Programmez !
www.programmez.com
Mensuel - Mars 2008 - N°106
LE MAGAZINE DU DÉVELOPPEMENT

100% Open Source !

Développer 100 % en Open Source

Plongée au cœur des projets ouverts : KDE, Gnome, Ubuntu, PHP, Eclipse EMF...

Model Driven

Quand le modèle pilote le code

Linux
Découvrir KDE 4.0

Web
Le futur de Javascript

C++
Blitz++ : des calculs surpuissants !

Un labyrinthe en C++

SQL Server 2008
Les index spatiaux

Eclipse
Modélisation avec Process Framework Composer

Vista
Maîtriser l'API Windows Contacts

Printed in France - Imprimé en France - BELGIQUE 6,45 € - SUISSE 12 FS - LUXEMBOURG 6,45 € - Canada 8,95 $ CAN DOM Surf 6,90 € - TOM 940 XPF - MAROC 50 DH
M 04319 - 106 - F: 5,95 €

WINDEV | WEBDEV | WINDEV Mobile

Vous êtes invité !

TDF TECH 2008
Tour de France Technique
LSI Sponsor Officiel

Vous êtes invité !
11 villes du 27 Mars au 29 Avril 2008
100% technique

PLATEFORME PROFESSIONNELLE DE DÉVELOPPEMENT (AGL)

Et vous, connaissez-vous WINDEV 12 ?
Ici présentation de WINDEV à Paris, une des 12 villes du Tour de France WINDEV.

UN CODE MULTI-PLATEFORME:
Windows, .Net, Java, PHP, J2EE, XML, Internet, Ajax, Pocket PC, SmartPhone, Client riche ...

Des applications superbes sans compétence graphique grâce aux gabarits fournis. Ergonomie assurée par le correcteur d'IHM intégré.

Parmi les 500 nouveautés:

Accès natif à **SAP**

Fond de page PDF

Débogage à **distance**

Compilation «JITc»

Gestion des exigences

Fonctions d'administration Réseau **SNMP**

Réplication automatique

Sauvegarde à chaud

Héritage de modèle

Nouveau RAD

100 Nouvelles fonctions **Java**

39 Nouvelles fonctions **PHP**

50 Nouvelles fonctions **Linux**

Débogueur PHP

Web 2.0/Ajax

500 NOUVEAUTÉS

112 pages de témoignages prestigieux, à nous demander (gratuit)

Demandez le dossier technique gratuit (en couleurs, en français), accompagné de 112 pages de témoignages et d'un DVD. **Version Express Gratuite.**

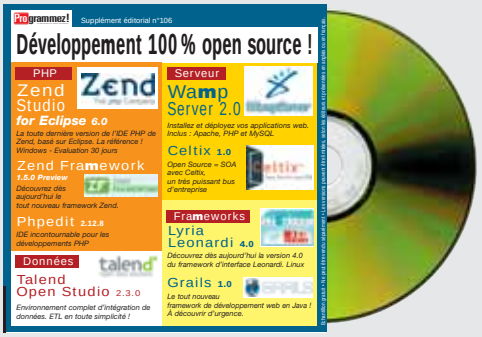
Tél Province **04.67.032.032** Tél Paris **01.48.01.48.88** **info@pcsoft.fr**

www.pcsoft.fr

Fournisseur Officiel de la Préparation Olympique

XII

SOMMAIRE N°106



CD-Rom 106 PROGRAMMEZ !

Développement 100 % Open Source !

PHP

Zend Studio for Eclipse 6.0
La toute dernière version de l'IDE PHP de Zend, basé sur Eclipse. La référence ! *Windows- Evaluation 30 jours*

Zend Framework 1.5.0 Preview
Découvrez dès aujourd'hui le tout nouveau framework Zend.

Phpedit 2.12.8
IDE incontournable pour les développements PHP

Données

Talend Open Studio 2.3.0
Environnement complet d'intégration de données. ETL en toute simplicité !

Serveur

WampServer 2.0
Installez et déployez vos applications web. Inclus : Apache, PHP et MySQL

Celtix 1.0
Open Source = SOA avec Celtix, un très puissant bus d'entreprise

Frameworks

Lyria Leonardi 4.0
Découvrez dès aujourd'hui la version 4.0 du framework d'interface Leonardi.
Linux

Grails 1.0
Le tout nouveau framework de développement web en Java ! À découvrir d'urgence.

_LE JOURNAL DE NOTRE INFRASTRUCTURE

_69e JOUR : Tout ce que nous voulons, c'est une information bien précise. Gilles la tenait presque, mais il a fait un faux mouvement. Comment trouver des informations commerciales fiables dans des montagnes de données contradictoires qui s'accumulent sans fin ?

_Gilles vient d'attraper un panda en peluche !

_71e JOUR : J'ai trouvé ! Grâce aux solutions IBM de gestion de l'information, nous rendons l'information plus claire et partageons une définition commune des données sources pour gagner en cohérence et en fiabilité. Je peux créer une seule et unique banque d'informations à partir des différents systèmes sources. Et ainsi tout le monde peut prendre de meilleures décisions.

_Ouf... On était à court de jetons.

IBM

Information Management

Téléchargez notre livre blanc sur la gestion des données de référence
IBM.COM/**TAKEBACKCONTROL**/ACCURATE/FR

L'info permanente

Programmez!
LE MAGAZINE DU DÉVELOPPEMENT

Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky
Rédacteur en Chef : François Tonic
Ont collaboré : F. Mazué, C. Pierre de Geyer, A. Goncalves, F. Brachfeld, X. Mehaut, D.Gageot, C. Fondacci, L. Laforets, S. Souane, J. Benois, J. Vidames, L. Goubet, C. Brun, S. Drapeau, M. Chauvin, J.M. Maman, J. Descombes, C. Robert, J.B. Boisseau, L. Guillois, R.S. Mompelat, V. Perdereau.
Dessin : Michel Piedoue, Jissey
Couverture : illustration Loïc Guillois
Maquette : AJE Conseils
Publicité : Régie publicitaire, K-Now sarl
Pour la publicité uniquement : Tél. : 01 41 77 16 03
coordination@programmez.com
Editeur : Go-02 sarl, 6 rue Bezout - 75014 Paris Coordination@programmez.com - Dépôt légal : à parution - Commission paritaire : 0707K78366 - ISSN : 1627-0908 - Imprimeur : ETC - 76198 Yvetot
Directeur de la publication : Jean-Claude Vaudecrane

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10 - abonnements.programmez@groupe-gli.com Tél. : 01 55 56 70 55 - Fax : 01 55 56 70 20 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.
Tarifs abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 45 € - Etudiant : 39 € - CEE et Suisse : 51,83 € Algérie, Maroc, Tunisie : 55,95 € Canada : 64,33 € Tom : 79,61 € - Dom : 62,84 € Autres pays : nous consulter.
PDF : 30 € (Monde Entier) souscription en ligne.

Édito

Et si on passait aux choses sérieuses ?

Depuis quelques années, éditeurs et autres spécialistes ès méthodes nous le martèlent : il faut passer de l'artisanat à l'ère de l'usine à logiciels !
Sur le papier, l'idée est séduisante. Il faut dire que notre industrie du logiciel reste paradoxalement artisanale. Certes, on automatise certaines tâches. On utilise de plus en plus de la modélisation, de la génération de code technique.
Bref, nous ne sommes plus dans l'artisanat, où le développeur sculptait avec amour son oeuvre autour d'une pizza, avec café et coca. Mais, nous ne sommes pas non plus entièrement dans l'usine à logiciels, car la démarche n'est pas menée à son terme. La fabrication demeure mi-artisanale, mi-industrielle.

Tout le défi des prochaines années est là. Comment peut-on arriver à mettre en place de véritables usines à logiciels, que ce soit en entreprise, dans une SSII ou bien chez tout développeur ? Il y a là une véritable révolution (je pèse mes mots) à faire. Il faut changer la méthode, la planification, la manière même de penser le code et l'application.
Comment aller vers cet objectif sans tout casser ? Le développement par modèle (le fameux Model Driven) nous aide à générer une partie du code. Les éditeurs disent " tout le code". Sans aller aussi loin, nous nous concentrons sur la génération du code technique. Ensuite, il faut intégrer, dès la conception, les tests, les cas de tests afin de valider très rapidement le code, les fonctions. Et si cela permet de détecter un bug d'architecture, l'architecte dira avec un plaisir réel (si, si) : "développeur, je t'aime". Je sais, on se croirait dans "Amour, gloire et code". Mais pour que tout cela tienne, il faut des outils totalement intégrés se greffant sur un process de méthodologie comme XP, RUP, etc. Il est très important de pouvoir mener un projet sur un modèle de méthode défini et mis en place.

Qui dit équipe, géographiquement distribuée ou non, dit communication. Et c'est là que le terme communication unifiée prend un sens développeur. Car cela permet de mieux gérer la communication entre les personnes en proposant tout un panel de techniques : messagerie instantanée, téléphonie IP, vidéo conférence, etc. Et surtout, on y ajoute une couche de collaboration et de partage. Il faut communiquer, échanger, expliquer, travailler ensemble.

On entend régulièrement dire que le modèle open source de développement est à prendre en exemple. Car là on a des dizaines de développeurs de par le monde, codant. Que l'on ne s'y trompe pas. Ce n'est pas improvisé. Il y a une infrastructure collaborative et surtout des outils de source pour collecter, tester, intégrer. Et là, la gouvernance du projet tient tout son rôle. Les "boss" du projet surveillent, aiguillent les développeurs, le code. Car, sans direction forte, encadrant fermement les développeurs de la communauté, le projet partirait rapidement à la dérive.
Mais, ne bridons pas trop notre ami le développeur !
Heureux, il est plus efficace.

François Tonic - ftonic@programmez.com

PROCHAIN NUMERO

N°107 - Avril - Parution : 29 mars 2008

Ruby

Découvrez toutes les facettes de Ruby et de Rails, les nouvelles références du Web 2.0

Toujours plus de JAVA !

En route vers Java 7 : toutes les nouveautés, la guerre Eclipse NetBeans...

PERSPICACITÉ

Souriez avec Jissey

www.programmez.com
chaque semaine, un gif animé

EN ARRIVANT, MARTIN VIT TOUT DE SUITE QUE SON CENDRIER AVAIT DISPARU !

JAVA

Événement : Paris a enfin son Java User Group !

Alors que le Club Java n'a plus d'activité majeure depuis plus de 3 ans, la communauté Java se désespérait d'avoir enfin son premier Java User Group français ! Oui vous avez lu, le premier JUG a été inauguré mardi 12 février ! Paris JUG remplit donc un vide indéniable pour les développeurs Java en France et particulièrement à Paris. Il est en effet étonnant que la France soit privée de JUG alors que l'Italie en possède une quinzaine. L'un des plus gros est brésilien avec 25 000 membres ! L'ambition de Paris JUG est d'organiser tous les mois, le 2e mardi, une session autour d'un thème. Cela pourra prendre la forme d'un " cours " magistral, d'une table ronde. Comme le rappelle Sun, chaque JUG est indépendant de l'éditeur, même si Sun soutient un minimum le User Group en fournissant conseils, contacts et goodies. Pour les premiers pas de Paris JUG, les fondateurs (Antonio Goncalves et Davis Dewalle) ont pu compter sur le soutien de plusieurs SSII. L'ambition est de développer Paris JUG et de grossir mois après mois, avec pourquoi pas, l'organisation d'une conférence plus importante… Pour le début, les ambitions sont modestes, réunir chaque mois au moins 30-35 personnes. L'adhésion à Paris JUG est gratuite.
La prochaine réunion aura pour thème : Selenium, un outil de tests fonctionnels Java. Date : mardi 11 mars, inscription sur le site.
Site : http://www.parisjug.org

PHP

Zend se met à Eclipse

Zend, leader des outils de développement PHP, vient de dévoiler la version 3.6 de Zend Plaform, un serveur d'application PHP. Elle apporte des améliorations notables, selon l'éditeur, sur le cluster, la disponibilité. *" La réactivité et la fiabilité des applications web en PHP dépendent de plusieurs éléments. Les développeurs PHP et les administrateurs web sont souvent confrontés à différentes technologies pour identifier et résoudre les goulets d'étranglement et les causes des erreurs, "* commente Mark de Visser, Directeur Marketing chez Zend Technologies. Zend Platform 3.6 comporte trois volets : PHP Intelligence pour la surveillance des applications (détection des erreurs, surveillance des événements Java, http, Apache…), gestion des performances (mise en cache du code pré-optimisé et le contenu), gestion des clusters. Sur Zend Studio for Eclipse, l'éditeur vient de le rendre disponible. L'outil se base sur le projet Eclipse PDT et est avant tout destiné aux développeurs professionnels voulant s'appuyer sur Eclipse. Il comprend un ensemble d'outils complets pour l'édition, le débogage, l'analyse, l'optimisation du code, et la gestion de bases de données. De même, il offre le support des méthodes de développement agile telles que le test unitaire, le refactoring, le code coverage et le profilage. Eclipse est une plate-forme multi-langage de référence, basée sur une architecture modulaire et soutenue par une communauté dynamique. Zend Studio for Eclipse est commercialisé au prix de 399 € H.T. par développeur. Zend propose un prix spécial de lancement de 299 € HT pendant une durée limitée. Les utilisateurs de Zend Studio Professional, qui possèdent un contrat de support en cours, recevront Zend Studio for Eclipse gratuitement.

OUTIL

Twiki en version 4.2

Twiki est un moteur de wiki réputé qui arrive maintenant en version 4.2. Les développeurs se sont concentrés sur l'éditeur visuel pour simplifier le travail. On bénéficie de la disponibilité d'une recherche programmable, d'une installation améliorée, de l'intégration d'une authentification externe (via LDAP comme exemple), et d'un meilleur support des plug-in API. Hormis cela, cette version est une release mineure de l'outil.
Site : http://TWiki.org

JAVA

NetBeans 6.0.1 est de sortie

NetBeans est maintenant disponible en v6.0.1, release de maintenance. Elle ajoute plusieurs nouveaux langages (japonais, brésilien, chinois simplifié), support de Glassfish v2 UR1, bug fix. L'équipe travaille activement à la version 6.1 qui doit sortir fin avril, la version 7.0 est toujours prévue pour fin novembre prochain !

SGBD

PostgreSQL 8.3 est disponible !

Début février, la communauté PostgreSQL était heureuse d'annoncer la sortie de la version 8.3 du SGBD. Et la liste des nouveautés est assez impressionnante, on notera notamment : support SQL/XML, support de Visual C++, intégration de Tsearch, support de SSPPI et GSSAPI (pour la sécurité. La recherche textuelle est maintenant intégrée ainsi que la gestion des ENUM. On trouve aussi :
- le " Heap Only Tuples " (HOT) ou " Tuples en mémoire seule " élimine jusqu'à 75% du temps de maintenance des tables fréquemment modifiées ;
- l'étalement des points de vérification (checkpoints) réduit l'impact des points de vérification sur les temps de réponse ;
- l'auto-optimisation du démon d'écriture sur disque (background writer) ;

l'option de validation asynchrone raccourcit considérablement les temps de réponses de quelques transactions.
Site : http://www.postgresql.org

And the winner is...

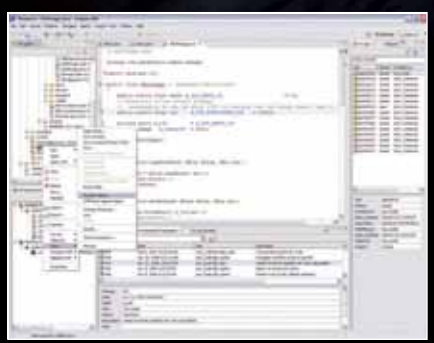
Le distributeur international ComponentSource vient d'annoncer la liste de ses best-sellers 2007. Les " Awards 2007" reflètent, indique-t-il, ses ventes de l'année. La liste des 100 produits est en ligne sur programmez.com. Ci-dessous les 10 premiers :

1 NetAdvantage for .NET - Infragistics
2.ActiveReports for .NET - Data Dynamics
3.DXperience - Developer Express
4.ComponentOne Studio Enterprise - ComponentOne
5.InstallShield by Macrovision (formerly InstallShield
6.Janus WinForms Controls Suite - Janus Systems
7.RadControls for ASP.NET - Telerik
8.Spread for Windows Forms - FarPoint Technologies
9.Altova® XMLSpy - Altova
10.TX Text Control .NET - The Imaging Source

Perforce | Le système rapide de gestion de configuration logicielle

Présentation du module d'extension de Perforce pour Eclipse

Pour travailler avec Perforce dans une interface IDE Eclipse.



Module d'extension de Perforce pour Eclipse

Le module d'extension de Perforce pour Eclipse permet aux développeurs d'accéder facilement au système de GCL Perforce depuis leur interface IDE Eclipse. Il propose les fonctionnalités suivantes :

- Accès rapide à l'historique complet des fichiers
- Prise en charge complète du développement collaboratif, avec possibilité de fusionner les fichiers
- Possibilité de travailler hors ligne lorsque la connexion avec le serveur Perforce est indisponible
- Outil de comparaison des fichiers et prise en charge du suivi des défauts intégrés
- Prise en charge de la fonction de refactoring de l'environnement Eclipse

Le module d'extension de Perforce pour Eclipse prend en charge les systèmes d'exploitation Windows et Linux. Et ce n'est que l'un des nombreux outils intégrés dans le système de GCL Perforce.

eclipse FOUNDATION MEMBER

PERFORCE SOFTWARE

Téléchargez sans conditions une copie gratuite de Perforce sur www.perforce.com. Vous bénéficiez d'une assistance technique gratuite pendant votre période d'évaluation.

SYSTÈME

Google : 90% d'audience en janvier

Selon l'étude mensuelle Xiti Monitor sur les parts de marché des moteurs de recherche auprès des utilisateurs du web, en janvier, Google se maintient au-dessus des 90% de part de visites (90.03%), mais perd 0,8% par rapport à décembre 2007. Il distance de très loin le N°2, Yahoo! (2.88%). Ensuite viennent Live Search (2.49%) et Orange (1.56%). Fait notable, AOL rejoint le top 5 des moteurs de recherche de janvier 2008 et détrône ainsi Free de sa 5ème place, avec une part de visites de 1.54%.

SYSTÈME

Un système open source écrit en C# !

Voilà un projet open source très intéressant : Cosmos. Il s'agit d'un système d'exploitation écrit en C#, bootant donc en .Net. Il fonctionne pour le moment sur PC mais le iPhone est une des cibles futures… Cosmos est donc du code managé (dans le sens .Net du terme), incluant la pile .Net nécessaire. Si le projet utilise C#, tout autre langage .Net est possible. La première pré-version est disponible. Bien sûr, tout cela n'est pas encore " friendly " mais le projet espère packager plus simplement Cosmos dans les prochains mois. Actuellement, il fonctionne en virtualisation. Il utilise un micro noyau, des modules de configuration et est suffisamment souple pour être multi-plate-forme… Il est sous licence BSD. Un projet à suivre de près ! Site : http://www.gocosmos.org

WEB

Grails : votre nouveau framework web

La version 1.0 de Grails est maintenant disponible. Il s'agit d'un framework de développement web en Java et Groovy en licence Apache. Il ressemble fortement à Rails mais si Rails est en Ruby, Grails permet de coder en Java… Ce n'est pas un hasard s'il utilise d'ailleurs des librairies comme Spring ou Hibernate. Grails propose dans un package unique tous les aspects d'une application web : interface, données. L'architecture de base de Grails est l'idée de convention par configuration. Cela doit permettre de réduire le code purement technique au profit d'un code plus métier, plus fonctionnel. Des plug-in NetBeans et Eclipse existent mais attention à leur qualité… Grails n'en est qu'à ses débuts, il faudra voir comment évoluera le framework et son support par les éditeurs et les outils. Pour un développeur Java, cela peut être intéressant, mais il rentre tout de même en concurrence avec les autres solutions comme Rails. Alors faut-il choisir Grails et non Jruby ? À vous de voir !

WEB

IDE de nouvelle génération ?

Connaissez-vous Global System Builder ? Il s'agit d'un IDE léger. Il permet de coder son application et surtout de la déployer rapidement sur un système. Il faut d'abord localiser le serveur cible sur lequel on veut déployer son application, puis activer une connexion distante avec le serveur cible. Téléchargez ensuite une application WinForm (déploiement par ClickOnce), puis lancez l'application. On possède alors une console et un navigateur de classe. Vous pouvez commencer la programmation pour le serveur cible, dans un éditeur de code accessible. Attention : il se limite aux langages dynamiques. Site : http://globalsystembuilder.com/

ARCHITECTURE

JBoss lance sa SOA intégrée

Red Hat annonce la disponibilité mondiale de JBoss Enterprise SOA Platform, afin d'aider les entreprises à accélérer leur migration vers la SOA, grâce à une offre exhaustive d'intégration d'applications, de services et de processus. Commercialisée sous la forme d'une simple distribution d'entreprise, cette solution regroupe toutes les fonctionnalités d'intégration de SOA, d'applications et de processus métier. L'objectif est d'aider les entreprises à gagner en performance de manière plus simple, libre et rentable qu'avec des plates-formes propriétaires. A l'ensemble des briques JBoss déjà existantes, vient s'ajouter l'ESB JBoss, un avantage de poids qui va lui permettre de s'imposer sur un marché très concurrentiel. JBoss Enterprise SOA Platform offre davantage de flexibilité de déploiement, s'adaptant aussi bien aux projets d'intégration de SOA à petite échelle qu'aux projets plus étendus. Digne des meilleurs projets de développement Open Source, comme JBoss ESB, JBoss jBPM et JBoss Rules, JBoss Enterprise SOA Platform est une solution légère, simple à installer et économique à exploiter.

SOLUTION

Sun se renforce dans la virtualisation

Alors que l'on attend la sortie de Hyper-V de Microsoft dans les 180 jours, Sun ne veut pas laisser la concurrence trop avancer. Après avoir dévoilé son approche xVM, le constructeur met la main sur Innotek, éditeur de VirtualBox, environnement en open source !

COLLABORATION

Novell rachète SiteScape

Le monde de la collaboration évolue lui aussi rapidement. Alors que l'on attend avec impatience Jazz d'IBM, Novell rachète SiteScape, environnement collaboratif en open source. Il permet de créer des espaces de travail temps réel. Ainsi, Novell va pouvoir mettre en place une solution de communication unifiée, au moins sur la partie collaborative, partage de documents.

Agenda

MARS

Paris la Défense, Du 12 mars au 13 mars 2008. **DOCUMATION,** 14e rendez-vous de la gestion de contenu et du document
http://www.documation.fr/

Le 20 Mars 2008, Paris 8e, Elysée Biarritz, **VUME 2008,** évènement interactif sur le futur des Applications Internet Riches (RIA) et des outils de Visualisation. www.ilog.fr/vume

La quatrième **Conférence FileMaker** francophone se tiendra au Grand Hôtel Aston à Nice du 28 au 30 mars 2008 http://www.fmconf.com:80/

AVRIL

1er, 2 et 3 Avril 2008, Paris Expo Porte de Versailles Hall 8 **RTS EMBEDDED,** 16e salon des solutions informatiques temps réel et des systèmes embarqués
www.ni.com/embedded/fr/

Les 15 et 16 Avril 2008, CNIT Paris La Défense. **COIP/VOIP 2008.** Toutes les communications sur IP !
http://www.salon-coip.com/

Les 15 et 16 Avril 2008 au CNIT Paris La Défense.**5èmes Rencontres du Management de Projet,** pour faire le point sur les outils et méthodes. www.groupesolutions.fr/

ETRANGER

Du 04 Mars 2008 au 09 Mars 2008 **CeBIT 2008,** le plus grand salon mondial dédié à l'électronique grand public. Hanovre, Allemagne, Hannover exhibition Center
http://www.cebit.de/homepage_e

Du 19 Mars 2008 au 20 Mars 2008 **Infosecurity Belgium 2008.** Bruxelles (Belgique), Brussels Kart Expo
http://www.infosecurity.be/sites/www_infosecurity_be/en/index.asp

FRAMEWORK

Leonardi en open source !

L'éditeur français a dévoilé la version 4.0 de son environnement de génération d'interface homme-machine, Leonardi. Il s'agit d'un framework piloté par métier. Surtout, il est disponible en open source. C'est une phase importante dans l'évolution du framework et pour la société. Ce choix vise, selon l'éditeur, à accélérer la diffusion du framework, et surtout à constituer une communauté autour de Leonardi. Il est disponible sous licence GPL et le modèle économique est celui de la double licence. Les évolutions incluses dans la v4 sont nombreuses et importantes : support de Java 5, introduction d'un dashboard avec onglets, nouvelle interface pour les saisies et la gestion des filtres complexes. Au-delà du framework, l'éditeur a aussi travaillé sur le studio de conception : Leonardi Studio.

Là aussi, beaucoup de nouveautés, on retiendra en particulier : un éditeur graphique de formulaire, un module de metadata, un générateur de schémas de bases de données, un diagramme de classes, etc.

Site : www.lyria.com

MODÈLE

Softfluent part à l'offensive

Durant les TechDays 2008, l'éditeur français Softfluent a lancé une grande promotion autour du Model Driven et de son outil CodeFluent. Trois versions sont désormais disponibles : *Expression, Developer* et *Architecte*. Les versions comprennent l'ensemble des fonctionnalités, la différence venant de la complexité des applications générées qu'on peut réaliser. L'édition Express est gratuite et vise les étudiants et développeurs " amateurs ". L'édition Développeur vise plutôt le développeur indépendant. L'éditeur souhaite aussi développer une communauté : CodeFluent Expert Network. D'autre part, CodeFluent fait son entrée dans le centre technique de Microsoft à Paris : MTC. Permettant ainsi aux clients de tester les solutions maison !

MODÉLISATION

Acceleo 2.2.0 : génération JEE, Php et Python

Acceleo, qui est un générateur de code Open Source basé sur les technologies Eclipse et EMF, est sorti en version 2.2 courant janvier. Ce projet qui a maintenant bientôt deux ans, permet de transformer des modèles UML ou DSL vers du code, selon les principes de l'approche MDA. Cette version apporte encore un lot de nouvelles fonctionnalités. L'éditeur de template a été refondu pour améliorer la lisibilité des templates et faciliter leur saisie, l'interface à désormais été totalement localisée et de nouvelles fonctionnalités de fond ont été ajoutées telles que les scripts génériques ou encore la gestion de l'encoding des templates. On peut citer l'apparition de Spring dans le module JEE accompagné d'une nouvelle documentation de mise en oeuvre, une meilleure gestion des héritages dans le module PHP, diverses améliorations dans le module Python ainsi que la première livraison, expérimentale, de WISSS : outillage dédié à la modélisation et génération d'application web PHP. Côté communautaire, l'activité est soutenue sur les différents canaux de communication, en particulier sur la planète Acceleo et sur le forum. Acceleo était également présent aux dernières conférences Eclipse, "Eclipse Summit Europe" en Allemagne et "Eclipse Now You Can" à Paris par exemple.

Site : http://www.acceleo.org/pages/nouveautes-d-acceleo-2-2-0

SonarJ et Maven surveillent votre architecture Java

SonarJ permet à un architecte de décrire à un haut niveau d'abstraction l'architecture logique de son projet Java, et d'indiquer instantanément à son équipe de développement l'introduction d'une violation d'architecture dans le code. Cette faculté d'informer le développeur en temps réel est le grand point fort de l'outil : aucun autre outil ne peut le faire. Le logiciel, distribué en France par pcMetric, calcule aussi toute une série de métriques (complexité d'une méthode, degré de couplage entre classes, dépendance entre packages...) et vous prévient aussitôt lorsque le seuil que vous autorisez est dépassé. Grâce à son plug-in Maven, SonarJ peut intégrer ces informations directement dans le rapport Maven, à chaque compilation du projet. Pour exemple, prenons l'application Java Petstore tirée du livre " Java EE 5 " aux éditions Eyrolles. Cette application Internet/Intranet, développée en Java EE 5, est découpée en plusieurs couches. Son architecture logique peut être décrite de manière graphique grâce à SonarJ Architect : Dans ce diagramme, les couches sont représentées comme des éléments logiques enrichis de stéréotypes. Par exemple :

- <<public>> : la couche est accessible par toutes les autres
- <<bottom>> : la couche n'accède à aucune autre couche (sauf les <<public>>)

Les flèches indiquent le sens des dépendances autorisées (ex. la couche cliente peut accéder à la couche service et non l'inverse). Après cette étape, il faut faire correspondre la définition logique à la réalité physique : vous affectez avec SonarJ Architect les packages Java de votre application aux différentes couches (ex. com.yaps.petstore.swing.** pour la couche Swing). L'outil se chargera alors de contrôler qu'il n'y a pas de violations d'architecture.

Mise en œuvre du plug-in Maven

Avec le plug-in Maven de SonarJ, vous automatisez cette vérification et vous générez un rapport que vous intégrez à votre cycle de build Maven. Il vous suffit de déclarer le plug-in maven-sonarj-plugin dans le pom.xml de votre projet, comme ceci :

```
<project>
   (...)
   <build>
      <plugins>
         <plugin>
            <groupId>com.hello2morrow.sonarj</groupId>
            <artifactId>maven-sonarj-plugin</artifactId>
            <version>3.2.0-SNAPSHOT</version>
            <executions>
               <execution>
                  <goals>
                     <goal>sonarj</goal>
                  </goals>
               </execution>
            </executions>
         </plugin>
      </plugins>
   </build>

   <reporting>
      <plugins>
         <plugin>
            <groupId>com.hello2morrow.sonarj</groupId>
            <artifactId>maven-sonarj-plugin</artifactId>
            <version>3.2.0-SNAPSHOT</version>
            <configuration>
               <reportDir>target/sonarj</reportDir>
               <reportName>sonarj-report</reportName>
            </configuration>
         </plugin>
      </plugins>
   </reporting>

   <pluginRepositories>
      <pluginRepository>
         <id>hello2morrow repository</id>
         <url>http://www.hello2morrow.de/maven/repository</url>
      </pluginRepository>
   </pluginRepositories>

<project>
```

Une fois votre pom.xml modifié, deux possibilités s'offrent à vous pour générer un rapport : soit vous utilisez le goal défini par SonarJ (mvn sonarj:sonarj), soit vous laissez le cycle de vie Maven s'opérer. En effet, ce plug-in est rattaché à la phase verify du cycle de build Maven. Vous pouvez donc continuer à utiliser *mvn verify* ou, plus communément, *mvn install.* Pour que ce rapport soit rajouté au site Maven, il faut définir le plug-in dans la section reporting du pom.xml. Ainsi, lorsque vous exécutez la commande *mvn site,* un lien SonarJ est automatiquement rajouté au menu et vous pouvez alors consulter le rapport généré. Maintenant que votre architecture est définie et que le plug-in est intégré au pom.xml de votre application, essayez d'introduire une dépendance non autorisée dans votre code. Votre architecture logique vous informe que le client Swing et le client Web peuvent invoquer la couche service. Par contre, le fait qu'il n'y ait pas de flèche entre Swing et Web indique à SonarJ que cette dépendance n'est pas autorisée. Laissons donc notre développeur appeler une classe de la couche client Swing à partir de la couche Web. Exécutez *mvn install,* SonarJ détecte alors une violation d'architecture et la signale par le biais du rapport. Si vous voulez que ce rapport s'intègre au site Maven, exécutez *mvn install,* (notez le message " Architecture violations ") :

SonarJ est un outil efficace pour contrôler l'intégrité de votre architecture de manière très simple. Avec son plug-in Maven, vous pouvez l'intégrer de manière transparente à votre cycle de livraison et être informé du bon respect de votre architecture via un rapport.

Antonio Goncalves
Architecte logiciel
http://www.antoniogoncalves.org

Frédéric Brachfeld
pcMetric
http://www.pcmetric.com

NetAdvantage® for .NET

NetAdvantage for ASP.NET

Des composants haute performance pour interfaces utilisateur. Une expérience supérieure pour vos utilisateurs.

Des composants haute performance pour interfaces utilisateur
Recevez une boite à outils complète pour multi-plateformes, en influençant même les applications les plus exigeantes incluant des grilles riches et de haute performance pour Windows Forms et ASP.NET

Compatibilité Visual Studio®
Utilisez la version de Visual Studio qui vous convient - VS 2005 ou VS 2008 avec un support complet pour NetAdvantage

Créez des tableaux de bord professionnels
Présentez des graphiques et gauges haute fidélité pour ASP.NET et Windows Forms afin d'obtenir une présentation limpide des Key Performance Indicators

Design Once, Style Everywhere
Donnez instantanément un style professionnel à vos applications créées avec NetAdvantage pour imiter Office® 2007, Vista® ou bien créez et appliquez vos normes corporatives à travers les contrôles et vos applications

Accélérez vos applications web
Des composants rapides et légers, basés sur notre nouveau framework AJAX Aikido™, sont essentiels pour des applications agiles pour Web 2.0 et SharePoint® Web Parts

Donnez du pouvoir à vos utilisateurs
Délivrez des interfaces utilisateur riches et productives à vos clients, pour ASP.NET et Windows Forms, également disponible pour WPF ou pour JSF

Pour de plus amples informations: infragistics.com/dotnet

Infragistics®
Powering The Presentation Layer

sales-europe@infragistics.com

N° Vert 0800 667 307

Infragistics Sales 800 231 8588
Infragistics Europe Sales +44 (0) 800 298 9055

WINDOWS FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors & more

TechDays 2008 : C'est parti pour Windows Server 2008 !

Windows Server 2008 est disponible. La France a découvert cette version dès début février durant les TechDays 2008, avec les versions françaises de Visual Studio 2008. Plus de 15 000 inscrits, plus de 11 000 personnes présentes, 250 sessions, 3 jours, 100 exposants, l'événement Microsoft est plus que jamais la conférence informatique professionnelle la plus importante de France et sans doute d'Europe, surclassant les TechEd Europe !

Durant ces 3 jours, les visiteurs pouvaient voir et utiliser les dernières versions des outils et technologies Microsoft et des partenaires. Deux grandes tendances étaient présentes sur les stands éditeurs : la virtualisation et la communication unifiée. Le marché Windows connaît un nouveau souffle avec l'arrivée, dans les 180 jours, de Hyper-V, l'hyperviseur de Microsoft. Tout le monde semble se positionner avant la sortie officielle. Autre axe très fort, la communication unifiée que Microsoft pousse fortement depuis la sortie de son serveur de communication. La démonstration durant la dernière keynote avec l'intégration du Speech Server, de Windows Server 2008, de Communication Server fut des plus impressionnantes sur l'interaction dynamique des différents environnements serveurs.

