

PatternsBox–Ptidej, intégration de deux outils de conception et de rétro-conception à Eclipse

Yann-Gaël Guéhéneuc*, **Hervé Albin-Amiot[†]** et **Pierre Cointe**

Projet OBASCO

École des Mines de Nantes / INRIA Rennes

4, rue Alfred Kastler – BP 20 823

44 307 Nantes Cedex 3

France

{guehene|albin|cointe}@emn.fr

Nous nous intéressons aux phases de conception et de maintenance de programmes à objets. Nous pensons que l'utilisation des motifs (tels les motifs de conception de [2]) facilite, d'une part, la conception et la documentation [1] et, d'autre part, la compréhension et l'amélioration [3] de ces programmes.

Nous avons développé à l'occasion de nos travaux de thèse de doctorat, PATTERNSBOX [1] et PTIDEJ [3], deux outils de conception et de maintenance basés sur les motifs. PATTERNSBOX permet (i) d'appliquer les solutions recommandées par un ensemble de motifs de conception et (ii) d'identifier dans l'architecture d'un programme à objets les solutions complètes correspondant à des motifs donnés. PTIDEJ permet (i) d'identifier des solutions à des motifs donnés et leurs variantes et (ii) d'identifier des défauts de conception dans l'architecture du programme.

Ces deux outils présentent un intérêt pour les développeurs essentiellement lorsqu'ils sont intégrés avec les outils utilisés habituellement pendant les phases de conception et de maintenance. C'est pourquoi nous les avons intégrés à des environnements de développements intégrés (EDI).

Dans cet article, et après avoir présenté l'EDI d'OTI / IBM ECLIPSE [5], nous décrivons notre première expérience relative à l'intégration de PTIDEJ avec ECLIPSE, puis nos projets quant à l'intégration de PATTERNSBOX.

1 Présentation de la plate-forme Eclipse

Après VISUAL AGE pour Smalltalk (199–), VISUAL AGE pour Java (1996) et VISUAL AGE MICRO EDITION (1999), la plate-forme ECLIPSE représente une nouvelle génération d'environnement de développement.

*Ce travail est en partie financé par Object Technology International, Inc. – 2670 Queensview Drive – Ottawa, Ontario, K2B 8K1 – Canada

[†]Ce travail est en partie financé par Soft-Maint – 4, rue du Château de l'Éraudière – 44 324 Nantes – France.

Il s'agit d'une “*platform that has been designed from the ground up for building integrated web and application development tooling*” [6]. ECLIPSE a donc pour ambition d'être “*une plate-forme universelle pour outils intégrés*” [4].

ECLIPSE est conçue pour faciliter le développement, l'intégration et le partage d'informations entre outils développés par différentes entreprises. Elle veut apporter une solution aux problèmes d'interopérabilité (importation et exportation des données) entre outils et contribuer à limiter le foisonnement des interfaces graphiques utilisateurs. Pour cela, elle fournit un ensemble limité de services basiques communs à tous les outils et des mécanismes d'extension, à partir desquels il est possible de développer et d'intégrer des outils de différentes natures.

1.1 Modèle

ECLIPSE définit un environnement *générique* et *ouvert* pour construire des outils *intégrés* à la plate-forme et interopérables.

Générique ECLIPSE sait uniquement gérer des ressources. Les ressources sont soit des fichiers soit des dossiers; organisés hiérarchiquement les uns par rapport aux autres. Elles sont accessibles grâce à un identifiant unique et indépendant de la localisation physique des fichiers (locale, sur l'intranet, sur l'Internet).

Elle offre des services d'ajout, de retrait, de modification (édition, attribut), de construction (analyse, compilation), d'exécution et de contrôle de version (locales) de ces ressources.

Elle offre également des mécanismes de marqueurs (pour marquer les ressources ou des points dans les ressources); de deltas (pour comparer deux ressources); de perspectives (pour rassembler en un tout cohérent des vues, des éditeurs, des menus...); de vues; d'éditeurs; de menus et menus contextuels; de barres d'outils; de pages de préférences; de pages d'aide.

Oouvert ECLIPSE offre un modèle d'extension simple et consistant. Une extension (*plug-in*) est en ensemble de classes qui vient étendre la plate-forme et les extensions déjà présentes en se greffant sur des points d'extension (*extension*) définis par celles-ci. Une extension est automatiquement activée au démarrage de la plate-forme si nécessaire ou à la demande.

