Un environnement XP de tests unitaires répartis

Michel Babri* - Ibrahim Lokpo* - Gérard Padiou**

* Département Mathématiques et Informatique
Institut National Polytechnique Félix Houphouet-Boigny
Yamoussoukro, Côte d'Ivoire
michel_babri@yahoo.fr, lokpo@hotmail.com

** Institut de Recherche en Informatique de Toulouse, UMR CNRS 5505
INPT-ENSEEIHT
2 rue Camichel, BP 7122
F-31071 Toulouse cedex 7
padiou@enseeiht.fr

RÉSUMÉ. La programmation eXtrême dite XP (eXtreme Programming) est une méthodologie efficace de développement rapide de logiciels qui exige que les membres d'une équipe de développeurs soient physiquement proches les uns des autres. Des travaux de recherche tentent d'étendre les principes de cette méthode à un contexte distribué. Le défi est alors de préserver les qualités fondamentales de l'approche tout en s'affranchissant de la contrainte de proximité physique des développeurs. Notre travail s'inscrit dans cette logique. De façon plus précise, nous proposons de fonder cette extension de XP à un environnement distribué sur l'assistance à la réalisation des tests unitaires, pierre angulaire de la méthode XP.

ABSTRACT. eXtreme Programming (XP) is a methodology based on principles and practices for quickly developing software. However, this approach requires the programmers to be co-located. Many research projects propose how to extend XP to a distributed environment. Therefore, the challenge is to sue the XP approach without conflicting with the distributed constraints. Our work is a step towards this goal. More precisely, we propose an extension based on assistance for supporting distributed unit testing, one of the key principles of XP.

MOTS-CLÉS : Programmation XP, CSCW , Tests unitaires, XP Distribué.

KEYWORDS : eXtreme Programming, CSCW, unit testing, distributed XP.

1. Introduction

La programmation eXtrême ou eXtreme Programming (XP) [1] est une méthode de production de logiciels qui connaît un large succès depuis quelques années. Avec cette méthode, il s'agit de mettre à disposition le plus tôt possible, un produit opérationnel de qualité. Elle repose sur le principe du développement guidé par les tests unitaires et la programmation par paire [2]. Dans sa forme originèle, cette méthode présente toutefois quelques sévères limitations dont les principales sont la localisation centralisée et le nombre limité des développeurs.

Des travaux de recherche ont été menés pour lever ces limitations. Ces travaux ont permis d'aboutir au concept de Programmation eXtrême Distribuée (Distributed XP) [6]. Le défi posé est de préserver les qualités fondamentales de la méthode tout en s'affranchissant de la contrainte de proximité physique des développeurs. Notre travail s'inscrit dans cette logique. C'est une contribution à la préservation des fondements de l'approche XP dans un contexte distribué en se fondant sur l'assistance au test unitaire, principe essentiel de la méthode. Dans ce document, nous analysons une mise en œuvre des tests unitaires dans le contexte d'une équipe distribuée de développeurs.

Dans une première section, nous évaluons les problèmes liés au concept de XP distribué. La section suivante expose quelques solutions. Nous développons ensuite les principes et la mise en œuvre de notre solution. Nous terminons par un état d'avancement de notre projet.

2. XP et la distribution

La méthode Distributed-XP (DXP) [6] se définit comme une extension de la méthode XP dans laquelle la contrainte de localisation géographique des membres d'une équipe de développeurs a été levée. En tant que telle, cette approche DXP doit relever de nouveaux défis notamment dans les pratiques XP fortement liées à cette contrainte de localisation tels que la programmation par paire, le jeu de rôle et l'intégration continue.

2.1. La programmation par paire

La programmation par paire est une des principales pratiques XP dans laquelle deux programmeurs travaillent sur une même machine. Ainsi, les binômes peuvent apporter des changements à tout endroit du code. Dans un binôme, les rôles sont interchangeables dans la même journée ; ceci amène le binôme à une parfaite maîtrise de son sujet. Une question essentielle que l'on peut se poser est de savoir quel serait le comportement d'une paire de

programmeurs si ceux-ci ne sont plus sur la même machine, voire s'ils sont géographiquement distants.

2.2. Les tests unitaires

On distingue deux types de tests: - le test fonctionnel dans lequel on ne vérifie que les résultats en sortie obtenus avec des paramètres valides. - le second type de test consiste à tenir compte du fonctionnement interne du module et des liens entre modules. Le test donne également des informations sur l'état interne du module ou de la classe.