Virtualisation, parallélisation, interopérabilité

La virtualisation constitue aujourd'hui pour Microsoft un terrain à conquérir même s'il arrive bien après Xen ou VMware. Et cela concerne le poste de travail (l'éditeur estime à 1 % la virtualisation dessus), le serveur et maintenant l'application avec Softgrid. Dans la stratégie de l'éditeur, la virtualisation s'intègre dans une fonctionnalité bien plus large : le datacenter dynamique. Il s'agit de pouvoir allouer dynamiquement de nouvelles ressources, d'instancier de nouveaux serveurs (données, applications, etc.) selon la charge. Tout cela doit pouvoir se faire via un modèle de fonctionnement. La modélisation est au cœur de la vision de Microsoft pour le futur, et notamment avec le DSL… D'autre part, Windows doit devenir totalement modulaire pour pouvoir adapter Windows à ses besoins propres, c'est le projet MinWin. La programmation multicore a été longuement abordée dans les sessions et les keynotes. L'enjeu est important, car il s'agit de simplifier la parallélisation du code pour le développeur. Autre tendance visible sur le salon, l'interopérabilité. Terme magique ! Ce n'est pas un hasard si Sun, IBM, Talend, Zend, Novell étaient présents sur l'événement dans les sessions et les stands. Cela montre comment Microsoft souhaite profiter de l'open source sur la plate-forme Windows. La session sur l'interopérabilité des services Web Sun – Microsoft a été particulièrement suivie.

.Net 3.5 : LINQ à l'honneur

Linq a été largement abordé durant les 3 jours, ainsi que le projet Astoria. SQL Server 2008, qui ne doit pas arriver avant septembre prochain, a eu droit à plusieurs sessions techniques pour mieux connaître les nouveautés. Le développement et le design Web étaient aussi à l'honneur. Une trentaine de sessions permettaient d'aborder PHP, Silverlight, la gamme Expression, Cardspace, etc. La partie Live a eu aussi droit à plusieurs présentations, une des plus réussies fut celle sur Virtual Earth du premier jour. Cela a permis de comprendre, en profondeur, comment interagissaient Map, Virtual Earth et les données. Plusieurs stands proposaient d'ailleurs des solutions autour de Virtual Earth.

Mais les TechDays étaient aussi le lieu pour découvrir plus de 600 offres d'emplois, qui ont connu un immense succès ! Il faut dire que le monde .net manque de compétences et les développeurs .Net ne courent pas les rues… Microsoft a d'ailleurs rappelé aux jeunes étudiants que c'était le bon moment pour s'y mettre. C'était aussi une occasion unique pour rencontrer les grandes communautés .Net / Windows / SQL Server… ainsi que les nombreux groupes utilisateurs.

Et le futur alors ?

Hormis le Datacenter dynamique, l'une des rares excursions dans le futur s'est concentrée sur l'interface et notamment sur l'avenir de la souris qui vit ses dernières années. Le projet Surface et surtout l'interface dite multi-touch ont été longuement présentés et expliqués. Les laboratoires Microsoft travaillent activement sur cette nouvelle génération d'interface totalement interactive, manipulable avec le doigt, la main. Mais malheureusement, hormis l'actuelle version de Surface, aucune application concrète n'a été montrée, aucun système à la iPhone ne sera disponible sous Windows avant des mois.

Plusieurs centaines de webcast seront visibles sur le site afin de rejouer les sessions que vous auriez ratées.

Rendez-vous en 2009 !

François Tonic

LEONARDI

L'ihm

EN TOUTE SIMPLICITÉ !

ATELIER "QUICK START" LEONARDI VEN. 28 MARS 300€ HT REMBOURSABLES.

Votre application de gestion multilingue avec plus de 100 vues de 20 types différents, en DHTML/Ajax, en Swing ou en plugin Eclipse, connectée à un SGBD et un bus JMS.

« il vous faut combien de temps pour la réaliser ? »

Si votre réponse est moins d'une semaine, inutile de vous rendre sur notre site, ni de télécharger la version gratuite de LEONARDI, sinon il est temps de passer à la vitesse "*Model-Driven*"...

NOUVELLE VERSION!

LEONARDI V4.0 OPEN SOURCE

LYRIA

Pour en savoir plus sur la solution Leonardi, rendez-vous sur notre site www.lyria.com ou envoyez-nous un courriel à info@lyria.com

Eclipse Process Framework Composer

EPFC, outil développé dans le cadre du sous-projet officiel Eclipse Process Framework (EPF), est une application riche construite sur le framework Eclipse et utilisant abondamment les plug-in de haut niveau de l'écosystème.

Le but d'EPFC est d'offrir un outil de modélisation de processus complexes aux méthodologues, aux chefs de projets, aux directeurs de projet qui ont en charge la maintenance et l'implémentation d'un processus quelconque. Accessoirement, EPFC est utilisé au sein de l'outil Jazz pour modéliser les processus utilisés par l'application.

De quoi s'agit-il ?

EPFC est un outil permettant à divers intervenants de décrire un processus, de quelque nature qu'il soit, même si l'intention première est de fournir un outil de support méthodologique à des approches telles que RUP, OpenUP, Scrum, XP ou encore EclipseWay. Les objectifs du projet EPF (la base de l'EPFC) sont :

1. Servir de base à la création d'un écosystème vivant autour des problématiques de description de processus complexes.

2. Fournir l'outillage, un méta-modèle unifié de description de processus, ainsi qu'un contenu de base qui puisse être utilisé pour la modélisation d'une vaste famille de processus principalement, mais pas exclusivement, informatiques.

3. Utiliser la communauté Eclipse comme un vecteur d'acceptation de ce framework.

Eclipse EPF permet la description de processus de manière claire et structurée. Il permet notamment :

1. Aux Méthodologues de décrire une méthode de base et des méthodes personnalisées pour un besoin, une division, ou un client particulier.

2. Au Management d'avoir une vue cohérente et réaliste des processus internes de la société et de pouvoir améliorer ces processus et donc le ROI(1) de l'entreprise.

3. Aux Editeurs de logiciels de vendre autour/au dessus d'EPF du service et des produits, ainsi que de construire des outils.

4. Aux Universitaires d'enseigner le développement de processus, d'utiliser EPF comme support pour des projets étudiants et d'effectuer des recherches sur les processus méthodologiques.

5. Aux Fournisseurs de services de proposer des offres de formation, de conseil, de monitorat et de tutorat.

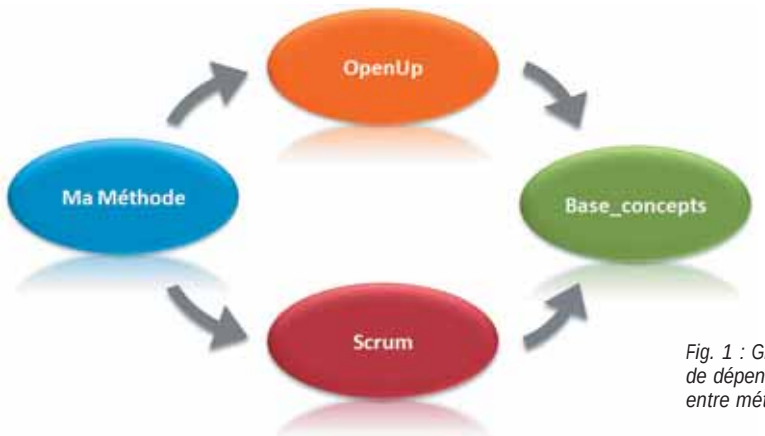


Fig. 1 : Graphe de dépendances entre méthodes

Approche générale

EPFC implémente un méta-modèle de définition de processus de développements qui se base sur le méta-modèle de l'OMG(2) SPEM2 (Process Engineering Meta-model). Il en reprend le vocabulaire et la terminologie. Pour rappel, EPFC permet de définir de nouveaux processus, d'étendre ou de modifier des méthodes existantes. Une méthode est appelée un plug-in ou encore une library. Ce terme de library est intéressant car il indique qu'une méthode peut faire référence à d'autres bibliothèques ou parties de bibliothèques pour sa définition. En cela on se rapproche d'un développement logiciel où une application est composée d'une ou plusieurs bibliothèques de code, l'application elle-même pouvant être réutilisée, le cas échéant dans d'autres applications (Fig.1).

EPFC définit une bibliothèque par défaut, nommée Base_concepts, proposant des conseils, des termes et vocabulaire pouvant être réutilisés dans les méthodes que vous définirez.

EPFC distingue deux aspects dans la description d'une library :

4. L'aspect définition des concepts manipulés que l'on appelle Method Content (contenu de la méthode)

5. L'aspect enchaînement de ces concepts que l'on appelle Process (processus) (Fig.2).

Method content / Eléments de la méthode

EPFC propose un canevas permettant de décrire toutes les entités manipulées dans une méthode.

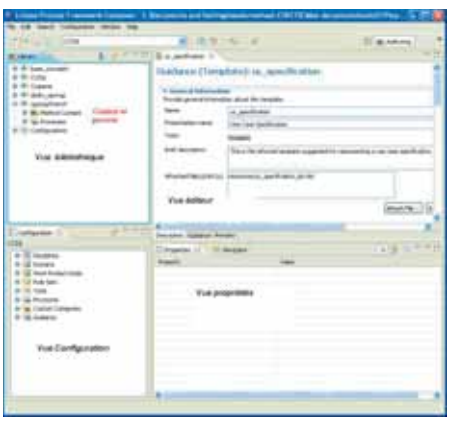


Fig. 2 : Application Eclipse EPFC

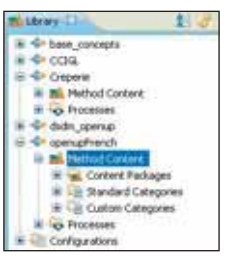


Fig. 3 : Sous-sections proposées pour la partie Method Content

Cette section est décomposée en trois sous-sections que nous décrivons ci-après :

6. Content packages (paquetages de contenu)

7. Standard categories (catégories standard)

8. Custom categories (catégories personnalisées)

Content packages

L'outil offre tout d'abord d'organiser dans des packages les éléments de la méthode, tout comme on organiserait des classes dans des packages et sous-packages Java. La section concernée est appelée Content packages. Il s'agit d'un moyen de hiérarchiser l'information pour mieux la retrouver. Ceci est important, en particulier quand on construit une méthode à partir d'éléments d'autres méthodes.

(1) ROI, Return of Investment, retour sur investissement
(2) www.omg.org

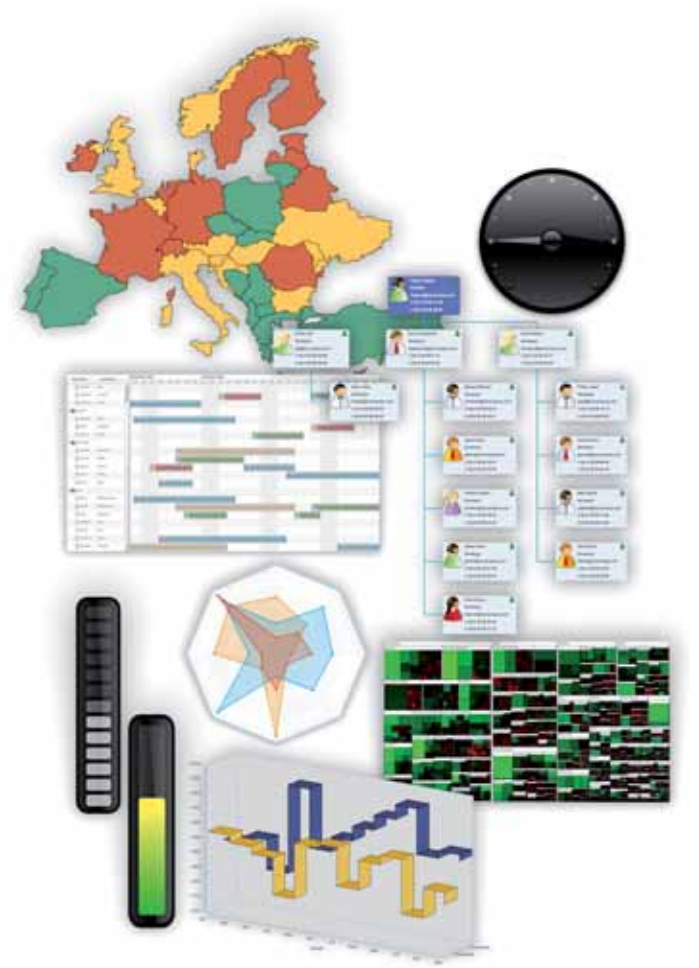
VUME 2008

ILOG
Changing the rules of business™

Elysée Biarritz – 75008 Paris
Jeudi 20 mars 2008

Participez à VUME 2008 et découvrez le futur des Applications Internet Riches (AIR)

- Table ronde sur les AIR
- Présentation des solutions de Visualisation ILOG
- Sessions techniques
- Rencontres et échanges avec les communautés Business et Techniques



Davantage d'informations :
www.ilog.fr/vume/

Adobe

Microsoft

Chaque package est décomposé en quatre sections à remplir ou non :

9. roles (rôles) : tous les intervenants du processus, circonscrits au package

10. tasks (tâches) : toutes les actions possibles du package

11. work products (éléments utilisés) : tout ce qui peut être utilisé ou produit et qui est relatif au package. On distingue les artifacts (artefacts), les outcome (produits) et les deliverable (livrable)

12. guidance (conseils) : la liste de conseils relatifs au package

Il est à noter qu'un élément ne peut se trouver que dans un package à la fois ! Un work product de type *artefact* est un élément tangible qui est consommé, produit ou modifié par une tâche. Il peut être composé d'autres artefacts. Un work product de type *outcome* est un élément intangible qui peut être un résultat ou un état. Les outcome peuvent également être utilisés pour décrire un autre work products qui n'est pas formellement défini. Un *deliverable* est un ensemble de work product, habituellement des artefacts. Il est utilisé pour prédéfinir un contenu typique ou recommandé sous la forme de work products packagés pour livraison. Les deliverable sont utilisés pour présenter à un client, utilisateur ou référent métier une sortie d'un processus qui a une valeur, matérielle ou non.

Standard categories

La section Standard categories permet d'ordonner le contenu de la méthode sous un autre angle, c.a.d. celui de la méthode, du processus. Pour faire simple, l'organisation proposée ici est déjà orientée méthode, ce qui n'était pas le cas dans la phase précédente.

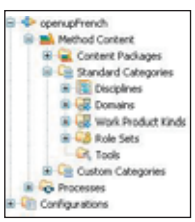


Fig. 4 : Sous-sections de la partie Standard Categories

Les informations sont regroupées en cinq catégories standard, à savoir :

13. Disciplines : il s'agit d'un ensemble de tâches qui peuvent être regroupées afin de réaliser une phase dans le processus. Par exemple, dans un cycle de vie en V, les disciplines sont les suivantes : Spécification, Analyse, Conception, Codage, Test de vérification et Test de Validation. Dans cette section, il n'est pas encore question d'ordonner les phases entre elles ! Ceci sera fait dans la perspective Process.

14. Domains : il s'agit du pendant des disciplines pour les Work products et non plus les tâches. Un domaine est une hiérarchie logique, raffinable de Work products regroupés entre eux sur des critères temporels (timing), de ressources ou de relations entre eux. Tandis qu'un domaine regroupe plusieurs work products, un work product n'appartient qu'à un seul domaine. Les domaines peuvent être par ailleurs décomposés en sous-domaines. En pratique, on retrouve souvent la même hiérarchie que celle que l'on aura définie dans la section disciplines.

15. Work Product Kinds : il s'agit d'une autre façon de regrouper les work products entre eux. Cette fois-ci un work product peut se retrouver dans plusieurs Work Product Kinds.

16. Role Sets : sert à regrouper des rôles ayant un certain nombre d'*acquaintances* entre eux.

17. Tools : peut fournir une description générale des outils et de leurs caractéristiques.

Custom categories

En pratique, la raison d'être de cette section est d'organiser les informations en vue de la génération du site WEB. C'est généralement ces informations et la façon dont elles sont organisées qui serviront dans la navigation du site intranet généré. Les deux copies d'écran ci-dessous vous montrent comment une description dans EPFC peut se retrouver dans le site WEB généré (Fig. 5 et 6).

Fig. 5 : Custom categories dans EPFC pour la méthode OpenUp

Il est à noter que c'est lors de la configuration de la méthode que l'on choisit les catégories à afficher dans le site web.

Process / Processus méthodologique

Une fois le contenu de la méthode décrit, il faut organiser ces éléments entre eux pour constituer un processus qui se tient. L'utilisateur doit définir, dans la terminologie de l'outil, des Capability patterns ainsi que des Delivery Processes.

Delivery processes

Un Delivery Process est le processus qui couvre l'ensemble du cycle de vie de développement du début à la fin. Il peut être utilisé comme un template pour la planification et l'exécution du projet. Il fournit un modèle de

Fig. 6 : Mêmes catégories dans le site web généré

cycle de vie avec ses phases, ses itérations et ses activités. Ce processus est construit à partir des groupes d'activités définis dans la section Capability patterns, séparés généralement par des jalons (milestones), c'est-à-dire des évènements déterminant des fins ou des débuts de phases (Fig.7).

Fig. 7 : Phases constituant le processus OpenUp

Il est possible de définir plusieurs Delivery processes, et ce sera la partie configuration qui décidera quel processus sera généré et comment.

Capability patterns

Un Capability pattern est un groupe d'activités qui partagent une même problématique. On décrit dans cette section un ensemble de phases de développement sans décrire les relations entre ces phases, un Delivery process s'occupant d'établir ce lien. On peut considérer que les Capability patterns sont les use cases de haut niveau servant à décrire la méthode, et donc décrits avec des verbes, contrairement aux catégories qui sont décrites par des noms. Tous les groupes d'activités identifiés ne sont pas forcément utilisés dans le même Delivery process, mais peuvent être propres à un processus ou partagés entre plusieurs cycles de vie. C'est la principale utilité des Capability patterns que de pouvoir mutualiser l'information. Nous reviendrons plus en détail sur EPF dans un prochain article et notamment sur l'implémentation OpenUp.

Xavier Méhaut
Consultant senior Edis-Consulting

Extrême Java

Seam

UML

Hibernate

Gestion de projet

Scrum

XML

.Net

valtech training

Au plus court vers vos nouvelles compétences

90 formations au développement logiciel

chez vous
ou à Paris, Toulouse, Lyon, Grenoble, Genève, Bruxelles, Luxembourg

Architecture et intégration

- Introduction au logiciel libre (1 jour)
- La persistance dans les applications Java (1 jour)
- Ingénierie logicielle objet (3 jours)
- Introduction aux architectures et technologies du Web (1 jour)
- Architectures .Net multi-niveaux (3 jours)
- Intégration d'applications (EAI, B2B) : les technologies et le projet (3 jours)
- Urbanisation du système d'information (2 jours)
- Architecture orientée service (3 jours)
- Architecture d'entreprise avec Java EE (4 jours)
- Du Mainframe au serveur d'applications (1 jour)

Développement Java et C++

- Introduction technique à Java (1 jour)
- Programmer en utilisant les aspects et les Design Patterns (3 jours)
- Java et la conception objet (5 jours)
- Développement d'un client riche avec SWT et Eclipse RCP (3 jours)
- Atelier Java avancé (5 jours)
- Eclipse, créer son environnement de développement intégré (2 jours)
- Programmation intensive avec Java (5 jours)
- Extrême Java (4 jours)
- Développer une application Corba (4 jours)
- L'essentiel de C++ et la conception objet (5 jours)
- Programmation efficace et avancée en C++ (5 jours)

Microsoft .Net

- C# et la conception objet (5 jours)
- Programmation avec Visual Basic .Net et conception objet (5 jours)
- Programmation intensive avec le Framework .Net (5 jours)
- Développement d'applications Web avec ASP.NET (5 jours)
- Développement d'applications Windows Forms sur la plate-forme .Net (5 jours)
- Développer des applications Web Services avec .Net (4 jours)
- Développer des applications avec C# et le Framework .Net 3.0 (5 jours)

Gestion de projet

- Gérer des projets avec un processus itératif (4 jours)
- Les méthodes agiles de développement logiciel (1 jour)
- Le Processus Unifié de développement logiciel (2 jours)
- Du recueil des besoins aux exigences : rédiger le cahier des charges (2 jours)
- Gestion de projet (3 jours)
- Manager des hommes dans le cadre d'un projet (2 jours)
- Management de projet (5 jours)
- MSProject (3 jours)
- Gérer les projets agiles avec Scrum (2 jours)
- Gérer les projets agiles avec XP (2 jours)

Frameworks Java EE

- Concevoir et développer des EJB 2 (5 jours)
- Développer une application Java EE avec les EJB 3 (5 jours)
- Gestion de la persistance avec Hibernate (3 jours)
- Mise en œuvre du Framework Seam (3 jours)
- Développement avec le Framework Spring (3 jours)
- Gestion avancée de la persistance avec Hibernate (2 jours)

Analyse, conception et modélisation avec UML

- Introduction technique à l'analyse, la conception et la programmation objet (1 jour)
- Introduction à UML (1 jour)
- Concevoir avec les Design Patterns (5 jours)
- La modélisation métier avec UML (3 jours)
- Analyse et conception avec UML (5 jours)
- La modélisation des systèmes complexes avec UML 2 et SysML (3 jours)
- La modélisation efficace des exigences avec les cas d'utilisation (2 jours)
- Analyse orientée objet avec UML (2 jours)
- D'UML 1 à UML 2 : quoi de neuf, docteur ? (1 jour)
- Modéliser les besoins et analyser avec UML (4 jours)

Oracle

- Introduction technique (1 jour)
- Exploitation (4 jours)
- SQL (3 jours)
- PL / SQL (3 jours)
- Optimisation des requêtes (2 jours)
- Administration (5 jours)
- Tuning (3 jours)

XML et Web Services

- Introduction à la technologie XML (1 jour)
- Introduction aux technologies Web Services (1 jour)
- Développer avec XML (3 jours)
- Développer une application XSL (2 jours)
- Développer des applications Web Services en Java (4 jours)
- Développer des applications XML avec Java (2 jours)

Développement Web

- Développement de pages Web avec HTML, CSS et JavaScript (3 jours)
- Développement, déploiement et administration d'applications Web (Java EE) avec WebSphere (3 jours)
- Développement d'applications Web avec PHP (3 jours)
- Ajax, pour dynamiser le poste client (2 jours)
- Hacking des applications Web (1 jour)
- Développer des applications avec Adobe Flex 3 (5 jours)
- Conception d'applications Web d'entreprise avec Java EE, les Servlets, JSP et Struts (5 jours)
- Développement d'applications Web avec Ruby on Rails (3 jours)
- Développement d'applications Web avec JavaServer Faces (3 jours)
- JavaServer Faces avancé (2 jours)
- Utilisation du Framework Struts pour le développement d'applications Web (3 jours)
- Développer une application Web avec Ajax et le Google Web Toolkit (3 jours)
- Développer des applications pour Adobe Integrated Runtime (2 jours)

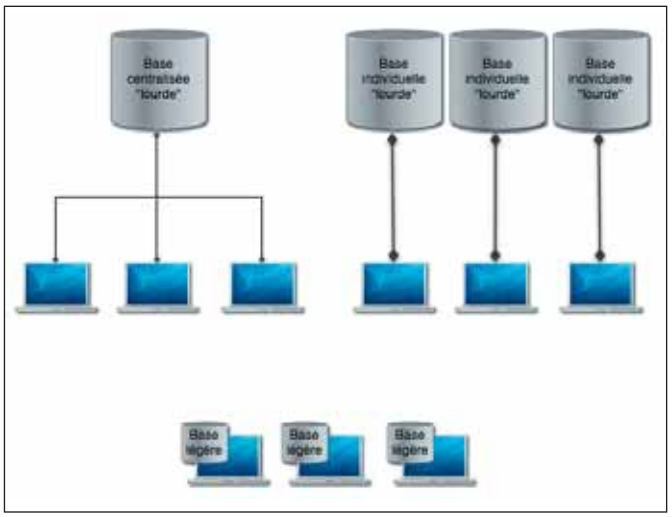
Stratégies de développement logiciel

- Test Driven Requirement ou la gestion des exigences dirigée par les tests (2 jours)
- Test Driven Development ou la programmation pilotée par les tests en Java (3 jours)
- Stratégie de test, vérification et validation (3 jours)
- Les fondements de l'IT Infrastructure Library (ITIL) (3 jours)
- Introduction au CMMI (3 jours)
- L'usine logicielle, des concepts à la pratique (3 jours)

www.valtech-training.fr
+33 (0)1 41 88 23 00 - +33 (0)5 62 47 52 00
info@valtech-training.fr

Les bases de données légères

Demandez à un développeur quelles bases de données il maîtrise, il y a de bonnes chances qu'il cite l'une des suivantes : Oracle, SQL Server, MySQL, PostgreSQL. Il existe pourtant de nombreuses bases de données, dites "légères" qui présentent une série d'avantages méconnus.



Ces outils permettent par exemple de faciliter l'automatisation des tests, de fluidifier le développement à plusieurs ou de libérer du temps à un Dba pour le laisser se pencher sur les problématiques de performance. C'est ce que nous allons tenter de découvrir, en nous concentrant sur le monde du développement Java.

Principe des bases de données légères

La plupart des bases de données légères présentent les mêmes caractéristiques. Elles n'ont pas besoin d'être installées, demandent une configuration minimale voire nulle, ont un encombrement mémoire minimal. Dans le cas des bases Java, elles sont souvent packagées en un seul fichier JAR. Elles savent toutes stocker les données dans des fichiers, certaines savent aussi fonctionner uniquement en mémoire, les données sont alors perdues en cas d'arrêt de la JVM. Parmi ces bases de données légères, on peut citer HSQLDB qui est la base légère la plus connue dans le monde Java mais aussi la base utilisée par OpenOffice.org pour proposer des fonctionnalités à la MS Access. HSQLDB tient dans un fichier JAR, sait fonctionner en mode client/serveur autonome ou en mode embarqué dans une application. On y accède à travers JDBC. Elle sait stocker ses données dans des fichiers ou bien utiliser uniquement la mémoire. Elle ne nécessite aucune configuration et propose pourtant la majorité des fonctionnalités d'une base poids lourd pour des performances mono utilisateur bien supérieures (voir le benchmark PolePosition http://www.polepos.org/).

Il faut tout de même citer les limitations des bases légères. La plupart ne supportent pas les triggers ou les procédures stockées. Dans le cas contraire, la syntaxe est très différente des bases standard du marché, ce qui peut être un frein à l'apprentissage et à la portabilité. Dans le cas des bases en mémoire, le volume des données traitées est vite limité par la mémoire disponible pour la JVM et réduit d'autant la mémoire disponible pour l'application, à cause des JVM 32 bits limitées à 2 Go d'adressage mémoire. Les avantages des bases légères ont fait tache d'huile depuis longtemps et c'est pourquoi Oracle propose une version XE légère et Microsoft une version SQL Server Express.

Attention, la qualification " légère " ne fait pas tout. Pour donner un exemple, Oracle XE pèse 150Mo qui paraissent peu par rapport à une installation complète, mais beaucoup comparés aux 624 Ko d'HSQLDB…

Simplifier les environnements de test

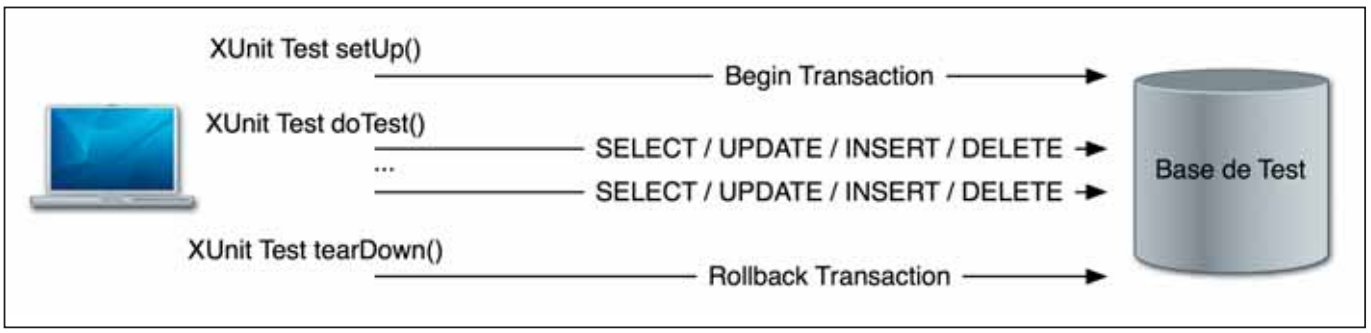
Toute personne impliquée dans des développements en équipe sait que la gestion des environnements de test est une problématique non triviale. Il faut soit installer une base de test par développeur, soit gérer les accès concurrents ou sérialisés à une base commune.

Un pattern à connaître est le " Database Sandbox Pattern " (http://xunitpatterns.com/Database%20Sandbox.html). Afin d'isoler les développeurs entre eux pour éviter les conflits lors des tests, on choisit d'installer une base par développeur, et pour chaque test, on charge des données de test, on exécute le test, on vérifie les résultats, enfin on supprime les données de test.

Un raffinement est d'utiliser le " Transaction Rollback Teardown Pattern " (http://xunitpatterns.com/Database%20Sandbox.html) qui consiste à exécuter le test dans une transaction terminée par un rollback. On s'assure ainsi que la base de données n'est pas réellement impactée par le lancement du test. Le framework Spring (http://www.springframework.org/) propose par exemple une classe utilitaire pour mettre en place ce Pattern : *AbstractTransactionalDataSourceSpringContextTests*. Ces patterns présentent quelques inconvénients. Il faut dans un premier temps installer puis maintenir autant de bases que de développeurs ; cela incombe en général à un Dba qui devrait se concentrer sur des tâches plus avancées comme l'optimisation des performances. La disponibilité du serveur de base de données est également un point à surveiller pour ne pas faire perdre du temps à toute l'équipe. C'est là que les bases légères permettent d'installer un environnement de test complet, base comprise, sur le poste de chaque développeur. La propriété " zéro configuration " permet d'en simplifier l'installation et la maintenance. De par leur taille réduite, elles passent inaperçues sur le poste déjà bien chargé d'un développeur. Enfin, le fait de disposer d'une base personnelle rend inutile la coordination avec les autres développeurs.

Utilisation dans le cadre de tests unitaires

Si l'on continue à se pencher sur la problématique des tests, l'écriture de tests unitaires pour des composants de types DAO est un problème en soi. Doit-on tester sur une base réelle ou doit-on utiliser des mocks pour mimer la



base ? Généralement on préconise l'utilisation de mocks puisqu'aucun environnement de test n'est alors nécessaire. Cependant, il est toujours intéressant de vérifier que nos DAO interagissent bien avec la base, surtout dans le cas de requêtes complexes. C'est bien là que les bases fonctionnant entièrement en mémoire (comme H2 database ou HSQLDB) sont les plus utiles. Elles sont capables de traiter des requêtes SQL très (très) rapidement, sans installation préalable, avec le maximum d'isolation entre les tests. On est capable d'écrire des milliers de tests unitaires ou même d'intégration qui seront exécutés en quelques minutes, en utilisant une base de données très proche de la base de production.

Peut-on tout faire avec une base légère ?

Il faut en général oublier les fonctionnalités trop spécifiques à telle ou telle base. Oublier les triggers, les procédures stockées ! Oublier les types spécifiques à telle ou telle base ! Cela peut paraître rédhibitoire. Pour moi, c'est plutôt le signe que l'on a souvent trop tendance à s'enfermer très tôt dans le choix d'une base de données alors que des mécanismes comme JDBC devraient nous permettre au contraire de faire des choix tardifs, voire pas de choix du tout.

Pour bien comprendre l'impact du choix précoce d'une base de donnée lourde, il faut bien comprendre les avantages d'un choix tardif : Le fait de rester agnostique est réversible. L'inverse l'est rarement. Si l'on s'est enfermé dans le choix d'une base et que l'on détecte un problème à un stade avancé du cycle de développement, il est déjà trop tard. Si l'on a choisi de rester hors des mécanismes spécifiques à telle ou telle base, il sera toujours temps de changer d'avis pour résoudre tel ou tel problème. La flexibilité vaut toujours mieux que la performance a priori. Le code qui s'exécute sur une base légère a toutes les chances de s'exécuter sur une base plus lourde. Ces bases sont en cela un élément forçant positivement à se limiter à du SQL standard. En pratique, on associera notre base légère à un framework de persistance comme Hibernate qui grâce à son mécanisme de dialecte, assurera une compatibilité avec toutes les bases.

Le choix d'une base légère à étonnamment deux impacts positifs sur les performances : elles sont extrêmement rapides pour exécuter des tests unitaires et le dba gagne du temps pour optimiser les requêtes avec les développeurs au lieu d'administrer les environnements de test.

Et en dehors du monde Java ?

Il existe beaucoup de bases légères en dehors du monde Java, et ce, depuis longtemps. On peut citer SQLite qui est utilisée par Google Gears pour créer des applications web off-line. Par exemple Google Reader qui utilise Google Gears, permet de récupérer les flux RSS en local, de les stocker dans la base SQLite et ainsi de fonctionner en consultation en mode déconnecté. Dans le monde .Net, la base VistaDB vise à fournir un remplacement isofonctionnel mais plus léger de SQLServer. Il devient facile de créer du code dédié à cette base, sans avoir à l'installer.

Les axes futurs

L'initiative de VistaDB est sûrement à prendre en exemple. Si l'on pouvait proposer en Open-Source, une base légère capable de mimer le comportement d'Oracle ou Sybase, en termes de syntaxe, de triggers, de procédure stockées, l'on disposerait d'un outil parfait pour développer le maximum de tests automatisés, sans avoir à se soucier de l'installation et de la maintenance d'un environnement complet de base de données lourde, et sans avoir à renoncer aux fonctionnalités spécifiques d'Oracle ou Sybase qui apportent dans certains cas des gains de performance non négligeables. Le futur est peut être aussi du côté de JDBC qui pourrait standardiser la manipulation de schémas comme il a standardisé le requêtage. On disposerait alors d'un mécanisme de DDL standard masquant encore plus le choix de la base sous-jacente.

