Outils d'assistance et les difficultés d'enseignement / apprentissage de la programmation, quelle aide ?

Souleiman Ali Houssein[12], Yvan Peter[1]

[1] Université de Lille, CNRS, Central Lille, UMR 9189, CRIStAL, F-59000 Lille,France
yvan.peter@univ-lille1.fr
[2] Université de Djibouti, Djibouti
s.alihoussein@ed.univ-lille1.fr

Résumé. L'apprentissage de la programmation est considéré comme une tâche complexe, surtout dans les cours d'introduction. Nous examinons dans cet article comment les outils d'assistance de l'enseignement/apprentissage traitent les différentes difficultés et quel genre d'aide ils apportent. Cette étude nous montre que la plupart des outils sont orientés vers une assistance des erreurs syntaxique et sémantique et qu'il y a un manque d'assistance dans la résolution de problème.

Keywords. Outil d'aide, évaluation, apprentissage de la programmation

1 Introduction

L'apprentissage de la programmation est une tâche complexe. J.Kasboll rapporte que le taux d'échec ou d'abandon des cours d'introduction à la programmation en premier cycle universitaire varient de 25 à 80% de part le monde [1]. Cette difficulté est due au fait que la programmation nécessite plusieurs compétences : (i) capacité de résolution des problèmes ; (ii) connaissance des outils de programmation ; (iii) maîtrise de la syntaxe et de la sémantique d'un langage de programmation. D'autre part, les cours d'introduction à la programmation, peuvent regrouper un nombre important d'étudiants qui ont des niveaux hétérogènes (certains étudiants ont une base de la programmation, acquise au cours de leurs enseignement secondaire alors que d'autres sont novices) [2]. Cela complique la tâche de l'enseignant pour aider les étudiants qui ont des difficultés.

Nous nous intéressons aux outils qui visent à assister l'enseignement/ apprentissage de cours d'introduction et présentons dans ce qui suit comment ces outils répondent aux différentes difficultés d'enseignement / apprentissage de la programmation et quel genre d'aide ils apportent aux apprenants.

1 Difficultés d'Enseignement/Apprentissage de la Programmation

Bien qu'il existe une diversité de travaux traitant la difficulté de l'apprentissage de la programmation dans le cours d'introduction, ces travaux convergent sur trois types de difficultés qui sont décrites dans la suite.

1.1 Problème de Compétence dans la Méthode de Résolution de Problème

Différentes études identifient la capacité de résolution de problème comme une difficulté pour les débutants en programmation [3],[4],[5].
Pour Annabela les compétences nécessaires à la résolution de problèmes sont [2] :

- La compréhension des problèmes : les étudiants essaient de résoudre le problème sans comprendre complètement : soit ils ont des difficultés d'interprétation, soit ils ont tendance à écrire les réponses avant de penser soigneusement ;
- Transfert de connaissances : beaucoup d'étudiants n'établissent pas d'analogie correcte entre les anciens problèmes rencontrés pour le transfert de connaissance sur le nouveau problème.

1.2 Problème de Compétence de Codage

Dans leurs études respectives, [2], [6], [7], [8], [9], rapportent que les difficultés de codage des étudiants novices, sont due à :

- des difficultés à détecter ou trouver des simples erreurs de programmation syntaxique ou logique ;
- difficultés pour décrypter les messages d'erreur (commentaire) du débogueur;
- le manque de compréhension de la façon dans le programme s'exécute : les étudiants ont des difficultés sur les concepts liés à la mémoire (pointeur, liste chaînée…) ;
- Le manque de connaissances et de stratégie de programmation. Pour Winslow, un grand nombre d'étude concluent que les programmeurs débutants connaissent la syntaxe et la sémantique des constructions basiques du langage, mais ils ne savent pas comment combiner ces connaissances en un programme valide [7]. Même quand ils savent comment résoudre les problèmes à la main, ils ont du mal à traduire cette solution dans un programme équivalent. Difficulté également soulignée par Lahtinen, qui dit que les problèmes des étudiants n'est pas la compréhension de concept de base, mais plutôt le manque de stratégies de programmation (difficulté à mobiliser ou à utiliser des connaissances acquises, pour résoudre une tâche) [6].