Les programmeurs sont amenés à écrire toute une batterie de tests automatiques des classes ou modules de l'application, et ceci avant qu'ils n'écrivent les différents codes. Lorsque ces tests de non régression sont satisfaits, les développeurs sont assurés que leurs modifications n'ont pas introduit d'erreurs dans les jeux de tests précédents. Le comportement d'ensemble de l'application sera, quant à lui, pris en charge par des tests dits de recette. Ceux-ci permettent de capter l'état d'avancement de l'ensemble d'un projet.

Etendre les tests à un environnement distribué est difficile pour au moins deux raisons : - les problèmes liés au déploiement des composants et à la topologie du réseau sous-jacent ; - la difficulté liée à la synchronisation des tests. Différentes solutions ont été proposées avec des extensions de Junit au web (http://httpunit.source.forge.net).

2.3. Intégration continue et partage de responsabilité

Ces deux propriétés sont étroitement liées. Le partage de responsabilité indique que toute paire de programmeurs peut modifier et incorporer tout ou partie du code. Par intégration continue, XP exige que la suite de tests soit tout de suite exécutée. De cette façon, les éventuels problèmes induits apparaissent immédiatement. Toutefois, l'intégration continue ne peut se faire qu'à des instants précis. Ceci devient primordial dès lors que les binômes sont distants.

3. Solutions pour appliquer XP dans un contexte distribué

Le développement des télécommunications et plus particulièrement d'Internet a permis à de nombreuses équipes dispersées dans le monde de collaborer à un même projet(https://www.cvshome.org/). De telles équipes partagent le code sans pour autant être obligées de se rencontrer physiquement. La contrainte de proximité géographique imposée par XP, aux équipes de développement doit être levée pour permettre à des équipes virtuelles d'appliquer les autres recommandations XP.

La programmation par paire, telle que prescrite par XP, oblige les membres du binôme à travailler sur la même machine et donc dans un même bureau. Or, de nombreuses applications telles que, par exemple, le télé-enseignement, peuvent être organisées en binômes. Il peut arriver que les membres d'un binôme ne soient pas sur le même campus. Dans ces conditions, il paraît intéressant de leur donner tout de même les moyens de développer par paire distribuée. Internet offre de nombreux outils qui peuvent être mis à profit pour faire de la programmation par paire distribuée (DPP) : messagerie électronique, vidéo conférence, etc.

Des logiciels de gestion de versions concurrentes ainsi que des environnements de développement intégré sont également utilisés. La programmation par paire distribuée [8] offre finalement une lisibilité plus forte aux développeurs, mais l'éloignement physique nécessite beaucoup de temps d'explication verbale.

4. Un système de tests unitaires distribués

Les résultats de différents travaux de recherche permettent de s'affranchir des contraintes initiales de XP sans perte de son efficacité. Par exemple [6,7] ont proposé la notion d'équipes virtuelles pour garantir les fonctionnalités de CSCW (Computer-Supported Cooperative Work) dans un contexte distribué, notamment la programmation par paire. De telles approches nécessitent l'existence d'un réseau fiable. Nous proposons une approche qui tient compte d'un contexte dans lequel le réseau de connexion n'offre pas un haut débit. Nous nous intéressons aux tests unitaires dans un environnement distribué, comme base d'une application de XP dans un tel contexte. Notre outil s'adresse particulièrement aux équipes de développement dispersées ou mobiles qui souhaitent toutefois adopter XP comme méthode. Cet outil est un moyen destiné à des paires de programmeurs distantes qui veulent écrire des tests et les modules associés dans un projet auquel elles collaborent.

4.1. Choix d'un environnement de programmation

Cette expérimentation s'appuie sur l'environnement de programmation BlueJ (http://bluej.org). Il s'agit d'un environnement de développement intégré pour Java. Prévu pour l'enseignement des concepts orientés objet, il intègre un outil d'assistance aux tests unitaires, Junit (http://www.junit.org).

Nous avons aussi utilisé JML [3,4], un langage de spécification de modules Java permettant de faciliter le développement des tests par génération automatique. De plus, la gestion des versions concurrentes des modules à développer a nécessité l'utilisation d'un logiciel adéquat, en l'occurrence CVS (http://www.cvshome.org). CVS permet de suivre toutes les modifications opérées sur tout ou partie du code source d'un projet.

