Une approche didactique pour l'introduction de la Programmation Orientée-Objet en classe

Fahima Djelil, Maria Teresa Segarra Montesinos, Jean-Marie Gilliot

▶ **To cite this version:**

Fahima Djelil, Maria Teresa Segarra Montesinos, Jean-Marie Gilliot. Une approche didactique pour l'introduction de la Programmation Orientée-Objet en classe. DIDAPRO-8, Feb 2020, Lille, France. hal-02471954

HAL Id: hal-02471954
https://hal.science/hal-02471954

Submitted on 9 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Une approche didactique pour l'introduction de la Programmation Orientée-Objet en classe

Fahima Djelil, Maria Teresa Segarra Montesinos et Jean-Marie Gilliot

Lab-STICC, IMT Atlantique, 29 238 Brest, France
Fahima.djelil@imt-atlantique.fr
mt.segarra@imt-atlantique.fr
jm.gilliot@imt-atlantique.fr

Résumé. Ce papier s'intéresse à l'ingénierie didactique pour l'enseignement de la Programmation Orientée-Objets aux débutants. Il propose de définir une nouvelle approche didactique, dérivant d'une pratique d'enseignement répandue dans la communauté anglo-saxonne, l'approche « objet en premier ». Cette nouvelle approche, dite « par emboitement », décrit un processus à trois étapes permettant l'introduction de trois niveaux de concepts de manière progressive : utilisation d'objets, création de classes et conception. Nous nous focalisons sur le principe de l'emboitement, qui permet à l'apprenant de se concentrer sur un niveau de concepts en lui occultant les concepts du niveau suivant, dans le but de réduire les difficultés d'apprentissage. Nous montrons dans un premier temps, comment nous avons implémenté cette approche dans un nouveau micromonde de programmation. Ensuite, nous montrons la transposition de cette approche dans un environnement de classe en école d'Ingénieur. Enfin, nous examinons cette approche selon une dimension temporelle afin de couvrir l'ensemble d'un programme d'enseignement, en présentant une vision itérative et en spirale.

Mots-clés : Didactique de l'Informatique, Enseignement de la Programmation, Paradigme Orienté-Objet.

1 Introduction

L'enseignement de la Programmation Orientée-Objet (POO) aux débutants peut s'avérer difficile. En particulier, le changement de paradigme de programmation dans les enseignements introductifs est une source de difficulté souvent perçue chez les débutants [1]. Cela est confirmé par de nombreuses recherches en didactique de l'informatique [2] [3] [4], où l'enseignement de la programmation constitue l'un des défis majeurs [5].

Il existe plusieurs tendances pour l'introduction de la programmation et du paradigme Orienté-Objet (OO), qui ont été mises en pratique à grande échelle et qui se différencient par le type du premier paradigme introduit dans les enseignements [6]. L'approche « Objets en Premier » se distingue par un grand nombre de travaux dans la communauté anglo-saxonne, montrant l'intérêt développé pour cette approche en vue de réduire les difficultés d'apprentissage chez les débutants [3].

L'objectif de ce papier, est de décrire une nouvelle approche didactique pour l'introduction des concepts fondamentaux de l'OO, dite « par emboitement », et qui dérive de l'approche « Objets en Premier ». Nous montrons, par ailleurs, comment cette nouvelle approche est implémentée dans le micromonde de programmation PrOgO [7]. Nous montrons, ensuite comment elle peut se transposer pour être utilisée plus largement en classe, dans l'organisation des activités d'enseignement.

Par ailleurs, ce travail ne s'inscrit pas dans un paradigme méthodologique comparatiste. Notre objectif est de caractériser cette approche didactique, afin de montrer comment elle peut aider à structurer un programme d'enseignement introductif de la POO en classe, plutôt que de la comparer à une autre approche didactique.

Ce papier s'organise comme suit. La section 2 décrit comment l'approche « Objet en Premier » se met en pratique dans les curricula et comment elle est décrite dans la littérature. Nous décrivons ensuite notre nouvelle approche didactique qui en dérive (section 3), son implémentation dans l'environnement PrOgO (section 4), puis son élargissement en vue d'un usage en classe (section 5). Enfin, nous concluons par nos principales contributions (section 6).

