en place d'activités d'apprentissage expérientiel, et imposaient le développement d'occasions additionnelles (i.e. en sus des heures de classe et de travaux pratiques). Par ailleurs, la lacune en matière de référence stable sur l'UML déjà mentionnée s'avère aussi problématique pour les étudiants en début de programme.

Ce constat a mené à la mise en branle d'un projet visant à améliorer la qualité des apprentissages en analyse et conception de logiciels. Nous avons décomposé cet objectif en une hiérarchie de sous-objectifs :

- SO1 : Améliorer l'expérience d'apprentissage dans le cours introductif d'analyse et de conception de logiciels.
    - SO1.1 : Favoriser l'apprentissage autonome de l'UML.
    - SO1.2 : Accroître la composante expérientielle de l'apprentissage.
- SO2 : Fournir une référence stable et adaptée sur l'UML qui soit disponible tout au long du programme.

Ces sous-objectifs sont mis en correspondance avec deux moyens pédagogiques. Ainsi :

- La *mise en place de séances de travaux dirigés* dans le cours introductif d'analyse et de conception de logiciels vise à accroître la composante expérientielle de l'apprentissage en offrant l'occasion, chaque semaine, de mettre en pratique les notions couvertes à la séance précédente.
- Le *développement d'un nouveau site web portant sur le langage UML* permettra à la fois aux étudiants en début de parcours de pratiquer un apprentissage autonome du langage, et aux étudiants de tous niveaux de bénéficier d'une référence sur l'UML en français, qui soit stable, de qualité et adaptée au contexte de nos programmes.

5. LES SÉANCES DE TRAVAUX DIRIGÉS

La grande densité de matière à couvrir dans le cours d'analyse et conception de logiciels rendait très difficile l'intégration d'un aspect expérientiel dans les séances théoriques. L'équipe de projet s'est donc rabattue vers l'ajout d'une séance hebdomadaire de travaux dirigés. D'une durée d'une heure seulement, ces séances, menées par une équipe de deux auxiliaires d'enseignement, sont strictement orientées vers la résolution en petits groupes d'étudiants d'exercices portant sur la dernière séance théorique réalisée.

Les exercices sont de deux types: des problèmes simples indépendants les uns des autres, mais également des missions d'application des notions discutées dans le cadre d'un cas servant de fil conducteur tout au long du trimestre. L'équipe pédagogique a en effet développé, au fil des ans, un petit nombre de cas techniques représentant des situations s'approchant de la réalité et pour lesquelles un système logiciel a été « commandé » par le « client ». Ces cas comportent un document de vision qui synthétise le besoin à très haut niveau; des artéfacts d'analyse et de conception, principalement en UML, qui fournissent des exemples d'application d'une démarche de développement; et, dans certains cas, une implémentation en langage C++ ou Python. Par ailleurs, ces cas sont construits de manière à favoriser naturellement l'intégration des patrons de conception [10] vus en classe.

6. LE GUIDE UML

Le développement du site web constituant le Guide UML s'est amorcé avec le choix d'une plateforme technologique. Le Guide devait pouvoir être mis à jour et redéployé facilement; être accessible non seulement aux étudiants, mais au public en général, et être relativement à l'abri de l'obsolescence technologique. Le site web est disponible publiquement à tous à travers le projet GitHub et peut être accédé par les étudiants ou toute autre personne qui désire y avoir accès.

Nous avons choisi d'adopter le format Markdown (MD) pour la rédaction du contenu textuel. Chaque page du site est donc composée d'un ou plusieurs documents MD séparés en sections sur un sujet spécifique. Certaines sections sont également accompagnées de diagrammes UML pour mieux illustrer leur contenu.

Les diagrammes UML ont été élaborés à l'aide de l'outil PlantUML et convertis en images SVG pour leur

intégration dans les pages. Cette approche permet de définir le contenu à l'aide de texte simple, facilitant ainsi son édition sans l'aide d'outils supplémentaires. Il est possible d'exporter les diagrammes vers d'autres outils, pourvu que le format PlantUML est supporté par ceux-ci.

Le projet repose également sur l'utilisation de Jekyll, un générateur de sites statiques, pour transformer les fichiers Markdown en pages web et construire le site. Un menu de navigation est ajouté à chaque page et permet de naviguer à travers les différentes sections. Le code source et le site web sont hébergés sur GitHub, permettant un déploiement automatisé à chaque mise à jour du contenu. En effet, il suffit d'ajouter nos modifications sur GitHub pour déclencher une mise à jour du site via GitHub Actions, le processus prenant en moyenne moins de 60 secondes. Cette approche garantit une grande réactivité pour les corrections et les mises à jour du contenu.