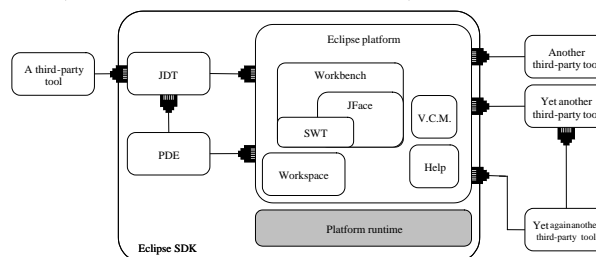
Les points d'extensions disponibles sont définis par la plate-forme et les extension existantes dans des fichiers XML associés (fichiers *plugin.xml*).

Une extension indique à la plate-forme les points d'extension qu'elle utilise en associant à chaque point d'extension la ou les classes mettant en œuvre l'extension (toujours dans le fichier *plugin.xml*).

Les services de bases de la plate-forme eux-même sont définis comme des extensions au noyau minimal de la plate-forme.

De plus, ECLIPSE est sous licence *Common Public Licence v1.0 (CPL)*. Cette licence facilite la diffusion d'outils intégrés à la plate-forme aussi bien sous la forme de code source libre que sous la forme d'outils à usage restreint (commerciaux, par exemple)

Intégrés À partir des points d'extension, les outils peuvent venir s'intégrer les uns aux autres, dans la limite où ils fournissent les points d'extension nécessaire. La figure ci-contre décrit brièvement l'intégration des extension.



1.2 Implémentation

La plate-forme ECLIPSE est implémentée en Java. Elle offre, entre autres, les concepts et les services génériques suivants :

- la fenêtre *Workbench* représente l’environnement de développement. L’objectif du *Workbench* est de réaliser “*seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of Workbench resources*” [6]. Plus d’une fenêtre *Workbench* peut exister à un moment donné. La figure 1(a) montre la fenêtre *Workbench* vide ;
- chaque fenêtre *Workbench* contient une ou plusieurs perspectives. La figure 1(b) montre la perspective ressource, par défaut. La perspective ressource présente la vision la plus simple des données manipulées par la plate-forme. Elle définit une vue *Navigator*, figure 1(c), qui présente les ressources sous une forme hiérarchique et permet de les manipuler ;
- par exemple, d’autres outils peuvent afficher et manipuler les ressources différemment de la vue *Navigator*. La perspective Java, qui offre un environnement complet pour le développement de programmes Java, définit une vue *PackageExplorer* dans laquelle les ressources sont affichées et manipulées sous la forme de paquetages et d’unités de compilation, figure 1(d).
- de nombreuses perspectives et vues existent ou sont en développement, celles fournies par défaut avec ECLIPSE v2.0 sont :
 - *CVS Repository Exploring* pour la gestion de versions ;
 - *Debug* pour le débogage de programmes Java ;
 - *Java Browsing* et *Java Type Hierarchy* pour le développement de programmes Java ;
 - *Plug-in Development* pour le développement de nouvelles extensions à ECLIPSE ;
- un aspect multi-langage pour écrire des environnements de développement pour des langages autres que Java (par exemple, C++, Cobol ou Prolog) par l’intermédiaire des points d’extension liés aux éditeurs et aux mécanismes de construction et d’exécution : il est possible d’écrire un éditeur pour un langage autre que Java et d’étendre le mécanisme de construction et d’exécution pour ce langage.
- une nouvelle bibliothèque de composants graphiques, SWT, plus efficaces (mémoire et vitesse de rafraîchissement) et mieux intégrés au système d’exploitation sous-jacent. Les composants SWT se manipulent d’une manière plus proche du langage C++ et très éloignée des abstractions de haut-niveau proposées par la bibliothèque SWING.

2 Intégration de Ptidej à Eclipse

2.1 Présentation de Ptidej

PTIDEJ est un outil de rétro-conception de l’architecture d’un programme à objets basé sur les motifs de conception. Il permet de modéliser et de visualiser l’architecture d’un programme à la UML en analysant les classes du programme, puis d’identifier les groupes de classes dont la structure et l’organisation ressemble à la solution d’un motif de conception donné, répertorié dans un référentiel.

PTIDEJ offre une meilleure compréhension du programme pendant la phase de rétro-conception en (i) modélisant l’architecture du programme à un plus haut niveau d’abstraction que son code source et en (ii) explicitant les motifs de conception utilisés.

De plus, PTIDEJ facilite l'identification de défauts de conception : en identifiant les groupes de classes dont l'organisation ressemble à un motif de conception ou à un défaut de conception donné.