Quels sont les freins à l'adoption plus large de ces bases ?

Nous avons vu que les bases légères ont des propriétés très intéressantes qui ne sont pas nouvelles, loin de là, et pourtant, leur utilisation reste trop confidentielle. Les raisons sont multiples : les grands comptes ont des réflexes éditeur et base de données unique qui ne laissent pas la place à des bases Open Source et spécialisées.

David GAGEOT
Directeur Technique
Valtech Technology Paris
david.gageot@valtech.fr
www.valtech.fr

Tableau récapitulatif

HSQLDB	Java	http://hsqldb.org/	Sans doute la plus connue des bases de données légères Java.
H2 Database	Java	http://www.h2database.com/	Créée par le développeur d'HSQLDB. Plus rapide. Meilleure gestion de la mémoire.
Derby	Java	http://db.apache.org/derby/	Base issue de l'ouverture du code de Cloudscape.
JavaDB	Java	http://developers.sun.com/javadb/	Base intégrée dans le JDK 6 de Sun. Basée sur Derby.
SQLite	C/C++	http://www.sqlite.org/	Base utilisée par Google Gears.
VistaDB	.Net	http://www.vistadb.net/	Base la plus proche possible de SQL Server.

Une vision industrielle du développement logiciel

SoftFluent est une jeune entreprise innovante créée par des experts du domaine du développement logiciel et de l'industrialisation des processus associés.

La vision de la société SoftFluent repose sur le constat que le métier du développement logiciel reste encore trop largement artisanal.

SoftFluent s'est donné la mission de contribuer à son industrialisation.

L'évolution rapide des technologies, le dynamisme de plus en plus fort des marchés, les transformations rapides de certains secteurs ainsi qu'un contexte d'internationalisation accru amènent des contraintes de plus en plus forte sur les équipes devant réaliser des développements logiciels.

Cela se traduit par un besoin d'adaptation des méthodes, que vous soyez un éditeur de logiciels, une grande entreprise devant réaliser un logiciel spécifique ou une société de service.

Pour tirer les bénéfices escomptés, cette adaptation des méthodes se doit d'être structurelle et de s'inscrire par rapport à ces enjeux, et non autour d'une simple approche articulée autour de la technologie du moment.

C'est pourquoi SoftFluent investit fortement en Recherche et Développement pour apporter des solutions structurellement innovantes, tout en capitalisant sur sa maîtrise des 3 composantes clés de la réussite d'un projet de développement logiciel :

- les hommes,
- la méthode,
- les outils.

SoftFluent s'avère donc le partenaire idéal de vos développements logiciels stratégiques et peut vous conseiller de manière pragmatique sur la manière de mener à bien votre projet, incluant le choix de partenaires compétents dans votre région.

SoftFluent mise et investit sur les compétences de ses collaborateurs

Fondée par des anciens de Microsoft, la société dispose d'une offre unique sur les technologies de développement de Microsoft, leur mise en œuvre et la stratégie de mise en œuvre associée.

CONSEIL

CONSEIL
Conseil aux éditeurs de logiciels sur de multiples dimensions

Conseil aux entreprises en technologies de l'information

CONSEIL TECHNOLOGIQUE
Expertise en développement sur les technologies microsoft

Audit d'applications .NET

Accompagnement de projets à fort enjeu technologique

PRODUITS LOGICIELS

USINE LOGICIELLE ORIENTÉ-MODÈLE
permettant de structurer les applications orientées données

SERVICE EN LIGNE
http://www.codefluent.com
permettant de générer à la volée des composants métier

DÉVELOPPEMENT

DÉVELOPPEMENT D'APPLICATIONS
Développements spécifiques
Développements de produits logiciels en externalisation de R&D d'éditeur de logiciels

Développements de prototypes à fort enjeu technologique

APPLICATIONS EN TANT QUE SERVICE
Spécifications et développement

Hébergement, exploitation et support inclus selon un forfait

Recourir aux consultants de SoftFluent est la garantie de disposer du meilleur niveau de compétences pour le développement sous plate-forme Microsoft.

Editeur du logiciel CodeFluent, SoftFluent dispose d'un savoir-faire d'industrialisation de niveau "produit".

SoftFluent réalise d'ailleurs des développements pour le compte d'éditeurs de logiciels, qu'il s'agisse de produits traditionnels déployés ou de produits hébergés sous forme de services.

PASSIONNÉ(E) DE DÉVELOPPEMENT LOGICIEL ?

« Compte tenu de son positionnement et de sa mission, SoftFluent investit en priorité sur les compétences de ses collaborateurs et collaboratrices. SoftFluent me procure un tutorat par des profils très expérimentés et l'immersion dans des projets innovants. Cette approche pragmatique me permet d'enrichir mon savoir-faire tous les jours et d'interagir en permanence avec des collègues d'excellent niveau ».

Naïla ZEROUG, chez SoftFluent depuis 2006

SoftFluent SAS – 46, rue Auguste Blanqui – 94250 Gentilly – France – Tel : +33 1 46 16 05 02 – Fax +33 1 46 16 06 71 – www.softfluent.com

Microsoft | PROGRAMME IDÉES

La première fabrique logicielle entièrement pilotée par les modèles

CodeFluent est un produit de génie logiciel qui permet d'industrialiser la fabrication d'applications professionnelles manipulant des données sur la plate-forme .NET en automatisant la création des composants à partir d'une modélisation de votre métier.

CODEFLUENT PROCURE LES BÉNÉFICES SUIVANTS

Un gain très significatif de productivité par l'automatisation des tâches répétitives et la mise en œuvre du modèle métier selon une architecture orientée services.

La limitation du risque par la structuration du travail des développeurs autour d'une modélisation objet évolutive qui garantit une mise en œuvre selon les meilleures pratiques d'implémentation SOA sous plate-forme Microsoft.

Une maintenabilité et une qualité accrues grâce à l'approche intrinsèque de génération qui évite les erreurs et permet d'effectuer des mises à jour sur toutes les couches à l'aide d'une modification centralisée.

L'évolutivité de l'application car la prise en charge de nouvelles technologies et de nouvelles versions de plate-forme est assurée par la mise à disposition régulière de nouveaux producteurs (Windows Presentation Foundation, Office Business Applications, mobilité).

Les clients de **CodeFluent** gagnent sur leur marché

ENTREPRISES

118 218 LE NUMERO

118218 a choisi CodeFluent pour le développement accéléré et pérenne d'une application spécifique à son métier. 118218 dispose ainsi d'une application métier de type Web 2.0 tirant à la fois partie des technologies ASP.NET et Ajax.

ÉDITEURS DE LOGICIELS

VCS TIMELESS
Alternative retail software

VCS Timeless est un éditeur d'envergure internationale dans le secteur de la distribution spécialisée. La société a retenu CodeFluent pour le développement de son progiciel de gestion dédié à son secteur. Colombus.Next repose sur la technologie .NET 3.0 incluant WPF et WCF. La roadmap comprend le développement de nombreux modules sur plusieurs années.

SSII

SOGETI

Sogeti Application Consulting Services a sélectionné CodeFluent comme outil privilégié pour la réalisation d'applications spécifiques sous .NET. Sogeti propose à ses clients d'utiliser la fabrique logicielle CodeFluent pour la réalisation de leurs développements afin de garantir productivité, limitation du risque et maintenabilité.

DÉVELOPPEURS

CodeFluent EXPERT NETWORK

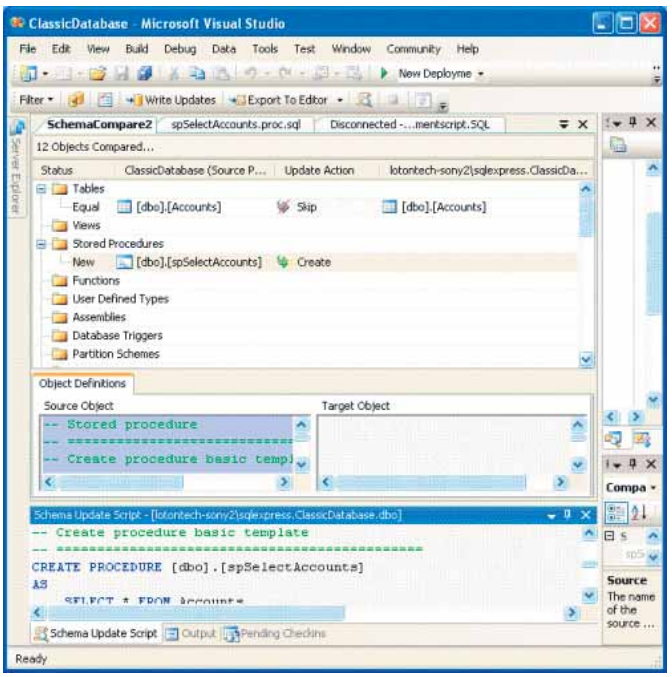
Richard Clark, 1er MVP .NET reconnu par Microsoft sur le marché français, a acquis la licence CodeFluent pour optimiser ses développements. Il est entré par ailleurs dans le réseau des experts CodeFluent, un label créé par SoftFluent pour reconnaître l'implication des développeurs dans la promotion et l'évolution du produit.

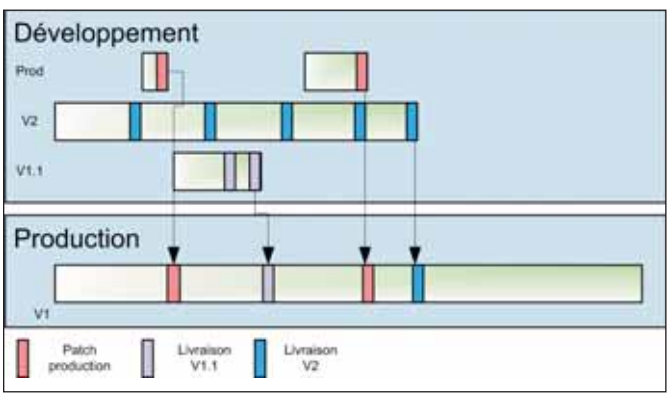
L'usine à build et les bases de données

Le dossier du n°104 était consacré aux usines à build et plus généralement à ce que l'on peut nommer " l'industrialisation logicielle " couvrant le build, le cycle de vie et le contrôle qualité du logiciel. Mais lorsque le logiciel comprend une base de données – comme c'est le cas pour les applications Web ou pour les intranet de gestion – comment les développements SGBD peuvent s'intégrer au processus automatisé de construction du logiciel ?

Les solutions évoquées dans le n°104 (Maven2, Ant principalement) permettent d'automatiser la construction du logiciel pour la rendre récurrente, voire quotidienne, via des " planificateurs " évolués comme CruiseControl, LuntBuild ou Continuum. Couplées à des exécutions de tests unitaires (Junit), fonctionnels (FitNesse, GreenPepper) et d'outils de revue de code (PMD, CheckStyle), elles permettent de valider le bon fonctionnement du logiciel et sa non-régression tout au long des développements, les dysfonctionnements étant détectés au plus tôt dans le processus de développement de l'application.

Mais ces solutions n'adressent en fait que le tiers applicatif du logiciel. Lorsqu'une entreprise automatise la construction de ses applications elle se rend alors vite compte d'un nouveau " noeud " dans le processus : les développements SGBD. Car s'ils représentent une partie moindre des développements, ce qui a permis aux développeurs SGBD de packager et de déployer leurs évolutions sans difficulté pendant que les équipes applicatives se mobilisaient pour réussir à construire leur partie, ils ne peuvent pour autant, plus suivre un rythme de déploiement quotidien. Les " Night Builds " applicatifs produits utilisent alors généralement les bases de données de développement qui ne sont pas le reflet de ce qui sera livré. L'exécution et la validation des tests (notamment fonctionnels) sont donc biaisées et on détectera encore des erreurs au moment de la livraison, faute de validation complète du logiciel de manière continue. Pire : les éventuelles erreurs remontées par le processus de validation nocturne seront ignorées par les développeurs reléguant cela à un " problème de SGBD ", celui-ci ne pouvant être considéré comme fiable. Cette problématique, ce " trou " dans l'automatisation peut mettre à mal l'ensemble de l'usine à construction du logiciel.





Les spécificités du build SGBD

On peut donc bien parler de build SGBD, opération qui consiste à packager et à déployer les évolutions développées sur un environnement cible. Mais ce build a une spécificité importante par rapport au tiers applicatif : il est systématiquement incrémental. Chaque livraison, chaque déploiement n'est qu'un delta permettant de faire passer l'application d'une version N à une version N+1. Intégrez à cette problématique le parallélisme constant des versions en développement, décrit dans le diagramme ci-dessous, et la situation devient très complexe.

Le diagramme ci-contre montre un exemple typique où la chronologie des versions en développement est différente de la chronologie des mises en production. Or cela pose un problème pour les livraisons incrémentales : la V2 sera mise en production sur une V1.1 patchée. Mais lorsque les équipes livrent pour la première fois cette V2 la version 1.1 n'existe pas, encore moins le patch de production. Et encore, ce diagramme ne présente pas les environnements d'intégration, de recette, voire de pré-production...

Face à ce problème, le build et le packaging des développements SGBD est aujourd'hui considéré comme une opération nécessairement manuelle. Mais avec l'industrialisation croissante de la production logicielle, il se pourrait bien que le SGBD devienne le prochain enjeu de l'automatisation. Ajoutez à cela qu'avec l'externalisation progressive de divers modules logiciels composant une application, le manque de norma-

lisation des livraisons SGBD devient également une problématique bien réelle.

Quelles solutions, quel avenir ?

Malheureusement les solutions sont rares, voire inexistantes, les éditeurs ou initiatives open-source ayant préféré se concentrer sur les aspects " client " SGBD plus ou moins évolué (Oracle SQL Developer, projet Data Tools Platform d'Eclipse) plutôt que de construire des plates-formes de développement modernes adressant ces problématiques. L'élément clé de la réflexion est sans aucun doute la gestion de version, à cause de l'aspect incrémental de l'ensemble des évolutions. Or les systèmes de versioning actuels sont quasiment tous orientés fichiers, ce qui correspond mal aux besoins de gestion de version des entités du SGBD (tables, vues, index, procédures stockées). Alors on essaie de faire rentrer le SGBD dans les modèles existants, en choisissant de versioner les fichiers SQL correspondant aux entités du SGBD. Mais la fusion des évolutions (ou merge) devient alors une fusion de fichier, nécessitant généralement une intervention humaine pour interpréter le contenu des requêtes à fusionner.

Alors à quand la production SGBD moderne où les développements peuvent être déployés automatiquement, où chaque développeur peut recréer à volonté la base de l'application dans une version donnée, où les livrables sont standardisées pour permettre à différentes équipes de travailler sur une même application, où le retour arrière serait automatiquement géré, où les tests unitaires seraient normalisés et capitalisés, où le refactoring de nom, la migration de données seraient assistés par une plate-forme de développement puissante ?

N'y a-t-il pas d'enjeux ? Pourtant, il n'est pas rare de voir un intégrateur travailler pendant des jours avec un développeur SGBD pour réaligner une plate-forme de recette sur laquelle on finit par découvrir que plusieurs patchs n'ont pas été appliqués. Comment améliorer la qualité d'un logiciel lorsque l'on ne peut pas avoir confiance en la plate-forme de validation ? Microsoft semble l'avoir bien compris et a pris une large avance sur ce domaine avec son extension " Team Edition for Database Professionals ". Référentiel déconnecté, refactoring, tests unitaires, packaging automatique, gestion du cycle de vie sont quelques unes des fonctionnalités prises en charge par cette extension. Tenteraient-ils une fois de plus de prendre des parts de marché sur ses concurrents (Oracle notamment) en misant d'abord sur des outils performants et séduisants ? Possible, mais la solution reste architecturée autour d'un versioning fichier qui peut avoir ses limites. Oracle a arrêté le combat depuis des années, se reposant sur son outil Oracle Designer, proposant un versioning natif des entités du SGBD (Oracle SCM). Cet outil, bien qu'extrêmement performant, a cessé d'évoluer depuis 2004 et Oracle prévoit de le remplacer peu à peu par Jdeveloper. Pour le moment, les outils d'Oracle ne répondent pas aux nouvelles problématiques exposées dans cet article. A noter également un projet basé sur la plate-forme Eclipse RCP qui devrait voir le jour dans les prochains mois. Affaire à suivre sur http://www.nextep-softwares.com. Nous sommes donc bien loin du tout automatique, les solutions restent pauvres ou propriétaires. Mais les besoins sont là, il y a donc fort à parier que les solutions vont suivre...

Christophe Fondacci
cfondacci@nextep-softwares.com
neXtep Softwares

Model Driven : quand le modèle pilote le code

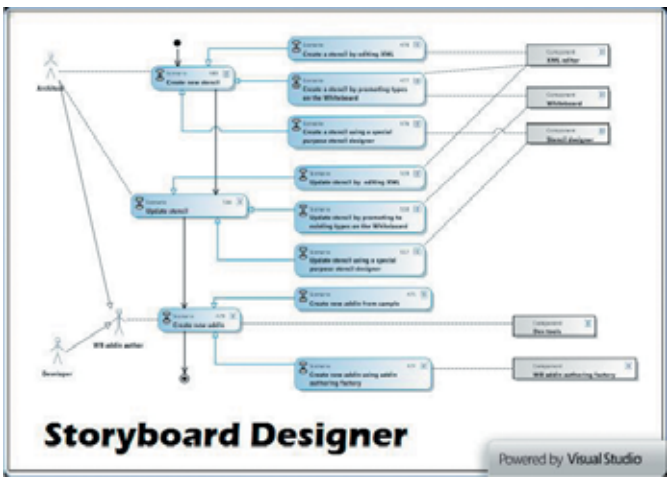
Le succès de la récente journée MD Day à Paris prouve que le sujet suscite un intérêt croissant surtout du côté de l'entreprise. Le développeur se sent sans doute (encore) un peu éloigné de ces concepts ou tout du moins, les voit-il trop abstraits, trop compliqués à mettre en œuvre.

Il est certain que l'approche Model Driven, ou pilotage par modèle, peut apparaître abstraite au regard des différentes déclinaisons MD que l'on trouve. Les principales se nomment : MDA, MDD, MDT, MDE. La base reste la même : le modèle. Dès que l'on fait de la modélisation, d'une certaine manière, on fait du Model Driven. Ce MD est censé apporter une rigueur de conception à partir d'un modèle défini et c'est sur ce modèle, que l'on génère tout ou partie du code, d'une couche métier, d'une interface, d'une application, etc. On peut en effet appliquer le concept MD à tout et n'importe quoi.

Cependant, après une vague d'enthousiasme il y a environ 3 ans, les éditeurs puis les utilisateurs se sont calmés. Car si sur le papier le MD, et tout particulièrement MDA, répond à des besoins réels et séduit, inutile de foncer tête baissée. Il y a un problème de choix.

Uniquement par le modèle ou autoriser le round-trip ?

Là se pose en effet un choix architectural et technique. Aujourd'hui, nous savons bien faire de la génération à partir d'un modèle. En quelque sorte, on part du haut vers le bas (le code). *" Le MDA propose une approche top-down (haut en bas), le modèle faisant foi. Dans la pratique, nous recommandons une approche " Model Driven " (top-down), où le modèle est maître, et où les modifications de code ne se font que dans des parties définies et instrumentées par le générateur. Cette approche fonctionne depuis plus d'une décennie et est fiable "* précise **Philippe Desfray** (direction R&D Softeam).



Mais la tentation est forte de vouloir faire du bidirectionnel. C'est-à-dire la possibilité de partir du code pour générer un modèle ou encore modifier le code généré, les modifications étant ensuite appliquées au modèle. *" Il est pourtant souvent constaté que les utilisateurs veulent bénéficier du mode round trip, pour pouvoir adopter une approche mixte, mêlant à la fois le top-down et le bottom-up (bas en haut) ".*

Et c'est là l'un des problèmes. Aujourd'hui, dans le cadre de projets lourds et complexes, la synchronisation bidirectionnelle entre code et modèle (bref, le round trip) n'est pas assurée au niveau qualitatif. Si les techniques évoluent, il ne faut pas s'attendre à des miracles à court terme. L'approche " haut vers le bas " est à privilégier pour au moins une partie du code à partir du modèle. C'est ce que l'on voit souvent dans les outils de génération de la couche d'accès aux données (= mapping).

Aujourd'hui, seul MDA émerge réellement, même si MDE ou MDD commencent à prendre de l'importance. L'avantage du MDA est que l'on dispose désormais de très nombreux outils, commerciaux ou open source et que ces outils reposent sur des frameworks et standards (MOF, UML, EMF, etc. *" Le foisonnement d'outils et de méthodologies intégrant MDA est un signe fort que MDA a le vent en poupe dans un monde toujours plus avide d'automatisation et d'industrialisation. Pour ceux qui seraient gagnés par la confusion, il faut se concentrer sur la compréhension du MDA tel qu'il est défini par l'OMG, ainsi que les standards sur lequel il repose. On sensibilisera les développeurs et les architectes en montrant la plus-value qu'apporte le MDA, comme l'automatisation, l'abstraction, l'industrialisation, la capitalisation du savoir-faire, la clarification des rôles, la maîtrise des environnements techniques toujours plus complexes, ou encore l'opportunité de se concentrer sur le métier"* analyse Philippe Desfray.

Et le langage spécifique alors ?

Enfin, on peut se poser la question de l'opportunité d'utiliser un DSL (Domain Specific Language) à la place de MDA ou d'un quelconque MD. L'approche DSL peut être intéressante, même si on perd l'abstraction offerte (en principe) par le MDA. Surtout, le DSL n'est pas standard et aucune spécification n'existe, chaque éditeur faisant ce qu'il veut. *" Les DSL ne sont que des modèles spécifiques non standard, un cas particulier de mise en œuvre MDA. L'appui sur les standards permet de bénéficier de personnes formées, de capacités d'échange de modèles, d'une offre outillée importante",* conclut Philippe Desfray.

Mais ce n'est pas pour autant qu'il faut écarter le DSL. Dans des environnements précis, sur des plates-formes données, le DSL peut rendre des services afin d'adopter un modèle de développement, certes dépendant, mais adapté à ses besoins propres. Alors pour ou contre le DSL ? La question se pose.

■ **François Tonic**

L'approche MD

Que recouvre exactement le terme Model Driven (MD) ? Est-ce un concept abstrait ou une méthode utilisable clé en main ? Je vais tenter de répondre à ces questions en présentant les déclinaisons possibles de l'approche Model Driven.

En deux mots, l'approche MD est une démarche de construction des logiciels visant à placer la modélisation au cœur de leur processus d'élaboration. Les livrables étant pour l'essentiel dérivés des modèles (programmes, documentations, tests, etc.). Mais pour répondre plus précisément il faut d'abord savoir ce qu'est un modèle.

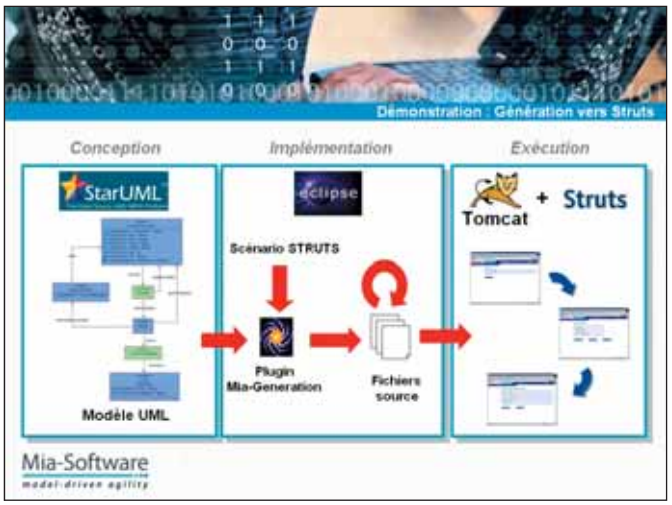
Modèle ? Vous avez dit modèle ?

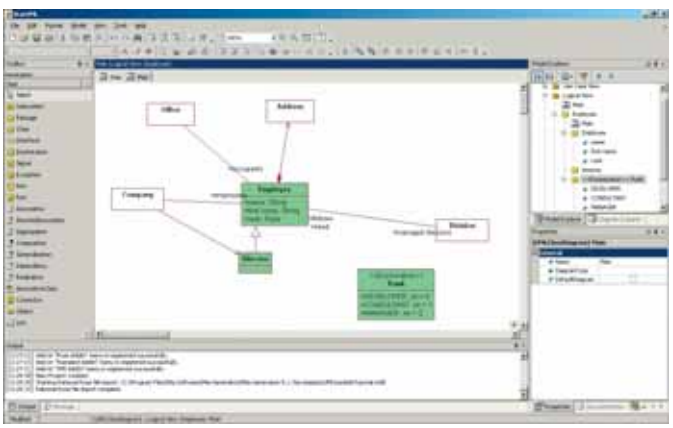
Parmi les nombreuses définitions de ce terme, celle de l'académie française m'apparaît la plus appropriée : *" Représentation, physique, graphique ou, plus généralement, mathématique, qui formalise les relations unissant les différents éléments d'un système, d'un processus, d'une structure, en vue de faciliter la compréhension de certains mécanismes ou de permettre la validation d'une hypothèse ".*

Un modèle est donc une abstraction, une simplification, une idéalisation d'une réalité existante ou à venir, permettant d'en maîtriser la complexité. Il s'agit d'une construction intellectuelle censée être représentative de l'objet de l'étude.

Il est le fruit d'une démarche intellectuelle de distanciation du flux des expériences quotidiennes. Ainsi, en prenant le recul nécessaire aux généralisations et aux rationalisations, le modèle est le moyen d'échapper à l'empirisme.

Disposant d'un modèle représentatif du réel, l'ingénieur comme le scientifique se dotent d'un outil de compréhension et d'action extrêmement puissant puisqu'il leur devient possible de faire des prévisions fiables et des produits reproductibles.





Nous connaissons tous des modèles : celui de la mécanique céleste pour prévoir le mouvement des planètes, de l'ingénierie mécanique pour calculer les contraintes supportées par une pièce d'un moteur.

Les modèles informatiques sont également des modèles d'ingénierie. L'informatique devenant plus mature, sa modélisation peut s'envisager comme plus représentative des systèmes réels. Cette pertinence est la condition sine qua none de l'adoption de l'approche MD. Ou plutôt des MD, le pilotage pouvant être très variable en fonction du périmètre couvert. Afin de s'en faire une idée, sans doute convient-il de situer le modèle dans son écosystème.

Le modèle dans son écosystème

La figure page suivante représente les acteurs de l'écosystème et les principaux flux d'informations entre eux. Schématiquement l'approche sera plus ou moins MD selon l'ouverture des vannes Modélisation, Capitalisation et Dérivation. Plus elles seront ouvertes, plus le processus sera Model Driven. A contrario plus les vannes connectées à la démarche empirique le seront, moins les modèles gouverneront le produit.

La nature et les volumes des flux d'informations conditionnent les déclinaisons possibles de l'approche MD.

Les contraintes de coûts, de délais, réglementaires, humaines et autres forment l'environnement de l'écosystème. Ils poussent à une maîtrise toujours plus grande des processus de développement. Donc objectivement à une place croissante d'une modélisation représentative.

Le flux Modélisation

Le périmètre MD dépendra de la portée et du degré de liberté du formalisme. Il varie du " clé en main " comme c'est le cas des méthodes formelles (méthode B) à l'entièrement libre, comme pour les DSL (*Domain Specific Language*). L'initiative **MDA** (Model Driven Architecture) de l'OMG est à ce titre dans une position intermédiaire. Centrée sur la modélisation **UML**, elle propose une approche en trois niveaux d'abstraction. Le CIM (Computation Independent Model) dédié à la formalisation du besoin métier indépendamment de toute mise en œuvre informatique. Le PIM (Platform Independent Model) : modèle de la solution informatique indépendante de la technologie. Et un PSM (Platform Specific Model) modélisant la solution dans un environnement technique. Les CIM et PIM sont le plus souvent uniques. En revanche les PSM peuvent être multiples afin de se cumuler (objets répartis + Java par exemple) ou de viser une

couche particulière du système (schéma de la BD). Pour converger vers la solution tout en assurant la traçabilité, des transformations sont opérées, des modèles les plus abstraits vers des modèles de plus en plus concrets. Une autre dimension est le focus de modélisation. Il peut être porté sur l'amont (les spécifications externes avec ou sans CIM), l'aval (les spécifications internes détaillées - PSM) ou uniformément réparti. Pour des raisons de moyens, une situation courante consiste à ne maintenir qu'un seul modèle hybride entre PIM et PSM.

Quelques types d'outils

Type d'outil MD	Description	Principaux cas d'utilisation
Modeleur	Construction / Visualisation d'un modèle en mode graphique	• Analyse • Conception • Compréhension d'un logiciel existant (combiné avec T2M)
Model-to-Text (M2T)	Production automatique de texte (code source, fichiers de paramétrage, documentation) à partir d'un modèle	• Industrialisation des développements logiciels (génération de code) • Documentation (génération documentaire)
Model-to-Model (M2M)	Production automatique de modèles à partir d'autres modèles	• Changement de niveau d'abstraction (analyse vers conception) • Changement de paradigme (objet vers relationnel). • Migrations d'outils (outil Merise vers outil UML, outil UML1 vers outil UML2, ...)
Text-to-Model (T2M)	Production automatique de modèles à partir de l'analyse de texte (code source ou fichiers de paramétrage).	• Rétro-documentation • Cartographie • Analyse qualité • Refactoring (combiné avec M2M et M2T)

Le flux Dérivation

Il s'agit ici de produire tout ou partie d'un livrable final directement à partir d'un modèle. On pense bien entendu à la génération de code applicatif (**MDG**) ou à la transposition manuelle.

En fait les artefacts produits peuvent être également la documentation ou les mises à jour des schémas de BD. Mais aussi le référentiel des fonctionnalités, les cas de tests, la liste des tests de non-régression à jouer suite à des évolutions (**MDT**). Ou bien encore l'IHM au moins pour les éléments les plus simples (fenêtre de propriétés d'un objet par exemple).

Vous l'aurez compris cette liste n'est pas exhaustive.

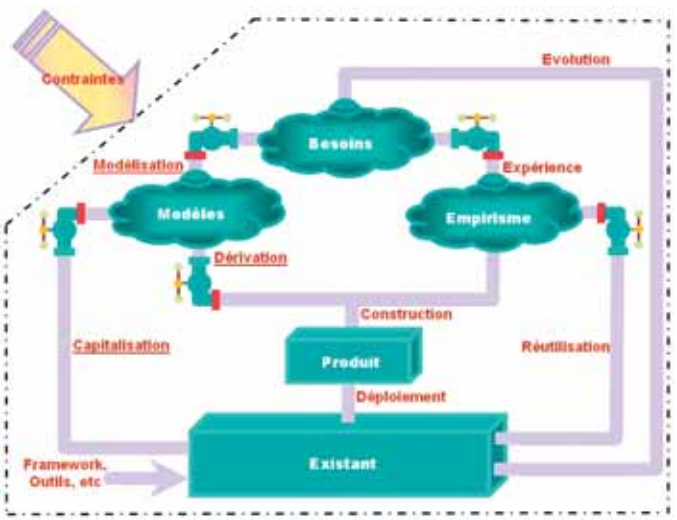
La génération peut être réalisée directement avec l'outil de modélisation, via un générateur sur étagère ou bien à l'aide d'un méta-générateur. Avec un modeleur (qui donne souvent accès au modèle par scripting) le générateur est à réaliser intégralement et est lié à sa technologie. Dans le second cas le générateur agit comme une boîte noire dont les produits sont tout au plus paramétrables. Le méta-générateur, ou générateur de générateurs, cumule les avantages puisqu'il permet de personnaliser entièrement le code généré tout en étant totalement transparent. Une chose est certaine, plus les produits dérivés automatiquement des modèles sont nombreux et impliqués dans le produit principal, plus les modèles prennent de l'importance dans le projet. C'est un levier significatif pour l'adoption progressive de la démarche MD.

Le flux Capitalisation

Last but not least le flux capitalisation est celui de la reprise d'un existant déjà produit, d'un existant incorporable disponible sur étagère ou d'un outillage. On capitalise ici sur le travail des équipes internes ou externes pour un gain de 100%, ou via des outils pour un gain décuplé.

L'existant déjà produit servira de base aux versions suivantes, notamment pour les analyses d'impact. Il pourra aussi être rétro-modélisé automatiquement à partir du code en l'absence d'un modèle fiable (**MDM**).

Parmi l'existant incorporable figurent bien sûr les framework et les progiciels. Mais la capitalisation métier passera aussi par la réutilisation de services et de composants métiers d'entreprise. La réutilisation des services métiers n'est possible qu'au travers de leur description dans un repository implanté dans le modèle de l'entreprise (**MD-SOA**).