Ces choix s'inscrivent dans certaines tendances récentes en la matière. Par exemple l'ouvrage de Furhman et Ross sur l'analyse et la conception de logiciels [11] utilise également le format Quarto Markdown (QMD), qui est une variante du format MD, ainsi que PlantUML pour la génération des diagrammes. L'ouvrage est également disponible sur la plateforme GitHub et accessible de la même manière que notre site web.

L'ouvrage de Pilone et Pitman sur l'UML [12] a été sélectionné comme référence pour la construction du site étant donné son degré intéressant de complétude ainsi que sa présence dans la liste des références recommandées par l'Object Management Group pour l'obtention de la certification UML 2 de niveau « intermédiaire ». Plus spécifiquement, la structure de navigation du site a été fortement inspirée de la table des matières de l'ouvrage. Elle a été complétée par l'ajout d'un cas maison appelé PolyAuto, construit autour de l'idée d'un système de réservation de véhicules en libre-service. Le cas comporte une description des fonctionnalités recherchées ainsi qu'un exemple d'application pour chacun des types de diagramme couverts par le site, soit les diagrammes de classes, de paquetages, de composantes, de déploiement, de cas d'utilisation, d'activités, d'interaction et d'états. La figure 1 montre l'exemple de diagramme de classes fourni.

7. RÉSULTATS ET DISCUSSION

Afin de mieux comprendre les impacts des mesures proposées, nous avons mené deux sondages auprès de deux groupes d'étudiants. Le premier groupe était formé d'étudiants ayant complété le cours de base en analyse et conception de logiciels qui avaient été exposés à la fois aux séances de travaux dirigés et au Guide UML. Le taux brut de participation à ce premier sondage (i.e. sans tenir compte de la non-complétion du sondage par certains répondants) est de 15 étudiants sur 70 inscrits. Le second groupe était formé d'étudiants à mi-parcours du cours-projet de troisième année en génie logiciel qui avaient été exposés au Guide UML. Le taux brut de participation à ce deuxième sondage est de 30 étudiants sur 126 inscrits.

Le faible nombre de répondants limite grandement la portée des conclusions qu'on peut tirer de ces sondages. Il permet toutefois de tirer des indications générales susceptibles de guider les travaux subséquents. Les commentaires reçus semblent pertinents à cet égard.

7.1. Séances de travaux dirigés

Les auxiliaires d'enseignement qui ont mené les séances de travaux dirigés ont fait état des commentaires généralement positifs des étudiants qui ont participé aux séances. Le sondage à l'intention des étudiants au cours de base en analyse et conception de logiciels a permis de préciser ces perceptions.

Les figures 2 et 3 illustrent respectivement l'achalandage aux séances de travaux dirigés et l'appréciation de celles-ci. Les personnes ayant déclaré n'avoir jamais ou presque jamais assisté aux séances ont été exclues de l'analyse touchant l'appréciation de celles-ci. La majorité des répondants (7 sur 10) ont « souvent » ou « très souvent » participé aux séances, et les participants ont apprécié leur expérience dans une forte proportion (5 sur 7 l'ayant apprécié « énormément » ou « beaucoup »).