2 Approche « Objet en Premier »

2.1 L'approche « Objet en Premier » dans les curricula

L'approche « Objet en Premier » (*Object-First*) est décrite dans le rapport CC2001(Computing Curricula 2001) [6] publié par l'ACM (*Association for Computing Machinery*) et l'IEEE Computing Society, sur la base des pratiques d'enseignement les plus répandues dans l'enseignement supérieur à l'échelle mondiale. Cette approche accorde une grande importance aux fondamentaux de la conception et de programmation OO dès le début des enseignements. En pratique, les premiers enseignements abordent rapidement les notions « d'objets » et « d'héritage », afin de confronter de manière précoce les apprenants à ces concepts. Après l'introduction de ces notions dans des programmes interactifs simples, les enseignements passent aux notions de programmation classique comme les structures de contrôle, tout en restant focalisés sur la conception OO. Le déroulement des enseignements permet ensuite d'introduire les algorithmes, les structures de données fondamentales et la conception logicielle avec plus de détails.

Le principal avantage de cette approche est la confrontation précoce des débutants à la POO [6]. Elle peut néanmoins présenter des inconvénients liés à la complexité des langages de programmation utilisés pouvant surcharger les débutants. Les spécificités des langages de POO tels que C++ et Java peuvent accentuer les difficultés d'apprentissage avec cette approche. C'est justement pour remédier à cela que l'usage des environnements visuels et interactifs comme les micromondes de programmation s'est beaucoup développé. La littérature montre que l'approche « Objet en Premier » a été portée par l'usage de ce type d'environnements [2]. L'objectif étant d'aider les débutants à construire une compréhension fine et rapide des concepts abstraits de l'OO, en leur offrant des activités attrayantes et signifiantes.

2.2 L'approche « Objet en Premier » dans la littérature

Dès le milieu des années 1990, l'approche « Objet en Premier » s'est beaucoup répandue en pratique et a émergé comme une réelle approche didactique dans l'enseignement introductif de la POO. À titre indicatif, l'interrogation de la base de données ACM Digital Library sur l'expression « *Object-First approach* » (approche Objet en Premier) retourne 208.739 résultats [recherche réalisée le 16 septembre 2019].

Une analyse de plus de 200 contenus a permis d'identifier trois étapes introduisant trois catégories de concepts par lesquelles cette approche se met en application [2]:

- Utilisation d'objets : l'apprenant utilise des « objets » préalablement implémentés. Il passe à la définition de « classes », une fois le concept d'objet maîtrisé. On se concentre alors sur l'usage de l'objet avant son implémentation.
- Création de classes : l'apprenant définit et implémente des classes et crée des instances de classes préalablement définies. On aborde alors cet aspect de programmation de manière concrète et créative.
- Concepts : l'apprenant apprend des principes généraux du paradigme OO, par la création de modèles OO. On se concentre sur les aspects conceptuels de l'OO.

De nombreuses expériences d'enseignements décrites dans la littérature peuvent se ramener à cette approche. Ainsi, Woodworth et Dann [8] décrivent une approche par laquelle l'apprenant est guidé dans la conception et l'utilisation d'abstractions, en modélisant des propriétés d'objets manipulés dans une précédente phase de résolution de problèmes.

Buck et Stucki [9] proposent de structurer un cours introductif de la POO de façon à permettre aux apprenants de débuter en modifiant un code préalablement implémenté, puis de concevoir des parties du système OO. Ils soutiennent une approche où les détails d'un langage de programmation doivent être introduits de façon progressive selon les besoins.

Kölling et Rosenberg [1] ont mis en application une démarche qui consiste à commencer par les objets et non par la programmation impérative. L'apprenant est amené à créer et à manipuler des objets à partir de classes existantes, puis à modifier des programmes de code existants au lieu de les créer par lui-même. Cette démarche incite à la lecture de code bien construit, afin de le reproduire et d'acquérir des bonnes pratiques de programmation. Ces programmes sont modélisés en OO et montrent des relations hiérarchiques entre classes. Cette approche est implémentée dans l'environnement Greenfoot [10].