1.3 Problème de Méthode d'Enseignement

Différentes études pointent également des problématiques liées à l'enseignement [2],[9] :

- Un enseignement non personnalisé : il est souhaitable que l'enseignant puisse aider les étudiants d'une manière individualisée, en leur donnant des explications plus détaillés sur les problèmes rencontrés. Cependant, en réalité il est impossible que l'enseignant puisse donner une telle personnalisation, en raison des contraintes de temps et du nombre important d'étudiants dans la salle de classe ;
- Les enseignants se concentrent plus sur l'enseignement d'un langage de programmation et de ses détails syntaxiques, au lieu de promouvoir ou d'insister sur la résolution problème.

2 Outils d'Assistance à l'Enseignement/Apprentissage de la Programmation

Pour répondre ou apporter une solution aux différentes difficultés citées ci-dessus. Nous avons trouvé dans la littérature, plusieurs outils, qu'on peut catégoriser comme outil de visualisation, outil d'évaluation automatique [10], et un ensemble d'outils qui vont au delà de l'évaluation. Nous présentons dans ce qui suit ces différentes catégories d'outils.

2.1 Outil de Visualisation

Comprendre comment la machine exécute un programme, est une étape importante pour l'apprenant, pour construire par la suite de bons programmes. C'est pour répondre à ceci que des outils de visualisation ont été développés. Ils sont destinés à assister les apprenants dans la compréhension de la façon dont les programmes s'exécutent. Cette visualisation peut prendre plusieurs formes : (i) une représentation sous forme graphique du déroulement de l'exécution du programme écrit par l'apprenant (par exemple Malba [11]); (ii) une représentation sous forme de simulation sur un jeu, ou les apprenants écrivent leur code et l'exécutent pour voir comment il agit sur le jeu (par exemple Prog&Play[12]).
Les articles suivants décrivent en détails les outils de cette catégorie : [13],[14].

2.2 Outils d'Évaluation Automatique

L'objectif de cette catégorie d'outils est d'évaluer le code de l'apprenant, en mettant en œuvre : (i) soit une évaluation sommative, qui a pour fonction de donner une note à l'apprenant en faisant le bilan de ses compétences et de ses connaissances après la réalisation d'une tâche ; (ii) soit une évaluation formative, qui a pour fonction de

réguler (régulation effectuée par rétroaction) l'apprentissage pour accompagner la réussite de l'apprenant sur une tâche donnée. La plupart des outils de cette catégorie effectuent des évaluations formatives. Afin d'évaluer (analyser) le code de l'apprenant pour lui donner une rétroaction détaillée, les outils collectent ou extraient dans les programmes des apprenants certaine informations qui seront comparées par la suite à certaine exigences ou à des solutions modèles.
Les articles [15],[16],[17] recensent et analysent ce type d'outils. D'après l'étude menée par Ala-Mutka, ces outils utilisent deux approches pour évaluer la production de l'apprenant : (i) **Une approche dynamique** basée sur l'exécution du programme. Elle permet d'effectuer une comparaison, sur un ensemble des données de test, des résultats de sortie ou de valeur de retour, entre la solution de l'apprenant et une solution modèle. Elle est souvent utilisée pour évaluer la fonctionnalité et l'efficacité de celle-ci (par exemple BOSS [18], ASSYST[19]) ; (ii) **Une approche statique** basée sur l'analyse du programme (code) sans l'exécuter. Elle permet de révéler la qualité du code ou de sa compréhension. Plusieurs types d'analyses de code sont utilisés [15]:

- Une analyse faite sur le style de programmation basé sur les erreurs (syntaxique, sémantique, logique,..) , par exemple Gauntlet [20];
- Une analyse faite avec des valeurs métriques (complexité, longueur des lignes de code, durée de l'exécution..) , par exemple Proust[21], Talus [22].