2.2 Notre expérience

Nous avons donc intégré l'outil PTIDEJ à la plate-forme ECLIPSE. Cette expérience nous a permis de constater que :

- les services de gestion des ressources (fichiers Java, fichiers de projets) sont bien fournis par ECLIPSE et la perspective Java. Toute l'implémentation relative à la lecture/écriture de fichiers est donc simplifiée ;
- la visualisation de l'architecture du programme et des motifs de conception est intégrée à la perspective Java, le développeur n'a donc pas besoin de changer d'outil pour obtenir tantôt une vision Java, tantôt une vision architecturale de son programme.

Par contre, nous avons également remarqué que :

- l'organisation et le fonctionnement de ECLIPSE est difficile à comprendre au début. Il faut investir du temps pour avoir une vision claire du fonctionnement global de la plate-forme et des inter-actions entre ses services ;
- la documentation de la plate-forme, de ses services basiques et des perspectives par défaut sont parfois trop succinctes. Cependant, les listes de discussions autour de ECLIPSE sont très actives et permettent d'obtenir une solution généralement rapide à tout problème (plus de 50 000 messages).
- la bibliothèque graphique SWT, utilisée par la ECLIPSE pour améliorer ses performances, oblige à réécrire l'interface utilisateur et les bibliothèques graphiques dédiées (pour la visualisation de l'architecture du programme, par exemple). L'utilisation du motif de conception *Usine abstraite* facilite néanmoins cette réécriture ;
- ECLIPSE requiert beaucoup de ressources mémoire et un processeur rapide. Son utilisation peut donc se révéler frustrante sur des ordinateurs anciens. Elle est implémentée en Java, ce qui garantit une portabilité néanmoins limitée par l'utilisation de la bibliothèque graphique SWT. La version WIN32 fonctionne bien, les versions LINUX sont moins optimisées et robustes ;
- les points d'extension utilisés par un outil sont décrits dans un fichier XML. Ce fichier, généré automatiquement à partir de la perspective *Plug-in Development*, permet de découpler implémentation et intégration de l'outil. Mais ce découplage est pénalisant lorsqu'un problème survient : il n'est pas toujours clair si le problème vient de l'implémentation ou de la description de l'extension dans le fichier ;
- ECLIPSE intègre par défaut l'environnement de test JUNIT. L'environnement JUNIT facilite grandement le développement et le test des programmes et des extensions.

En conclusion, l'intégration d'outil à la plate-forme ECLIPSE nécessite, au départ, un important travail de compréhension. Une fois la plate-forme bien en main, l'intégration d'outils se révèle aisée et permet de rapidement développer et expérimenter des idées en se basant sur les services déjà existants. L'intérêt majeur de ECLIPSE est donc la possibilité d'étendre les services fournis par la plate-forme et par les outils déjà existant pour construire rapidement de nouveaux outils intégrés.

3 Intégration de PatternsBox : nos attentes

3.1 Présentation de PatternsBox

PATTERNSBOX est un outil permettant l'application et la détection semi-automatisée des solutions de motifs de conception. Il est conçu pour assister le développeur pendant la phase d'implémentation en s'appuyant sur un environnement de développement intégré (VISUALAGE pour Java, à l'heure actuelle).

Contrairement à PTIDEJ, la détection proposée par PATTERNSBOX n'est pas destinée à la rétro-conception mais à la documentation et à la compréhension de bibliothèques de classes à l'aide de motifs de conception. En particulier, elle n'inclue pas la détection de formes *approchées* de motifs et ne propose pas de mécanisme de rétro-action pour corriger les formes *approchées* identifiées.

La phase d'application est prise en charge par un moteur de transformation source-à-source construit au dessus de l'interface *API Tool Integrator* de VISUALAGE pour Java. Cette interface constitue en quelque sorte le seul point d'extension de l'EDI. Les limitations de cette interface nous ont conduits à mettre en place un analyseur syntaxique Java et une interface de manipulation du source dédiés à PATTERNSBOX, alors que VISUALAGE pour Java inclut déjà ces fonctionnalités.

La phase de détection se base, pour partie, sur une analyse de code octal. Notre choix d'une analyse de code octal s'explique par notre volonté d'aider à la compréhension de bibliothèques de classes dont le code source n'est pas nécessairement disponible ou distribué. VISUALAGE pour Java ne propose pas d'accès direct au code octal des classes présentes dans son référentiel ; l'accès doit être précédé d'une phase d'exportation coûteuse en temps.