Becker [11] adopte une approche qui se consacre en premier à la création et à l'utilisation d'objets, avant d'introduire les notions conceptuelles de l'OO par l'extension de classes existantes. Les notions de la programmation impérative (structures de contrôle conditionnelles et itératives, expressions booléennes, …) sont introduites en dernier de manière progressive. Cette approche a été implémentée et mise en expérience dans les environnements Karel et ObjectKarel [12].

Joel et Frens [13] présentent une méthodologie en trois phases. La première (objets en premier) consiste à apprendre à résoudre un problème en identifiant les objets à

déclarer dans un programme. La seconde (classes et méthodes) se consacre à l'introduction de méthodes par leur déclaration et définition dans une classe. Cette phase permet d'introduire des notions de programmation impérative. La troisième phase (héritage) permet d'introduire les concepts d'héritage et de polymorphisme.

Cooper, Dann et Pausch [14] ont adopté l'approche « Objet en Premier » en utilisant l'environnement Alice. Leur démarche consiste à introduire les objets dans un contexte visuel et significatif. Les notions de programmation impérative sont introduites de manière progressive en se focalisant sur le comportement des objets.

On peut observer que l'intérêt développé pour cette approche est induit par le changement de paradigme de programmation en enseignement de l'informatique, source de difficultés de compréhension des fondamentaux de l'OO. Cette approche préconise l'OO dès le début des enseignements, et une découverte progressive de la programmation impérative avec une focalisation sur l'OO. Or, en pratique le choix du paradigme dans l'introduction des concepts de programmation reste très varié. À titre indicatif, le rapport CC2001 liste six approches différentes d'enseignement de la programmation de l'informatique aux débutants. Chacune présentant ses avantages et ses inconvénients.

Nous ne focalisons pas notre attention sur le paradigme utilisé en premier dans l'approche « Objet en premier », mais sur la manière dont cette approche introduit les concepts fondamentaux de l'OO. Nous ressortons une approche didactique par « emboîtement » des concepts OO.

3 L'approche didactique par emboîtement des concepts fondamentaux de l'OO

L'approche « Objet en Premier » nous a permis de définir une nouvelle approche didactique pour l'introduction des concepts de POO, par un principe « d'emboîtement » [7]. Nous retenons dans l'approche « Objet en Premier » l'ingénierie didactique par laquelle les concepts fondamentaux de l'OO sont introduits aux débutants. Celle-ci se caractérise par trois étapes qui se suivent par emboîtement :

1) Utilisation d'objets : l'apprenant apprend à se familiariser avec le concept de l'objet et ses propriétés. À ce stade, le concept de classe est volontairement occulté. On montre brièvement la relation qui lie l'objet à la classe (l'objet étant une instance de classe), sans se focaliser sur la classe. L'apprenant se concentre dans cette phase sur comment les objets sont utilisés dans la résolution d'un problème donné.
2) Création de classes : l'apprenant apprend à créer des abstractions ou des modèles à partir de propriétés d'objets, une fois ce concept maîtrisé. À ce stade les relations et associations entre différentes classes ne sont pas apparentes.
3) Conception : nous proposons de retenir la notion de conception, plutôt que le terme concept, dans le sens où les concepts introduits le sont dans une logique de conception d'application. L'apprenant apprend à construire des modèles OO et à organiser son programme en classes liées par des relations spécifiques.

Ces trois étapes emboitent les concepts OO de façon à amener l'apprenant, à garder à l'esprit qu'une classe permet de définir les propriétés d'objets et que la modélisation a pour objectif de spécifier la solution à un problème, que l'on peut concrètement résoudre au moyen d'objets. C'est un moyen d'aider les apprenants débutants à mieux mentaliser le paradigme OO.

Nous voyons une approche didactique se caractérisant par un emboîtement de trois catégories de fondamentaux de l'OO (Fig. 1). Il s'agit à chaque étape, de permettre à l'apprenant, d'interagir avec les propriétés d'une catégorie tout en lui occultant les propriétés de la catégorie suivante. L'objectif étant de s'affranchir de la complexité de la modélisation OO lors de la résolution de problèmes par les débutants, pouvant constituer un frein à leur apprentissage.