4.2. JUTE : un Environnement de Tests Distribués pour Java

JUTE (Java Unit Testing Environnement) est un outil qui permet aux paires de programmeurs d'appliquer un nombre important de tests unitaires sans se préoccuper ni de l'organisation de ceux-ci, ni de la manière dont ils s'exécutent. En programmation eXtrême, les tests unitaires doivent être exécutés de façon la plus systématique possible [5]. Si ceux-ci passent avec succès, les développeurs sont sûrs d'avoir obtenu une nouvelle version de leur application globale. Il est donc nécessaire de construire une suite globale de tests pour l'ensemble du projet. Dans cette optique, nous avons écrit un générateur automatique qui explore la hiérarchie de paquetages du projet afin d'engendrer une suite globale de tests.

La figure 1 montre la structure générale de notre outil. Chaque paire envoie son code après l'avoir testé; or le test n'est rien d'autre que du code en plus; celui-ci est donc également envoyé au dépôt. Chaque paire de programmeurs récupère ce dont elle a besoin (code et tests associés) à partir du dépôt central. Grâce à l'outil mis en place, la paire reconstruit sa suite de tests et l'exécute. Lors de cette exécution, l'échec d'un test précédemment concluant peut provenir de deux sortes d'erreurs : -un comportement marginal n'a peut-être pas été prévu ou des interactions entre modules n'ont pas été prises en compte ; - une erreur due au nouveau code qui vient d'être écrit. Lorsqu'un test est concluant, son résultat apparaît dans la hiérarchie des classes du projet. Dès lors, les principales fonctions, entre autres celles de cvs peuvent être appelées.

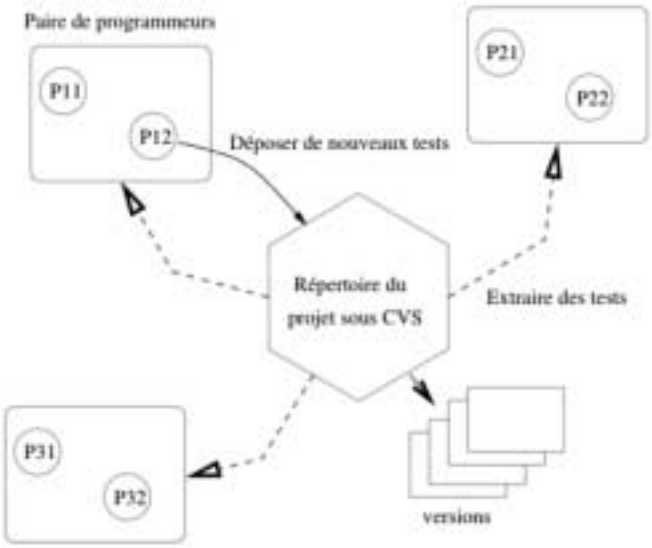

Figure 1 : Structure générale du système de tests distribués.

Un exemple de scénario associé à la figure 1 se présente comme suit : supposons que la paire P3 de programmeurs (P31, P32) désire écrire un module M3 qui fait appel aux modules M1 de (P11, P12) et M2 de (P21, P22). JUTE permet à P3 d'extraire du dépôt central du projet M1 et M2 ainsi que les tests associés et de construire de façon automatique les tests associés à l'ensemble des modules utilisés par la paire P3.

4.3. Etat d'avancement du projet

Dans la version actuelle du projet, les trois extensions à l'environnement BlueJ ont été réalisées. La figure 2 montre cette version étendue qui fait donc apparaître un menu à trois options :

- l'option test engendre la suite globale de test qui sera par la suite compilée par Junit.
- l'option jml permet d'engendrer des classes (par une commande jmlc) et une suite de tests unitaires (par une commande jmlunit).
- Avec la dernière option, on peut appliquer différentes fonctions cvs à des modules sélectionnés dans une liste affichée.

Ceci constitue une première étape d'assistance au test unitaire dans un contexte distribué indispensable à la pratique de la programmation eXtrême. Une autre caractéristique à développer est la programmation par paire.

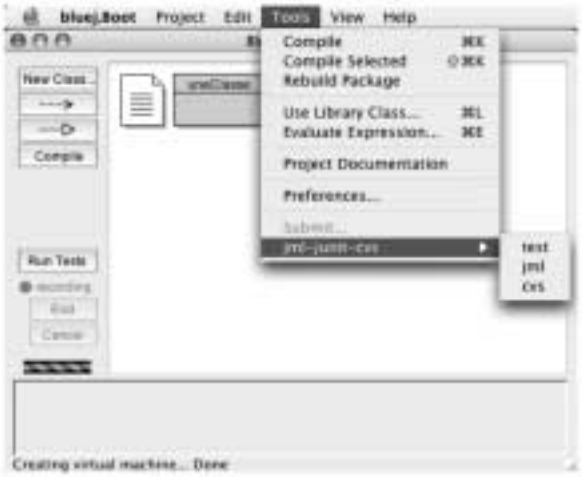

Figure 2 : Menu de base

5. Conclusion

Dans ce papier, nous avons décrit et présenté notre approche pour la pratique des tests unitaires dans un environnement distribué. L'environnement JUTE permet en effet de construire une suite de tests qui sont exécutés de façon quasi automatique même par des équipes distantes. Il s'appuie sur trois extensions apportées à l'environnement de développement Bluej.