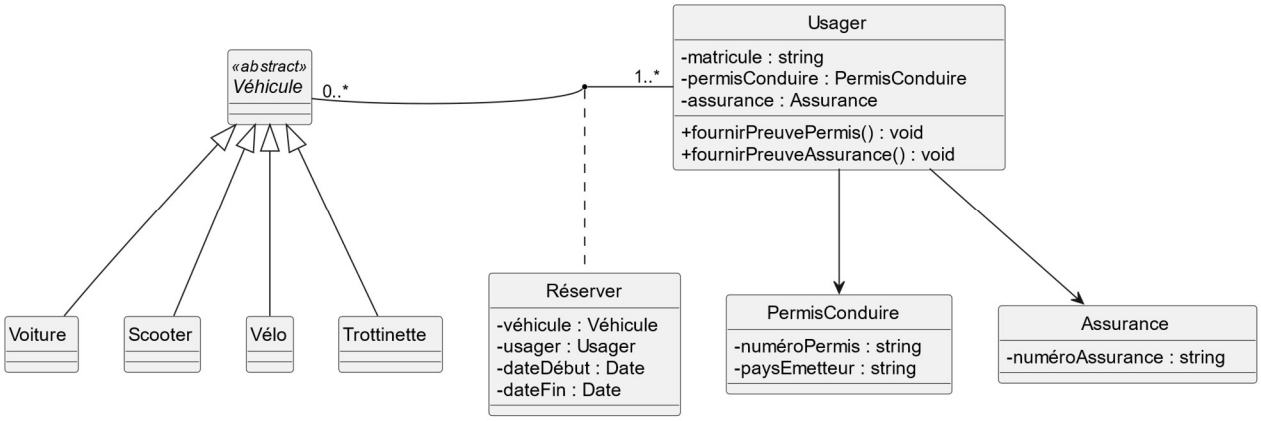

Fig. 1. Exemple de diagramme de classes fourni dans le Guide UML [13].

Les commentaires reçus font ressortir la pertinence des exercices par rapport à la matière vue en classe; l'utilité des rétroactions reçues; et la nature complémentaire de la contribution des auxiliaires d'enseignement par rapport à l'enseignant. La principale critique formulée porte sur la durée très courte d'une heure par période hebdomadaire. Le manque de formalisme des exercices a également été souligné. Ce dernier élément découle sans doute de la nature évolutive du contenu des séances, et devrait donc s'estomper avec l'arrivée en maturité de celui-ci.

Notre interprétation de ces résultats est que les séances de travaux dirigés sont appréciées mais méritent un certain nombre d'ajustements, principalement en ce qui a trait à l'horaire des séances.

7.2. Guide UML

Les figures 4 à 7 illustrent respectivement l'utilisation du Guide UML et l'appréciation de celui-ci pour les deux clientèles visées. Le portrait qui ressort de ces chiffres est plus mitigé que dans le cas des travaux dirigés.

Près de la moitié des étudiants du cours de base (4 sur 9) ont « souvent » ou « très souvent » utilisé le Guide UML. Cette proportion nous semble un peu faible étant donné que l'enseignant avait mentionné à plusieurs reprises l'existence du Guide, et que des hyperliens avaient été inclus au plan de cours ainsi que sur la page du cours. Une interprétation possible est que certains étudiants ont utilisé le Guide uniquement lorsqu'ils faisaient face à une échéance (par exemple, un travail pratique ou un examen) qui nécessitait une connaissance plus précise de l'UML. Par ailleurs, la même proportion d'étudiants considère que le Guide les avait aidés « un peu », alors que seulement le tiers (3 sur 9) juge qu'il les avait aidés « énormément » ou « beaucoup ». Les éléments jugés favorables incluent la pertinence des exemples fournis, l'apport d'une perspective complémentaire, et la facilité avec laquelle le Guide peut être consulté. La principale critique est à l'effet que le Guide manque de profondeur, avec des exemples en nombre insuffisant et des explications trop brèves.

Du côté du cours-projet, il faut savoir que seul un petit sous-ensemble des étudiants devait consulter ou produire des artéfacts en UML. Ces artéfacts figurent dans un document d'architecture logicielle dont la réalisation va généralement être prise en charge par une ou deux personnes par équipe de cinq ou six étudiants. Le taux d'utilisation du Guide (6 sur 14) doit être interprété en conséquence, quoiqu'il est fort possible que les personnes qui ne se sont pas senties concernées aient simplement choisi de ne pas ouvrir le sondage. En revanche, les trois quarts des utilisateurs (9 sur 12) semblent avoir « beaucoup » apprécié le guide. Les commentaires positifs sont de nature très variée, mais l'idée d'avoir des exemples de diagrammes ainsi que des explications sur ceux-ci semble partagée par les répondants. Les critiques sont très peu nombreuses. Parmi les suggestions figurent l'ajout d'exemples additionnels et de diagrammes plus complexes, ainsi que l'inclusion d'une section sur les outils disponibles pour faire de la modélisation en UML.