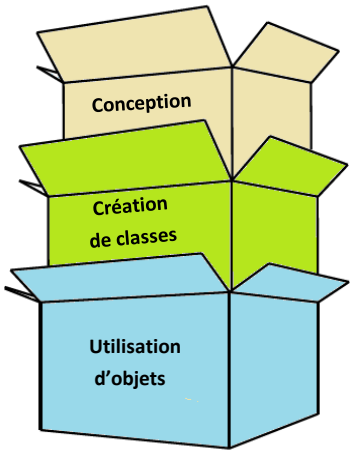

Fig. 1. Approche didactique par emboitement des concepts OO [7].

Cette approche didactique, nous a servi pour la conception de l'environnement PrOgO [7], qui implémente les deux premières phases, utilisation d'objets et création de classes. La troisième phase n'étant pas implémentée, l'enseignant peut l'introduire sous forme de débriefing avec ses apprenants après l'utilisation de l'environnement PrOgO.

4 Implémentation de l'approche didactique dans l'environnement PrOgO

4.1 L'environnement PrOgO

PrOgO[1] est un micromonde de programmation OO, conçu dans le but d'aider les débutants à apprendre par le jeu les fondamentaux de l'OO. Il repose sur une métaphore

[1] PrOgO est le résultat d'une thèse (prix AUF 2018) financée dans le cadre du Programme d'Investissement d'Avenir par le projet E-Education Tactileo (2013-2016). http://progo.iut-lepuy.fr

de jeu de construction et d'animation en 3D. Son interface graphique est essentiellement constituée d'une scène 3D dédiée à la création et à l'animation de constructions graphiques (Fig.2), et d'un éditeur à auto-complétion qui permet de visualiser ou de saisir des instructions de code en langage C++ entièrement synchronisé avec la scène 3D. Les constructions graphiques sont des représentations visuelles de systèmes OO.

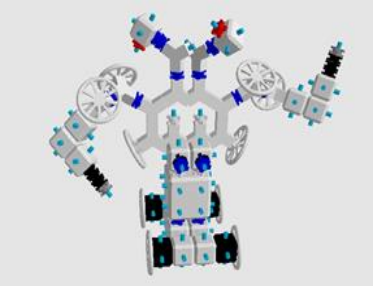

Fig.2. Une construction 3D dans PrOgO [7].

4.2 Modèle OO sous-jacent aux constructions 3D dans PrOgO

L'interface de PrOgO fournit des modèles graphiques 3D représentant des classes qui donnent lieu à des objets directement utilisables par l'apprenant. Chaque construction 3D réalisée par l'apprenant constitue une instance d'une nouvelle classe et consiste en un système OO qui peut être décrit par un diagramme de classes UML « *Unified Modeling Language* » (Fig.3). Deux types de classes permettent de modéliser les composants graphiques servant au jeu de construction [7]:

- La classe « `ComposantStructurel` » modélise l'ensemble des composants graphiques servant à la construction de la réalisation 3D. Cette classe possède un attribut « `couleur` » et deux méthodes « `connecter()` » et « `colorierPendant()` », qui permettent respectivement de connecter un objet à un autre et de colorier un objet pendant un temps donné.
- La classe « `ComposantActif` » modélise l'ensemble des composants structurels qui sont capables d'exécuter un mouvement (une action de rotation pendant un temps donné). Cette classe hérite de la classe `ComposantStructurel`, et possède deux autres caractéristiques, l'attribut « `angleDeRotation` » et la méthode « `tournerPendant()` ».

Les classes visualisées à l'interface dérivent soit de la classe « `ComposantStructurel` », c'est le cas de « `Base` », « `Cube` », « `BifurcationY` » et « `Ressort` », soit de la classe « `ComposantActif` », c'est le cas de « `Engrenage` » et « `Roue` ». La classe « `Base` » possède une unique instance et représente la base de toute construction dans la scène 3D.

Une construction 3D possède également deux méthodes : la méthode « `animer()` » regroupant les comportements d'animation programmés par l'apprenant lors de la réalisation de sa construction ; la méthode

« `réinitialiser()` » qui permet de réinitialiser une construction 3D à son état de création.