Cette catégorie d'outils aide souvent les apprenants par des rétroactions détaillées sur leur code (un conseil sur une erreur du code, des listes de données entrées en test et de résultat attendu,..). Ils permettent également de réduire la charge de travail de l'enseignant [15]. Ils sont en général spécifiques à un langage de programmation bien précis [17].

2.3 Autres Outils basés sur l'Évaluation

D'autres outils existent dont la finalité n'est pas uniquement l'évaluation. Ils intègrent un processus d'évaluation ou utilisent un outil d'évaluation automatique dans l'objectif d'orienter les apprenant à améliorer leur compétences ou d'assister l'enseignant dans cette tâche. *SmartLab* [23], outil pour le suivi de TP, fournit à l'enseignant à partir d'un moniteur la progression de chaque étudiant, sur une tâche de programmation en temps réel. Il utilise un outil d'évaluation automatique qui permet d'analyser le code de l'apprenant.
De-la-fuente propose pour sa part, un système d'orchestration d'activités basé sur la technologie de script couplé à un outil d'évaluation automatique pour donner l'accès à la tâche suivante, en fonction de l'avancement des groupes d'étudiants sur une tâche donnée. Cela réduit la charge des enseignants qui veulent mettre en œuvre une méthode de pédagogie par projet [24].
QuizPack utilise le résultat de l'évaluation automatique pour générer des exercices paramétrés en langage C [25].
CourMaker développé à l'Université de Nottingham, s'appuie sur l'outil Ceilidh: il permet d'apporter des rappels de cours en fonction des résultats de l'évaluation. Il supporte une variété de langage de la programmation [26].

PASS est un système qui intègre un processus d'évaluation, utilisant une approche dynamique pour comparer le résultat de l'exécution de production de l'apprenant à celle attendue. L'objectif est de leur fournir une rétroaction pour qu'ils puissent déboguer leurs erreurs. Le système prend en charge l'enseignement de plusieurs langages de programmation (C, C++, Java, Pascal,..) et propose également des activités de niveaux de difficulté différents [27].
D'autres outils trouvés dans la littérature, sont orientés vers l'assistance des compétences de résolution de problème. Ces outils utilisent les techniques d'évaluation automatique, mais se concentrent plus sur la résolution du problème et moins sur les erreurs syntaxiques et sémantiques. Ils utilisent un langage indépendant de tout langage de programmation, ou l'apprenant décrit la solution soit en organigramme soit en langage naturel (pseudo code) comme Allogène [28], algo+ [29].

3 Analyse et Conclusion

Dans cette étude des outils d'assistance à l'enseignement/apprentissage de la programmation, nous remarquons qu'ils suivent deux orientations. Ceux qui sont uniquement destinés à effectuer des évaluations avec des techniques différentes, et ceux ou l'évaluation fait partie du processus d'apprentissage. Peu importe l'orientation de ces outils, ils sont destinés à apporter une aide, chacun à sa manière, aux différentes difficultés de l'enseignement/apprentissage de la programmation. Beaufils et al. distinguent plusieurs niveaux d'aide [30]:

- Assistance : consiste à apporter une prise en charge partielle de la tâche. Mis en œuvre par des agents qui effectuent une partie de la tâche ou guident très fortement l'utilisateur ;
- Guidage : accompagne l'utilisateur dans l'accomplissement d'une tâche, en commentant certaines de ses actions et en lui suggérant des actions susceptibles de le faire progresser dans la tâche ou d'améliorer son efficacité ;
- Aide intelligente : relève du guidage, mais possède une caractéristique supplémentaire qui consiste à effectuer une analyse relativement élaborée de l'activité de l'utilisateur et permettre une adaptation du contenu à ses besoins particuliers ;
- Conseil : est analogue au guidage mais il ne propose pas nécessairement la meilleure marche à suivre ; il tient compte des imperfections de l'analyse et produit des informations plutôt méthodologiques;
- Explication : elle a pour fonction de détailler et d'expliciter le fonctionnement ou le résultat d'une action ou d'un raisonnement.

