Quels indicateurs pour l'utilisation en binôme d'un environnement de développement intégré ?

Gregory Dyke

Ecole des Mines de Saint-Etienne, 158 Cours Fauriel, 42023 Saint-Etienne, dyke@emse.fr

Résumé

Dans cet article, nous décrivons une expérimentation visant à améliorer le suivi d'étudiants effectuant en binôme des travaux pratiques de programmation. Nous avons mis en place un dispositif de traçage d'environnement de développement intégré permettant de collecter non seulement les erreurs de compilation, mais aussi les erreurs d'exécution ainsi que des indicateurs tenant compte de la présence de deux élèves derrière le même ordinateur (activités de souris et de clavier). En corrélant ces indicateurs avec le résultat d'évaluation humaine des TP et les notes individuelles lors d'évaluations sommatives, nous espérons mettre en évidence les prédicteurs de performance les plus fiables.

Introduction

La problématique de suivi de l'apprenant, permettant aux tuteurs de suivre de nombreux élèves ou groupes de manière efficace n'est pas nouvelle (e.g. May *et al.*, 2008). Dans le cadre de l'enseignement de la programmation, plusieurs auteurs ont tracé l'activité de programmation à divers niveaux afin de corréler cette activité avec la performance finale, et ainsi de proposer aux étudiants des pratiques optimales (e.g. Jadud, 2005 ; Johnson *et al.*, 2003 ; Fenwick *et al.*, 2009).

Or, dans ces études, les étudiants sont seuls derrière leur ordinateur. Cependant, pour des raisons pédagogiques ou de limitation de ressources, il est fréquent que les travaux pratiques de programmation se fassent en binôme, facteur qui n'est pas pris en compte. Il apparaît dès lors important de déterminer si l'activité du binôme mène à une bonne compréhension.

Dans cet article, nous examinons les indicateurs utilisés pour mesurer la qualité de programmation et y rajoutons des métriques indicatrices du travail en binôme. Nous présentons une expérimentation en cours qui vise à corréler ces traces d'activité de programmation avec les résultats collectifs (évaluation du travail en binôme) et individuels (évaluation par QCM). Bien que l'analyse de cette expérimentation n'ait pas encore débuté, nous considérons instructif de terminer par une présentation des analyses que nous ferons, et des buts que nous visons à l'avenir.

Contexte

L'expérience en cours se déroule sur une promotion en première année de Grande Ecole d'ingénieurs comprenant 108 étudiants. Lors du cours de *Programmation Orientée-Objet en Java et Modélisation UML*, les étudiants effectuent sept scéances de TP de 2 heures en Java, par groupe de deux élèves derrière chaque poste. C'est leur deuxième cours de programmation et leur première expérience avec le langage Java.

C'est aussi leur première expérience avec l'Environnement de Dévelopement Intégré (IDE ou Integrated Development Environment) Eclipse. Cet IDE effectue une compilation en continu et nous permet de tracer une variété d'informations que nous développerons dans les sections à venir. Les élèves sont évalués au travers de certains de leurs TPs (note de binôme) et d'un QCM final (note individuelle).

Indicateurs de Succès en Programmation

L'utilisation, lors de l'enseignement de la programmation, des « meilleures pratiques », telles que celles décrites par PSP (Personal Software Process ; Humphrey, 1995), peut mener à de fortes améliorations dans la qualité du résultat produit (Johnson *et al.*, 2003). Les métriques utilisées pour évaluer ces pratiques sont basées sur des indicateurs de bas niveau qui décrivent l'effort investi dans le programme, la taille du programme, ainsi que ses défauts. L'effort peut être évalué par le temps passé sur chaque fichier, à coder ou à naviguer.

L'approche prescriptive de PSP peut-être mis en contraste avec diverses approches descriptives, particulièrement lors de l'observation de programmateurs novices. Jadud (2005) montre que la persistance d'erreurs de compilation entre deux instance de compilation ainsi que la présence répétée d'erreurs de compilation identiques est positivement corrélée avec l'échec. Rodrigo *et al.* (2009) ajoutent à ces mesures des facteurs affectifs et sociaux tels que l'ennui, la confusion et la discussion au sujet de l'IDE (mais non en rapport avec la tâche à accomplir).

Fenwick *et al.* (2009), confirment les résultats de Jadud et indiquent que le travail en sessions courtes et multiples est aussi bénéfique à la note finale. Ils ajoutent que le temps passé sur une tâche n'est pas corrélé avec la note : les étudiants peuvent passer peu de temps par ennui ou par facilité, et beaucoup de temps par perfectionnisme ou par la faute d'échecs successifs. Ils notent aussi que les élèves surestiment à la fois leur fréquence de compilation, la qualité des tests auxquels leur code a été soumis, et le bon découpage de leur code en modules.

A ces indicateurs, nous pouvons ajouter quelques indicateurs hypothétiques issues de notre propre expérience d'enseignement : respect de conventions de nommage, utilisation des capacités de l'IDE (navigation, indentation automatique, *refactoring*, suggestions de corrections d'erreurs de compilation) et fréquence d'exécution (la compilation étant automatique sous Eclipse).