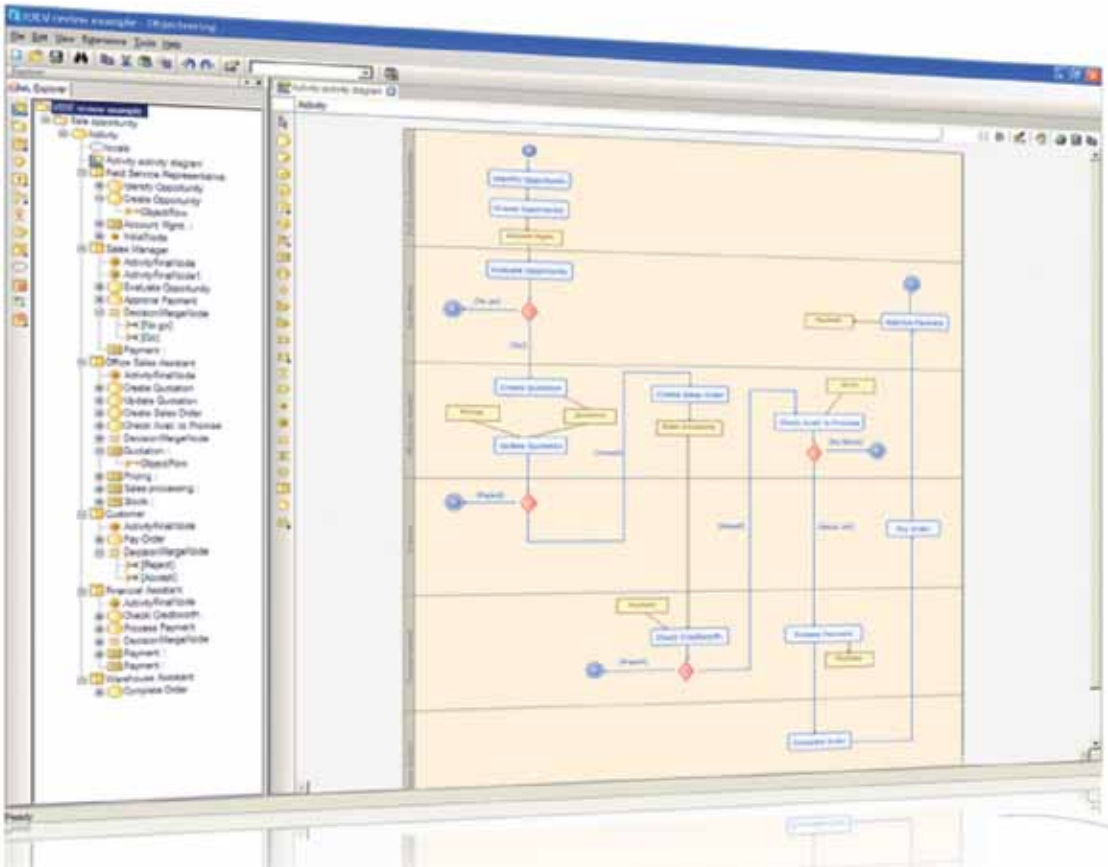
Les acronymes MD

Les concepts énoncés dans l'article se déclinent en de nombreuses filières. Le tableau ci-après en énumère quelques unes.

Les MD	Signification	Description
MDA	Model Driven Architecture	Architecture MD préconisée par l'OMG
MDD	Model Driven Development	MDE appliqué à l'informatique
MDE	Model Driven Engineering	Recours à des modèles comme source primaire dans un processus d'ingénierie
MDG	Model Driven Generation	Génération d'artefact à partir de modèles
MDI	Model Driven Integration	Intégration basée sur les modèles
MDM	Model Driven Modernization	Cas particulier de MT visant à transformer du code en passant par des modèles intermédiaires
MDO	Model Driven Offshoring	Mise à profit de l'approche MD dans la gestion des développements off-shore
MD-SOA	Model Driven SOA	Recueil et définition des services métiers d'entreprise dans un modèle.
MDT	Model Driven Testing	Définition des fonctions à tester et de cas de tests à partir des modèles. Automatisation des tests.
MT	Model Transformation	Opération consistant typiquement à traduire un modèle exprimé dans un langage (méta-modèle) en un autre modèle exprimé dans un langage différent (autre méta-modèle)

Your projects deserve a tool*

Objecteering Modeler



Objecteering 6.1

guide vos développements par les modèles sur toute la portée du système

Objecteering 6.1 est une solution de modélisation complète et ouverte supportant les standards OMG de modélisation UML 2.1, BPMN, et SysML, la modélisation de l'Architecture d'Entreprise et la modélisation d'une architecture SOA.
Il intègre le support de l'analyse des besoins et de la définition du dictionnaire, assurant ainsi une traçabilité complète sur tout le cycle de vie.
La technologie MDA associée à une ouverture de l'outillage en Java permet de guider le développement à chaque contexte utilisateur, chaque infrastructure technique cible. Ses générateurs sur étagère automatisent le développement d'applications Java, J2EE, .Net C#, C++, SQL...

Objecteering Software, éditeur de l'atelier Objecteering 6, est le spécialiste français UML/MDA pour le développement d'applications guidé par le modèle. Son offre modulaire couvre le cycle de vie de la gestion des exigences jusqu'au déploiement d'application.
Pour plus d'information sur Objecteering 6.1, pour télécharger Objecteering **Free Edition** ou Objecteering **Enterprise Edition**, rendez-vous sur : www.objecteering.com

Architecture d'Entreprise
UML2 BPMN
C# MDA SysML
Java SQL
J2EE SOA
C++

* Vos projets méritent un outil

www.objecteering.com - Tél. : 01 30 12 16 60 - sales@objecteering.com

Objecteering SOFTWARE

The model-driven development company

Quintre&sens • 4721

L'outillage concernera la modélisation bien sûr, la transformation de modèles (**MT**), la gestion des exigences que devra satisfaire le logiciel ou bien encore la génération de code.

Pour conclure

L'approche sera plus ou moins driven en fonction des volumes véhiculés par les flux d'informations Modélisation, Capitalisation et Dérivation. Les conditions pour une maximisation des volumes étant principalement l'outillage et l'automatisation, elle-même conditionnée par le degré de couverture d'un modèle de niveau N par le modèle N + 1 (taux de génération du code à partir du PSM par exemple). Les modèles formels, UML et DSL ne sont pas incompatibles. Chacun d'eux peut apporter sa contribution au développement d'une application. La mise en œuvre du MDA à la DSIV de la SNCF l'illustre par la création d'un DSL " DSIV PL " et l'emploi d'UML via une fusion des modèles PIM et PSM. Le succès de ce déploiement du MD repose d'une part sur une batterie d'outils de validation de modèles, de génération (48 à 68% du code) et de reprise de l'existant. Et d'autre part sur l'exclusion des parties les plus difficiles à modéliser (IHM, Règles de Gestion). L'ensemble de contraintes formant l'environnement de l'écosystème où implanter le MD étant toujours unique, pour réussir, c'est la démarche MD elle-même qu'il nous faut adapter à l'entreprise. C'est pourquoi il n'y a pas un MD, mais des MD.

Modeleur UML ou DSL ?

	Avantages	Inconvénients
UML UML (Unified Modeling Language) est une notation standardisée par l'OMG (Object Management group) pour analyser et concevoir des systèmes logiciels	• UML est standardisé • De nombreux outils de modélisation qui constituent un éventail très riche en termes de fonctionnalités et de tarifs. • UML intègre le mécanisme de profil (stéréotypes, tagged-values et contraintes) qui permet d'étendre UML pour l'adapter à des besoins non couverts nativement par la norme.	• Les concepts fournis par un modeleur UML sont largement plus riches que les besoins réels. • La création d'un profil et son outillage dans un modeleur UML peuvent s'avérer coûteux.
DSL / DSM Un DSL (Domain Specific Language) est un ensemble de concepts, généralement définis par un méta-modèle, permettant de décrire un domaine particulier d'un système. Pour créer des modèles selon un DSL il existe deux possibilités : une syntaxe concrète permettant de décrire textuellement un modèle ou un modeleur dédié (Domaine Specific Modeler) permettant de " dessiner " le modèle.	• Un DSL fournit un outil avec lequel on peut manipuler directement les concepts de son métier. • L'apprentissage du langage de modélisation est donc réduit par le fait que les concepts à appréhender sont déjà connus.	• Il existe encore très peu de DSL standardisés. • Si le DSL n'est pas déjà outillé par un DSM du marché, se lancer dans la construction d'un tel outil reste complexe malgré les solutions qui émergent (GMF pour Eclipse et DSL-Tools pour Microsoft) • Les fonctionnalités dans un modeleur créé " soi-même " resteront toujours largement inférieures à celles disponibles dans les meilleurs outils du marché.

Luc Laforets
Consultant / Mia-Software
llaforets@Mia-Software.com

AVIS D'EXPERT

Le modèle, édifice central de l'application

Questions à Jean-Loup Comeliau (Lyria)

Programmez ! : On entend beaucoup parler de MD comme le MDT, MDD ou encore MDA. Quel est aujourd'hui le MD le plus en vogue et pourquoi ?

Jean-Loup Comeliau : Il est naturel que le " Model Driven " fasse parler de lui. Jusqu'à récemment, il s'agissait surtout des concepts architecturaux du MDA pour aborder le génie logiciel de manière différente. Il s'agit d'une réelle avancée dans la manière de concevoir les applications, qui change notre approche et nos habitudes. Aujourd'hui, les standards sont mûrs et MDA est une expression de l'ingénierie des modèles parmi d'autres, les DSL ou MDE en sont d'autres. La démarche MD devient visible car elle porte ses fruits sur le terrain, en permettant notamment le découplage entre le monde métier et la technologie et en augmentant l'agilité.

P ! : L'un des soucis est de préserver la synchronisation bidirectionnelle entre le modèle et ce qui est généré. En MDA, le round trip sur des projets complexes est au mieux médiocre, au pire, quasi impossible à maintenir dans les cas extrêmes. La situation va-t-elle changer dans l'avenir ?

J.L.C. : Il reste bien sûr des problèmes à résoudre et le Gartner prévoit que la pleine puissance des concepts prônés par MDA ne s'exprimera qu'aux environs de 2010. Les outils vont continuer à s'améliorer, en augmentant notamment leur capacité d'adresser la totalité d'un projet complexe. Parmi les alternatives pour contourner cette problématique de la synchronisation bidirectionnelle, il y a celle choisie par Lyria pour son framework Leonardi, où le modèle, édifice central de l'application, est exécuté à la volée. L'indépendance vis-à-vis de la plate-forme cible est donc assurée sans générer de code. Le modèle peut alors évoluer librement sans que cette re-synchronisation soit nécessaire, puisque les changements sont interprétés dynamiquement par le moteur.

P ! : La profusion de Model Driven ne nuit-elle pas à sa compréhension ? Comment sensibiliser le développeur, l'architecte à cela ?

J.L.C. : La profusion des références au MD n'est qu'une expression de la richesse des concepts sous-jacents. Il s'agit de mettre en avant les principes de l'ingénierie des modèles (IDM) et cela peut s'exprimer de façon différente en fonction des contextes de mise en œuvre : outils de développement, de management, systèmes d'exploitation. Avec le MDE, cela dépasse même le cadre du logiciel.

P ! : Finalement, ne faut-il pas passer à l'étape suivante directement : les DSL ?

J.L.C. : Je ne crois pas que les DSL soient l'étape suivante. Ils constituent eux aussi une étape, l'un des moyens de mettre en œuvre l'IDM pour des tâches bien précises, d'aligner l'informatique avec les besoins de domaines spécifiques en bâtissant autour du modèle métier. S'ils accroissent l'agilité des applications spécifiques et permettent aux experts métier la validation au niveau du domaine, ils restent coûteux à borner fonctionnellement, et difficiles à mettre au point.

AVIS D'EXPERT

MDD est plutôt un concept

Questions à Grégory Weinbach *(Responsable pôle MDA, Objet Direct)*

Programmez ! : On entend beaucoup parler de MD comme le MDT, MDD ou encore MDA. Quel est aujourd'hui le MD le plus en vogue et pourquoi ?

Grégory Weinbach : MDA, MDD, MDT, MDE... il est difficile d'établir une hiérarchie de "popularité". MDD est plutôt un concept, finalement assez vague : on produit des modèles plutôt que du code, le code étant généré à partir de ces modèles. Leur contenu n'est pas précisé.

En ce sens, beaucoup de gens pratiquent ou ont pratiqué ! le MDD : générer des scripts SQL à partir d'un MCD, utiliser un AGL Cobol à la Pacbase ou faire de la génération de code à partir de schémas XML... c'est du MDD ! Et finalement, MDA c'est du MDD "normalisé". Il suppose un enchaînement de transformations types qui vont d'un modèle, souvent exprimé en UML, jusqu'au code. Les modèles intermédiaires sont identifiés, PIM et PSM, même si leur contenu précis n'est pas défini par la norme. En particulier, le PIM est une notion mal comprise et donc, sujette à interprétation : il est très rarement, vraiment "platform independent" et joue mal son rôle de pivot. Les outils permettant de faire de la transformation de modèle se disent "MDA compliant" mais ils ne proposent jamais sur étagère un ensemble de méta-modèles et de transformateurs/générateurs permettant effectivement de mettre en place un processus MDA. Le MDE est la discipline de l'ingénierie logicielle qui regroupe tout ce qui a trait au MD. Dès qu'on parle de Model Driven on est dans le MDE. Dès qu'on à affaire à une transformation d'un modèle en code, c'est du MDD, si le processus et/ou l'outillage se réfère(nt) à la norme on est dans le MDA.

P ! : L'un des soucis est de préserver la synchronisation bidirectionnelle entre le modèle et ce qui est généré. En MDA, le round trip sur des projets complexes est au mieux médiocre, au pire, quasi impossible à maintenir dans les cas extrêmes. La situation va-t-elle changer dans l'avenir ?

G.W. : Une des premières (et légitimes) questions posées par les gens à qui on présente un outillage MDA est : que se passe-t-il si je modifie le code généré ? Si certains outils MDA excellent dans le round-trip PSM <-> code, c'est souvent au prix d'un code illisible : des commentaires "balises" parsèment le code et ne doivent en aucun cas être modifiés sous peine de perdre la belle mécanique, "génération, modification manuelle, regénération...".

Dans tous les cas, dans une approche MDA, le source véritable étant le PIM modélisé en UML, ce round-trip n'a d'intérêt que s'il permet aussi la mise à jour du PIM à partir des modifications du PSM (PIM <-> PSM). Ce n'est presque jamais le cas, même quand l'outil le permet, ce qui est rare, peu de gens prennent la peine d'écrire les transformations inverses qui sont souvent très complexes. Je pense qu'il faut privilégier une approche pragmatique sans round-trip.

P ! : Et la notion de qualité dans tout cela ?

G.W. : C'est un problème difficile. Intrinsèquement, la qualité du code produit dans une approche MD est meilleure : conformité aux normes, à l'architecture, aux bonne pratiques, homogénéité. C'est d'abord la plate-forme de génération qu'il faut tester de manière fine. Il reste les problèmes de conformité aux exigences fonctionnelles. Les systèmes d'automatisation des tests fonctionnels apportent de bonnes réponses.

Pour aller plus loin : http://mdblog.fr/

AVIS D'EXPERT

"Si la démarche MDA est respectée, il n'y a aucune raison valable de désynchronisation"

Nous avons posé quelques questions à Guy Cartigny
(Responsable technique développement, Compuware)

Programmez : la synchronisation reste un problème majeur dans le développement MD. Qu'en pensez-vous ?

Guy Cartigny : Vrai et faux : Si la démarche MDA est respectée, il n'y a aucune raison valable de désynchronisation. Ce n'est que la conséquence de solutions incomplètes et de développeurs qui ne sont pas prêts à abandonner leurs lignes de code.

Il existe aujourd'hui des solutions qui garantissent la synchronisation bidirectionnelle.

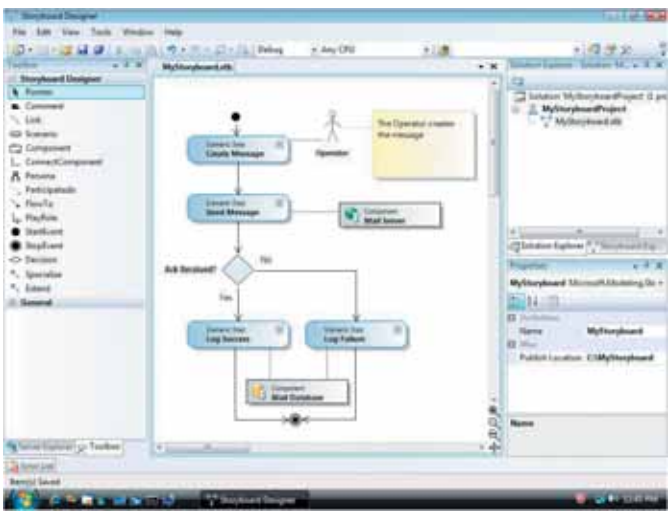
P ! : La profusion de Model Driven ne nuit-elle pas à sa compréhension ? Comment sensibiliser le développeur, l'architecte à cela ?

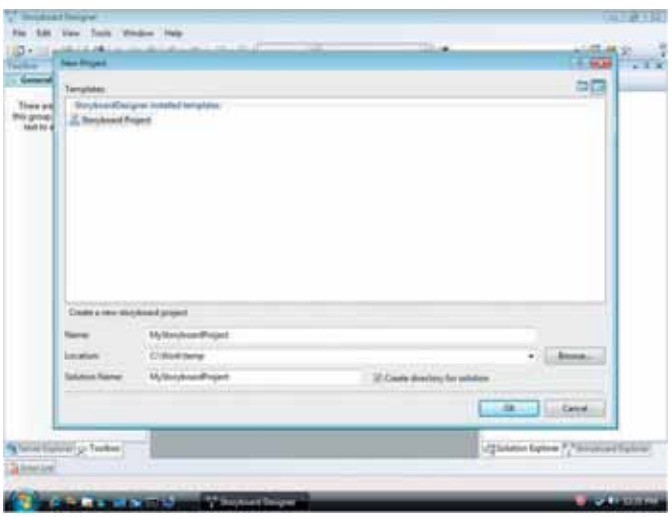
G.C. : Catégoriquement : Oui, elle y nuit ! Et il sera toujours très difficile de sensibiliser les développeurs. L'approche par les modèles, et quel que soit le nom utilisé: MDA, MDD, MDE, DSL, est synonyme de la fin du développeur. A terme, celui-ci devra synthétiser l'expression des besoins utilisateurs sous forme de modèle, puis générer l'application, effectuer les tests de performance et en assurer la maintenance fonctionnelle! Ce qui, reconnaissons le, n'est pas forcément dans ses aspirations. L'architecte a comme intérêt, par rapport à l'ingénierie des modèles, le fait de n'avoir qu'à concevoir, valider la ou les architectures optimales en fonction de la nature des applications qui seront développées. Aujourd'hui, une bonne partie de son temps est consacrée à expliquer l'utilisation de l'architecture retenue, à en vérifier le respect, voire à aider à la bonne fin du projet, et là encore, ce n'est pas forcement dans ses aspirations !

Introduction aux Domain Specific Language

Un langage peut être défini comme une grammaire, ainsi un DSL permet d'élaborer cette grammaire. C'est une programmation orientée langage (LOP) ou méta-programmation. Un DSL est conçu pour répondre à une problématique spécifique, contrairement au langage de modélisation UML qui adresse une généralité de problématiques. Une expression régulière est un exemple de DSL textuel. On peut également concevoir des DSL graphiques comme le diagramme de classes dans l'environnement de développement Visual Studio 2005 ou 2008.

Les règles fonctionnelles et techniques évoluent avec le temps. Nous sommes passés de langages procéduraux à des langages orientés objet. Chaque année, de nouvelles réglementations apparaissent et les marchés évoluent. L'entreprise se doit d'être réactive et faire preuve d'agilité et de souplesse. Idéalement, l'entreprise capitaliserait son savoir, son expérience et ses problématiques sous la forme d'un langage spécifique au domaine adressé. Nous pourrions avoir un jeu de symboles permettant de modéliser, par exemple dans le domaine financier une négociation, entre le vendeur d'un titre, une banque et le client qui souhaite acquérir le titre. On peut imaginer ajouter un symbole représentant la commission de la banque ainsi que toute la logique associée. Au final, après avoir défini un premier jeu de symboles compréhensibles par l'expert métier, ce même expert pourrait modéliser plus facilement un processus, de la même manière qu'il expliquerait à un développeur sur un tableau son métier. Une fois la logique la plus fidèlement modélisée et validée par l'expert métier et le développeur, il faudrait pouvoir générer à la demande des artefacts de toutes sortes : couche métier, client lourd, client léger, documentation, etc. Tout ce tableau décrit est possible grâce à la méta-programmation, c'est-à-dire le moyen qui permet de construire la grammaire d'un langage.





Théorie avec Martin Fowler

Martin Fowler pour décrire un DSL commence par décrire un framework et ses API puis un DSL (graphique ou non) au dessus d'une API pour le rendre plus facile à manipuler. L'alternative la plus commune est de définir un DSL en premier lieu en commençant à décrire un scénario. Si le langage à utiliser est contenu dans le DSL, il est intéressant de le faire en collaboration avec un expert du domaine en question pour utiliser le DSL comme vecteur de communication et combler ainsi le gap entre le développeur et l'expert métier.

DSL, le designer de designer

Les DomainSpecific Language Tools de Microsoft vous permettent de créer vos propres " designer " afin de concevoir dans Visual Studio vos propres modèles. L'objectif ici est d'exprimer une problématique métier ou technique dans Visual Studio, avec un langage qui sera compris par vos clients et vous-même, dans l'optique de générer le squelette de code de vos applications. En effet, après avoir élaboré et déployé votre DSL dans Visual Studio, vous pourrez modéliser votre application sur la base des éléments que vous avez préparés. Le modèle est sérialisé en XML dans un fichier avec l'extension DSL lors de son élaboration. Vous pourrez choisir une extension " maison " qui sera utilisé sur les postes où vous déploierez votre DSL. On peut y ajouter des modèles GAT de génération de code qui peuvent être appelés depuis votre modèle DSL.

L'investissement dans la construction d'un DSL est profitable s'il est intégré dans une démarche globale d'industrialisation des développements. Si le métier est connu et spécialisé, il est intéressant d'investir dans un langage " maison " vous permettant de modéliser vos applications et d'y associer de la génération de code. Au-delà du contexte métier, vous pouvez utiliser les DSL également pour des modélisations orientées technique (tels les diagrammes Archi-

BLUAGE™

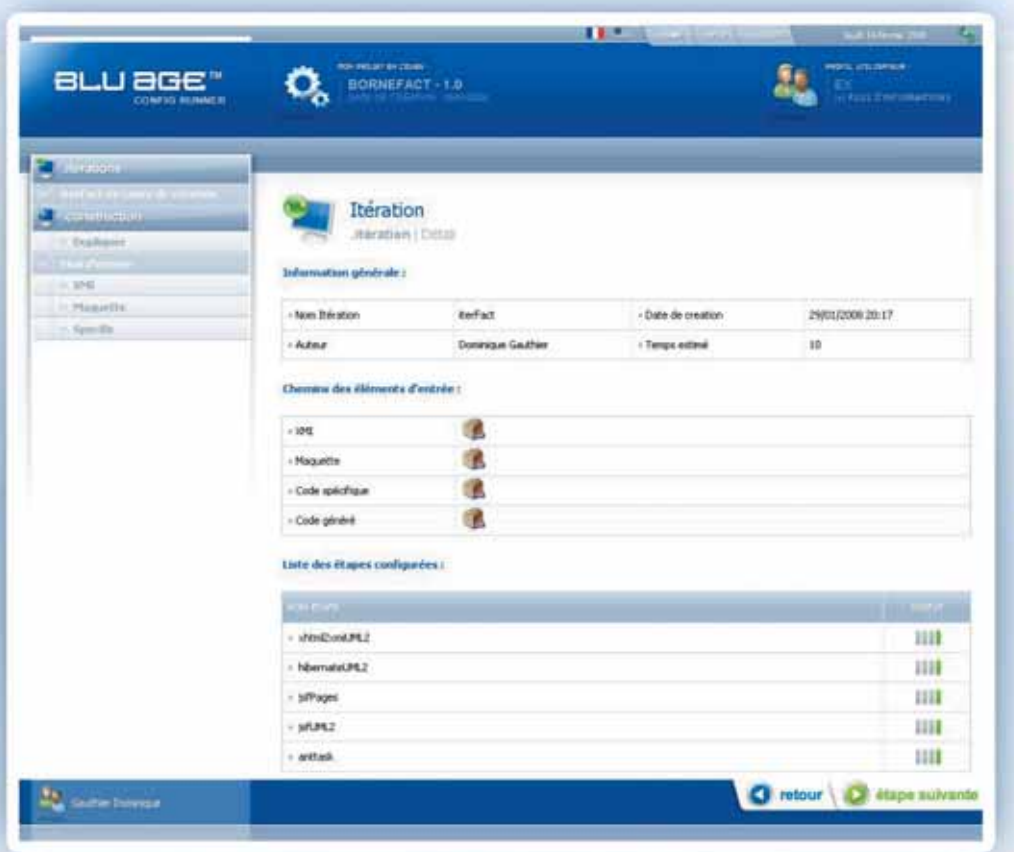
J2EE .NET APPLICATION GENERATOR

You Design, We Generate

www.bluage.com

Générateur MDD d'applications JAVA EE et .NET

BLU AGE™ est une suite logicielle MDA qui permet une transformation automatisée à 100% des diagrammes UML 2.0 en applications métiers Java EE et .Net. Grâce à BLU AGE™, les consultants analystes réalisent des applications SOA sans connaissance d'outils de développement, et réduisent de 30 à 50% les coûts et les délais de livraison de leurs projets.



Capitalisez sur le métier, pas sur la technique !

Sur la base de vos processus métiers modélisés en UML et des IHM maquettées en XHTML, BLU AGE™ génère automatiquement vos applications métiers dans le framework de votre choix. Un changement de framework ou une évolution fonctionnelle ? Vous modifiez les paramètres de génération (BSP) et/ou votre modélisation UML, et vous régénérez instantanément votre application !

Une solution ouverte, compatible avec les standards de l'Orienté Objet

Solution ouverte — aucun « run time » dans les applications générées - BLU AGE™ est proposé avec de nombreuses options de paramétrage préconfigurées (BSP - BLU AGE™ Shared Plugins), vous permettant de créer rapidement vos applications métiers dans des environnements professionnels (tels que ASPX, JSF, Struts, Spring, EJB, Hibernate, NHibernate…) pour les plateformes techniques telles que Websphere, Weblogic, IIS, Oracle, DB2. Vous utilisez des frameworks ou environnements de déploiement spécifiques ? Créez ou personnalisez vos propres BSP à l'aide des fonctionnalités de BLU AGE™ Software Factory (BSF), livrée nativement sous forme d'un Plugin Eclipse.

Travaillez suivant une méthode MDD, en mode itératif !

BLU AGE™ intègre Config-Runner, une interface web de gestion de projet, permettant aux consultants métiers (en charge de la modélisation UML) et aux architectes (en charge de la configuration des BSP) de travailler en parallèle et en mode itératif, jusqu'à la recette finale. Ces techniques accélèrent considérablement les délais de production des applications, tout en alignant les besoins fonctionnels avec les applications générées.

Des wébinars de démonstration de la solution BLU AGE™ sont disponibles en ligne.
Pour plus d'information, veuillez consulter notre site Web : http://www.bluage.com
Contactez-nous à l'adresse : **contact@bluage.com** ou par téléphone au : **01 56 05 88 00**

NETFECTIVE Creative Technology

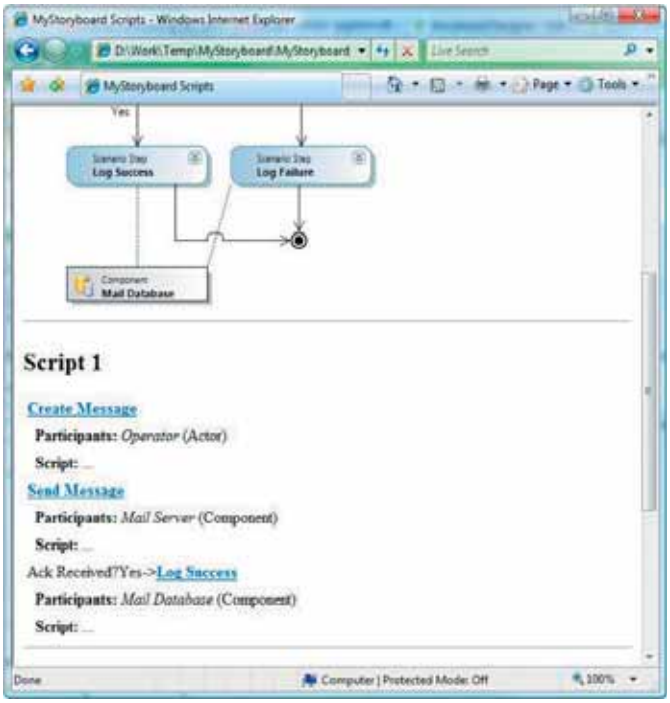
tectes proposés dans Visual Studio Team Suite). Les DSL Tools sont intégrés au SDK de Visual Studio 2005.

Un exemple de projet : StoryboardDesigner

Le projet StoryboardDesigner illustre comment utiliser les DSL Tools (déclinaison des DSL dans un environnement Microsoft) avec Visual Studio Shell (environnement de développement isolé et personnalisable depuis Visual Studio 2008). Il a été réalisé par la société sud-américaine Clarius Consulting ,connue pour son investissement dans l'extensibilité de Visual Studio et le développement d'usines logicielles.

L'objectif de ce projet est de définir un outil qui permet de construire un " storyboard " et de générer par la suite le scénario sous la forme de pages HTML avec les différents scripts possibles.

Pour cela, un Domain Model a été conçu. C'est le modèle de concepts décrit par le langage.



Pour ce DSL graphique, le Domain Model correspond en quelque sorte à la grammaire du langage et est représenté graphiquement par la surface de design.

Les briques de base sont les Domain Classes et les Domain Relationships. Les contraintes sont également définies pour s'assurer que le diagramme créé utilisant le langage est valide.

Enfin, il y a la possibilité de générer des artefacts en l'occurrence ici des pages HTML.

Bon à savoir, le designer DSL dans DSL Tools est lui-même un DSL, donc il est possible de construire sa propre méthode de génération de DSL. Entre tous ces éléments, des relations et cardinalités sont ajoutées. Elles permettent de préciser combien d'éléments peuvent être utilisés. Une relation peut être de type composition ou référence et peut contenir des attributs. Les éléments sont reliés à des représentations graphiques dans la partie droite du DSL : Diagram Elements.

Pour aller plus loin

Je vous invite pour ceux qui veulent aller plus loin dans les DSL à télécharger ce projet sur CodePlex : http://www.codeplex.com/storyboarddesigner et à visiter la communauté DSLFactory : http://www.dslfactory.org.

Sophiane Souane
Contact : contact@adiba.fr
Fondateur d'Adiba, société française qui propose du conseil et du service en informatique spécialisé sur la plate-forme Microsoft.

Les DSL, un standard dans 2 ans ?

Netfective, éditeur de BLU AGE (centré sur la notion de MDD) travaille activement sur la notion de MD et particulièrement de MDD. Nous avons posé quelques questions à **Imad Bernoussi,** directeur du marketing technique de l'éditeur.

Programmez : L'un des soucis est de préserver la synchronisation bidirectionnelle, notion de round trip, entre le modèle et ce qui est généré. A quoi sert le round trip quand on est capable de générer et déboguer en temps réel les modèles ?

Imad BERNOUSSI : En théorie, cela semble être une démarche intéressante. Mais dans des projets complexes cela est fastidieux à mettre en place pour la simple raison qu'il y a une multitude de langages d'implémentations. Il manque un élément pivot qui peut être un langage ou un modèle. La première solution nous permet d'apprécier à sa juste valeur la complémentarité entre MDA et SOA. En effet, la synchronisation bidirectionnelle entre le modèle et le XML généré est plus facile à assurer que celle entre un modèle et un langage d'implémentation. En pratique, il suffit de spécifier un schéma XML pivot et de définir la façon dont il est représenté en UML au travers d'un profil. Quant à l'approche dirigée par les modèles, elle se base sur la notion de transformation. Basée sur le rétro Engineering, elle permet le passage vers un modèle pivot qui sera utilisé ultérieurement pour la génération automatique de code ou d'applications.

P ! : Finalement, ne faut-il pas passer à l'étape suivante directement : les DSL ?

I. B. : De quoi avons-nous besoin concrètement ? De capitaliser sur les expertises métiers, de bâtir des applications utilisant un langage commun et en s'appuyant sur des briques de ces connaissances métier. A première vue, UML est suffisant pour exprimer les besoins fonctionnels des applications métiers. Mais, dans ses versions 1.5 et 2.0, ce dernier reste un standard assez généraliste. Pour l'utiliser de façon précise et spécialisée, nous devons passer par des extensions (profils UML). Les grands acteurs semblent prendre conscience de cette réalité. La déclinaison de MDE chez Microsoft s'appelle "Microsoft Software Factories". Elle est fondée essentiellement sur des langages de domaines de petite taille, facilement manipulables, transformables, etc. En un mot, ces DSL sont la base de l'automatisation MDE chez Microsoft. Quant à la stratégie MDE d'IBM, elle ne considère UML que comme un standard ouvert parmi tant d'autres. Le point crucial, c'est la possibilité de traitement automatique de modèles s'appuyant sur des standards précis autour de la plate-forme Eclipse. Donc, on voit, qu'à terme, on se dirige vers des plates-formes ouvertes de transformation. Mais on n'est pas encore dans une utilisation massive des DSL. Pour que cela devienne un standard industriel, il faudra attendre au minimum 24 mois...

Concilier Spring et MDA

Cet article introduit la mise en oeuvre du framework Spring à l'aide d'une approche dirigée par les modèles basée sur le générateur de code Acceleo. A partir d'un exemple, nous allons détailler la manière d'industrialiser la réalisation d'un composant métier d'accès aux données.

Nous utiliserons le formalisme UML pour modéliser l'exemple à l'aide de diagrammes de classes. Vous avez probablement entendu parler d'UML ou encore du MDA (Model Driven Architecture). Ces deux acronymes correspondent à des standards définis par l'OMG (Object Management Group) et des technologies dénommées IDM, Ingénierie Dirigée par les Modèles. L'IDM définit tout un ensemble de spécifications et de techniques permettant de définir, manipuler et transformer des méta-modèles ou modèles. Le but de cet article n'étant pas de vous présenter l'état de l'art de l'IDM, nous ne détaillerons donc pas plus ces concepts. Retenez néanmoins que l'approche est similaire à celle utilisée dans le bâtiment ou l'industrie, approche se traduisant par la réalisation de plans ou de spécifications détaillées avant de produire (par exemple : un plan de maison). L'idée est d'appliquer cette pratique au processus de construction du logiciel, les "plans" devenant dans ce contexte des "modèles".

UML et MDA

UML est le méta-modèle utilisé pour représenter des systèmes en se fondant sur différents types de diagrammes comme les diagrammes de classes, d'états, de séquences, ... Quant au MDA, il définit deux grandes notions : PIM et PSM. L'acronyme PIM signifie Platform Idependent Model et correspond à un modèle abstrait indépendant de toutes technologies. Dans notre exemple il s'agit de la représentation logique de notre application. L'acronyme PSM quant à lui signifie Platform Specific Model et fait référence à un modèle précis dépendant d'une technologie. Dans notre exemple, il s'agit de la représentation d'une partie d'une application constituée de classes, interfaces java et de fichiers de configuration de Spring. Ainsi notre but est d'effectuer une transformation " model-to-text " à l'aide d'Acceleo, afin de traduire notre PIM modélisé en UML en code source java et fichiers de configuration Spring.