On voit apparaître dans ce modèle OO trois relations de conception :

- L'héritage : les classes « `Engrenage` » et « `Roue` » héritent de la classe « `ComposantActif` ». Les classes « `Base` », « `Cube` », « `BifurcationY` » et « Ressort » héritent de la classe « `ComposantStructurel` ». Les classes dérivées partagent les mêmes propriétés que la classe de laquelle elles dérivent.
- La composition : la classe « `Construction3D` » se compose à la fois de la classe « `Base` » et de la classe « `ComposantStructurel` ». La suppression de la classe composée induit la suppression des classes qui la composent.
- L'agrégation : s'applique sur la classe « `ComposantStructurel` ». Tout composant structurel se trouve lié à un autre composant structurel, mais la suppression de l'un n'implique pas la suppression de l'autre.

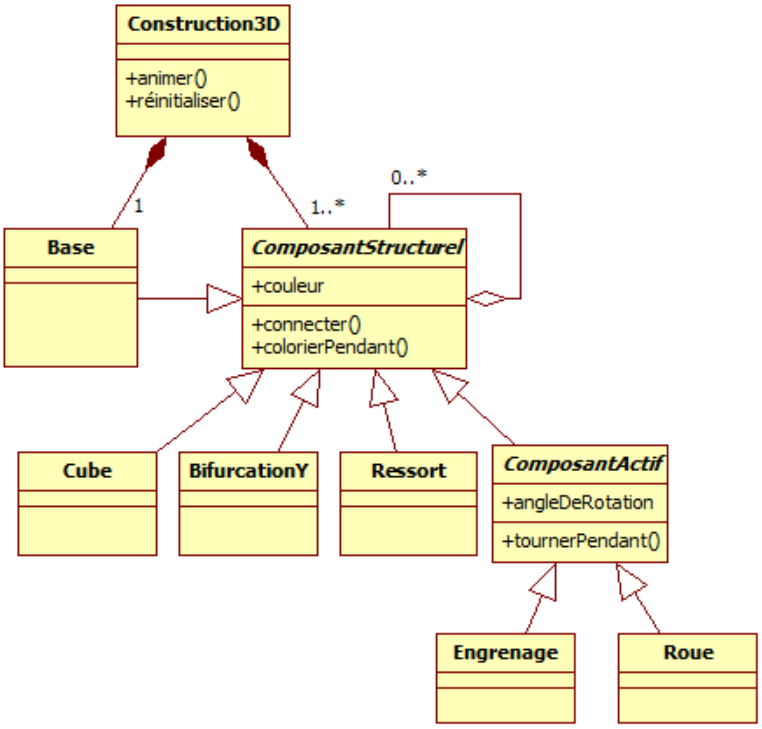

Fig.3. Diagramme de classes UML modélisant une construction 3D dans PrOgO [7].

4.3 Implémentation de l'approche didactique

Lors du jeu de construction, l'apprenant est amené à imaginer et à créer la structure qu'il souhaite, tout en accédant aux concepts OO sous-jacents. L'apprentissage se

déroule en deux phases selon l'approche didactique par emboîtement des concepts OO : utilisation d'objets et création de classes (Tableau 1). La troisième phase n'est pas implémentée dans la version finale de PrOgO, mais l'enseignant peut l'introduire en classe en expliquant le modèle OO sous-jacent aux constructions 3D réalisées par l'apprenant.

Tableau 1. Concepts OO introduits par l'approche didactique dans l'environnement PrOgO.

Étape didactique	Concept
(1) Utilisation d'objets	1.1 Lien existant entre l'objet et la classe : un objet est une instance de classe. 1.2 Caractéristiques d'objet : un objet se caractérise par des attributs et des méthodes. 1.3 Modification de l'état d'un objet : modifier la valeur d'un attribut d'objet ou réaliser un appel de méthode sur l'objet.
(2) Création de classes	2.1. Rôle d'une classe : une classe sert à décrire des propriétés d'objets. C'est un nouveau type de données. 2.2. Encapsulation dans une classe : une classe encapsule ses données et fonctions membres, afin de contrôler leur accès depuis l'extérieur. 2.3 Constructeur de classe : un constructeur de classes permet d'initialiser son instance, l'objet à sa création.