En comparant les différentes catégories d'outils présentées ci-dessus aux différents critères qui sont : quel difficulté traite t-il ? Et quel genre d'aide apporte t-i l? Nous remarquons que :
Pour les outils de visualisation, on peut dire qu'ils aident les apprenants sur les problèmes de compétences de codage et plus précisément leur donner un modèle mental clair de l'exécution de leur programme. Ils apportent donc une aide plus orientée vers l'explication.
Les outils d'évaluation automatique, quant à eux aident les apprenants sur les problèmes de compétence de codage et plus précisément dans la difficulté d'aider à retrouver leurs erreurs syntaxique et sémantique. L'aide se présentant sous forme de rétroaction, dépend de l'outil. Ceux qui mettent en œuvre une évaluation par une approche dynamique, donnent souvent des conseils ou du guidage. Quant à ceux qui utilisent une évaluation par une approche statique, ils proposent un guidage.
Les autres outils intégrant ou utilisant une évaluation automatique, sont classés en :

- Des outils qui aident l'enseignant dans ses tâches pour faciliter l'assistance aux apprenants ;
- Des outils de génération d'exercices qui sont plus orientés vers un guidage ;
- Les outils pour la résolution de problèmes, se concentrent plus sur l'évaluation et proposent des conseils ou du guidage.

La plupart de ces environnements prennent en charge l'apprentissage de la syntaxe et sémantique d'un langage de programmation [10]. Ils s'intéressent moins à la résolution de problème et proposent une forme d'aide. Nous remarquons également l'absence d'aide intelligent. Robins disait que pour passer d'un novice en expert il faut faire plus de pratique (faire beaucoup des exercices) [31]. Les outils de génération d'exercice permettraient d'améliorer les compétences des apprenants s'ils proposaient des activités personnalisées qui correspondent à une aide intelligente suite à une évaluation visant à détecter les difficultés de l'apprenant face à une activité donnée.
Dans notre futur travail on proposera un modèle d'évaluation, qui permettra de détecter les difficultés de l'apprenant et par la suite permettra à l'outil intégrant ce processus d'évaluation de générer une activité permettant d'améliorer les compétences manquantes.

References

[1] J. Kaasbøll, "Learning Programming," *Univ. Oslo*, 2002.
[2] A. Gomes and a. J. Mendes, "Learning to program - difficulties and solutions," *Int. Conf. Eng. Educ. – ICEE 2007*, no. January 2007, 2007.
[3] T. Jenkins, "Jenkins on the Difficulty of Learning to Program," 2002. [Online]. Available: http://www.psy.gla.ac.uk/~steve/localed/jenkins.html. [Accessed: 25-Apr-2016].
[4] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. Mccartney, J. E. Moström, K. Sanders, and O. Seppälä, *A Multi-National Study of Reading and Tracing Skills in Novice Programmers*. 2007.

[5] P. H. Tan, C. Y. Ting, and S. W. Ling, "Learning difficulties in programming courses: Undergraduates' perspective and perception," *ICCTD 2009 - 2009 Int. Conf. Comput. Technol. Dev.*, vol. 1, no. 2003, pp. 42–46, 2009.

[6] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *ACM SIGCSE Bull.*, vol. 37, no. 3, p. 14, 2005.

[7] L. E. Winslow, "Programming Pedagogy - A Psychological Overview," *ACM SIGCSE Bull.*, vol. 28, no. 3, pp. 17–22, 1996.

[8] I. Milne and G. Rowe, "Difficulties in Learning and Teaching Programming — Views of Students and Tutors," *Educ. Inf. Technol.*, vol. 7, pp. 55–66, 2002.

[9] V. Renumol, "Classification of cognitive difficulties of students to learn computer programming," *Indian Inst. …*, 2009.

[10] T. Rongas, A. Kaarna, and H. Kälviäinen, "Classification of computerized learning tools for introductory programming courses: Learning approach," *Proc. - IEEE Int. Conf. Adv. Learn. Technol. ICALT 2004*, pp. 678–680, 2004.