Introduction à Acceleo

Notre choix s'est portée sur la solution libre Acceleo (http://www.acceleo.org), un projet hébergé par OW2 et basé sur Eclipse. Il tire parti du projet EMF qui offre les mécanismes de manipulation de métamodèles et modèles. Acceleo est un générateur de code souple, modulaire, et très simple d'utilisation. Il permet de définir des générateurs à l'aide d'un langage de script particulièrement adapté à cet usage. Le script suivant permet de générer un fichier java pour chacune des classes du modèle :

```
<%
metamodel http://www.eclipse.org/uml2/2.1.0/UML
import fr.benois.acceleo.MyService
%>
<%script type="Class" name="filePath"%>
<%name%>.java
<%script type="Class" name="generateClass" file="<%filePath%>"%>
public class <%name%>{
  <%for (attribute) {%>
  /**
   * Attribute <%name%>
   * <%ownedComment.body%>
   */
  private <%type.name%> <%name.toL1Case()%>;
  <%}%>
}
```

La directive " metamodel " permet de spécifier le métamodèle auquel est conforme le modèle utilisé en entrée. En l'occurence, nous utilisons le métamodèle UML2.1. Le mot clé " import " joue le même rôle qu'en java, afin de permettre l'importation de classes java ou d'autres scripts. Cette approche permet l'implémentation des méthodes utilitaires et facilite la réutilisation. On notera que la suite Acceleo fournit un ensemble d'éditeurs graphiques, d'outil d'exécution et de débogage complètement intégrés à Eclipse. Ainsi l'édition de scripts est aussi simple que celle de fichiers java.

Installation d'Acceleo

Une archive " bundle 3.3 " d'Acceleo contenant les dernières versions d'Eclipse, d'Acceleo ainsi que le modeleur UML Topcased est disponible sur : http://acceleo.org/pages/telechargement-acceleo-2-2-0/fr

Modéliser en UML

A l'aide de diagrammes de classes, nous définissons un composant permettant d'accéder à des informations relatives à des articles stockés dans une base de données. Afin de différencier les concepts de service métier, de DAO (Data Acces Object) et d'entités métiers, nous allons utiliser un *profile* UML. Ce dernier correspond à un mécanisme d'extension d'UML permettant de définir ses propres concepts à l'aide de stéréotypes. Nous pouvons alors disposer des stéréotypes " Service ", " Dao ", et " Entity " applicables sur toutes classes modélisées. Nous avons également défini les stéréotypes " Remote " et " Transactionnal " applicables quant à eux aux opérations des services métiers.
L'image suivante illustre l'application du stéréotype " Spring::Transactional " sur l'opération *ajouterArticle* (Fig.1).

Fig.1

L'image de la figure 2 illustre la valorisation des propriétés " isolation " et " propagation " du stéréotype précédemment appliqué.
Nous allons maintenant créer une entité métier *Article* ainsi qu'une classe d'accès aux données *ArticlesDao* et un service métier *ArticlesServices*.

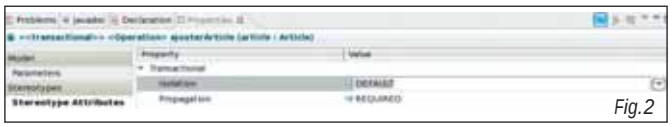


Fig.2

Notons que nous avons spécifié un aspect transactionnel pour les opérations : *ajouterArticle, modifierArticle* et *supprimerArticle* (Fig.3). Nous ajoutons enfin une dépendance entre le Service et le Dao afin de signifier que nous souhaitons injecter une instance d'ArticlesDao dans la classe *ArticlesServices* (Fig.4).

Fig.3

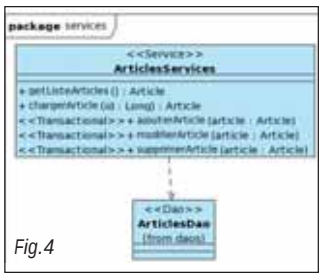


Fig.4


Fig.5


Fig.6

Génération du code

Nous devons maintenant définir une chaîne de génération constituée du modèle source, des scripts de génération à utiliser et des répertoires de destination. A cet effet, nous allons utiliser l'assistant d'Acceleo permettant de sélectionner une chaîne pré-paramétrée pour Spring. L'image de la figure 5 illustre la sélection du module de génération " Spring Business Layer Generator ". Après avoir répondu aux questions de l'assistant, un fichier portant l'extension .chain est créé. Ce dernier peut être exécuté en cliquant simplement sur " Launch " dans le menu contextuel pour lancer le processus de génération. L'image ci-contre présente l'ensemble des fichiers générés. Nous y trouvons les interfaces et implémentations du service métier et des Dao ainsi que l'entité métier, les fichiers de mapping Hibernate et le test unitaire JUnit validant le bon fonctionnement des méthodes CRUD *(Create-Retrieve-Update-Delete).* Les fichiers Spring permettant de gérer l'injection de dépendances sont générés quant à eux dans le répertoire META-INF/spring du projet (Fig.6). Le code ci-dessous illustre l'implémentation générée du service métier :

```
public class ArticlesServicesImpl implements IArticlesServices {
...
 private IArticlesDao articlesDao = null;
 public void setArticlesDao(IArticlesDao articlesDao) {
  this.articlesDao = articlesDao;
 }
 public IArticlesDao getArticlesDao() {
  return this.articlesDao;
 }
 public Article chargerArticle(Long id) throws TechnicalException{
  // Start of user code of Article chargerArticle(Long id)
    // TODO implement chargerArticle
   throw new UnsupportedOperationException("Not yet implemented");
  // End of user code
 }
...
```

On notera l'usage d'une section délimitée par *" Start of user code "* et *" End of user code "* laissée à l'usage des développeurs pour ajouter du code manuellement, préservé lors des prochaines générations. Le code suivant décrit la mise en oeuvre d'un tel bloc :

```
// Start of user code of Article chargerArticle(Long id)
 return articlesDao.find(id);
// End of user code
```

L'extrait suivant illustre une partie de la configuration Spring générée et permettant d'injecter le Dao dans le service métier. Le paramétrage du gestionnaire de transactions et des comportements transactionnels de Spring est également précisé pour les méthodes stéréotypées " Transactional ". Nous aurions pu également décrire l'usage du stéréotype " Remote " afin de générer le bloc de configuration Spring permettant d'exposer certaines méthodes en tant que WebService.

```
<bean id="articlesServices" class="fr.benois.articles.services.impl.Articles
ServicesImpl">
    <property name="articlesDao" ref="articlesDao" />
</bean>

<tx:advice id="txAdviceArticlesServices" transaction-manager="transaction
Manager">
    <tx:attributes>
        <tx:method name="ajouterArticle" />
        <tx:method name="modifierArticle" />
        <tx:method name="supprimerArticle" />
        <tx:method name="*" read-only="true" />
    </tx:attributes>
</tx:advice>
<aop:config>
    <aop:advisor pointcut="execution(* *..ArticlesServices.*(..))"
      advice-ref="txAdviceArticlesServices" />
</aop:config>
```

Conclusion

Dans cet article, après avoir énoncé quelques principes d'IDM, nous avons détaillé comment industrialiser la mise en oeuvre de composants métiers tirant partie du framework Spring. Pour ce faire nous avons utilisé l'outil de modélisation UML TopCased, couplé au générateur de code Acceleo. Il est a noter que nous aurions pu choisir une approche DSL (Domain Specific Language) afin de créer notre propre méta-modèle dans le but de définir nos propres concepts (" Service ", " Dao ", ...) Nous aurions pu ainsi réaliser des générateurs basés sur ce méta-modèle et non celui d'UML. Les sources de l'exemple sont disponible à cette adresse : http://www.benois.fr. Relecture technique assurée par **Thierry Templier.**

Jérôme BENOIS - *jerome@benois.fr - Responsable Technique – Argia-Engineering*

L'INFORMATION du DÉCIDEUR

Choisir, déployer, exploiter les softwares

Lisez le seul magazine offrant aux responsables informatiques une information et des témoignages focalisés sur le logiciel en entreprise.

Egalement au sommaire du numéro :

- *Lotus, au-delà de la messagerie*
- *Avis d'expert : ERP, génération 2.0*
- *Déploiement Vista : les outils d'administration*
- *La Virtualisation pour consolider ses serveurs*
- *Le tableau de bord de Windev 12*
- *Mener à bien son projet de développement*
- *Adopter la SOA : quels bénéfices pour l'entreprise ?*

L'actualité au quotidien :
• Sécurité • Projets et développement • Administration • Progiciels

Les Cas Clients

Prochainement : **Vidéos** *(Actualité et Cas Clients)*

N°1 - Février/Mars 2008
Chez votre marchand de journaux
Abonnez-vous (coupon ci-dessous)

www.solutions-logiciels.com

☐ **OUI**, *je m'abonne* (écrire en lettres capitales)

Envoyer par la poste à : Solutions-Logiciels, service Diffusion, 22 rue rené Boulanger, 75472 PARIS - ou par fax : 01 55 56 70 20

1 an : 25€ *au lieu de 40€, prix au numéro* *(Tarif France métropolitaine)* **- Autres destinations :** *CEE et Suisse : 30€ - Algérie, Maroc, Tunisie : 33€ , Canada : 39,50€ - Dom : 38€ Tom : 50€*

8 numéros. Prochaines parutions : N°2 Avril/mai - N°3 Juin/Jul/aout - N°4 Septembre - N°5 Octobre - N°6 Novembre - N°7 Décembre/Janvier - N°8 Février 2009 - N°9 Mars 2009

☐ M. ☐ Mme ☐ Mlle Société .....

Titre : ..... Fonction : ☐ Directeur informatique ☐ Responsable informatique ☐ Autre .....

NOM ..... Prénom .....

N° ..... rue .....

Complément .....

Code postal : ..... Ville .....

Adresse mail .....

☐ *Je joins mon réglement par chèque à l'ordre de SOLUTIONS LOGICIELS* ☐ *Je souhaite régler à réception de facture*

Editeur : K-Now sarl au cap de 8000€ siège social : 6 rue Bezout, 75014 Paris

PROG 106

Le quotidien des développeurs 100% Open Source

S'il y a 2 ans, on pouvait encore voir de la réticence ici ou là, désormais, l'open source fait partie du paysage informatique du développeur, des entreprises, parfois de l'utilisateur. Le débat pour ou contre le logiciel ouvert, pour ou contre le propriétaire / commercial, n'a plus lieu d'être, c'est un clivage dépassé. Désormais on pense pragmatisme, pratique. Que l'on soit en environnement mixte ou 100 % ouvert, un nouveau modèle informatique est né dans lequel, chacun peut trouver sa place. Un éditeur doit gagner de l'argent et les éditeurs libres l'ont bien compris soit en passant par du service, soit par une politique de double licence. Et le rachat de MySQL par Sun démontre aussi la pertinence de l'open source. Le modèle économique, majoritairement tourné vers les services, ou en double licence, ne constitue pas un handicap. De plus, l'émergence de nouveaux éditeurs ouverts, ou les rachats par des géants de l'industrie, crédibilise l'offre auprès des entreprises. Et le fait que SourceForge propose un marché géant des services, est une autre preuve de la bonne santé du monde ouvert ! Si en plus, on rajoute à cela la virtualisation, il y a de quoi aimer définitivement notre petit tux.

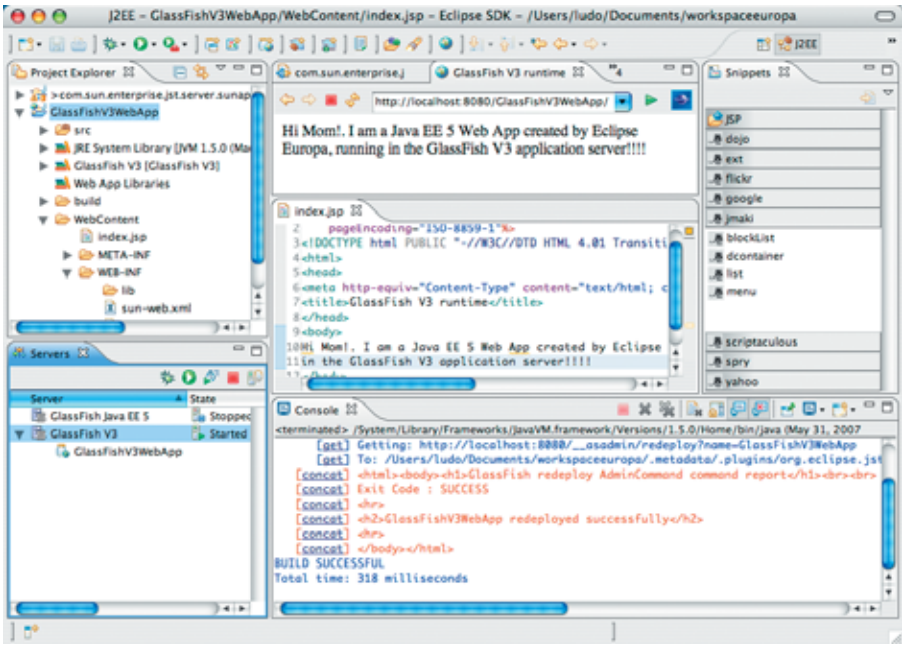
Dans ce grand dossier nous allons nous plonger dans les licences ouvertes. Sujet souvent oublié par le développeur, il devient critique pour les entreprises et éditeurs. Avec les multiplications des licences, les conflits potentiels entre les licences, les risques légaux existent bel et bien. Le respect de plus en plus scrupuleux de ces licences montre le souci des entreprises et développeurs de mieux intégrer et utiliser les outils ouverts. Mais nous verrons aussi que ce n'est pas un hasard si le recours à des avocats spécialisées et la prudence se généralisent. Autre thème fort du dossier, le poste 100 % Open Source du développeur ! Oui ce n'est pas une utopie. Bien au contraire… et il n'est plus l'apanage des seuls adeptes du libre de la première heure. Après le serveur, le poste de travail, qu'il soit bureautique ou de développement se convertit de plus en plus à l'open source. Mais là, deux cas de figures, complémentaires : le mixage commercial – ouvert, ou tout ouvert. Mais nous verrons aussi que selon la plateforme utilisée, les contraintes obligent à des choix. Enfin, 3e temps fort de notre grand dossier : au cœur des projets open source ! Vous verrez comment s'organisent, se gèrent, se développent quelques-uns des projets ouverts les plus connus, les plus utilisés. Nous plongerons au cœur d'un projet Eclipse de modélisation : EMF Compare, des environnements graphiques Gnome et KDE, ou encore de Ubuntu, OpenSUSE, PHP ! Pour terminer, Linagora nous a ouvert ses portes pour nous montrer comment une société de services libres travaille au quotidien ! Comme vous l'aurez remarqué au sommaire de ce numéro, l'open source est très largement présent : MDA en Open Source avec Spring et Acceleo, KDE 4.0 et ses nouveautés pour les développeurs. Que du bonheur de développeur ! *Bonne lecture… libre.*

François Tonic

Le poste du développeur libre

Oui, c'est possible, et dans de nombreuses sociétés open source, ils le sont déjà et depuis des années. Mais cette tendance se vérifie aussi maintenant chez les développeurs en entreprise, à la maison, à l'école ! L'autre tendance, qui existe depuis longtemps, c'est le poste de travail mixte : ouvert avec du commercial. Alors vive le LAMP et le WAMP !

Linagora est 100 % ouvert, Uperto aussi, Talend pareillement, sans oublier les éditeurs open source, les distributions Linux. Mais on ne peut pas tout passer en open source. Car pour certains projets, difficile de faire l'impasse sur certains outils, ou tout simplement, impossible de se passer de MacOS X ou de Windows, bien souvent la nature même du projet influence ses choix.



Les limites du 100 % Open Source

Le choix technologique va influencer l'outillage de développement. Si le projet fonctionne sous Windows, il faudra bien tester l'application Windows même si l'application est en Java. L'environnement de production est donc une contrainte forte pour le développeur. Certaines technologies, environnements serveurs, ou tout simplement les choix du client, imposent d'utiliser des outils propriétaires et/ou commerciaux. En .Net, si Mono existe, il ne permet pas de tout faire et surtout, les dernières évolutions ne sont pas prises en compte, pareillement pour un développeur Silverlight. Windows et les outils Microsoft demeurent incontournables. D'une manière générale, tout projet centré Windows nécessite Windows sur le poste de travail du développeur, même si on peut passer par de la virtualisation sur son Linux, mais est-ce assez productif ? Un élément a ne jamais oublier : TOUT DEPEND DE LA PLATE-FORME CIBLE !
Autre point délicat, la migration d'un poste propriétaire à un poste libre. Car migrer du code n'est pas toujours chose évidente ! Soyez très prudent sur ce point. Procédez à des tests préalables que ce soit pour le code, les fichiers projets, les modèles UML, MDA, etc.

Le mixte fonctionne très bien aussi

De très nombreux outils open source de développement fonctionnent sous Windows (et MacOS X), sans aucun problème. Les récents accords entre Microsoft, JBoss, Novell, Zend, Talend… permettent aujourd'hui de disposer d'une bonne optimisation du code sous plate-forme Windows Server. MySQL propose un plug-in Visual Studio, facilitant le travail du développeur. Et n'oublions pas que de nombreux développeurs PHP travaillent sous Windows, mais déploient sous Linux / Apache… Certains outils ne sont disponibles, par contre, que sous Windows, ce qui peut obliger le développeur à installer le système Microsoft. WAMP, même s'il est moins connu que LAMP, fonctionne lui aussi très bien. Un développeur habitué à VB, Visual Studio sera plus sensible à un IDE totalement intégré qu'à utiliser plusieurs outils non intégrés. En développement Java, hormis des outils très spécifiques dans les tests, le développement distribué, etc., Linux est tout aussi viable et puissant que Windows. Encore une fois, cela dépend de votre affinité avec le système, les outils. L'important est que le développeur soit à l'aise. Il ne faut pas imposer un système, un outil si le développeur n'est pas en phase avec. C'est contre- productif.

Quelle distribution ?

Sur ce point, il n'y a pas de règles. C'est à vous de tester et de voir quelle distribution répond le mieux à vos envies, vos attentes. Evitez tout de même d'installer des distributions trop

récentes ou exotiques, mieux vaut rester sur les valeurs sûres du marché : OpenSuSe, Fedora, Ubuntu / Kubuntu, Mandriva, etc. Installez uniquement des branches stables du système et évitez donc toutes les versions bêtas, en développement, expérimentales. En production de code, c'est toujours un aléa désagréable. Si vous voulez vraiment tester ces versions non stabilisées, ayez un autre poste ou passez par de la virtualisation. Même réflexe à avoir sur les outils : installez uniquement les versions stabilisées. La virtualisation est un formidable outil pour le développeur que ce soit sous Linux ou Windows (attention : machine puissante requise). Sous Linux, il existe plusieurs outils virtuels fonctionnant parfaitement comme Qemu ou KVM. Le fait d'utiliser des outils répondant à des standards peut aussi rassurer le développeur sur l'interoperabilité, mais que cela ne fasse pas oublier le test, le test et encore le test. Car même dans une application Java, on a des différences entre une JVM sous Windows et sous Linux. Donc prudence. De nombreux développeurs Python ou Ruby utilise un simple éditeur et non des IDE graphiques comme ceux de Microsoft, CodeGear ou Zend. Par contre, un développeur habitué à développer sous Windows ou MacOS X aura sans doute plus tendance à chercher un environnement totalement intégré, ce dont on ne dispose pas en open source. Pour faire de l'Ajax, du Web 2, du Flex etc. il ne sera pas difficile de trouver les outils nécessaires, même s'il n'existe pas d'équivalent aux outils Adobe comme Photoshop ou Dreamweaver.

Le 100 % open source de Talend mais ouvert à tout !

Programmez!: En tant que société Open Source, " imposez-vous " à vos collaborateurs l'utilisation exclusivement de logiciels Open Source ?

Cédric Carbone (C.T.O. Talend) : je pense qu'il y a une alternative Open Source pertinente à tout logiciel propriétaire, mais il faut sélectionner un logiciel avec pragmatisme, parce qu'il correspond le mieux à nos attentes et non pas uniquement parce qu'il est Open Source !

P!: Quel est le système d'exploitation sélectionné pour les ordinateurs de l'équipe technique ?

C. C. : A Talend, on a le choix de son OS! On préfère que nos ingénieurs disposent du système qui correspond le mieux à leurs besoins et leurs connaissances. La majorité des personnes de la R&D Française est maintenant sous Linux (Debian/Ubuntu

OpenSUSE). L'arrivée de Windows Vista a été un catalyseur à l'adoption de Linux !

P !: Qu'en est-il des logiciels que vous utilisez ?

C.C. : Beaucoup d'Open Source comme Mozilla/Filezilla, OpenOffice, Eclipse, MySQL, Talend Open Studio ;)... On utilise cependant quelques solutions propriétaires comme Framemaker pour la documentation ou des sharewares pour faire des tutoriels flash (DemoBuilder).

Tableau général

	Faisabilité	Limites
Web 2	Oui (Ajax, Rails, Python, quelques éditeurs). De nombreux frameworks	Pas d'outils équivalents à Dreamweaver ou à la gamme Expression
.Net	Mono principalement, quelques outils disponibles. Outils pour développeurs disponibles (notamment sur le refactoring, Nant, nhibernate, etc.)	Support incomplet de .Net, IDE pas aussi puissant que Visual Studio
Java	Oui, pas de problèmes, tous les outils sont disponibles !	Hormis quelques outils pointus sur les tests, le tuning et les performances, peu ou pas de limites
Serveur d'applications	Pas de problème (Jonas, JBoss, Geronimo, Glassfish)	Aucune, bien au contraire !
C, C++	Oui sans aucun problème	Aucune
Silverlight 1.0 / 2.0	Aucune actuellement, hormis Moonlight	Moonlight non encore disponible
Flex / Flash	Oui, éditeurs disponibles (dont celui d'Adobe), clones disponibles de Flash	Certaines librairies sont payantes chez Adobe.
Développement web	Oui possible, nombreux petits outils et frameworks	Environnements intégrés non disponibles en open source, dépendra de la technologie voulue
SGBD	Oui, aucun problème, embarras du choix ! Nombreux outils (ETL, requetages, etc.) disponibles	Dépendra de la plate-forme cible, du SGBD.
Virtualisation	Oui, aucun problème, offre performante (Qemu, Xen, Kvm)	Aucune, les outils ouverts fonctionnent aussi sous Windows
SOA	Oui sans problème, offre en constante évolution (Eclipse, Iona, OW2, Sun, etc.)	Dépend des technologies et standards éventuels mais l'interopérabilité atténue les problèmes
Mobilité	Possible dans certains systèmes embarqués, assez peu d'outils ouverts hors de ces systèmes (voir Android), Eclipse assez présent	Tout dépendra de la plate-forme cible. Offre fragmentée

Double licence et licence d'utilisation

Dernier point non négligeable, n'oubliez pas de regarder la licence du ou des outils, frameworks, etc. mis en œuvre dans votre projet. D'autre part, dans certains cas, la licence d'utilisation interdit ou limite l'usage, par exemple pour des API, librairies. C'est le cas des API Google ou Qt de Trolltech (en double licence). Pour un usage professionnel / commercial, il faudra acquérir une licence. Bref, comme pour le poste de travail 100 % Open Source, le poste de développement possède une alternative fiable et puissante dans beaucoup de développements. Et l'outillage open source se complète peu à peu, comme avec le futur Jazz, technologie collaborative d'IBM Rational et Novell en rachetant récemment SiteScape. N'oubliez pas non plus le rôle et l'importance de la formation et du support. Il peut être nécessaire de se mettre à niveau en passant quelques jours dans une formation spécifique ou de compréhension de l'open source. Le support est un autre point sensible, surtout en entreprise. Aujourd'hui, il existe des assurances logiciels (voir Linagora) et des supports spécifiques développements open source (voir SourceForge, IBM pour Eclipse, Sun, etc.). Point à ne pas négliger. À vous de jouer !

François Tonic

Les outils du développeur

Les logiciels open source ne constituent plus seulement l'apanage des passionnés de l'informatique. Ils sont rentrés de plain-pied dans le milieu professionnel et notamment dans le domaine du développement informatique.

Cet article s'attache à vous présenter les logiciels open source régulièrement utilisés par les développeurs, en évoquant les avantages et inconvénients liés à leur utilisation. Nous verrons également au travers de cas pratiques si le développeur peut disposer ou non d'un poste complètement open source selon les différentes typologies de projets.

1 L'open source dans les projets

Avantages

L'utilisation de l'open source dans le développement de logiciels offre de réels avantages :

- **Productivité :** grâce à la réutilisation de briques open source déjà développées.
- **Fiabilité du code :** l'ouverture du code offre une transparence totale sur le fonctionnement de celui-ci.
- **Réactivité de la communauté :** les logiciels open source bénéficient très souvent d'importantes communautés, aptes à fournir rapidement les réponses aux éventuels problèmes que vous pourriez rencontrer.

Inconvénients

Pour autant, l'open source n'est pas la panacée du développement. Malgré les avantages précédemment évoqués, le concept présente quelques lacunes :

- **Abandon du projet :** au fil du temps, certains projets open source peuvent se voir délaissés par la communauté au profit de projets plus innovants technologiquement (la pérennité est un élément important pour utiliser un projet, ndlr).
- **Support diffus :** pas de hotline à appeler en cas de problème majeur et trouver des informations peut parfois s'avérer très fastidieux (voir par exemple de support officiel d'IBM sur Eclipse ou du côté de Sun et de la place de marché service de SourceForge, ndlr).
- **Coûts imprévus :** développer les fonctions manquantes du logiciel open source choisi peut parfois s'avérer coûteux, par exemple dans le cas d'un code de mauvaise qualité ou bien insuffisamment documenté.

2 L'atelier du développeur

De nos jours, les projets informatiques sont de plus en plus complexes. Le développeur doit donc disposer d'outils efficaces aptes à l'assister pleinement dans les différentes phases de son projet. Cette section propose de vous présenter quelques uns de ces outils.

Conception

Lors de la phase de conception, le développeur élabore les diagrammes UML pour générer le code source associé. Le monde open source dispose à cet effet d'outils de qualité :

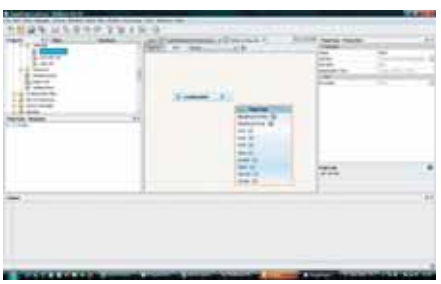
- **ArgoUML :** permet de créer les diagrammes de classes pour générer par la suite le code Java, C++ et PHP associé. Il est également utilisable en ingénierie inversée : c'est-à-dire qu'à partir des fichiers sources, ArgoUML génère le diagramme de classes. A noter que cet outil est directement utilisable en ligne, via Java Web Start.

- **StarUML :** permet également la génération de code pour de nombreux langages à partir d'un diagramme de classes. Il propose en outre de produire une documentation du projet très efficace. A noter que cet outil ne fonctionne que sur la plate-forme Win32.
- **Omondeo :** plug-in Eclipse. Le diagramme est réalisé dans Eclipse et le code des classes associées est directement disponible dans la vue développement. Ce plug-in offre au développeur un environnement unique pour le diagramming UML et le développement.
- **Du côté MDA,** on pourra citer Acceleo, OpenMDX, AndroMDA, etc.

Environnement

La construction représente la majeure partie du travail assigné au développeur. Il lui faut donc disposer d'un environnement complet qui bénéficie de fonctionnalités efficaces et ergonomiques, facilitant son travail au quotidien. L'open source offre aux développeurs parmi les meilleurs dans le domaine :

- **Eclipse :** est un environnement de développement intégré (EDI) multi-plate-forme disposant de nombreuses fonctionnalités : auto complétion du code, console, surlignage des erreurs, etc. Il dispose d'une bibliothèque d'extensions très riches dont la plupart open source. Pour ces raisons, Eclipse est largement utilisé dans le monde professionnel.



- **Netbeans :** est également un EDI multi-plate-forme d'origine Sun, disposant de fonctionnalités similaires à Eclipse et qui présente une finition assez remarquable.
- **Mono IDE :** est l'EDI proposé par la communauté open source pour le développement d'application .NET. Il supporte l'auto complétion du code et offre une gestion des classes intuitives.

Gestion de projet

Pour pouvoir travailler en équipe efficacement, les développeurs doivent s'appuyer sur un système de gestion de version du code source. Les deux principaux outils disponibles dans le monde open source sont :

- **CVS :** toujours très utilisé dans le monde professionnel, il permet de fusionner les différents changements opérés par les déve-

loppeurs, comparer les versions historiées, créer des branches pour le développement parallèle de sous-projets, …
- **Subversion :** proche de CVS, il propose toutefois des différences intéressantes : possibilité de déplacer, renommer un fichier tout en gardant son numéro de version, affectation d'un numéro de commit,…

Build

Pour construire rapidement et efficacement son application, le développeur dispose d'outils open source très complets :
- **Ant :** est utilisé pour construire l'application. En fonction de règles définies par le développeur, Ant s'occupe d'appeler les outils de compilation, d'édition des liens, de génération de documents. Il offre en outre la possibilité de nettoyer les fichiers intermédiaires générés lors de la construction.
- **Maven :** proposé par la fondation Apache, il s'utilise de la même façon qu'Ant, mais propose un fonctionnement plus simple.
- **Continuum :** associé à Maven pour les projets importants, il assure la construction de l'application, et l'exécution de tests unitaires à des dates prédéfinies. Continuum produit des rapports complets sur les résultats de la compilation et l'exécution des tests.

3 Les frameworks

Le développeur dispose aujourd'hui de très nombreux frameworks dans tous les domaines de développement. Vous n'avez que l'embarras du choix… Voici quelques exemples connus :
- **Struts:** Pour Java, il implémente le modèle MVC en étendant les servlets. Le code s'articule autour de classes Action pour le fonctionnement de l'application. Ces classes Action sont elles mêmes liées aux classes Form et FormAction pour la gestion de l'interface. Le lien entre ces différentes classes s'effectue au travers d'un fichier XML.
- **Java Server Faces (JSF) :** développé par le créateur de Struts, il vise à simplifier le développement en proposant un fonctionnement évènementiel (ex. clic sur bouton). JSF met également à disposition du développeur une bibliothèque de composants graphiques, simple à implémenter. Moins mature que Struts, JSF séduit toutefois de plus en plus les développeurs par sa simplicité et son efficacité.
- **Zend Framework :** Pour PHP, il propose une implémentation MVC. Il met en outre à disposition du développeur un certain nombre de fonctionnalités intéressantes : génération de fichiers PDF, lecture de flux RSS, création de logs, accès aux web services, etc.

- **Mono :** Mono est un projet ambitieux visant à recréer le framework .NET de Microsoft. Il fournit, entre autres, un compilateur C#, une machine virtuelle pour l'exécution d'application .NET, un serveur Web compatible ASP.NET.

4 Etude de cas

Comme nous venons de le voir, le monde open source dispose d'une large bibliothèque d'applications et frameworks à destination du développeur d'aujourd'hui. Cette richesse est-elle pour autant synonyme d'indépendance ? Le poste du développeur saurait-il être 100% open source dans tous les cas ? Pour tenter de répondre à ces questions, nous vous proposons d'étudier quelques configurations types en fonction de la nature du projet.

Cas du développement JEE

• L'environnement de développement : Eclipse

Eclipse est un outil très efficace. On dispose aujourd'hui de très nombreux plug-in pour tous les domaines de développements (gratuits ou payants).

Eclipse dispose d'environnement graphique de conception comme les RAD. Il est également possible d'accéder aux bases de données de façon intuitive grâce au projet DTP (Data Tools Platform) d'Eclipse Europa.

En outre, Eclipse intègre un outil de Business Intelligence : BIRT, qui s'il n'est pas aussi puissant que Business Object n'en reste pas moins une alternative open source suffisante dans de nombreux cas. Il existe aussi des distributions répondant à des profils de développements précis.

• Serveur d'application : JBoss

Le serveur d'application est l'élément central dans l'architecture d'une solution JEE. Il se doit donc d'être robuste et performant pour garantir une production de qualité. Le serveur JBoss répond à ces exigences et est aujourd'hui utilisé dans la production de services importants, comme par exemple le portail fiscal de la Direction générale des impôts. Il existe aussi d'autres serveurs : JOnas, Geronimo ou encore Glassfish (tous n'implémentent pas JEE 5).

• Frameworks : Struts et Spring

Le choix du framework est primordial pour une bonne réalisation du projet. Ce choix est fixé au début du projet, il est donc important d'étudier l'ensemble des exigences liées à l'application. L'open source dispose d'une multitude de frameworks mais tous ne se valent pas. Pour garantir le succès du projet, il est préférable d'utiliser un framework mature qui a fait ses preuves, comme Struts et Spring, plutôt qu'un framework innovant mais trop jeune.

• Tests de performance : jMeter

Les tests de performance peuvent être réalisés à l'aide des outils Badboy et jMeter. Badboy est à utiliser pour créer le script de tests simplement. Il suffit de surfer sur le site et badboy enregistre l'ensemble des requêtes émises. Pour réaliser les tests (injecteurs), le développeur/testeur peut utiliser jMeter et le script enregistré avec badboy. Malgré l'austérité de l'interface, jMeter constitue un outil très efficace pour ce genre d'opérations.

Et n'oubliez pas l'incontournable Junit pour le test unitaire.

Cas du développement Web Client/Serveur

• Le système d'exploitation : GNU Linux

Le développeur dispose d'un grand choix. Chaque distribution a ses différences, ses petits rajouts. C'est une affaire de choix personnel. Difficile de conseiller une distribution plus qu'une autre.

• Le serveur Web : Apache

Apache est aujourd'hui le serveur HTTP le plus utilisé avec une part de marché tournant aux alentours de 55%. Serveur de référence. Il s'intègre dans le concept LAMP (Linux Apache MySQL PHP)

• **Le langage web : PHP**

Alliant simplicité et efficacité, PHP est un des langages les plus couramment mis en œuvre pour le développement d'applications Web. Depuis la version 5, il permet un développement objet complet à l'instar du langage Java, ainsi que la gestion des exceptions.