Utilisation d'objets. Dans PrOgO, chaque graphique élémentaire 3D est une représentation visuelle d'un objet informatique. Les objets s'obtiennent par instanciation des modèles graphiques fournis à l'interface. Chaque objet possède la même apparence visuelle que sa classe à la création. L'apparence d'un objet est définie par sa position (son emplacement par rapport à un autre objet), sa couleur ou sa rotation. L'apprenant peut ensuite changer l'apparence de l'objet en modifiant ses attributs et en exécutant des appels de méthodes. Le comportement d'un objet est défini par deux fonctionnalités, la possibilité de changer de couleur pendant un temps donné et/ou la possibilité de réaliser une rotation pendant une certaine durée. Enfin, chaque objet peut être assemblé à un autre objet afin de construire une structure plus complexe. L'ensemble de ces opérations sont possibles à la fois dans la scène 3D et dans l'éditeur de code en langage C++. Dès que l'apprenant a terminé sa construction, il est invité à créer une nouvelle classe avec sa réalisation dans une seconde phase.

Création de classes. Toute nouvelle construction dans PrOgO constitue une nouvelle classe que l'apprenant peut créer, nommer et ajouter à sa liste de classes. Cette nouvelle classe peut alors être instanciée et visualisée dans la scène 3D. C'est un nouveau modèle qui regroupe les propriétés choisies par l'apprenant lors de sa réalisation. La création d'une nouvelle classe entraîne l'apparition d'un nouveau programme principal contenant une fonction « `main()` » vide invitant l'apprenant à instancier sa nouvelle

classe. Deux nouveaux fichiers apparaissent également, l'un contient la déclaration de la nouvelle classe « `*.hpp` », et le second contient sa définition « `*.cpp` ».

L'apprenant est amené à manipuler ces fichiers afin de découvrir le concept d'encapsulation. Certaines données et fonctions membres de la classe nouvellement créée sont privées (précédées par le mot clé « `private` ») et d'autres sont publiques (précédées par le mot clé « `public` »). À titre d'exemple, l'apprenant accède depuis la fonction « `main()` » aux fonctions publiques de sa nouvelle classe, mais n'accède pas aux membres privés qui ne sont visibles que depuis le fichier de définition de classe.

La classe nouvellement créée possède également un constructeur qui permet d'initialiser un objet lors de sa création. L'apprenant peut également observer le code du constructeur. À titre d'exemple, il peut remarquer que le constructeur est publique, porte le même nom que sa classe, et n'a pas de type de retour.

L'environnement PrOgO montre l'implémentation de l'approche didactique par emboîtement de concepts OO sur un cas d'utilisation unique (le modèle OO sous-jacent aux constructions 3D dans PrOgO). En pratique, dans un programme d'enseignement en classe, plusieurs principes de conception sont couverts. La section suivante montre comment cette approche peut se transposer en classe pour couvrir un large spectre de concepts OO.

Par ailleurs, l'environnement PrOgO a été utilisé avec des publics de différents niveaux, à savoir, le secondaire et le supérieur. Cela montre le potentiel de transférabilité de cette approche didactique à des publics différents.

5 Extension de l'approche didactique en vue d'un usage en classe

Au sein de IMT Atlantique, dans le programme de formation d'Ingénieurs, l'organisation de l'enseignement introductif de la POO s'effectue également selon l'approche « par emboîtement » en trois étapes. Elle a été mise en œuvre dans le cadre du module « Modèles et Programmation Orientée-Objet » en Tronc Commun. Ce module comprend 40 heures en face à face pédagogique, dont 22 heures de cours respectant cette approche didactique et 18 heures de projets. Les activités de programmation s'effectuent avec le langage Java. Les élèves possèdent des prérequis en programmation procédurale en langage Python. Leur effectif varie entre 40 et 60 élèves. Le Tableau 2 montre les concepts introduits aux élèves dans ce module en les associant à chacune des étapes didactiques : utilisation d'objets, création de classes et conception.

Tableau 2. Concepts introduits dans l'enseignement introductif de la POO au sein de IMT Atlantique.

Étape didactique	Concept
(1) Utilisation d'objets	1.1 Objet : abstraction d'une entité du monde réel participant à l'exécution d'un programme.