La contribution de JUTE est : d'une part, d'engendrer de façon automatique la suite globale de tests à partir de la hiérarchie des paquetages d'un projet en cours de développement et d'autre part, de permettre de relâcher la contrainte de proximité physique imposée aux programmeurs, grâce à l'utilisation d'un gestionnaire de versions concurrentes.

Les choix des différents environnements et logiciels utilisés se justifient essentiellement par leur caractère "open source" et, d'un point de vue performance, par l'absence de moyen de communication à haut débit.

D'autres approches complémentaires sont en cours de développement, en particulier, le projet XPWeb (http://xpweb.sourceforge.net/) de serveur Web offrant des outils de base de gestion de la coopération (CSCW) dans un développement XP et le projet associé EcliWPWeb (http://sourceforge.net/projects/eclixpweb/) permettant une connexion sous forme de plug-in avec l'environnement de développement intégré Eclipse (http:/www.eclipse.org). Par rapport à ces travaux en cours, notre approche s'en distingue par sa focalisation sur le test avec en particulier l'intégration de JML. Cependant, selon la même démarche que pour l'environnement Eclipse, l'utilisation du service Web de type CSCW offert par XPWeb pourrait être intégré en tant qu'extension de l'environnent BlueJ.

Le projet est encore en développement. La prochaine étape sera l'évaluation du serveur XPWeb pour l'assistance à la programmation par paire à partir de l'environnement BlueJ étendu actuel.

6. Bibliographie et biographie

6.1 Bibliographie

[1] Kent Beck. Extreme Programming Explained : Embrace Change. The XP Series. Addison Wesley Publishing Company, 2000.

[2] Kent Beck. Test-driven Development by example. The Addison Wesley Signature Series. Addison Wesley Publishing Company, 2001.

[3] Y. Cheon and G.T. Leavens. A Runtime assertion checker for Java Modeling Language (JML). In H.R. Arabnia and editors Y. Mun, editors, International Conference on Software Engineering Research and Practice (SERP '02), pages 322--328. CSREA Press, Las Vegas, 2002.

[4] Yoonsik Cheon and Gary T. Leavens. A simple and practical approach to unit testing : The JML and JUnit way. In Hern´andez Juan (Eds.), editor, 16th European Conference ECOOP 2002 Workshops and Posters, volume 2548 of Lecture Notes in Computer science. Springer Verlag, 2002.

[5] Erich Gamma and Kent Beck. Test infected : Programmers love writing tests. Java Report, 3(7), July 1998.

[6] F. Maurer. Supporting distributed extreme programming. In Don Wells and Laurie A. Williams, editors, XP/Agile Universe 2002, Second XP Universe and First Agile Universe Conference Chicago, IL, USA, August 4-7, 2002, ProceedingsXP/Agile Universe, volume 2418 of Lecture Notes in Computer Science, pages 13--22. Springer, 2002.

[7] H. Skaf-Molli, P. Molli, G. Oster, Cl. Godart, P. Ray, and F. Rabhi.Toxic farm: A cooperative management platform for virtual teams and enterprises. In 5th International Conference on Enterprise Information Systems ICEIS03. Angers, France, April 2003.

[8] David Stotts, Laurie Williams et al: Virtual teaming: Experiments and Experiences with Distributed Pair Programming, http://rockfish-cs.cs.unc.edu/pubs/

6.2 Biographie

Michel Babri est enseignant-chercheur en Informatique à l'Institut National Polytechnique Félix Houphouet-Boigny de Yamoussoukro (Côte d'Ivoire). Ses centres d'intérêt sont l'algorithmique, les systèmes d'exploitation et bien sûr les approches de travail coopératif appliquées à la programmation.

Ibrahim Lokpo est enseignant-chercheur en Informatique à l'Institut National Polytechnique Félix Houphouet-Boigny de Yamoussoukro (Côte d'Ivoire). Ses centres d'intérêt sont entre autres les systèmes répartis et les applications de travail coopératif distribué.

Gérard Padiou est enseignant à l'ENSEEIHT et chercheur à l'IRIT. Ses centres d'intérêts sont l'algorithmique répartie et la vérification de programmes réactifs répartis.