La solution de packaging LAMP (Linux Apache MySQL, PHP) permet aux développeurs d'installer facilement un système complet et efficace pour la réalisation d'applications Web Client/Serveur. Idéal pour le développeur !

Cas du développement Web simple (HTML, Javascript)

• **L'éditeur HTML : KompoZer**

Éditeur HTML disposant d'une interface WYSIWYG. Il permet de générer facilement les tableaux et les formulaires d'une page web et le code HTML produit est compatible avec les principaux standards. Malgré ces qualités certaines, KompoZer ne peut dans la plupart des cas être utilisé seul dans un cadre professionnel. Il ne saurait, en effet, rivaliser avec la référence du marché Adobe Dreamweaver ou même un Microsoft Expression Web.

• **Librairies graphiques : DHTML goodies**

Avec DHTML Goodies, le développeur Web dispose d'une librairie de composants graphiques riches et innovants qui s'appuie sur DHTML et AJAX. Il lui est très facile de produire des pages intuitives aux allures modernes. Il vous suffit pour vous en convaincre d'afficher les exemples proposés sur le site de l'éditeur.

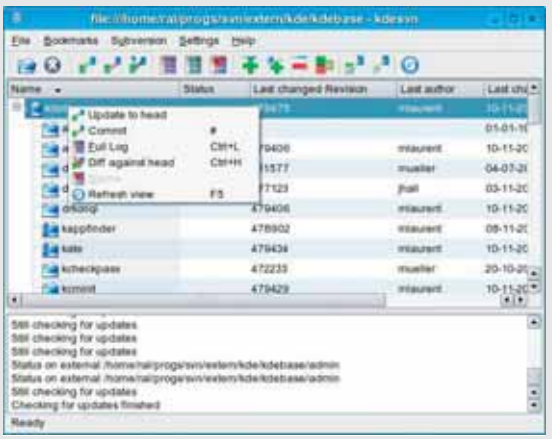
Cas du développement .NET

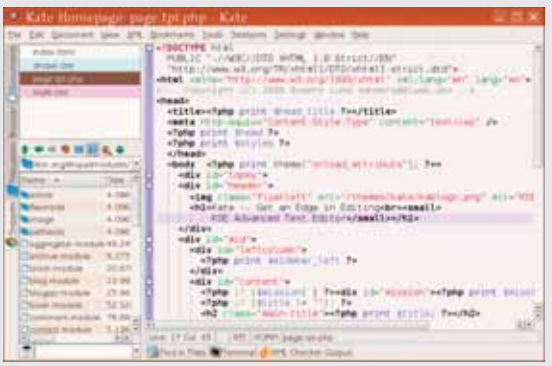
• **Framework : Mono**

Pour le développeur .NET, la bibliothèque open source est nettement plus réduite. Mono est un projet très avancé: il s'agit de reproduire une plate-forme dont on ne dispose pas du code source et dont les spécifications sont loin d'être rendues publiques dans leur globalité. Par la nature même du projet, Mono ne pourra que tenter de suivre les évolutions du framework .Net. Le projet commence à implémenter les librairies .Net 3.0.

Le développeur Uperto est open source

La société Uperto, spécialisée dans l'open source, a passé tous les développeurs en open source. Et ils sont heureux ! Voici quelques conseils, quelques outils utilisés en interne :





Pour un développeur C / C++

- Editeur : "vim" / "emacs" / "kate" ;
- Compilateur : "gcc" ;
- Débogueur "gdb" , parfois avec interface graphique de type "ddd" ;
- Détection de fuite mémoires/profiling : "valgrind";
- Gestion des build : "make", "cMake" ;
- Vérification de cohérence de code : "rats" (en exécution), "splint" (code statique);
- Documentation de code : "doxygen" ;
- Constructeur d'interface : "QTDesigner", "Glade", "Gorm" ;
- De nombreux outils de scripting pour automatiser des tâches de projets (déploiement / tests de non régression (TODO) etc.) ;
- Nombreux codes disponibles en licence libre et réutilisable ;

Du côté outillage

Les outils disponibles sont nombreux. Sous Unix et Linux, il n'y a pas de grands environnements unifiés mais plutôt un assemblage d'outils divers et variés :

- IDE : Eclipse est de loin le plus utilisé (même en dehors de Java) avec de nombreux plug-in : epic (Perl), phpeditor (PHP) , plug-in UML de factoring etc.
- Outils gestion de Projet : centralisé (Subversion ou CVS), décentralisé (Git ou Mercural), statistique/reporting de gestionnaire de source (Statsvn)
- Application de suivi de bugs : Mantis , Bugzilla , Flyspreay
- Test de performances (Web) : JMeter / Funkload ;
- Les fonctionnalités de gestion de projets peuvent être regroupées dans des portails internes de type Gforge, Trac, Redmine

■ **Fabien Vallon** *(Uperto)*

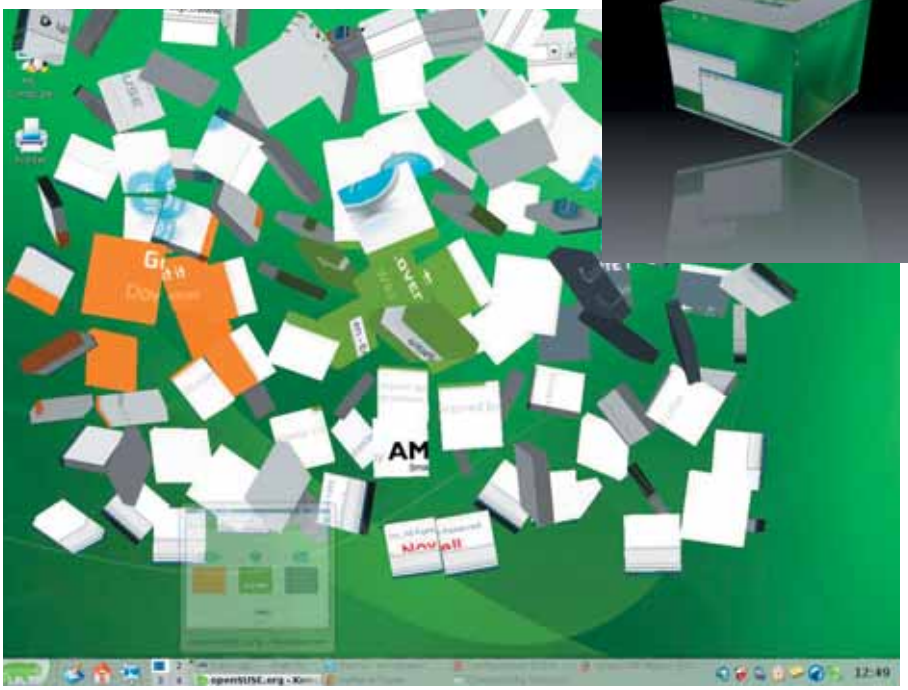
5 Conclusion

Comme nous venons de le voir, la logithèque open source propose plus ou moins d'outils selon les technologies ciblées. Le développeur Java ou PHP (et même Ruby, Python…) dispose d'un poste de développement totalement open source, tandis que le développeur .NET ou le développeur / designer Web peut difficilement se passer de certains produits payants. Donc oui, le poste 100 % open source existe et est une réalité.

■ **Vincent Perdereau** - *ads-COM*
vincent.perdereau@ads-com.fr - www.ads-com.fr

Les coulisses d'OpenSUSE

Rachetée par Novell, SUSE constitue aujourd'hui une des distributions majeures du monde Open Source. Sa déclinaison OpenSUSE, la version communautaire, comme l'est Fedora pour Red Hat, rencontre un franc succès.



OpenSUSE repose sur deux éléments vitaux : le wiki et le service Build. À cela se rajoutent des mailings lists, des forums, des IRC. Tout cela sert aux différents projets composant OpenSUSE. Les projets présents dans le service Build pourront ensuite être intégrés dans la distribution, comme par exemple One-Click Install, pour simplifier la gestion et l'installation des packages… Selon Zonker (alias Joe Brockmeier, openSUSE Community Manager), il y a entre 500 et 1000 développeurs actifs (un chiffre difficile à cerner), pour plus de 4 000 inscrits sur le service Build (https://build.opensuse.org). *" il est difficile de dire avec certitude l'ampleur de la communauté openSUSE. Nous estimons qu'il y a entre 4 et 5 000 installations par jour, soit plus d'un million d'utilisateurs, au total, par version. D'autre part, sur www.opensuse.org, on compte 55 000 utilisateurs enregistrés "*.

Un financement diversifié

Comme tout projet d'envergure, il faut financer la structure, l'animation. OpenSUSE ne déroge pas à la règle. *" openSUSE est largement soutenu et financé par Novell qui fournit personnel et infrastructure "* précise Michael Loeffler (Product management SUSE Linux). Pour compléter le budget, des sponsors donnent de l'argent, du matériel ou des services. Il y a ainsi AMD, IP Exchange dans le conseil (le Board). *" AMD a fourni beaucoup de matériel et IP Exchange procure de la bande passante pour la distribution et assurer le service Build "* poursuit Michael.

Quelle gestion ?

Pour assurer la communication entre la gouvernance et la communication, plusieurs canaux sont disponibles. Le plus important est la mailing list officielle dans laquelle les fonctions, évolutions, sont évoquées et discutées. Il y a aussi des meetings réguliers sur IRC pour parler des développements en cours, des statuts, des évolutions futures. *" Beaucoup de décisions sont prises là "* indique Michael.
À la tête de la distribution, on trouve le Board, le conseil de direction. Il contient des personnes salariées, ou non, par Novell. Le Board assure la bonne conduite du projet et est la voix de la communauté pour Novell.

Devenir contributeur

Oui, un développeur peut devenir contributeur d'OpenSUSE ! Le point de départ est la page : http://en.opensuse.org/How_to_Participate.
Toute nouvelle contribution est la bienvenue et même encouragée, c'est comme cela que vit et évolue un projet open source, par la vivacité de sa communauté. *" Nous dépendons beaucoup des développeurs pour améliorer les logiciels que nous livrons avec la distribution. "* martèle Zonker. *" OpenSUSE Build Service est un excellent point de départ pour contribuer. Tout le monde peut créer un logiciel",* poursuit Michael. Mais les contributions concernent aussi, et en premier lieu, les remontées de bugs, la documentation, la maintenance et le développement du wiki ou encore l'amélioration de la qualité logicielle des packages. Pour les artistes, vous pouvez aussi proposer vos talents au " département graphisme " pour les logos, etc. La localisation consomme beaucoup de ressources nationales (OpenSUSE est disponible en 52 langues).
Et si vous voulez voir votre création logicielle incluse dans la distribution, tentez votre chance en créant un projet dans le Build Service que tout le monde pourra faire évoluer, critiquer. Si tout se passe bien, il pourra faire partie des packages. Sur les outils utilisés, là aussi du grand classique, en privilégiant l'open source : Gcc, Python, Ruby, Perl, Java, Mono, PHP…

Et en France ?

Le projet ne tient pas de comptabilité communautaire par pays, tout juste peut-on estimer la population. Selon Michael, il y aurait plus de 2 000 utilisateurs enregistrés. Un wiki français est disponible sur www.fr.opensuse.org, tout comme une mailing list. Alors, développeurs français, participez ! Vous êtes les bienvenus…

François Tonic

Ubuntu : la distribution qui monte

Les distributions GNU Linux bougent ! Aujourd'hui, la plus en vogue est sans conteste Ubuntu. Le dernier salon Solutions Linux a été marqué par ce système. Activement soutenu par la communauté et les éditeurs, Ubuntu est rapidement devenue une référence.

Pour mieux connaître le fonctionnement d'Ubuntu, nous avons posé quelques questions à Cyril Lavier, développeur actif de la communauté française et internationale (alias davromaniak). Il s'occupe de plusieurs paquets présents sur les dépôts officiels, dont QTPFSGUI, et est " ubuntu-member " depuis septembre 2007. Il mouille aussi le tee-shirt dans les Ubuntu-Party. En terme de population, il est difficile de donner des chiffres. *" Cependant, nous pouvons assurer qu'il y a plusieurs millions d'utilisateurs, et que le nombre de ceux inscrits sur le forum francophone (http://forum.ubuntu-fr.org) avoisine les 65 000. D'ailleurs, beaucoup d'utilisateurs ne sont pas enregistrés, donc pas comptabilisés (on estime qu'il y a 4 fois plus d'utilisateurs non-enregistrés, que d'utilisateurs enregistrés).* " précise immédiatement Cyril.

Programmez ! : Comment se finance Ubuntu ? Qui soutient le projet ?

Cyril Lavier : *Ubuntu vit grâce à Mark Shuttleworth, ou plus exactement grâce à Canonical, l'entreprise créée pour soutenir le projet Ubuntu. Canonical est financée en grande partie par l'argent de Mark Shuttleworth. Cependant, l'objectif est qu'aux alentours de 2010, Canonical soit rentable.*

P ! : Comment se structure le projet, un projet Ubuntu ? Comment est-il dirigé de la tête à la base ?

C.L. : *comme dans tout projet ouvert, il y a la communauté, les développeurs, utilisateurs, tous bénévoles. Ensuite, on peut diviser ces bénévoles en plusieurs classes : "Contributeur" pour la partie contribution de code, de documentation ou d'assistance sur IRC. "Ubuntu-member", ces membres ont déjà contribué et pour les remercier, la communauté leur décerne ce titre, comme ce fut mon cas il y a quelques mois. Enfin, nous trouvons les "MOTU" (Master Of The Universe). Ce sont des personnes déjà Ubuntu-Member mais qui ont contribué fortement au projet. Certains MOTU sont chargés de valider les packages déposés sur le dépôt Universe.*

P ! : est-il possible de contribuer, si oui comment ? Quelles sont les conditions ? Quel processus suivre ?

C.L. : *Bien entendu on peut tous contribuer. Pour cela, il suffit de proposer ses services sur le forum de la communauté francophone. Par ce biais, vous faites connaître vos compétences, vos domaines de prédilections. Surtout, c'est là que le futur contributeur pose sa candidature pour tel ou tel projet. Il peut aussi aider les utilisateurs sur le forum. Il n'existe pas de réelles conditions, hormis respecter les autres membres et leur travail. Détail important : il faut aussi écrire dans un français correct… Sans être une obligation, tout contributeur est vivement invité à être régulièrement présent sur les canaux IRC de la communauté (au moins francophone). C'est un moyen d'être connu des autres membres, des utilisateurs. Pour contribuer au niveau "international", une connaissance de l'anglais est nécessaire en plus des conditions précédentes.*

P ! : Comment un nouveau projet rejoint-il Ubuntu ?

C.L. : *Tout dépend de la signification que l'on donne aux termes "projet" et "rejoint". Si on parle de projet au sens d'une application libre, "rejoint" signifiant qu'un paquet deb contenant ce logiciel est inclus, il faut qu'un "empaqueteur" volontaire se manifeste pour empaqueter le programme (tel a été mon cas pour QTPFSGUI (http://qtpfsgui.sourceforge.net), ou que le développeur du logiciel ouvre un bug sur le launchpad (http://launchpad.net) demandant s'il y a un, ou des volontaires pour empaqueter son logiciel.*

P ! : Quels outils de développements mettez-vous en oeuvre pour vos projets ?

C.L. : *Pour empaqueter les logiciels, nous utilisons les outils fournis par Debhelper, qui sont les mêmes que pour Debian à quelques différences près. Pour gérer les projets, il y a un système de versioning nommé Bazaar, similaire à SVN ou CVS. Le launchpad nous permet aussi de gérer les projets en centralisant les bugs, les traductions et les demandes de fonctionnalités.*

P ! : Quelle est la place de la communauté en France ? Son importance, son dynamisme ?

C.L. : *La communauté en France est représentée par l'association nommée Ubuntu-fr (http://www.ubuntu-fr.org). C'est une des plus actives. Elle organise tous les 6 mois, après la sortie d'une nouvelle version, une Ubuntu-Party. Cela a commencé en Juin 2006, avec une install-party réunissant 50 personnes durant un samedi après-midi dans une salle du 12ème arrondissement, gracieusement prêtée par la Fédération Française des clubs UNESCO. A l'heure actuelle, une Ubuntu-Party est un évènement qui dure 2 jours, la dernière s'est déroulée en Mai 2007, au Carrefour Numérique à la Cité des Sciences et de l'Industrie de la Villette. On y trouve des conférences, un atelier, et pour la prochaine (les 7 et 8 Juin 2008), nous organiserons des sessions de cours d'une heure, sur des thèmes variés.*

■ **Jean Vidames**

KDE : le succès d'une interface libre !

Les utilisateurs de GNU Linux, et occasionnellement sous Windows et MacOS X, connaissent bien KDE, un des environnements graphiques les plus utilisés. Sa 4e version majeure est disponible depuis janvier dernier. Nous vous proposons une plongée au cœur de son fonctionnement…

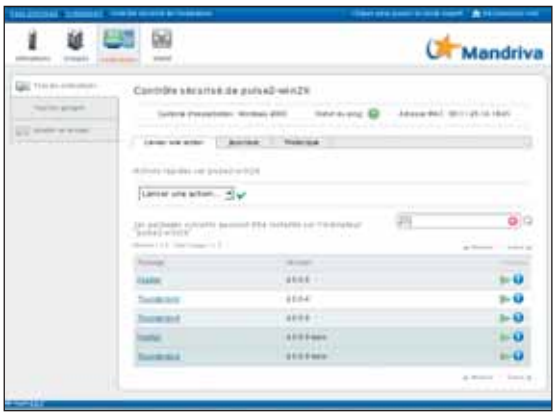
Énorme projet, KDE regroupe de nombreux sous-projets et inclut de nombreuses applications pour proposer aux utilisateurs, et développeurs, un ensemble complet et cohérent. Helio Chissini de Castro, ingénieur chez Mandriva Brésil et très actif dans le projet KDE a répondu à nos questions.

Programmez ! : Comment se structure le projet KDE ? Existe-t-il des filiales dans différents pays et une maison mère pour coordonner l'ensemble ?

Helio Chissini de Castro : KDE est un projet international collaboratif pour lequel il n'existe pas réellement de site central. Il dispose d'un statut d'association non gouvernementale, appelée KDE e.V.. L'association a été créée sous la loi allemande, pour gérer les aspects légaux, marketing et financiers, comme l'accord FreeQt, mais aussi la collecte de donations auprès de diverses entreprises. Les membres du comité de direction sont répartis dans le monde entier.

P ! : Comment un développeur peut-il contribuer à KDE ? Quelle procédure suivre ? Qui décide ou non de l'arrivée d'un nouveau contributeur ? Comment se vérifie la pertinence du code du nouveau venu ?

H.C.C. : Le projet KDE repose avant tout sur la confiance que nous plaçons dans nos développeurs, traducteurs et artistes. Pour devenir un contributeur KDE, il suffit de présenter des arguments construits pour prétendre entrer dans l'équipe. C'est essentiellement lié à la fourniture d'un travail somme toute raisonnable, des idées qui sont présentées via nos canaux de communication traditionnels, comme les listes de diffusion. Le processus habituel est relativement simple et naturel. D'abord un membre de l'équipe commence à intégrer vos contributions. Puis si celles-ci deviennent fréquentes et de qualité, vous pouvez obtenir alors un accès au principal serveur de développement.



P ! : La gouvernance est un élément important. Comment cela se passe-t-il chez KDE ? Qui fait quoi et comment ? Qui contrôle ?

H.C.C. : Le modèle du développement Open Source est plutôt un management à la méritocratie. KDE ne diffère pas sur ce point. Chaque application a ses mainteneurs et/ou son manager (communément appelée lead, ndlr). Ceux-ci ont gagné leur statut du fait de la création de la dite application ou des contributions apportées. Dans le cas de la maintenance des services critiques nécessaires au fonctionnement du projet (plate-forme de travail, serveurs web…) KDE dispose d'une équipe dédiée choisie généralement pour des raisons de proximité avec les dits serveurs, ou pour leur longue expérience de gestion de ce type de service. Chacun peut alors postuler. Il n'existe pas de contrôle centralisé. KDE dispose toutefois de groupes qui conseillent et fournissent les grandes directions du projet en matière technique et de marketing. En cas de conflit de management, par exemple deux parties de logiciel avec le même objectif sans parvenir à un accord, le groupe de conseil technique tranchera la question et donnera une réponse finale. Mais ce cas de figure est plutôt rare et se produit généralement lors de gros changements dans le logiciel. Tous ces groupes sont élus durant les conférences officielles du projet et chaque membre de l'association a également le pouvoir de participer et poser certaines restrictions.

P ! : Les distributions incluent KDE. Mais alors, comment se déroulent les relations avec les éditeurs, les développeurs l'implémentant ? Comment l'éditeur intègre-t-il KDE dans son Linux par des contributeurs présents en interne ?

H.C.C. : La plupart des distributions disposent d'une équipe de mainteneurs KDE qui font eux-mêmes partie du projet KDE. Cela procure donc un accès direct au projet et permet la prise de décisions collaboratives pour aboutir au planning le plus favorable pour l'ensemble des distributions. Dernièrement, des développeurs de Mandriva, OpenSuse et Kubuntu ont eu l'opportunité de s'asseoir à la même table lors du lancement de KDE4 aux Etats-Unis et de discuter ainsi des besoins des distributions pour la future version KDE 4.1 afin de permettre une meilleure intégration. Ainsi les efforts sont mutualisés lors de la communication avec les développeurs qui ne travaillent pas pour une distribution en particulier, lorsque ceux-ci souhaitent " commiter " leurs contributions dans les arbres de développement de KDE.

Site : www.kde.org

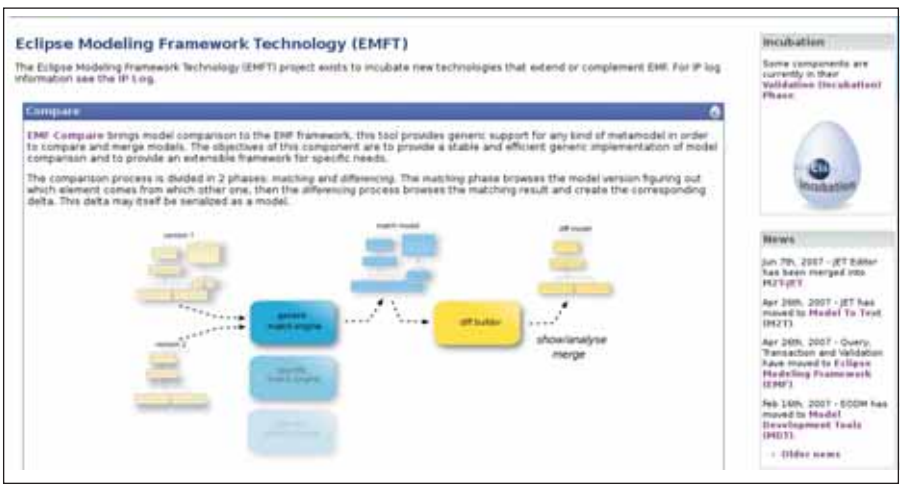
Jean Vidames

EMF Compare : Le quotidien d'un projet Eclipse

La fondation Eclipse héberge des dizaines de projets affiliés au cœur même d'Eclipse : son IDE, sa plate-forme. Nous avions dans notre précédent dossier Open Source abordé la communauté Eclipse, ce mois-ci, plongeons au cœur d'un projet, raconté par les créateurs et animateurs.

Le projet EMF Compare (http://www.eclipse.org/modeling/emft/?project=compare) est un exemple assez représentatif du processus de création d'un nouveau composant Eclipse. Son histoire remonte à 2006, lors du symposium "Modeling" organisé pendant "l'Eclipse Summit Europe". Il s'agit d'une journée dédiée aux échanges entre les différents acteurs d'un projet de haut niveau d'Eclipse, cette journée est articulée autour d'un thème précis et les différents composants/sous projets échangent entre eux pour adresser les diverses problématiques rencontrées.

En 2006, il apparaît clairement que les outils proposés par le projet "Modeling" sont de qualité mais qu'un énorme frein à leur adoption concerne le travail en équipe. Il était impossible d'utiliser des modèles avec des solutions de gestion de configuration sans devoir à un moment donné réaliser des comparaisons/fusions directement à l'aide du fichier sérialisé en XML. Deux spécialistes de l'open source interviennent pour tenter de répondre à ce défi : Intalio et Obeo. Chacun ayant une première implémentation, un "proof of concept" permettant la comparaison/fusion de modèles EMF. Les deux contributions seront fusionnées et formeront un nouveau composant au sein du projet modeling.



EMFT : Accueil du composant EMF Compare

Proposition à la communauté

La création d'un nouveau composant passe tout d'abord par la rédaction d'une "proposition" synthétisant les objectifs du projet, les contributions, et les organismes intéressés. Elle doit être claire sur sa portée car c'est à travers cette proposition que la communauté décidera si ce composant "empiète" sur un composant existant et si les objectifs ont une chance d'être tenus. Pour EMF Compare, les objectifs étaient évidents : permettre la comparaison/fusion de modèles EMF, mais l'accent est déjà mis sur l'extensibilité qui permettra de définir des moteurs de comparaison spécialisés. En terme d'organisation, le composant démarre en incubation et fait partie du projet EMFT (EMF Technology) jusqu'à ce qu'il témoigne d'une qualité et d'une efficacité suffisante pour être intégré à EMF.

Une fois la proposition publiée, les discussions se font sur deux canaux: le bugzilla Eclipse dans lequel les contributions initiales en terme de code source sont publiées, et les groupes de discussions. Ces dernières permettent à toute la communauté de commenter la proposition, d'obtenir des informations complémentaires ou de déclarer son intérêt pour le projet. Les discussions autour d'EMF Compare ont été relativement limitées : le support de la comparaison de modèles devait venir aussi vite que possible et les impératifs d'extensibilité étaient évoqués dès la proposition.

Processus IP

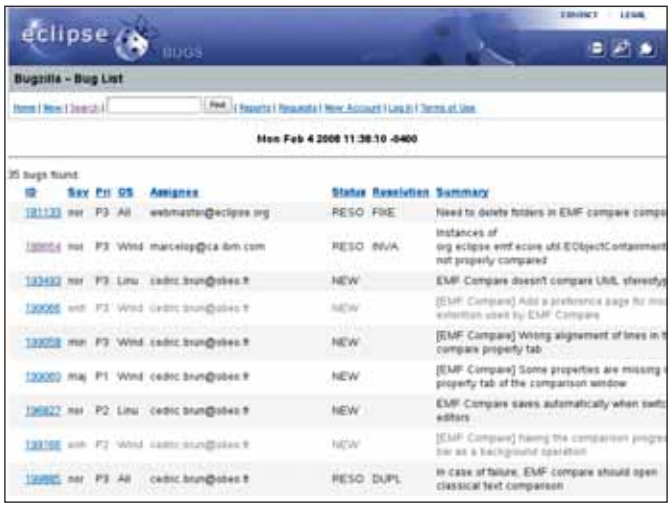
La fondation apporte un intérêt tout particulier aux problématiques liées à la propriété intellectuelle. Toutes les contributions doivent être validées en terme légal par une équipe dédiée à la fois en terme de licence, de brevets et de marques déposées. Cette équipe parcourt la totalité du code source d'une contribution et identifie les fichiers pouvant être problématiques. Les dépendances sont également validées: une liste de bibliothèques valides existe, et si la contribution dépend d'une autre bibliothèque il faut demander sa validation. Par exemple, la contribution EMF Compare dépendait de Log4J: le processus de création de composant a été figé pendant que le code a été refondu pour le débarrasser de cette dépendance.

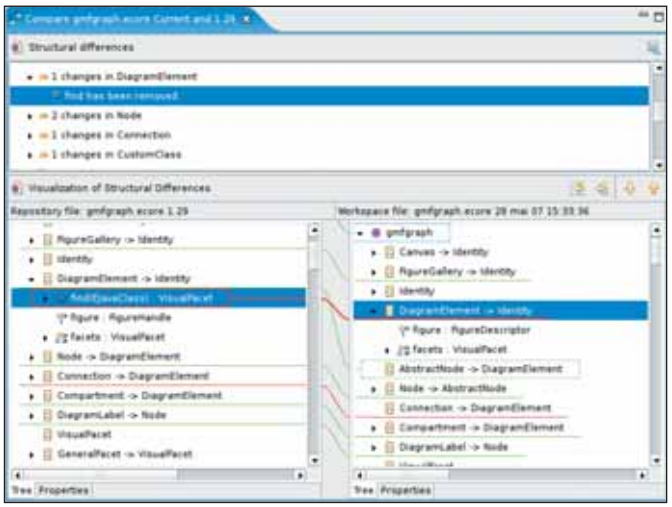
Passée sa création, le composant sera régulièrement soumis à des revues de validité en terme de propriété intellectuelle, en particulier au passage à la version 1.0.0.

Contributeurs initiaux

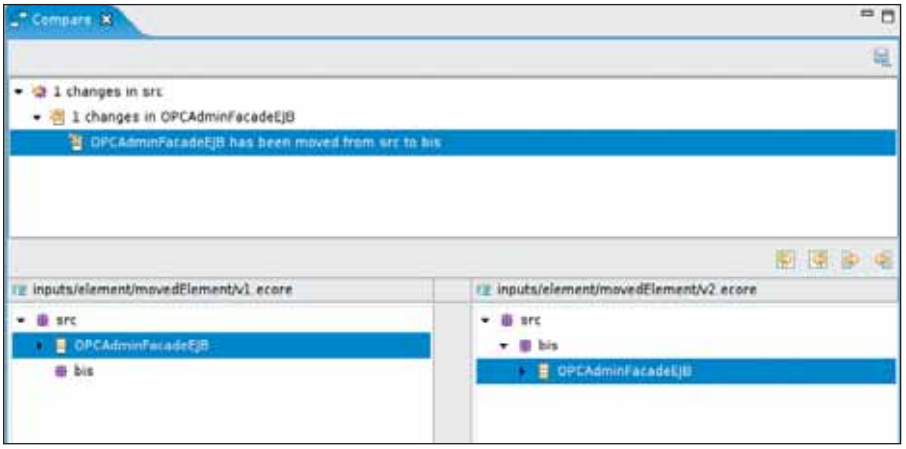
La proposition de projet prévoit des "commiteurs" initiaux ainsi qu'un responsable de projet. Pendant le processus de validation de la création du composant, ces personnes deviennent "commiteur Eclipse" et peuvent ainsi accéder au code source via CVS.

Le responsable du composant, bien que son projet fasse partie d'un projet de plus haut niveau, reste maître à bord et régit la conduite des développements.





Bugzilla Eclipse



Création effective du projet

La totalité de ce processus administratif a pris 5 mois, un processus simplifié existe désormais: le code peut être ajouté dans le dépôt d'Eclipse avant que la validation finale soit effective. Une fois accepté, le projet dispose d'un espace sur le dépôt de code source d'Eclipse, d'un newsgroup et des infrastructures de la fondation Eclipse pour le site web et la compilation du projet.

Vie d'un projet

Communauté

Pour chaque projet, la communauté peut se découper en 3 communautés distinctes mais liées. Tout d'abord la communauté des utilisateurs qui correspond à l'ensemble des personnes utilisant le projet Eclipse comme application finale. Ensuite la communauté des développeurs (commiteurs) et contributeurs qui rassemble les personnes qui travaillent directement sur le projet.

Enfin la communauté des "adopters" regroupe les acteurs d'autres projets qui utilisent le projet comme framework, ou bibliothèque. Le "project leader" doit veiller à maintenir les communautés actives. Différents vecteurs de communication sont ainsi mis à disposition par la fondation pour l'interaction au sein de chaque communauté et entre les communautés. Ainsi pour obtenir de l'aide, les utilisateurs bénéficient d'un groupe de discussion (ou newsgroup). Les développeurs utilisent quant à eux une liste de diffusion pour la communication interne du projet. Enfin, chaque projet correspond à une catégorie dans le système de suivi de bugs (bugzilla https://bugs.eclipse.org/bugs/) qui permet la remontée de problèmes ou de demandes d'amélioration.

De manière plus globale, le wiki Eclipse (http://wiki.eclipse.org) est la principale source d'informations concernant Eclipse, la fondation, ses procédures, ... Il regroupe la documentation et notamment bon nombre de FAQ (questions fréquentes). Une autre source d'information importante est la page Eclipse resources (http://www.eclipse.org/resources/) qui agrège livres, articles, podcasts, webinars, présentations, et exemples de code. Plusieurs canaux IRC (messagerie instantanée) sont disponibles tant pour Eclipse que pour ses projets de haut niveau. Utilisateurs avancés et développeurs des différents projets concernés y sont présents et répondent dans la mesure du possible aux questions et discussions les concernant.

Ces divers vecteurs d'interaction entre les utilisateurs et les développeurs ont pour autre avantage de permettre aux utilisateurs de proposer des patchs (morceaux de code destinés à corriger ou améliorer le projet cible). En effet, un utilisateur ayant désir de s'investir plus avant dans la vie du projet est à même de récupérer une version du code source à un moment donné puis de développer corrections et fonctionnalités par lui-même. Après discussion, le patch proposé sera bien souvent appliqué sur la version de production du projet cible si il ne lui apporte aucune nouvelle anomalie.

Conventions

L'ensemble des conventions préconisées par la fondation se trouve énoncé sur le wiki Eclipse (http://wiki.eclipse.org/Development_Conventions_and_Guidelines). Les conventions pour la documentation et pour le style du code diffèrent peu des recommandations officielles de Sun. Les conventions de nommage sont courtes, simples et aident à structurer un projet. La création des interfaces graphiques est quant à elle encadrée par un grand nombre de directives. Elles concernent le choix des couleurs, le choix des icônes, l'utilisation des assistants. Eclipse étant une plate-forme des-

tinée à être étendue, un certain "conformisme" est nécessaire et attendu afin de ne pas troubler l'utilisateur.
Toutes ces conventions, depuis le formatage du code au nommage des interfaces, reste bien entendu propre à chaque projet et le project lead est à même d'adapter ces conventions pour son projet. Dans ce cas, les fichiers de configuration Eclipse vérifiant ces nouvelles conventions sont souvent accessibles et distribués avec le projet.