L'intégration d'un outil comme PATTERNSBOX doit *préférentiellement* se faire en utilisant une interface graphique utilisateur semblable à celle de l'EDI auquel il est intégré. VISUALAGE pour Java ne propose que très peu d'accès à son interface graphique.

Pour toutes ces raisons et de part la complémentarité de PATTERNSBOX et de PTIDEJ, il nous semble intéressant d'en proposer une version intégrée à ECLIPSE.

3.2 Attentes

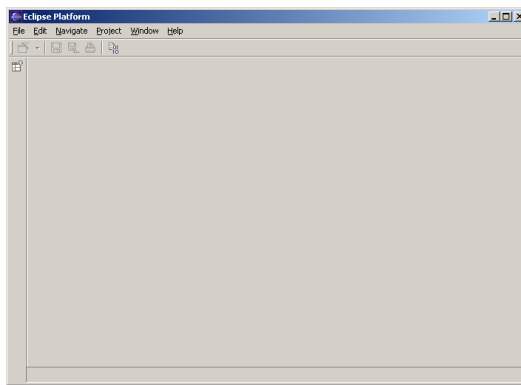
Par rapport à notre *expérience* d'intégration de PATTERNSBOX dans VISUALAGE pour Java, nous notons certains éléments qui devraient faciliter son intégration à la plate-forme ECLIPSE ; en particulier :

1. un accès *simple* à l'arbre de syntaxe abstraite des classes dont le code source nécessite une modification ;
2. un accès au code octal de chaque classe à analyser ;
3. un accès au système graphique pour assurer une intégration *harmonieuse* de l'outil ;
4. des interfaces de programmation graphiques assez proches des standards Java afin de limiter les efforts d'adaptation. Nous pensons en particulier au système événementiel Java et à la notion de *modèle* dans les SWING ;
5. des capacités d'interopérabilité (points d'extension) entre outils pour pouvoir, notamment, faire communiquer PATTERNSBOX et PTIDEJ.

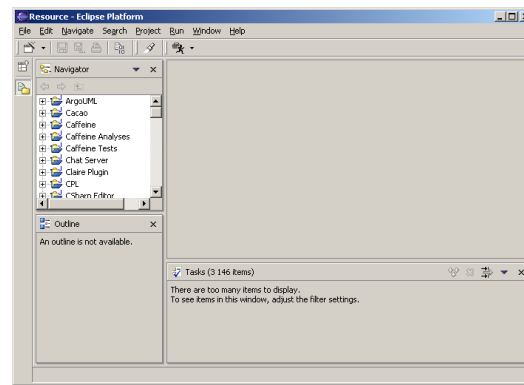
En conclusion, l'intégration des outils PATTERNSBOX et de PTIDEJ à la plate-forme ECLIPSE, offrira à ceux-ci une meilleure interopérabilité.

Bibliographie

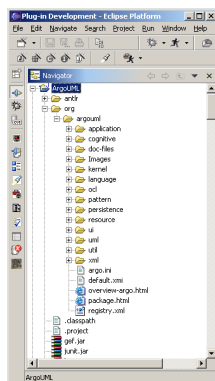
- [1] Hervé Albin-Amiot. *Idiomes et patterns Java : application à la synthèse de code et à la détection*. Thèse de doctorat, université de Nantes, février 2003. À paraître.
- [2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
- [3] Yann-Gaël Guéhéneuc. *Traçabilité de Motifs pour la Compréhension et la Qualité – Application aux Motifs et aux Défauts de Conception*. Thèse de doctorat, École des Mines de Nantes, mai 2003. À paraître.
- [4] Philippe Mulet. Eclipse – une plateforme universelle pour outils intégrés. Thomas Ledoux, éditeur, *actes de la Journée OCM*, pages 60–82, mars 2002.
- [5] Object Technology International, Inc. / IBM. Eclipse platform – A universal tool platform, July 2001.
Available at: www.eclipse.org.
- [6] Object Technology International, Inc. / IBM. Eclipse help system, November 2002.



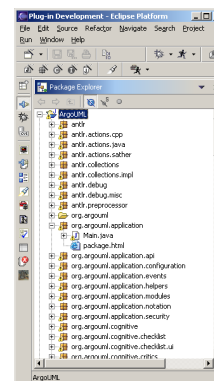
(a) La fenêtre *Workbench* vide.



(b) La perspective ressource.



(c) La vue hiérarchique des ressources.



(d) La vue Java des mêmes ressources.

FIG. 1 – Captures d'écran de ECLIPSE.