Indicateurs du Mode de Collaboration

Dans un désir de ne tracer que ce qui est aisément collectable, c'est à dire les évènements dans l'IDE Eclipse, nous passons nécessairement à côté de toute l'interaction langagière et gestuelle. Nüssli *et al.*, 2009 ont cependant montré des résultats encourageants sur l'analyse de collaboration basée sur des données de bas niveau (oculométrie et présence/absence de parole). Ces résultats suggèrent qu'il pourrait être bénéfique de tracer l'activité de la souris (indicatrice de déictiques) ainsi que celles du clavier et des changements d'onglet (i.e. changement de fichier), qui pourraient être indicatrices de potentielles pertes d'alignement et d'accord (Baker, 2002) au sein du binôme (bien qu'il ne soit pas possible d'identifier l'auteur de ces opérations).

Collecte et Analyse de Données

Plusieurs outils ont été proposés pour récolter des données sur l'activité de programmation, dont ClockIt (Fenwick *et al.*, 2008) et HackyStat (Johnson *et al.*, 2003). Nous avons avons modifié ce dernier pour inclure le traçage de l'activité de souris, de l'activité du clavier, l'activité d'exécution, l'activité d'utilisation de fonctions propres à l'IDE, en plus des éléments tracés à la base tels que le nombre d'erreurs, la taille des fichiers et le temps d'activité. L'outil HackyStat se présente sous la forme d'un plugin pour Eclipse et d'un serveur de récolte de données.

Une fois les données récoltées, nous les organiserons comme conseillés par Mostow et Beck (2006) en préparation à l'application d'algorithmes de fouille de donnée. Ainsi, chaque évènement (ou indicateur calculé à partir de cet évènement) peut être associé avec le résultat en sortie (la note du binôme et les notes individuelles). Nous espérons ainsi montrer quels facteurs sont les prédicteurs les plus importants d'échec ou de réussite.

Au delà de ces premières analyses, nous souhaitons aussi identifier les binômes disparates (où seul un des élèves obtient une bonne note) et tenter de classifier des binômes rencontrant des problèmes similaires afin de proposer une meilleure répartition en salle de TP. Il sera aussi intéressant d'essayer de minimiser la quantité de données nécessaires pour prédire le résultat d'apprentissage avec une fiabilité similaire. Si cette quantité est suffisamment faible, nous pourrons reconduire cette expérience l'année prochaine en tentant d'utiliser nos résultats pour détecter les binômes en besoin de soutien. En effet, toutes les données étant récoltées automatiquement, le calcul de prédiction pourra se faire d'une séance à l'autre.

Enfin, l'utilisation de l'outil Tatiana (Dyke *et al.*, 2009) nous permettra d'examiner de plus près certaines séances de programmation (c'est à dire reproduire et annoter l'activité de production de code à une date ultérieure) afin de générer des hypothèses sur de nouveaux indicateurs qui nous permettraient d'améliorer la performance de notre modèle prédictif. En particulier, l'évaluation par QCM nous permettra d'établir, pour chaque étudiant, un modèle d'apprenant basé sur ses connaissances en programmation et modélisation orienté objet. Il sera intéressant d'examiner s'il est possible de prédire ce modèle à partir des données collectées.

Conclusion

Dans cet article, nous avons présenté une expérience en cours sur des étudiants programmant en binôme avec le langage Java et l'IDE Eclipse. Les données récoltées ont été choisies d'après les résultats d'études similaires et ont été complétées par des données de bas niveau qui, nous espérons, nous informeront sur le mode de travail collaboratif et seront aussi prédictrices de résultats lors d'évaluations de travail de groupe et de connaissances individuelles. Nous avons enfin présenté certaines des analyses que nous ferons, ainsi que de l'application que nous espérons faire des résultats.

Références

Baker, M., and Lund, K. 1997. Promoting reflective interactions in a computer-supported collaborative learning environment. *Journal of Computer Assisted Learning*, 13, 175–193.

Dyke, G., Lund, K., and Girardot, J.-J. 2009. Tatiana : an environment to support the CSCL analysis process. *CSCL 2009*. Rhodes, Greece, 58–67.

Fenwick, J. B., Norris, C., Barry, F. E., Rountree, J., Spicer, C. J., and Cheek, S. D. 2009. Another look at the behaviors of novice programmers. *SIGCSE '09*. ACM, New York, NY, 296–300.

Humphrey, W. S. 1995. *A Discipline for Software Engineering*. Addison-Wesley.

Jadud, M. 2005. A first look at novice compilation behaviour using BlueJ. *Computer Science Education*, 15(1):25–40.

Johnson, P. M., Kou, H., Agustin, J., Chan, C., Moore, C., Miglani, J., Zhen, S., and Doane, W. E. 2003. Beyond the Personal Software Process: metrics collection and analysis for the differently disciplined. *In Proceedings of the 25th international Conference on Software Engineering*. IEEE Computer Society, Washington, DC, 641–646.

May, M., George, S., and Prévôt, P. 2008. A closer look at tracking human and computer interactions in web-based communications. *International Journal of Interactive Technology and Smart Education*, 5(3): 170–188.

Mostow, J., and Beck, J. 2006. Some useful tactices to modify, map, and mine data from intelligent tutors. *Natural Language Engineering (Special Issue on Educational Applications),* 12(3):195–208.

Nüssli, M.-A., Jermann, P., Sangin, M,. and Dillenbourg, P. 2009. Collaboration and abstract representations: towards predictive models based on raw speech and eye-tracking data. *CSCL 2009*. Rhodes.

Rodrigo, M. T., Baker, R. S., Jadud, M. C., Amarra, A. M., Dy, T., *et al.* 2009. Affective and behavioral predictors of novice programmer achievement. *ITiCSE '09*. Paris, France. 156-160.