La réalisation du Guide UML nous a amenés à actualiser notre réflexion sur la place de l'UML dans nos programmes d'enseignement. Petre souligne la valeur pédagogique associée à l'enseignement de l'UML :

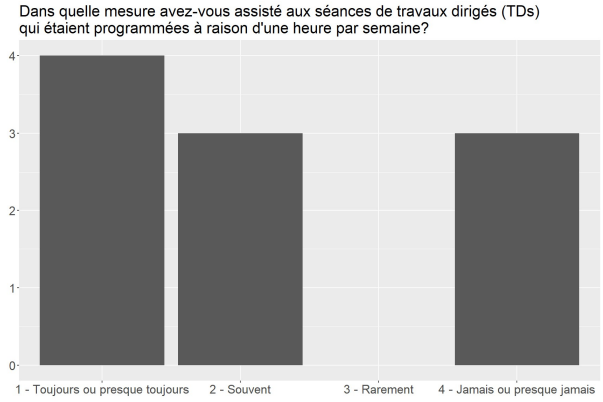

Fig. 2. Achalandage aux séances de travaux dirigés du cours de base.

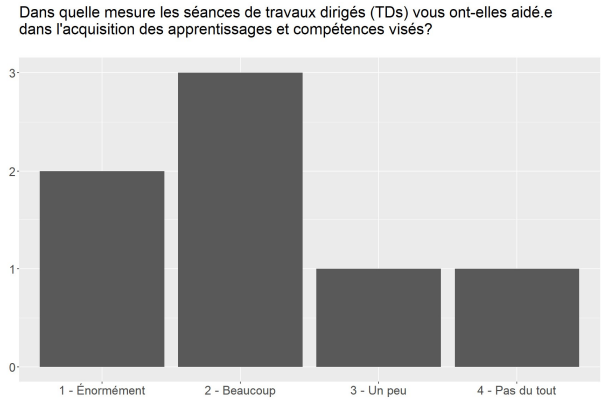

Fig. 3. Appréciation des séances de travaux dirigés du cours de base.

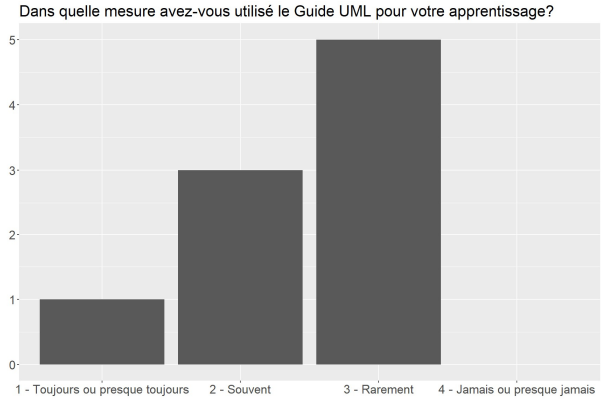

Fig. 4. Utilisation du Guide UML par les étudiants du cours de base.

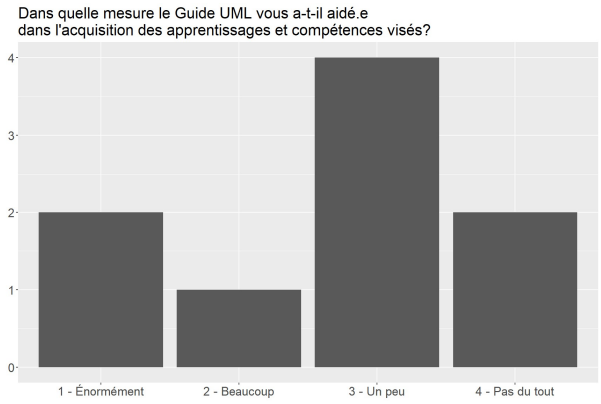

Fig. 5. Appréciation du Guide UML par les étudiants du cours de base.

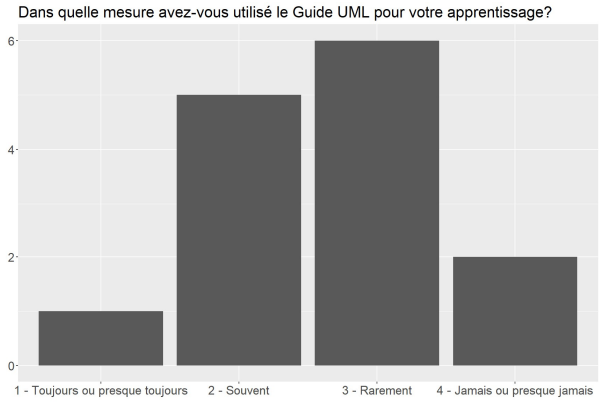

Fig. 6. Utilisation du Guide UML par les étudiants du cours-projet.