Construire et distribuer le projet

Afin de fournir aux utilisateurs une version utilisable et maintenue d'un composant, il faut réaliser un "build". Le build consiste à compiler le code source du composant afin de générer les archives jar correspondantes. Il est réalisé à partir des plug-in et des features constituant le composant. Les plug-in contiennent déjà la description de toutes les informations nécessaires à leur compilation. Les features regroupent les plug-in par lot, spécifiant les dépendances à résoudre lors du build, ainsi que les informations relatives aux update-sites. Les update-sites permettent aux utilisateurs de télécharger et installer directement les composants sous Eclipse. Le build est généralement réalisé par un système basé sur des scripts *ant* et *bash*, permettant de gérer le processus depuis la compilation jusqu'à la publication du composant, en passant par le lancement de tests unitaires.
Une fois le build d'un composant effectué, sa publication peut avoir plusieurs statuts : Intégration, Maintenance, Stable, Release. Le statut de Release n'est attribué qu'au terme d'une revue complète du composant par un comité, l'EMO. La "Release review" peut prendre plusieurs semaines. L'EMO vérifie plusieurs critères, par exemple si les bugs sont régulièrement corrigés, la conformité du code source aux standards Eclipse, l'utilisabilité de l'outil, de l'interface... Si la revue est positive, le composant est publié et devient accessible sur les serveurs de téléchargement d'Eclipse.

Maturation d'un projet

Incubation

La phase d'incubation est la phase dans laquelle entre tout projet fraîchement créé. Son but est de permettre de poser les bases du projet open-source Eclipse en adoucissant le processus IP. Ce contexte permet de développer la communauté autour du projet tout en posant la technologie sous-jacente. Cette phase permet par ailleurs aux nouveaux projets de se familiariser avec la communauté et les procédures Eclipse. La phase d'incubation d'un projet se termine par une revue nommée "Graduation review". Cette revue permet de s'assurer que le projet se plie bien aux normes d'Eclipse, qu'il a une base de code de bonne qualité et fonctionnelle, que sa communauté est suffisamment active et qu'il est convenablement intégré à la plate-forme Eclipse. Le projet passant cette revue avec succès entre dans une phase où il est considéré "mature".

Accueil du Wiki Eclipse.

Maturation

Un projet arrivé en phase *mature* a fait la preuve qu'il est projet open-source en phase avec les processus et valeurs Eclipse. Son équipe de développement est active, de même que sa communauté, qui continue souvent de croître. Les projets entrés dans cette phase ont pour nécessité de passer une "Release review" pour chaque distribution majeure.
La phase de maturation d'un projet ne s'achève que dans le cas ou ce projet passe haut-niveau (ou top-level) via une "Promotion review" ou qu'il est terminé via la "Termination review". Les projets passent top-level dans les rares cas ou le projet a pu démontrer qu'il était constamment leader dans un domaine technique donné et qu'il possède une importante communauté de développement.
La revue "Termination review" a, quant à elle, pour objectif de fournir aux développeurs et utilisateurs une dernière opportunité pour démontrer leur intérêt dans le projet avant qu'il ne soit retiré ou archivé.

Archive

Un projet devenu inactif, que ce soit dû à l'épuisement de ses ressources, ou a son achèvement, est archivé. Un projet peut atteindre son achèvement "naturel" de plusieurs façons, il peut par exemple être absorbé par un autre projet. Un projet archivé peut toutefois être réactivé par la communauté si un intérêt suffisamment important lui est porté et que les nouveaux développeurs peuvent prouver que les raisons de l'archivage ne sont plus valides pour ce projet. Un projet réactivé de cette façon l'est à travers une revue de création ou "Creation review". La revue de création pour la réactivation d'un projet est plus sévère que pour la création d'un nouveau projet. Ceci pour s'assurer que les ressources sont suffisantes pour reprendre le flambeau et que le plan de développement est toujours conforme aux objectifs initiaux ou recadrés.

Conclusion

Eclipse offre ainsi un cadre et des procédures rôdées afin de permettre aux entreprises comme aux individus de contribuer à une communauté grandissante, ou de se lancer dans l'aventure open-source. Les groupes de discussions et salons IRC sont ouverts et accessibles pour répondre à toute question que vous pourriez encore vous poser concernant la fondation Eclipse. Serez-vous le prochain à rejoindre la communauté francophone?

Laurent Goubet, Cédric Brun, Stéphane Drapeau, Mariot Chauvin
Consultants MDA - Obeo (http://www.obeo.fr)

PHP : comment participer au langage libre le plus populaire ?

PHP, vous connaissez ? Langage phare du monde web depuis plusieurs années, vous vous dites " j'aimerais bien y participer en tant que développeur ".

Aujourd'hui PHP repose sur près de 1000 bénévoles travaillant ponctuellement à améliorer cette plate-forme phare. Ces bénévoles n'ont pas tous la même implication et on peut estimer qu'en temps de travail effectif il y a près de 100 personnes fortement impliquées. Ce groupe est géré au quotidien par un noyau dur composé de près de 10 personnes avec à leur tête le mythique Rasmus Lerdorf.

Cette communauté de développeurs se répartit ainsi :

- équipe de développement (500 personnes) ;
- équipe qualité (250 personnes) ;
- équipe de documentation (120 personnes) ;
- équipe de traduction (120 personnes).

Étant donné que de nombreux contributeurs participent à plusieurs équipes, on estime leur nombre total à 1000. Ce qui est considérable. Plus globalement, au niveau mondial, on estime à plus de 4,5 millions de développeurs l'utilisant et à des milliers, les projets basés sur ce langage. En France, grâce à l'association française des utilisateurs de PHP (AFUP) la technologie à une grande visibilité. La communauté PHP française est reconnue et appréciée à travers le monde entier. Composée de plus de 250 professionnels utilisateurs de PHP, l'AFUP est aujourd'hui le référent des professionnels et des particuliers pour tout ce qui touche à PHP.

Les développeurs de PHP travaillent bénévolement. En général, ils travaillent dans une société qui utilise massivement PHP et leur implication permet d'optimiser leur travail.

On peut citer Rasmus Lerdorf, le créateur de PHP, qui travaille pour Yahoo! et qui en parallèle optimise PHP pour répondre au besoin de son employeur.

La structure des projets PHP

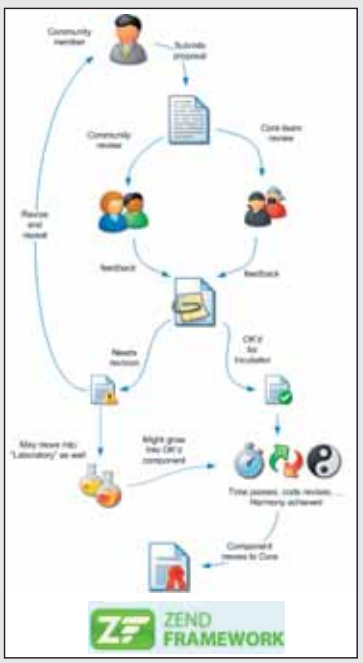
Les sorties (releases) sont généralement gérées par un RM (Release Master) jouant le rôle de l'organisateur. Il est éventuellement aidé par un RMB (Release Master Bitche), dont le rôle est de gérer les tâches ingrates : servir d'avocat du diable, recueillir les critiques et les bogues, etc. La désignation d'un RM se fait sur une base de volontariat et par approbation de ses pairs. Les développeurs utilisent l'outil CVS pour gérer les différentes versions. Chacun d'entre eux propose ses sources à son rythme via cet outil CVS.

Oui, je veux participer au projet PHP !

Tout développeur peut participer au projet. Pour cela, il suffit de télécharger les sources via CVS et de développer un patch. Il propose ensuite celui-ci à la communauté (via les mailing list, par exemple). Le patch est alors évalué par l'un des " anciens ". Si le PHPGroup considère le patch comme bon, il intègre la version officielle. Sinon il peut être mis dans une bibliothèque externe (la SPL), voire totalement refusé. Une fois que le " commiteur " a proposé plusieurs patchs, il lui est donné du 'karma'. Plus un développeur a de 'karma', plus il aura le champ libre. Il pourra donc, à terme, " commiter " des patchs sans passer par une lourde procédure de tests/validation.

Cyril Pierre de Geyer
Consultant / Architecte Open Source
http://www.anaska.com
Auteur du livre "PHP 5 avancé"

Et pourquoi pas devenir contributeur à Zend Framework ?



PHP vous passionne, vous utilisez Zend Framework ? Alors pourquoi ne pas y contribuer ? Le framework est l'un des deux projets Zend ayant une communauté au sens open source, les autres outils de l'éditeur n'étant pas ouverts. La première chose à faire est de s'inscrire sur la mailing list officielle. Si on veut renseigner un bug, il suffit de le faire sur le bugtracker du framework, voire de rendre disponible un patch dans le système. Ensuite il est analysé, pour l'inclure ou non dans le framework.

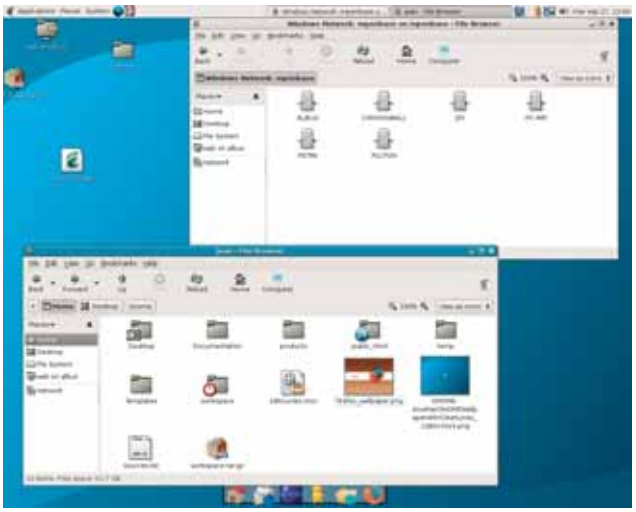
Si on souhaite devenir un contributeur, il faut tout d'abord proposer et discuter de son idée, inscrire sa proposition dans la mailing list, puis ouvrir une nouvelle page dans le wiki. Là, la communauté et Zend viennent discuter avec vous du projet, l'amender, l'accepter ou le refuser. Le projet proposé sera mis en incubateur et intégré aux versions non stables du framework. S'il est accepté et qu'il fonctionne, il peut ensuite faire partie du package stable du framework. Il faut que le contributeur démontre sa capacité de développeur, d'écoute envers la communauté. On ne devient pas un lead d'un composant du framework sans avoir contribué au code avant… À noter que le contributeur doit signer un agrément stipulant que le code est garanti sans copyright. La responsabilité du contributeur étant engagée en cas de problème. Sur la gouvernance, elle se répartit entre Zend et la communauté mais c'est l'éditeur qui peut trancher, décider en cas de désaccord ou pour des éléments importants du framework. Il est garant de la cohérence de l'ensemble.

Site : http://framework.zend.com/

François Tonic

Gnome : l'autre interface open source

Dans l'univers GNU Linux, deux environnements graphiques dominent : KDE et Gnome. Intéressons-nous ici à Gnome. Nous avons posé quelques questions à Vincent Untz, membre de la Release Team et du " board of directors " de la Gnome Foundation. Il fait donc partie des " têtes pensantes " du projet.



Gnome c'est :
- une centaine de petits projets : code, documentation, tests, gestion de bugs, etc.
- plusieurs centaines de contributeurs dont un grand nombre de traducteurs. Sur le code pur, il y a entre 50 et 100 développeurs actifs et réguliers.

La traduction est un élément clé d'un projet de l'ampleur de Gnome. Car il faut localiser l'interface, les applications liées et surtout localiser la documentation (utilisateur et technique), ce qui demande beaucoup de travail.

Comme d'autres grands projets, Gnome possède sa fondation : Gnome Foundation, qui gère les aspects légaux et financiers.

Elle reçoit l'argent d'organisations, souvent membres de l'Advisory Board, en quelque sorte un comité de conseil qui n'a aucun pouvoir décisionnel. Une organisation ne peut être membre que sur invitation de la fondation. Leurs " cotisations " vont de zéro (comme pour Debian ou la FSF) à 10 000 dollars pour les grandes entreprises. *" Cet argent permet de faire la promotion de GNOME, de participer à l'organisation d'évènements, de conférences, etc. Il y a aussi des frais administratifs. Nous organisons également des "hackfests", qui sont de petites rencontres avec un but précis "* explique Vincent Untz. Détail important : Gnome ne rémunère ni développeur, ni responsable.

Les éditeurs ne sont pas absents du projet. Car certains de leurs développeurs travaillent sur Gnome, tout ou partie de leur temps. Phénomène connu, mais *" que personne n'a jamais étudié de près, donc on ne sait pas exactement combien de personnes travaillent sur GNOME dans le cadre de leur activité. J'estimerais le nombre entre 20 et 50, mais c'est vraiment impossible de le savoir, ni combien de temps elles y passent."* analyse Vincent. Côté outils, on retrouve Subversion, Bugzilla. Pour le code, la liberté est laissée, par contre la compilation / intégration est généralement assurée par autotools, GNU make et gcc.

Programmez : comment se structure le projet, un projet gnome ? Comment est-il dirigé de la tête à la base ?

Vincent Untz : Commençons par le plus simple : un projet faisant partie de GNOME. Cela peut très bien être un projet en terme de code, ou un autre projet (promotion, sysadmin, traduction, documentation, etc.). Chaque projet possède un ou plusieurs chefs, généralement appelés mainteneurs, qui vont en assurer la gestion. Ainsi, c'est le chef qui va décider de la direction future, rendre vivante l'équipe qui tourne autour du projet, etc. Une personne devient mainteneur d'un projet par ses compétences : elle a prouvé par le passé qu'elle en est capable.

Au niveau du projet GNOME dans son ensemble, c'est plus complexe. Il n'y a pas une forme d'autorité unique, et c'est quelque peu volontaire. La Fondation est dirigée par un board de 7 personnes, élues annuellement par les membres de la Fondation. Tout contributeur au projet peut devenir membre de la Fondation. Les 7 membres du conseil d'administration supervisent donc le bon fonctionnement de la Fondation, la poussent dans de nouvelles directions ambitieuses, etc. Bref, tout le travail classique d'un conseil d'administration dans une organisation à but non lucratif. Ensuite, il y a la Release Team de 8 personnes. Elle vise à s'assurer que le développement technique du projet se déroule bien. Elle fournit un planning pour les versions de GNOME, avec différentes périodes. En quelque sorte, elle assure donc la direction technique du projet. Néanmoins, elle vise aussi à déléguer ce rôle en partie à la communauté et donc toutes les décisions sont discutées sur les listes de diffusion, et un consensus est généralement rapidement atteint. Les personnes fortement impliquées dans le projet ont une voix plus importante que les autres.

P ! : est-il possible de devenir contributeur, si oui comment ? Quelles sont les conditions ? Quel processus suivre ?

V.U. : Tout le monde peut être contributeur. Il suffit d'envoyer des patchs pour corriger des bugs ou ajouter des fonctionnalités dans Bugzilla. Participer aux discussions sur les listes de diffusion est aussi souvent un premier pas. Il n'y a aucune condition d'entrée, si ce n'est le respect des autres.

P ! : Comment un nouveau projet rejoint-il gnome ?

V.U. : il y a deux possibilités. Un projet qui fait officiellement partie de GNOME, doit être proposé pour inclusion au début d'un cycle de développement. Le projet sera débattu et étudié en profondeur. La personne s'en occupant doit pouvoir répondre aux questions et critiques et continuer à être active. Pour un projet qui fait officieusement partie de GNOME, les seules contraintes sont une bonne qualité et une bonne intégration avec le reste de GNOME.

François Tonic

Reportage : Linagora, une société open source à 100 %

Comment une société de services en logiciels libres comme Linagora fonctionne-t-elle ? Comment travaillent les développeurs ? Comment participe-t-elle aux communautés ? Alexandre Zapolsky, son PDG, nous a ouvert les portes.

Linagora s'articule autour de plusieurs pôles d'activités : services Pro (services, formations, pour les entreprises), OSSA (centre d'expertise, assurance), Formations, et une partie Editeur open source (obm, sécurité, Feder ID, Nareto).

Dans l'antre du service d'assurance ouverte

Il est environ 9h. Nous débutons notre visite par la grande salle où l'OSSA travaille. Là, une dizaine de personnes s'activent à résoudre les problèmes, les bugs sur les logiciels ouverts supportés (plus de 200 !).

Michel Loiseleur nous accueille (ingénieur de son état). OSSA est la cheville ouvrière de l'offre assurance qualité sur les logiciels libres pour les entreprises. Ce service existe depuis 2006. Un client peut soumettre une demande par téléphone, mail ou fax. La demande est alors rentrée dans un logiciel de suivi (écrit en Rails). Il faut tout d'abord décrire le problème, récupérer la configuration du client, reproduire le problème. À partir de là, les différents ingénieurs, développeurs de l'OSSA regardent de plus près. Il s'agit alors de comprendre le problème et d'en découvrir la cause. Cela débouche souvent sur le développement de correctifs (patchs) pour corriger l'anomalie ou la contourner. Il faut fouiller très profondément dans le code pour trouver la cause du dysfonctionnement. Cela permet aussi de toucher au plus près la qualité du code des projets ouverts. Et là, de l'aveu même de Michel, on y trouve de tout. *" Le code des projets dépend beaucoup de la manière de travailler des développeurs. Par exemple, chez Sun (pour OpenOffice), c'est la culture de spécifications. C'est à la fois un bien et un mal ".*

La rapidité d'action et la qualité de la correction sont les 2 critères d'excellence de l'OSSA. La qualification de la demande est une étape cruciale pour savoir où, quand et comment apparaît l'anomalie. *" On installe une machine virtuelle contenant l'environnement du client puis on cherche à le reproduire "* explique Michel. En interne, la virtualisation est assurée par KVM, Xen et parfois VMware. Pour être au plus près de l'environnement utilisateur, il y a une phase de compilation avec les spécifications clientes. Le binaire est souvent livré avant la source du correctif *" car nos clients sont en production "* précise Michel. Mais parfois, le client exigeant va jusqu'à fournir une machine contenant les applications à la cellule d'expertise de Linagora. L'OSSA met à jour régulièrement les " masters " clients avec les modifications apportées pour être à jour. Dans certains cas, il est nécessaire de garder un arbre des sources précompilées, par exemple avec OpenOffice et ses 2 Go de source qui nécessite entre 8 et 24h de compilation, un délai impossible à tenir pour les équipes de l'OSSA… Car les clients paient pour avoir de la réactivité entre 45 000 et un million d'euros.

" Nous avons différents experts apportant leur propre sensibilité, leurs propres compétences. On a ainsi un packageur Fedora, un autre sous Kubuntu. Ils sont impliqués dans la communauté. Nous avons aussi des profils de développeurs généralistes, ou d'experts en annuaire ", précise Michel. Chaque demande est examinée par plusieurs experts, aux sensibilités différentes. *" l'openspace s'avère très pratique pour communiquer, échanger. Chacun peut apporter son expertise, sa vision. Par exemple, sur un problème de plantage aléatoire sur un pilote Intel, ce n'est pas un développeur bas-niveau qui a résolu l'anomalie mais un administrateur impliqué dans la communauté… "* poursuit Michel.

Le travail en équipe est une des forces de l'OSSA. Mais surtout, il faut être capable de changer de sujet en quelques heures, de passer d'un bug OpenOffice à un problème LDAP ou de synchronisation de données. L'équipe a aussi une contrainte de temps. Le chronomètre tourne quand les experts travaillent, par contre, quand des informations ne dépendent pas d'eux, on fige le chronomètre. Seul le temps effectif de travail interne est pris en compte. Le travail de reversement est très important pour l'OSSA, les patchs mis au point sont proposés aux diverses communautés. Et ils sont pris en compte plus ou moins rapidement.

Chaque client sait qu'il y a une phase de reversement après avoir résolu le dysfonctionnement. Mais celui-ci n'est pas aussi évident que l'on pourrait le croire… *" chaque communauté à ses propres spécifications : formatages différents du code, sur la présentation des commentaires, etc. "* nous raconte Michel. D'autre

part, chaque communauté a son agrément de contribution qu'il faut signer. Il précise la cession des droits sur le code, le patch…

" Nous ne faisons pas un travail simple ! Il n'y a pas de tabou et il peut y avoir des contradictions. On est des passionnés, et ravis de travailler là. On est là pour réparer les choses. C'est passionnant et assez flatteur (pour l'ego) de résoudre (un problème inattendu, complexe). On se sent un peu comme un sauveur. " tient à préciser Michel, mais tout va dans le même sens. L'ambiance est studieuse, et la vie se cale sur les alertes des clients. Notons aussi que Linagora laisse du temps aux personnes pour contribuer aux projets open source. Pas tout le monde bien entendu. C'est un élément important pour la société. Cela permet aussi de rester en contact avec les communautés.

Comment travaillent ces experts OSSA ? Chacun a sa configuration, ses outils préférés. Un des mots d'ordre est d'utiliser une distribution stable et d'éviter les branches non stables. Car il ne faut pas oublier que l'on est là pour les clients avant toute chose. Ensuite, on retrouve les classiques, gdb, sysprof, etc. Sur les horaires, il y a une permanence assurée de 7h à 20h. Mais l'équipe reste joignable 24h / 24 en cas de contrat H24. Il faut alors se connecter de partout au réseau OSSA (très sécurisé cela va sans dire)…

Et le recrutement n'est jamais terminé. Si en plus le candidat est un contributeur actif à l'open source, c'est un argument non négligeable lors des entretiens et sur le salaire.

À l'étage avec Sophie Gauthier

Le temps passe vite à l'OSSA. Il faut déjà monter au 6e étage pour continuer notre visite. Nous laissons donc le grand bureau ouvert. Au 6e, nous attend Sophie Gautier. Connue pour son rôle dans Open Office, sur la partie francophone et dans le Comity Council, elle travaille aujourd'hui chez Linagora, pour justement apporter son expertise OpenOffice et de la communauté. Linagora souhaite se renforcer sur le bureau de travail open source et sur OpenOffice. Car les entreprises n'ont pas forcément les compétences nécessaires pour migrer, maintenir, former. Un contributeur actif travaille aussi chez Linagora (avec une vingtaine de patchs produits en 2007). *" Mon rôle est sur l'expertise OpenOffice sur les projets auxquels on répond. Et faire en sorte que Linagora soit un bon citoyen avec la communauté",* précise Sophie. Et il y a aussi un peu d'ordre à mettre dans les idées reçues. Le passage à OpenOffice n'est pas seulement une migration…

Le projet Assemblée Nationale

Un des plus gros projets fut celui de l'Assemblée Nationale : fourniture d'un environnement de travail entièrement open source ! Le challenge fut grand pour Linagora, car le projet devait se réaliser en 3 mois. Si le choix fut en premier lieu économique, Sophie rappelle un point essentiel : *" à court terme, on n'économise pas (migration des documents, formations…) mais à long terme, si. ".* Le peu de recul ne permet pas d'évaluer concrètement les gains financiers. Pas encore.

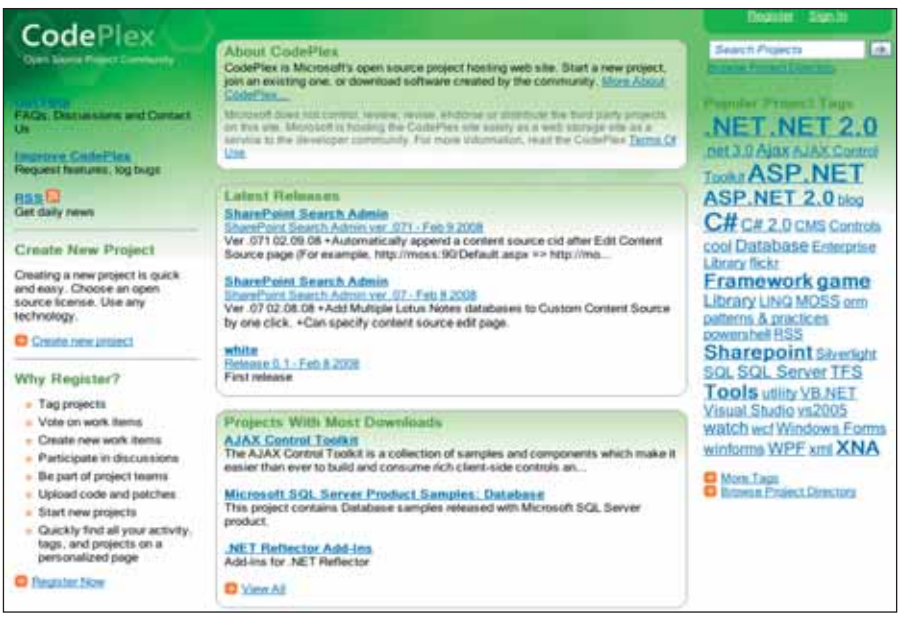
" Le 2e challenge était de fournir un bon package, prendre en compte la diversité des terminaux mobiles. Des services web et la chaîne TV n'étaient pas adaptés à Linux, mais là, on n'y peut pas grand chose. Le principal travail fut l'intégration des différentes couches entre elles. Il manque encore (à l'open source) une telle couche d'intégration ", poursuit Sophie. Les applications demandent en effet d'échanger, communiquer ensemble, or un OpenOffice, Firefox Thunderbird, etc. ne le font pas naturellement. Il a donc fallu, en interne, concevoir une "glue" pour assurer cette communication, mobilisant 2 à 3 développeurs. C'est au total plusieurs milliers de packages qu'il a fallu inclure, repackager, compiler pour fournir une distribution adaptée à l'Assemblée Nationale. *" La migration s'est réalisée en binôme* (Linagora, Assemblée, ndlr), *avec une période de tests. Chaque nouvelle version est testée à L'OSSA avant intégration dans la distribution utilisateur (Kubuntu). Nous avons aussi organisé des ateliers utilisateurs sur les fonctionnalités, les sécurités, l'interopérabilité "*. Sophie nous quitte, elle part en réunion. Il est presque l'heure du déjeuner. Il fait beau, pas mal de collaborateurs s'installent en terrasse. Pour notre part, nous partons à la recherche d'**Erwan Le Gall** qui fait partie des services Pro. Après une nouvelle tasse de café (libre bien entendu), Erwan arrive. On commence à discuter. Ingénieur, il est plus administrateur que développeur, même si dans les services Pro, tout le monde fait un peu de développement, connaît le code, surtout le script, utile en réseau :*" il y a plus de 40 personnes dans le service. On a un pôle sécurité, un autre supervision, et une partie éditeur "* commente Erwan. La partie édition fait beaucoup de Java, de C. Avantage des petites structures, on acquiert rapidement des responsabilités, notamment en allant seul chez les clients… *" Une de mes dernières missions consistait, durant 5-6 mois, à stabiliser les logiciels sur le matériel du client. J'étais en contact avec l'OSSA, je développais des interfaces. "* Comme tous les autres salariés à Linagora, Erwan connaît le Libre, les outils, il participe aussi à des communautés. Une des particularités du service Pro est que l'on est souvent sur le terrain, chez les clients. C'est pour cela qu'il n'y a pas de réels bureaux attribués. Quand on est entre deux missions, on s'installe là où il y a de la place. Linagora a l'ambition d'être un leader, si ce n'est **le** leader, de l'open source en France, mais aussi de peser au niveau européen. Cela passe par un élargissement de l'offre, comme sur le desktop ouvert, mais aussi par le renforcement des équipes. Au dernier salon Solutions Linux, plusieurs dizaines de candidats ont été reçus dans le cadre de nouvelles embauches. Si vous êtes développeur avec une culture open source forte, pourquoi pas vous ?

François Tonic

Comment Microsoft perçoit le mouvement Open source

Depuis quelques mois, Microsoft semble vouloir jouer un rôle croissant dans le mouvement Open source. Pourtant, en discutant avec quelques développeurs, il n'est pas rare de s'entendre dire que Microsoft veut faire de l'Open Source pour mieux l'étouffer. Cette réaction, quasi épidermique, fait écho au célèbre adage " Embrace, extend and extinguish ". Cette réputation acquise à la fin du siècle dernier, vient de la stratégie adoptée, entre autres dans le monde du Web pour la domination du marché. En réalité, les rapports qu'entretien Microsoft avec ce mouvement se sont sérieusement complexifiés.

Depuis son éclosion dans les années 90 et l'amplification continuelle de l'univers Open source, force est de constater que cette vague de fond a obligé Microsoft à adapter graduellement sa stratégie. Face à ce modèle économique, les débuts furent critiques et poussifs. On peut se rappeler de WIX, un des premiers projets Open Source de Microsoft, ou de WTL (Windows Template Library). A l'époque, Microsoft traînait clairement les pieds. Aujourd'hui, l'éditeur signe des accords de partenariats avec de grands acteurs du monde Open Source. Et très récemment, Microsoft France vendait à Renault, une solution à base de serveurs Linux Open Suse de Novell, capables de supporter des machines virtuelles Windows.



Une évolution naturelle ?

Pour bien comprendre cette évolution, il faut garder à l'esprit que Microsoft n'est pas une société philanthropique. Son but est de vendre des logiciels et particulièrement ses produits phares, Windows et Office. Cependant, avec son modèle économique clairement basé sur la vente de licences logicielles, l'arrivée de l'Open source a causé un véritable tremblement de terre et a été considéré comme une menace sérieuse.

Dans l'inconscient collectif, les logiciels Open source sont souvent associés à gratuité, accès au code et communautés de développeurs importante, pour des produits souvent de qualité mais n'ayant pas toujours une finition excellente. Bien que cette vision de l'Open source ne soit que partielle, dans les faits, ces caractéristiques se retrouvent dans de nombreux produits concurrents de Microsoft. Que ce soit, dans la bureautique avec Open Office et les produits Mozilla, dans les serveurs avec entre autres les très connus Linux, Apache, MySQL ou encore dans les outils de développement avec Eclipse.

Par ailleurs, les concurrents commerciaux classiques ont bien compris l'intérêt à profiter de la vague Open Source. Elle leur permet de bénéficier à la fois d'une image de sympathie, et du soutien d'une communauté de développeurs potentielle. En intégrant les apports du modèle Open Source à leurs produits commerciaux, on estime que certains éditeurs font de réelles économies. En effet, une communauté dynamique et étendue, c'est autant de développeurs et de testeurs que l'on ne paye pas. Mais c'est aussi une " évangélisation " à moindre frais, le marketing viral étant relativement efficace dans cet univers. Evidemment, cette stratégie comporte des risques, car un code ouvert peut être récupéré aisément et l'absence de support dans certains cas, peut s'avérer pénalisante.

Il s'agit ici d'un point critique. A l'exception d'un cercle de partenaires réduit, Microsoft a toujours rechigné à livrer le code de ses produits phares. Ceux-ci ayant une certaine avance technologique, Microsoft n'a pas trop intérêt à en dévoiler le code. Il en a encore moins l'intérêt, que ses moyens financiers lui permettent de recruter les équipes nécessaires à son développement. Ce faisant, il garde une totale maîtrise du code et de son évolution. Au final, maître à bord, il est plus libre dans la stratégie commerciale et technologique, et il peut faire évoluer chacun de ses produits à sa guise.

Seule une contrainte extérieure, a pu amener Microsoft à publier son code. Elle peut être

judiciaire. Ce fut le cas, par exemple d'un des procès mené par la Communauté Européenne. Procès durant lequel l'Europe exigeait la mise à disposition du code source. Mais elle peut également être le fait de l'activité d'un concurrent. Ainsi, dans le monde du développement, on aura noté que l'ouverture de certaines parties du code de .NET arrive après le classement en Open Source de celui de Java. Mais, là encore, on peut considérer que l'impact technologique sera limité, car certains outils comme Reflector, rendaient déjà possible la reconstruction du code source de .NET.Ou tout simplement pour des impératifs commerciaux, par exemple avec le CNITSEC (China Information Technology Centre). Cet accord devant faciliter l'accès au marché chinois de l'OS Microsoft face à son concurrent Linux et améliorer l'image de l'entreprise.

Une forge et des licences !

Il faut dire que dans le domaine de l'Open source, les initiatives de Microsoft ces trois dernières années, ont été tous azimuts. Parmi toutes ces initiatives, on citera en premier lieu, la création du déjà très célèbre forge Codeplex. Il remplace son prédécesseur GotDotNet désormais en fin de vie. Il s'agit du site de référence pour tous les projets Open source en rapport avec les technologies Microsoft. Beaucoup de projets internes y sont gérés.

Ensuite, on trouve le portail web " Shared Source Initiative " destiné au partage conditionné des sources des produits Microsoft ou ceux de la communauté des développeurs maison, avec des clients, partenaires, le monde éducatif, les administrations ou les développeurs du monde entier. On y trouve notamment, la section " Reference Source Service " qui permet, à partir de Visual Studio 2008, de downloader le code source Microsoft en mode débogage. Ou encore, la section " Community Source " qui regroupe un ensemble de projets Open source développés par les équipes de Microsoft. Essentiellement des liens vers des entrées de Codeplex. Microsoft propose également le portail " Open Source at Microsoft " et le site " Port 25 ". Le premier présentant la stratégie Open source de l'entreprise est plus généralement, le point d'entrée de tout ce qui est en rapport avec le mode Open source. Le second est plus spécifique aux travaux des laboratoires de Microsoft.

Parallèlement, Microsoft propose deux types de licence Open source qui ont été approuvées, en octobre dernier, par l'OSI (Open Source Initiative) : Microsoft Public License (Ms-PL) et Microsoft Reciprocal License (Ms-RL) . Elles contiennent très peu d'articles, comparées à la GPL, par exemple. La première fixe les conditions d'utilisation et de distribution d'un code source libre de droit. Tandis que la seconde impose une forme de réciprocité dans la distribution des sources originelles.

Des accords d'optimisation

Enfin, et probablement l'initiative la plus significative est la signature d'accords de collaboration et de partenariat avec de grands acteurs du monde Open Source. Parmi les nombreux accords passés, on trouve par exemple celui de Novell, Zend, ou de JBoss (et tout récemment Talend).

Toutes ces initiatives peuvent surprendre à première vue. Toutefois, Microsoft se devait de réagir car avec le temps, le modèle Open Source est apparu comme durable et perce de plus en plus le monde de l'entreprise. Aussi, plutôt que de s'enfermer, Microsoft choisit de contre-attaquer sur 3 points.