	1.2 Attribut, attribut d'instance : variable d'un objet. Sa valeur représente une partie de l'état de l'objet. 1.3 Méthode, méthode d'instance : traitement associé à un objet.
(2) Création de classes	2.1 Classe : construction d'un langage OO qui définit la structure d'une famille d'objets et son comportement. 2.2 Instanciation : action de créer un objet à partir d'une classe. 2.3 Constructeur : méthode d'une classe permettant de créer des objets de la classe. Elle initialise l'état de ces objets. 2.4 Méthode de classe : traitement associé à une classe, qui ne peut pas manipuler des attributs d'instance. 2.5 Encapsulation : principe par lequel l'état d'un objet ne peut être accédé directement que par l'objet lui-même.
(3) Conception	3.1 Association : lien qui dure dans le temps entre deux objets. L'identité d'un objet est mémorisée dans un autre objet. 3.2 Agrégation : association non symétrique, qui exprime une relation de type « ensemble/élément ». Une instance d'élément agrégé peut exister sans agrégat (et inversement). 3.3 Composition : association non symétrique, qui exprime un couplage fort et une relation de subordination. Détruire l'agrégat détruit les éléments agrégés. 3.4 Héritage : concept propre aux langages de programmation pour la réalisation d'une relation de type classe mère/classe fille où les classes filles partagent des membres avec la classe mère. Il s'agit d'une relation de type « est-un ». Etc.

L'utilisation de cette approche didactique en classe montre qu'elle nécessite d'être pensée sur le long terme, selon une dimension temporelle, afin de couvrir l'ensemble du programme pédagogique d'enseignement.

En pratique, un retour de l'étape « conception » vers l'étape « utilisation d'objets » est nécessaire, afin de permettre aux apprenants de mieux visualiser les éléments conceptuels sur le comportement des objets. Les classes étant le moyen de modéliser les objets, l'étape « création de classe » intervient de nouveau dans les activités. Cela dit, pour chaque nouveau concept OO, les trois étapes didactiques s'enchainent successivement. Ainsi, au sein d'IMT Atlantique, l'exemple de la problématique de gestion de stocks en magasin est pris comme cas d'utilisation pour introduire successivement des concepts OO comme « l'association », « la composition », « l'agréation » « l'héritage », etc.

Les trois étapes didactiques sont renouvelées de manière itérative et continue (en spirale) pour couvrir un grand nombre d'aspects de conception enseigné au travers d'un cas d'utilisation (*Fig. 4*).

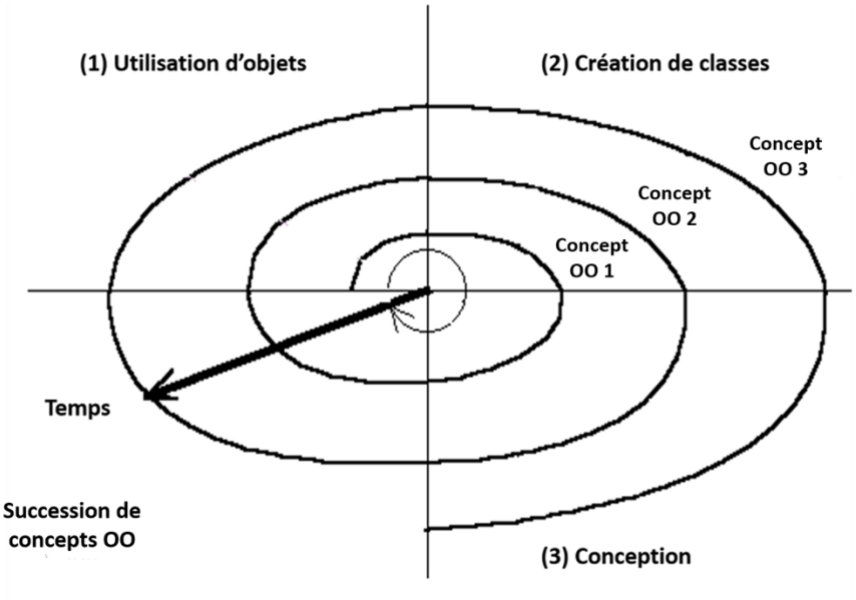

Fig. 4. Vision itérative en spirale de l'approche didactique pour l'enseignement de la POO en classe.

6 Conclusion

Ce papier propose d'étudier la question de l'introduction de la POO dans les programmes d'enseignements aux débutants, et cherche à établir une ingénierie didactique autour de l'enseignement des concepts OO en classe.