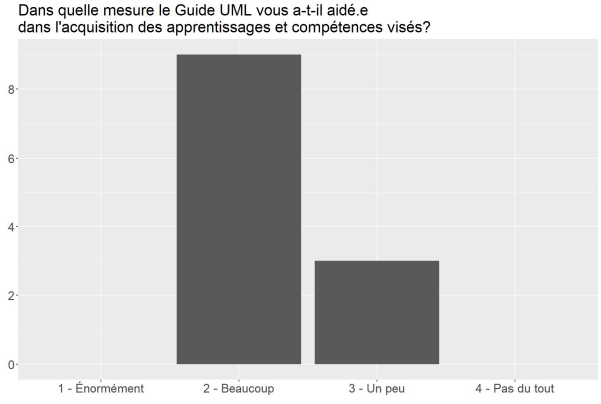

Fig. 7. Appréciation du Guide UML par les étudiants du cours-projet.

> Whether or not UML is a dominant representation in professional practice, there is evidence that it plays an important role in software engineering education, providing a common representation from which to drive discussion and build a shared model of the problem context and design proposals. It provides a medium for "model-based thinking" for student who may not already have a repertoire of representations and reasoning tools. The typical use of UML in education introduces key concepts, directs attention, and frames student exploration and practical engagement with problems and design. It could be argued that UML's value in education lies in that intellectual development, rather than in mirroring industry practice per se. [14, p. 1234]

Le Guide UML nous semble conforme à cet esprit de développement intellectuel et à cette vision de l'UML comme outil d'aide à l'acquisition d'une pensée orientée modèles. Il présente les notions de manière accessible et succincte et les illustre à l'aide de diagrammes simples et d'un exemple unifié. Il appartient à l'enseignant de bâtir sur cet échafaudage et de définir les activités pédagogiques permettant à l'étudiant d'acquérir les compétences visées en analyse et en conception.

Le choix du sous-ensemble de la norme UML 2.5.1 à couvrir dans le Guide est une autre décision qui méritait réflexion. Une couverture exhaustive était évidemment hors de question; les deux parties de la norme [4, 5] totalisent 960 pages. Nous nous sommes rabattus sur un sous-ensemble recoupant partiellement le niveau « intermédiaire » de la certification 2.0 du Object Management Group [15], tout en gardant à l'esprit que le niveau « fondation » semble plus approprié pour les étudiants au cours introductif en analyse et conception de logiciels. Ces choix comportaient une part importante de subjectivité. Il n'était pas question de former des experts compte tenu du niveau d'études visé, ce qui nous a amenés à éliminer d'emblée le niveau « avancé ». En revanche, le niveau « fondation » omettait certains aspects que nous jugions important de couvrir dans une perspective de développement des compétences des étudiants au fil des trimestres de nos programmes d'études. Par ailleurs, le choix de nous baser sur un ouvrage de référence [13] plutôt que directement sur un sous-ensemble de la norme découlait du fait que l'ouvrage faisait déjà un tri des idées les plus importantes parmi le très grand éventail de notions de niveau « intermédiaire » présentées dans [15].

La question de l'utilisation optimale du Guide au sein des divers cours demeure ouverte. À court terme, nous prévoyons de nous concentrer sur l'apport d'améliorations incrémentales au Guide suite à la rétroaction des étudiants et, en parallèle, à l'évolution de notre réflexion. Par la suite, nous procéderons à la révision progressive du contenu du cours d'analyse et de conception de logiciels afin à la fois de l'alléger et d'introduire des activités d'apprentissage autonome en classe. Nous aimerions mettre en œuvre une démarche d'amélioration itérative et incrémentale du cours qui s'étalerait sur plusieurs années et qui mettrait à profit les nouvelles séances de travaux dirigés. En ce qui concerne les cours-projets intégrateurs de deuxième et troisième année, nous avons présenté le

Guide aux étudiants et les encourageons à s'en servir comme première référence sur l'UML aux fins de leurs travaux. Une réflexion approfondie sur la manière d'utiliser UML dans ces cours-projets a été amorcée et devrait également se poursuivre au cours des prochaines années.