[11] N. Guibert, L. Guittet, P. Girard, N. Guibert, L. Guittet, and P. Girard, "Initiation à la Programmation « par l ' exemple » : concepts , environnement , et étude d ' utilité," *Acte Colloq. EIAH'05, Montpellier*, pp. 461–466, 2005.

[12] M. Muratet, "Conception, réalisation et évaluation d'un jeu sérieux de stratégie temps réel pour l'apprentissage des fondamentaux de la programmation," 2010.

[13] J. Sorva, V. Karavirta, and L. Malmi, "A Review of Generic Program Visualization Systems for Introductory," vol. 13, no. 4, 2013.

[14] J. Hidalgo-céspedes, G. Marín-raventós, V. Lara-villagrán, U. D. C. Rica, S. José, and C. Rica, "Learning principles in program visualizations : a systematic literature review," 2016.

[15] K. M. Ala-mutka, "A Survey of Automated Assessment Approaches for Programming Assignments A Survey of Automated Assessment Approaches for Programming Assignments," no. October 2013, pp. 37–41, 2007.

[16] C. Douce, "Automatic Test-Based Assessment of Programming : A Review," vol. 5, no. 3, pp. 1–13, 2006.

[17] P. Ihantola and O. Seppälä, "Review of Recent Systems for Automatic Assessment of Programming Assignments," 2010.

[18] M. Luck and M. Joy, "A Secure On-line Submission System," pp. 1–33.

[19] D. Jackson and M. Usher, "Grading Student Programs using ASSYST," pp. 335–339.

[20] T. Flowers, C. a. Carver, and J. Jackson, "Empowering students and building confidence in novice programmers through Gauntlet," *34th Annu. Front. Educ. 2004. FIE 2004.*, pp. 10–13, 2004.

[21] W. L. Johnson, "Understanding and Debugging Novice Programs," vol. 42, no. 1990, pp. 51–97.

[22] W. R. Murray, "Talus : Automatic Program Debugging for intelligent Tutoring Systems," *Thèse Dr.*, p. 40, 1986.

[23] A. Alammary, A. Carbone, and J. Sheard, "Implementation of a Smart Lab for Teachers of Novice Programmers," *14th Australas. Comput. Educ. Conf.*, pp. 121–130, 2012.

[24] L. De-La-Fuente-Valentín, M. Pérez-Sanagustín, P. Santos, D. Hernández-Leo, A. Pardo, C. D. Kloos, and J. Blat, "System orchestration support for a flow of blended collaborative activities," *Proc. - 2nd Int. Conf. Intell. Netw. Collab. Syst. INCOS 2010*, pp. 415–420, 2010.

[25] P. Brusilovsky and S. Sosnovsky, "Individualized Exercises for Self-Assessment of Programming Knowledge : An Evaluation of QuizPACK," vol. 5, no. 3, pp. 1–22, 2006.

[26] C. Higgins, T. Hegazy, P. Symeonidis, and A. Tsintsifas, "The CourseMarker CBA System : Improvements over Ceilidh," pp. 287–304, 2003.

[27] F. L. Wang, "Designing Programming Exercises with Computer Assisted Instruction," no. February 2015, 2008.
[28] J.-P. Fournier and J. Wirz, "ALLOGENE: Un environnement d'apprentissage de l'algorithmique.," *Troisième rencontre Francoph. Didact. l'informatique. Assoc. EPI (Enseignement Public Informatique)*, pp. 101–113, 1992.
[29] A. Bey and T. Bensebaa, "ALGO+, an assessment tool for algorithmic competencies.," *EEE Glob. Eng. Educ. Conf. (EDUCON). IEEE*, pp. p. 941–946, 2011.
[30] B. A. Blondel, F. Marie, and L. Dominique, "Aide, conseil et explication dans les logiciels éducatifs.," *Rapp. synthèse*, 1998.
[31] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, 2003.