Pour ce faire, nous nous sommes intéressés à une approche très répandue dans la communauté anglo-saxonne, se focalisant sur l'enseignement du paradigme OO dès le début des enseignements, l'approche « Objets en Premier ». Cette approche nous a permis de ressortir trois étapes didactiques importantes dans le processus d'introduction des concepts OO aux débutants, associées à trois niveaux de concepts qui s'introduisent par emboîtement. Cette nouvelle approche didactique, ne se focalise pas sur le type de paradigme à introduire en premier comme dans l'approche « objet en premier », mais sur l'emboitement des trois niveaux de concepts OO, comme élément d'ingénierie didactique.

Cette approche, nous a servi dans la conception du micromonde de programmation PrOgO. Cela démontre la faisabilité de son implémentation dans un environnement numérique et ludique d'apprentissage. Par ailleurs, nous avons vérifié la transposition de cette approche en classe, en se basant sur le programme d'enseignement dispensé à l'école d'Ingénieurs IMT Atlantique. Nous représentons cette approche selon une dimension temporelle, afin de couvrir l'ensemble d'un programme d'enseignement en classe sur le long terme. Il en résulte une vision itérative en spirale, montrant l'application des trois étapes constitutives de cette approche à chaque itération, visant

à introduire de nouvelles notions de conception OO au travers d'un cas d'utilisation réel.

Enfin, cette approche contribue aux questions de didactique de l'informatique induites par la généralisation des compétences informatiques à l'école, comprenant l'initiation à la programmation et le paradigme OO. Cette approche, pose en effet un cadre pouvant servir à chaque enseignant s'interrogeant sur la façon d'introduire la POO en classe.

Références

1. M. Kölling et J. Rosenberg: Guidelines for teaching object orientation with Java.*ACM SIGCSE Bulletin,* vol. 33, n13, pp. 33-36 (2001).
2. J. Bennedsen et C. Schulte: What does Objects-First mean ? : An international study of teachers' perceptions of Objects-First. *Seventh Baltic Sea Conference on Computing Education Research* (2007).
3. J. Bennedsen : Teaching and learning introductory programming : a model based approach. Thèse de doctorat. University of Oslo. Faculty of Mathematics & Natural Sciences (2008).
4. A. Robins, J. Rountree et N. Rountree: Learning and teaching programming. A review and discussion. *Computer science education,* vol. 13, n° 12, pp. 137-172 (2003).
5. A. Mcgettrick, R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove et K. Mander: Grand challenges in computing: Education—a summary. *Computer Journal,* vol. 48, n° 11, pp. 42-48 (2005).
6. G. Engel et E. Roberts : Computing curricula 2001. Computer science, The Joint Task Force on Computing Curricula (2001).
7. F. Djelil : Conception et évaluation d'un micromonde de Programmation Orientée-Objet fondé sur un jeu de construction et d'animation 3D. Thèse de Doctorat. Université Blaise Pascal - Clermont II (2016).
8. P. Woodworth et W. Dann: Integrating console and Event-Driven models in CS1. *ACM SIGCSE Bulletin,* vol. 31, n°11, pp. 132-135 (1999).
9. D. Buck et D. J. Stucki: Design early considered harmful : Graduated exposure to complexity and structure based on levels of cognitive development. *ACM SIGCSE Bulletin,* pp. 75-79 (2000).
10. M. Kölling: Introduction to programming with Greenfoot, Pearson Education: Upper Saddle River, New Jersey, USA (2010).
11. B. W. Becker: Teaching CS1 with karel the robot in Java. *ACM SIGCSE Bulletin,* vol. 33, n°11, pp. 50-54 (2001).
12. S. Xinogalos, M. Satratzemi et V. Dagdilelis: An introduction to Object-Oriented Programming with a didactic microworld : objectKarel. *Computers & Education,* vol. 47, n°12, pp. 148-171 (2006).
13. J. Adams et F. Jeremy, «Object centered design for Java : teaching OOD in CS-1,» *ACM SIGCSE Bulletin,* vol. 35, n°11, pp. 273-277 (2003).
14. S. Cooper, W. Dann et R. Pausch: Alice : a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges,* vol. 15, n°15, pp. 107-116 (2000).