8. CONCLUSION

Ce projet a permis de mettre en place deux innovations qui, quoique perfectibles, semblent valables pour plusieurs étudiants. Les séances de travaux dirigés semblent répondre à un besoin réel en matière d'apprentissage actif et d'obtention d'une rétroaction complémentaire à celui de l'enseignant. Nous prévoyons de poursuivre notre réflexion quant aux ajustements à apporter à la formule, notamment quant à la logistique des séances ainsi qu'à la formalisation des exercices utilisés. Quant au Guide UML, il devra faire l'objet d'un développement plus poussé avant que nous puissions porter un jugement définitif. Un travail d'approfondissement du contenu et d'ajout d'exemples plus poussés pourra être complété par une revue du contenu du cours de base en analyse et conception de logiciels ainsi que des cours-projets et ce, afin d'assurer un arrimage de qualité entre le Guide et ces cours.

Au-delà de la difficulté de mesurer la valeur de ces innovations, l'idée même de réaliser des améliorations incrémentales touchant l'enseignement de la modélisation en génie logiciel conserve une pertinence certaine. Cette idée est d'ailleurs au cœur des activités d'amélioration continue des programmes de formation de l'ingénieur au sein du Département de génie informatique et génie logiciel. Par ailleurs, les compétences en modélisation renforcent les qualités de l'ingénieur liées à l'analyse, la conception et la communication, qui sont trois piliers de ces programmes. Ce constat, à lui seul, milite en faveur de la poursuite de nos efforts.

Remerciements

Les auteurs aimeraient remercier Justin Lachapelle, Michel Lominy et Achille Saint-Hillier, qui ont réalisé le site web Guide UML; Amélie Simard, qui a travaillé à la révision du Guide et a subséquemment accepté de contribuer activement aux nouvelles séances de travaux dirigés, d'abord comme répétitrice puis comme responsable; ainsi que Wajiha Bissola Badirou, Fedwin Chatelier et Octav Sucuturdean qui ont également assumé la responsabilité de ces séances. La création des cas utilisés dans les séances ainsi que le développement du Guide ont bénéficié de subventions du Fonds d'actions pédagogiques stratégiques de Polytechnique Montréal.

Références

[1] Object Management Group, « UML ». https://www.uml.org (visité le 9 février 2024).

[2] P. N. Robillard, P. Kruchten et P. d'Astous, Software Engineering Process with the UPEDU. Boston, MA : Pearson Education, 2003.

[3] P. Kruchten, The Rational Unified Process : an Introduction. Reading, MA : Addison Wesley Longman, 2000.

[4] ISO/IEC 19501:2005 – Information Technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2, Organisation internationale de normalisation (ISO), 2005.

[5] ISO/IEC 19505:2012 [two parts] Information Technology—Object Management Group Unified Modeling Language (OMG UML), Organisation internationale de normalisation (ISO), 2012.

[6] P. Bourque et R. E. Fairley (dir.), Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0. Washington, DC.: IEEE Computer Society Press, 2014.

[7] The Joint Task Force on Computing Curricula, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical Report. IEEE Computer Society /Association for Computing Machinery, 2014.

[8] D. Budgen, Software Design : Creating Solutions for Ill-Structured Problems, 3rd edition. Boca Raton, FL: CRC Press, 2021

[9] J. Whittle, J. Hutchinson et M. Rouncefield, « The State of Practice in Model-Driven Engineering », dans IEEE Software, vol. 31, no. 3, p. 79-85, Mai-Juin 2014, doi: 10.1109/MS.2013.65.; cité dans [8, p. 322]

[10] E. Gamma, R. Helm, R. Johnson et J. Vlissides, Design patterns: elements of reusable object-oriented software. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[11] C. Fuhrman et Y. Ross, Analyse et conception de logiciels. Montréal : École de technologie supérieure, 2023. Sous licence CC BY.

[12] D. Pilone et N. Pitman, UML 2.0 in a Nutshell. Sebastopol, CA: Oreilly Media, 2005.

[13] A. Saint-Hillier, J. Lachapelle et M. Lominy, « Guide UML ». https://gigl-uml.github.io/Guide_uml_polymtl/ (visité le 9 février 2024)

[14] M. Petre, « "No shit" or "Oh, shit!": responses to observations on the use of UML in professional practice », dans Softw Syst Model, vol. 13, p. 1225–1235, Août 2014, doi: 10.1007/s10270-014-0430-4.

[15] Object Management Group, Unified Modeling Language® 2 (UML® 2) Certifications. https://www.omg.org/ocup-2/ (visité le 9 février 2024)