D'abord, l'interopérabilité. Les accords de partenariat et de collaboration, vont tous dans ce sens. Ils vont permettre aux produits Microsoft d'être beaucoup plus interopérables et de rester au centre de l'échiquier. En se servant ainsi de l'engouement pour l'Open Source, Microsoft peut alors continuer à vendre plus sans nécessairement faire de cadeaux aux concurrents, tout en limitant les conséquences potentielles de sa position dominante. L'exemple du projet Renault à base de serveur Novell cité précédemment est sur ce point assez emblématique.

Ensuite, capter la communauté des développeurs autour des produits Microsoft et éviter une hémorragie vers d'autres produits tiers. C'est ici qu'intervient le site Codeplex, qui en reposant sur Team Foundation Server (TFS) participe à faire connaître la plate-forme. Mais aussi la série Express, qui représente une offre concurrente à l'Open source gratuit. Et surtout, la sortie très régulière de nouveautés significatives, dans tous les secteurs (Serveurs, Bureautique, Développement), contraint le développeur à suivre un rythme effréné, lequel se retrouve donc moins disponible pour les alternatives.

Enfin, réduire l'impact marketing et commercial des concurrents en ouvrant sous conditions certaines parties des sources de ses produits, comme c'est le cas dans le match Java/.NET. Il en profite pour améliorer du même coup son image.

En conclusion, il apparaît pour Microsoft que l'Open source est un modèle économique dont se servent ses concurrents pour conquérir des parts de marché. Dans cette concurrence sans merci, la stratégie de Microsoft consiste donc à accompagner le mouvement et offrir des alternatives suffisamment attractives et interopérables pour, au final, continuer à vendre ses produits. L'avenir nous dira donc, si cette stratégie suffira à garantir les positions commerciales du géant de Seattle.

Références

http://www.microsoft.com/resources/sharedsource/default.mspx
http://www.microsoft.com/opensource/default.mspx
http://www.opensource.org/licenses/ms-pl.htm
http://www.opensource.org/licenses/ms-rl.html

Joël Descombes - *Architect .NET - SQLI*

Licences ouvertes : cerner les problèmes et comprendre les enjeux

La question des licences se pose de plus en plus. Les problèmes apparaissent lorsque l'on souhaite respecter les termes de ces licences. Mais cela ne va pas toujours de soi. Nous avons posé quelques questions à Benjamin Jean, juriste au groupe Linagora, et fin connaisseur du sujet.

Programmez !: Cette profusion de licences ne pose-t-elle pas un problème aux développeurs, éditeurs, utilisateurs ? Et les problèmes sont-ils spécifiques à chacun ?

Benjamin Jean : *Du jour où les entreprises se sont intéressées au Logiciel Libre, la multiplication des licences fut impossible à réfréner : chacune y allant de sa propre politique open source, et de sa licence maison. Ainsi, je ne suis pas certain que la multiplication de licences soit la cause de tous les maux : au pire, ce n'est que le catalyseur d'un problème bien plus général qui est celui de l'incompatibilité - lorsque l'on est en présence de licences, souvent similaires, dont les termes respectifs s'opposent d'une façon telle qu'il est impossible de les respecter simultanément. Cette complication avérée, on s'aperçoit qu'elle influe directement sur les développeurs/éditeurs qui sont amenés à renoncer à la combinaison de diverses briques logicielles et sur les distributeurs qui se retrouvent dans l'impossibilité d'exploiter certaines solutions logicielles. Phénomène intéressant, l'Open Source Initiative a récemment entamé une politique de regroupement et de classification des licences : conseillant certaines licences, comme celles " populaires ou largement utilisées ", et celles " destinées à une utilisation spécifique ", et en déconseillant d'autres. À mon sens, si cette initiative était destinée à résoudre les problèmes rattachés à la multiplication des licences, elle supprime l'impartialité de l'OSI à ses débuts. À ce titre, je préfère presque la classification incomplète de la FSF, au moins n'y recherche-t-on aucune objectivité - d'ailleurs, les critères sont loin d'être objectifs : la GNU GPL v3 est-elle redondante à l'égard de la GNU GPL v2 ? Ou l'inverse?*

P ! : Comment distingue-t-on les types de licences ?

B.J. : *Une licence libre, quel que soit son degré de complexité, se conceptualise facilement une fois la logique commune saisie. En schématisant : deux attributs caractérisent les diverses licences : leur type (fonction des droits et obligations), et leur portée. Plusieurs classifications peuvent être retenues afin d'organiser les types de licences. J'aime pour ma part compléter le binarisme classique par un repérage historique. On oppose de manière traditionnelle, sans autre précision, les licences libres " copyleft " aux licences libres " permissives ". Aucun consensus n'existe réellement sur la notion de copyleft, mais on peut légitimement considérer comme telle toute licence assurant la pérennité des libertés/droits accordés : que ce soit en imposant la réutilisation d'une licence spécifique (à l'instar de la GNU GPL), ou, plus souplement, en imposant que la licence finale reprenne ces libertés (à l'instar de la CDDL). À l'inverse, une licence permissive confère le maximum de droits au développeur, qui est libre de redistribuer l'œuvre comme il l'entend. Historiquement, le premier mouvement des licences GNU en engendra d'autres, notamment de la part des universités (avec des licences très permissives) et des communautés. Plus tard, les entreprises souhaitant adhérer au mouvement choisirent de créer des licences correspondant à leurs besoins propres, comme la MPL créée en 1998. Enfin, des licences adaptées aux spécificités des administrations et personnes publiques (citons les licences CeCILL, la licence European Union Public licence, ou encore la Non-Profit Open Software licence) se sont récemment développées.*

Sans prétendre à l'exhaustivité, cette double approche permet de distinguer une licence " communautaire " non " copyleft " comme la licence Apache d'une licence " académique " " permissive " comme la licence BSD : la première comportant tout une série d'obligations en matière de brevets que la seconde ignore. On comprend aussi pourquoi certaines licences, d'origine communautaire, sont si difficile à saisir et à utiliser.

P ! : Pourquoi les éditeurs et organismes inventent-ils de nouvelles licences ?

B.J. : *Cette question est intéressante, et peut aussi être posée différemment : pourquoi les éditeurs et organismes devraient-ils utiliser des licences écrites par des personnes tierces, dont les objectifs peuvent être différents ? Je pense qu'il ne faut pas mélanger l'unité derrière laquelle les licences se rejoignent et par laquelle elles sont dites libres, ou open source, et la diversité des licences, qui peut être créatrice d'incompatibilité lorsque mal gérée. Un exemple parfait est la licence MPL, créée au côté de la NPL lors de la libération par Netscape en mars 1998 de Netscape Communicator 5. Rédigée par Mitchell Baker, cette licence permettait à l'entreprise de trouver un compromis entre sa volonté très pragmatique d'ouvrir son code, tout en assurant une relative sécurité juridique à ses partenaires et actionnaires. D'autres exemples, assez caractéristiques du dernier mouvement, sont les licences CeCILL et l'EUPL (European Union Public Licence, ndlr). La première est clairement créée comme une " GPL française ", adaptée aux particularismes français, dispositions d'ordre public, Loi Toubon, tandis que la seconde est une initiative européenne, destinée à doter l'administration d'un outil correspondant à ses contraintes. Leur spécificité est néanmoins difficilement reprochable puisqu'elles ont comme particularité de conserver une compatibilité à l'égard des autres licences : le contenu soumis à ces licences pouvant ainsi être redistribué sous de multiples autres licences : même si la réciproque n'est pas vraie. Inversement, on assiste à un certain retour en arrière d'entreprises qui, après voir adopté une licence propre pour leur produit, reviennent progressivement sur leur décision en y associant une licence plus connue, voire en changeant de licence au profit de cette dernière. Quant à l'opportunité de créer une nouvelle licence pour la distribution d'un logiciel particulier, dans un contexte particulier : j'avancerais que le problème de la mul-*

tiplicité des licences libres n'en sera plus un lorsque les rédacteurs de licences ne chercheront plus à imposer leur licence, et réfléchiront dans une logique globale. Le phénomène est encore jeune, mais je suis réellement convaincu que le développement du Libre et de l'Open source devra passer par une telle ouverture juridique. À cet égard, il est intéressant de constater que les licences non destinées aux logiciels semblent plus réactives à cette problématique - comme le montrent les efforts de compatibilité introduits par la Licence Art Libre 1.3, la CC-By-SA 3.0, et peut-être la prochaine GNU FDL.

P ! : Quand un développeur veut intégrer différentes briques utilisant des licences différentes, comment appréhender la question de la licence ? Quelle licence prime sur l'autre?

B.J. : *Je vais commencer par la dernière partie de la question, et revenir progressivement jusqu'à la première : oui, il existe un réel risque de conflit inter licence, qu'il convient de gérer et de maîtriser. Ensuite, aucune licence ne prime sur une autre : simple licencié, celui qui souhaite distribuer un logiciel ne peut le faire qu'en respectant l'ensemble des licences auxquelles il est soumis. La raison pour laquelle il est possible de distribuer sous une seule licence un projet comprenant des briques sous d'autres licences, c'est uniquement parce que cette distribution sous licence unique est autorisée par les autres licences présentes. Ainsi, le licencié respecte donc bien toutes ses obligations. De façon pragmatique, lorsque l'on exploite du contenu sous licence libre - en entreprise ou ailleurs -, il est nécessaire d'adapter la procédure de validation juridique et contractuelle : l'analyse devant être réalisée autant en amont des projets qu'en aval. En effet, l'expertise permettra de valider juridiquement la situation découlant du choix technique, et le travail prospectif du service juridique permettra d'envisager les licenciements possibles, ainsi que ceux les plus adaptés au logiciel. La première validation demande un travail conjoint entre l'équipe technique, seule apte à fournir les informations détaillées sur le logiciel, et l'équipe juridique. Les données brutes nécessaires pour l'étude par l'équipe juridique sont : le détail des briques logicielles utilisées, leur licence respective, comprenant leur version, et toute autre information influant sur ces licences : entête restrictif, interprétation, exception, etc., et enfin les interactions existantes entre chacune de ces briques, en pratique, un schéma. Un dialogue s'établit par la suite pour affiner les derniers détails. Si elle peut être contraignante, l'analyse juridique est nécessaire pour assurer la pérennité du logiciel, et l'étude prospective est d'autant plus importante que c'est le seul moment où l'auteur initial est amené à choisir la licence avec une minimum de contraintes, puisque l'accord des contributeurs sera postérieurement indispensable à une éventuelle modification.*

P ! : On évoque souvent ou parfois la question des licences. Constatez-vous encore des questions autour de cela ?

B.J. : *Je pense que les questionnements ne sont aujourd'hui plus les mêmes : les licences libres ne sont plus remises en question, tandis que leur gestion et leur utilisation concentrent tous les regards. Pour cela, les entreprises font très souvent appel à des " experts open source ", et/ou des cabinets d'avocats qui se sont spécialisés dans ces nouvelles problématiques. Ainsi, le principal enjeu est aujourd'hui celui de la diffusion des connaissances et du partage des expériences. De manière générale, je dirais que la situation s'améliore : les intérêts publics comme privés ayant mesuré l'importance de ces connaissances. Enfin, et c'est une des spécificités du phénomène du logiciel libre, les communautés du logiciel libre sont très présentes dans ce domaine, avec des initiatives comme celle de l'association " Veni, Vidi, Libri ", qui ont pour objet de diffuser les connaissances sur les licences et d'en faciliter l'utilisation. J'ajouterai qu'il reste encore une zone d'ombre : toutes les entreprises n'ont pas encore saisi l'importance du respect des licences libres. En effet, il n'y a plus aucune légitimité à exiger le respect d'une licence, d'un contrat, lorsque l'on ne respecte pas soi-même ce dernier - disons que c'est couper la branche sur laquelle on est posé.*

P ! : De plus en plus, les éditeurs et même des entreprises recourent à des avocats spécialisés pour auditer les licences. De quelle manière comprenez-vous cela ?

B.J. : *Ce que je comprends, c'est que le respect des licences libres est dorénavant parmi les priorités des entreprises. La complexité croissante des licences, proportionnée aux utilisations de plus en plus complexes qui sont faites des logiciels, et la volonté d'une assise juridique forte, sont certainement les deux facteurs principaux de ce phénomène. Néanmoins, il ne faut pas négliger deux autres intérêts dans cette externalisation de l'expertise juridique : la mutualisation des compétences, et le transfert de la responsabilité juridique vers ces professionnels. Pour finir, cette tendance n'est que le corollaire de la professionnalisation du secteur, et je ne serai pas étonné de voir se multiplier le nombre d'" experts " dans ce secteur. Une note nostalgique néanmoins, puisque le domaine perd l'une de ses forces : la facilité avec laquelle les auteurs pouvaient mettre leur logiciel sous licence libre. Il faut une équipe de techniciens et une équipe juridique là où il ne fallait qu'un entête... Aux vues des évolutions du secteur, je ne pense pas qu'il faille le regretter, mais il est certainement utile de surveiller ce glissement.*

P ! : La GPL 3 a suscité beaucoup de débats. Pouvez-vous nous expliquer les grands principes de cette licence, les différences par rapport à la v2 et quels conflits peut-elle créer ?

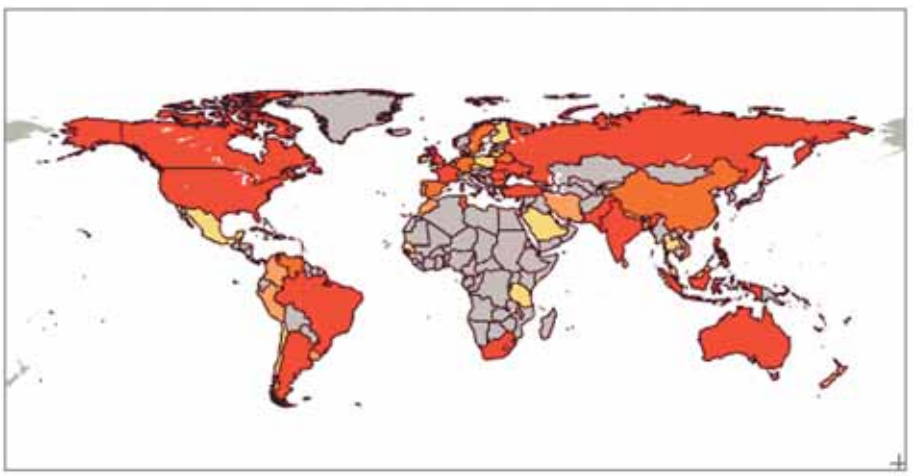
B.J. : *La licence GNU GPL est sortie, complètement rénovée dans sa troisième version, le 29 juin 2007. J'en tire pour ma part un bilan mitigé : la licence se complexifie et cherche à s'étendre à des situations qui n'étaient auparavant pas dans ses fonctions ; parallèlement, la licence prête l'oreille à des reproches qui avaient pu lui être adressés, favorisant par exemple la compatibilité avec les autres licences. Les nouveautés sont nombreuses, et ne peuvent toutes être énumérées dans l'espace réservé. Un rapide tableau : généralisation d'une " obligation de cohérence " : interdit à l'utilisateur de limiter par d'autres voies, ou droits exclusifs concurrents, les libertés qu'il concède par la licence ; un encadrement des accords parallèles portant sur des brevets, en réponse à l'accord Novell/Microsof ; l'intégration d'une modularité, afin de permettre d'intégrer d'autres briques logicielles sous licence a minima aussi libre que ces dernières (créant une compatibilité avec des licences comme la licence Apache, LaTeX, etc. ndlr) ; des adaptations technologiques (comme la distribution par peer to peer, ndlr). Tous les débats ne sont pas encore clos, et beaucoup adoptent une politique de réserve. Un groupe d'experts internationaux est actuellement en train de se réunir, aidé en ceci par la FniLL, dans l'intention de travailler sur ces nouvelles licences.*

■ *Propos recueillis par* **François Tonic**

THE CODING MACHINE : S'inspirer du modèle open source

Au premier regard, The Coding Machine est une petite société composée de 7 ingénieurs en informatique ! Créée il y a trois ans, elle propose des services d'ingénierie dans différents langages à l'image d'une SSII classique.

Et pourtant… The Coding Machine n'a rien d'un " petit ". En réalité, les sept ingénieurs cachent 349 développeurs qui, physiquement parlant, ne sont pas présents dans leurs locaux parisiens. Ils travaillent pour TCM depuis 61 pays. TCM est en contact permanent avec eux par le biais d'Internet et grâce à une plate-forme originale, efficace et moderne, développée par les deux associés de l'entreprise, **Jean-Guillaume Dujardin** et **David Négrier**. L'aventure commence en 2005. Après être passés par des cabinets de conseil (Accenture, Unilog Management et IBM Global Services), les deux associés ont l'intuition que les projets peuvent être réalisés par des développeurs qui ne sont pas nécessairement sur place. Pour cela, il suffit de développer une plate-forme collaborative basée sur le modèle de l'open source. Ce développement prendra six mois et sera pleinement opérationnel à la fin de l'année 2005. Les outils de collaboration en ligne, les réseaux sociaux qui se créent, poussés par la révolution technologique en cours, permettent l'émergence de nouveaux modes de travail à distance. Il est désormais possible de dématérialiser des organisations complexes, en gérant les projets de manière répartie, à l'image de Linux, Wikipedia et bien d'autres. Cette évolution technique permet de réinventer l'organisation d'une SSII, en s'affranchissant des frontières et en cultivant des talents dans le monde entier. Sa volonté est d'organiser les travaux de développement autour d'un chef de projet et d'une communauté de développeurs



Répartition mondiale

Loin des yeux mais pas loin du cœur

TCM s'appuie sur des chefs de projets, capables d'être responsables de l'architecture de la solution et du découpage des projets en composants. Ces composants sont ensuite proposés aux développeurs par le système d'appel d'offres de la plate-forme, la règle de découpage variant selon l'architecture choisie. Le principe adopté est de limiter les interactions entre les composants pour permettre au développeur de tester son composant. Aussi, les entrées et les sorties des composants sont très clairement identifiées. Il s'agit de simplifier la compréhension du projet pour le développeur. Le squelette de l'application est ainsi proposé en téléchargement sur le dépôt Subversion et l'ensemble des objets métiers sont décrits dans les spécifications détaillées. Lors de leur inscription, les développeurs remplissent un CV en ligne et peuvent alors répondre aux offres de la plate-forme qui correspondent à leurs compétences techniques. Après avoir été sélectionné, le développeur réalise le composant. Le chef de projet supporte la communauté du projet durant cette phase. Il apporte des précisions aux spécifications ou bien aide à résoudre des points techniques. Une fois le composant livré, il en contrôle la qualité et l'agrège aux autres. Des liens se tissent naturellement même à distance et, bien que loin des yeux, les membres de la communauté de TCM font partie de l'entreprise à part entière. Pour Jean-Guillaume Dujardin, gérant, *" les bons développeurs se trouvent dans le monde entier. Les sociétés offshore sont là pour nous le rappeler. Aussi, nous n'avons aucune préférence de nationalité. Nous recherchons simplement les meilleurs par domaines de compétence".*

Chef de projet, mais chef d'équipe avant tout

Le chef de projet est l'élément clé de cette nouvelle organisation. Il pilote l'organisation du développement en manager, gérant les demandes du client et les développeurs.

L'E-mail et le chat sont ses principaux moyens de communication. Malgré la distance géographique qui sépare les chefs de projets de ses développeurs, des liens se tissent naturellement. Les développeurs demandent souvent quels sont les feedback des clients ou bien les projets futurs.

Ainsi, David Négrier le Directeur Technique de TCM raconte : *" Un jour, un de nos premiers développeurs, qui est Indien, m'a posé cette question : quel est le temps en Hollande ? En fait, sur un de nos projets, il avait développé des compétences en Ajax, ce qui lui a permis d'avoir une proposition de travail en Hollande. Avec l'argent économisé grâce à son travail*

avec nous, il a même pu se payer son billet d'avion pour se rendre en Hollande. Aujourd'hui, il s'y est installé mais continue de travailler avec nous."

TCM ne propose pas des salaires au rabais. *" Notre modèle permet de bien payer les développeurs avec lesquels nous travaillons. C'est une grande avancée par rapport aux sociétés qui fonctionnent comme une place de marché. Cela nous permet de capitaliser sur les meilleurs développeurs."* explique Jean-Guillaume Dujardin.

En termes de salaire, la principale problématique à laquelle l'entreprise a été confrontée a été de comparer la productivité avec le temps passé. Dans le modèle, la rémunération d'un développeur ne dépend plus du temps qu'il passe sur une tâche mais bien du résultat qu'il produit. Cela avantage les meilleurs développeurs tout en mettant sur un plan d'égalité tous les développeurs de la communauté.

Rendre cohérents les morceaux du puzzle

Cependant, faire appel à une communauté de plusieurs dizaines de développeurs pour réaliser un projet informatique pose des problèmes évidents de cohérence. *" Nous avons dû réfléchir à cette problématique très tôt "* détaille David Négrier, *" nous avons industrialisé l'ensemble de la chaîne de développement. Notre savoir-faire se trouve dans le découpage puis l'agrégation de la solution en composants. Nous devons faire simple et efficace car il est indispensable que les composants fonctionnent bien les uns avec les autres. "*. Par ailleurs, l'utilisation des standards Open Source est quasiment systématique car ils permettent de mettre à disposition des développeurs un environnement sur lequel s'appuyer.

La solution développée par TCM offre un grand nombre d'avantages. Elle permet de réduire les délais en garantissant un développement des composants en parallèle sans les contraintes d'affectation de ressources. Le coût du développement est largement inférieur aux tarifs pratiqués par les SSII classiques, de 30 à 50 % moins important. Les chefs de projets développent une grande expertise dans la mise en place d'architecture de qualité en s'adressant à des développeurs très spécialisés qu'il serait impossible d'engager aussi nombreux et aussi compétents dans une SSII traditionnelle. Des limites existent. Par exemple, les documents de conception sont nécessairement très détaillés car ce document précise la prestation du développeur. Le temps passé à leur préparation est plus important que celui nécessaire à la mise en place d'une mission classique en France.

Enfin, le risque que certains développeurs s'arrêtent en cours de développement ou livrent un développement de très mauvaise qualité, est anticipé de manière simple : *"les développeurs avec lesquels nous travaillons régulièrement sont positionnés sur les éléments les plus critiques, tandis que les nouveaux sont testés sur les composants les plus simples. Si un de nos nouveaux développeurs nous fait défaut, nous pouvons facilement réattribuer ce composant à un autre développeur qui est en avance "* explique David Négrier.

Les projets

En moins de deux ans, plus de vingt projets ont été développés sur des technologies objets, principalement PHP et Java, et sont d'une taille variant de 30 à 300 jours homme. Ils vont de la mise en place d'une architecture SOA au streaming vidéo sur le web.

Pour faire vivre cette communauté de développeurs, The Coding Machine innove aussi en matière logicielle. *" Les développeurs sont souvent beaucoup plus sensibles à la technologie développée qu'au prix. C'est une des raisons qui ont fait que nous cherchons à développer les projets très innovants qui vont nous permettre de fidéliser les développeurs auxquels nous tenons ",* explique Jean-Guillaume. La dernière innovation en date est le développement d'un Framework Open Source reverse Ajax / PHP, THE XAJA MACHINE. C'est un Framework de type COMET qui permet de faire du push sur les navigateurs depuis le serveur. Le premier projet est un conseiller vidéo web qui peut être sollicité par l'utilisateur et qui " chat " en direct.

De la même manière que l'Open Source a bouleversé le monde de l'édition logiciel, The Coding Machine transforme le monde de l'ingénierie. Les échanges sont dématérialisés, répartis dans le monde entier et basés sur la confiance.

Jean Vidames

Les autres types d'organisation

La place de marché

Modèle phare du début du web, les places de marchés étaient censées tout révolutionner y compris dans l'organisation du travail. Le modèle proposé par Elance, GetACoder, ou bien encore RentACoder est simple. Il propose de mettre en relation des clients qui ont des besoins avec des développeurs freelance. A la manière d'eBay, ces sites reposent sur la réputation à la fois du client et du développeur. Les meilleurs développeurs peuvent prétendre à une meilleure rémunération et, de manière symétrique, les meilleurs clients peuvent espérer un prix de projet moins élevé. Ces deux facteurs devant se compenser pour permettre de définir harmonieusement un équilibre de marché.

Cependant, les écueils de ce modèle sont nombreux à la fois pour les développeurs et pour les clients. Pour les développeurs, les prix sont finalement tirés vers le bas ce qui rend difficile de rendre un travail de qualité pour une somme souvent ridicule. Et pour le client, l'issue de son projet est très incertaine, personne ne lui garantissant l'aboutissement du projet ou la simple disponibilité pour la correction des anomalies.

Au-delà, ce modèle pose le problème des projets d'envergure : les projets qui dépassent les 50 jours de développement sont difficilement réalisables dans ce mode d'organisation.

La compétition

Le modèle de Topcoder est assez intéressant. Il propose une bibliothèque de composants et fait évoluer cette bibliothèque en fonction des demandes des clients. Ces nouveaux composants sont développés sur le mode de la compétition. Le meilleur développeur gagne un prix. Les problèmes associés à ce modèle sont assez évidents pour le développeur. Il risque tout simplement de ne rien gagner, d'avoir travaillé pour rien. Pour les clients, le problème est de travailler sur un existant sans tout changer et de s'associer à une société qui est plus un éditeur qu'une société de services.

Référence

The McKinsey Quarterly - Eight business technology trends to watch – Décembre 2007
http://www.mckinseyquarterly.com/article_abstract_visitor.aspx?ar=2080&l2=13&l3=11

Géo localisation et géométrie avec les bases de données = Index Spatiaux

Un serveur de base de données sert habituellement à manipuler des données simples telles que des chaînes, des nombres, etc. Mais l'orientation prise par les différents moteurs de base de données semble aller en faveur de briques spécialisées dans le traitement de certaines données. Pourquoi et comment un moteur de base de données peut-il nous aider dans ces tâches ?

Pourquoi les index spatiaux ?

Regardons le rôle d'un moteur de base de données. Il sert à la fois au stockage des données et à leur traitement. Il est très efficace quand il s'agit de traiter une masse de données en bloc aidé en cela par des outils tels que les index.

L'index, à l'instar de ceux des livres, vous permet de ne pas lire l'intégralité du livre pour y trouver les informations recherchées. Prenons une carte : combien de temps vous faut-t-il pour trouver une ville sur une carte ? Et maintenant en connaissant la case où se trouve la ville sur cette même carte ? Voilà ! Vous venez de trouver l'intérêt des applications spatiales dans les moteurs de base de données.

Le principe n'est pas nouveau, il s'agit de stocker des informations dans un espace de stockage (les index étant généralement représentés par des arbres) pour filtrer le contenu d'un très gros volume de données. L'originalité vient du fait qu'au lieu de traiter des données de table (à une seule dimension) vous aurez deux dimensions à traiter.

Le laboratoire de recherche de Microsoft avait déjà mis en application ce principe sous SQL Server 2005 (à lire ici : http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-2005-122) pour de simples points. Les applications exemple vous permettent d'utiliser ce principe.

Microsoft n'est d'ailleurs pas le premier à entrer sur ce terrain, Oracle met déjà à disposition dans son moteur les index spatiaux sous forme d'option. La différence d'approche est assez notable, Oracle mise sur un module supplémentaire avec index spécifiques, alors que Microsoft exploite l'existant du moteur avec des index B-Tree et tables relationnelles classiques et les types .Net. C'est sans doute là le meilleur exemple de l'intégration de .Net à SQL Server.

Qu'est ce que nous apporte SQL Server 2008 ?

SQL Server 2005 fournit comme exemple le projet de recherche du laboratoire de recherche du groupe. C'est le premier pas qu'a su exploiter SQL Server 2008 en intégrant nativement ce type… En fait **ces** types puisqu'en passant, il est maintenant possible d'y stocker des points (comme le projet dans 2005), des lignes, des polygones, etc. Grâce à la levée de la taille limite des types .Net (passée à 2Go dans cette version). Autre nouveauté, SQL Server 2005 exploite dans son implémentation les données sur sphère, tandis que 2008 permet aussi le choix entre le mode sur plan ou sur sphère. Alors la Terre est ronde (type **geography**) ou plate (type **geometry**) ? Je ferais une réponse d'architecte : tout est question de taille, du type de données à traiter. Bien entendu la courbure de la Terre n'est pas à prendre en compte relativement à la taille de votre bureau, mais au niveau d'une ville, elle est impérativement prise en compte. Les index sont donc créés par dessus ces types et utilisent des tables internes permettant les requêtes de type : telle forme coupe-t-elle telle autre ? Quelle est la distance entre cette forme et celle-ci ? Mon point est-il inclus dans cette forme ? Etc. SQL Server 2008 apportera aussi ultérieurement l'intégration avec Virtual Earth (récupération de coordonnées par Web Services depuis le moteur ?).

Quels scénarios ?

La géo localisation bien sûr, c'est ce qui vient le plus rapidement à l'esprit, cela parle à tout le monde avec des exemples concrets. Mais pas uniquement, dans sa version SQL Server 2005, Microsoft Research l'a mise en place pour un site de catalogue de constellations (à tester ici : http://cas.sdss.org/dr6/en/).

En allant plus loin, tout ce qui est du domaine de la géométrie, de gestion de distances ou d'intersection d'éléments est utilisable. Prenez 2 axes, des données générant des points positionnés à l'intersection de valeurs sur ces 2 axes, et vous avez une application spatiale.

Exemple : je prends un axe " âge en mois ", et un autre de type " temps pour courir un 100 mètres ". Je crée des points grâce à ces données et je requête ces derniers dans un rectangle par exemple (forme plus complexe possible suivant votre imagination) pour déterminer les sportifs

Pour les .Net *addicted*

Demandez-le à votre marchand de journaux

de haut niveau de ma base de données. Je reprends un exemple souvent cité par l'un de mes chefs, regardez la taille des slips en magasin, vous avez un axe taille, un axe poids et un polygone qui marque la zone correspondant à votre taille de slip.

Exemple classique de géo localisation

Pour commencer, quelques pièges à éviter :

- Les cartes sont toutes des projections d'une partie d'une sphère sur un plan, il y a donc des erreurs en termes de mesures de distances induites par ces projections.
- Il est impossible de modéliser à l'aide de lignes ou de points la totalité du contour des côtes d'un pays, cette configuration étant infinie (les fractales vous connaissez ?).
- Attention à la notion d'être *à l'intérieur* ou *à l'extérieur* d'une zone (vous êtes peut-être actuellement en France, mais en dehors du reste du monde).

Une fois ces quelques notions connues, commençons par une table :

```
CREATE TABLE Clients
(
   id int identity(1,1) primary key,
   nom varchar(50) NOT NULL,
   localisation geography NOT NULL
)
```

Insérons quelques valeurs dans celles-ci (2 personnes totalement fictives) :

```
INSERT INTO Clients(nom, localisation)
VALUES ('Christian','POINT(48.834943 2.244349)'),
       ('Bill', 'POINT(47.640584 -122.129055)')
```

Notez en passant l'utilisation de l'insertion multilingue dans la clause INSERT disponible dans SQL Server 2008.
Maintenant je souhaite connaître la distance séparant Bill de Christian :

```
DECLARE @var1 geography, @var2 geography

SET @var1 = (SELECT localisation FROM dbo.Clients WHERE id = 1)
SET @var2 = (SELECT localisation FROM dbo.Clients WHERE id = 2)

SELECT @var1.STDistance(@var2)
```

Résultat en mètres : 8 050 538,90030601
Jusqu'ici pas d'index, la simplicité des calculs fait que ce n'est pas nécessaire. Admettons maintenant que je souhaite trouver qui habite dans un rayon de 1000 km autour de chez moi. J'ai bien entendu ajouté un peu plus de données que tout à l'heure.

```
DECLARE @var geography

SET @var = (SELECT localisation FROM dbo.Clients WHERE id = 1)

SELECT * FROM dbo.Clients WHERE localisation.STDistance(@var) < 1000000
```

Il me trouve, moi, bien sûr, dans le résultat de la requête et d'autres (tout dépend du volume de données que vous avez inséré dans la base de données).
On peut bien entendu exploiter les données calculées :

```
DECLARE @var geography

SET @var = (SELECT localisation FROM dbo.Clients WHERE id = 1)

SELECT nom, localisation.STDistance(@var) AS distance
FROM dbo.Clients WHERE localisation.STDistance(@var) < 1000000
ORDER BY distance
```

Auquel cas je trouve les personnes par ordre de proximité.
Je monte le nombre d'enregistrements présent dans ma table à cent dix mille environ. Je requête autour de mon domicile à 100 km, la requête met 8 secondes à s'exécuter. Maintenant ajoutons notre index :

```
CREATE SPATIAL INDEX Idx_Spatial_Loc
   ON dbo.Clients(localisation)
```

Tout simple, et maintenant la requête précédente s'exécute en moins d'une seconde ! Si le volume de données augmente encore il faudra passer par un index supplémentaire basé sur cet index principal.
Comment fonctionne l'index ? Il crée une grille, cette grille permet de découper ici la surface de mon globe, de faire une requête exclusivement à base de distances permet via l'index de supprimer énormément de possibilités d'enregistrements dans mon jeu de résultat. A l'instar d'un index classique, le fait d'augmenter mon rayon de recherche abandonnera l'utilisation de l'index et mon temps d'exécution reviendra à 8 secondes.
Maintenant, requêtons tous les points situés dans un polygone (celui de l'exemple est un triangle Paris, Londres, Bruxelles).

```
DECLARE @g geography
SET @g = geography::STPolyFromText('POLYGON((48.863558 2.34375,
 50.846108 4.356299, 51.52015 -0.101723, 48.863558 2.34375))',
 4326);

SELECT nom, localisation.STDistance(@g) AS distance, localisation.ToString()
FROM dbo.Clients WHERE localisation.STDistance(@g) < 1
ORDER BY distance
```

Vous trouverez alors tous les points présents dans cet espace. Mais comme dit plus haut, faites attention au sens du Polygone. Si en effet vous inversez juste les 2 premiers points, le moteur génèrera une erreur car vous inversez le polygone et celui-ci couvrira plus de la moitié de la planète. Notez le ToString() pour afficher la chaîne, sinon vous aurez à faire à la valeur binaire du type.

Nouvelles applications

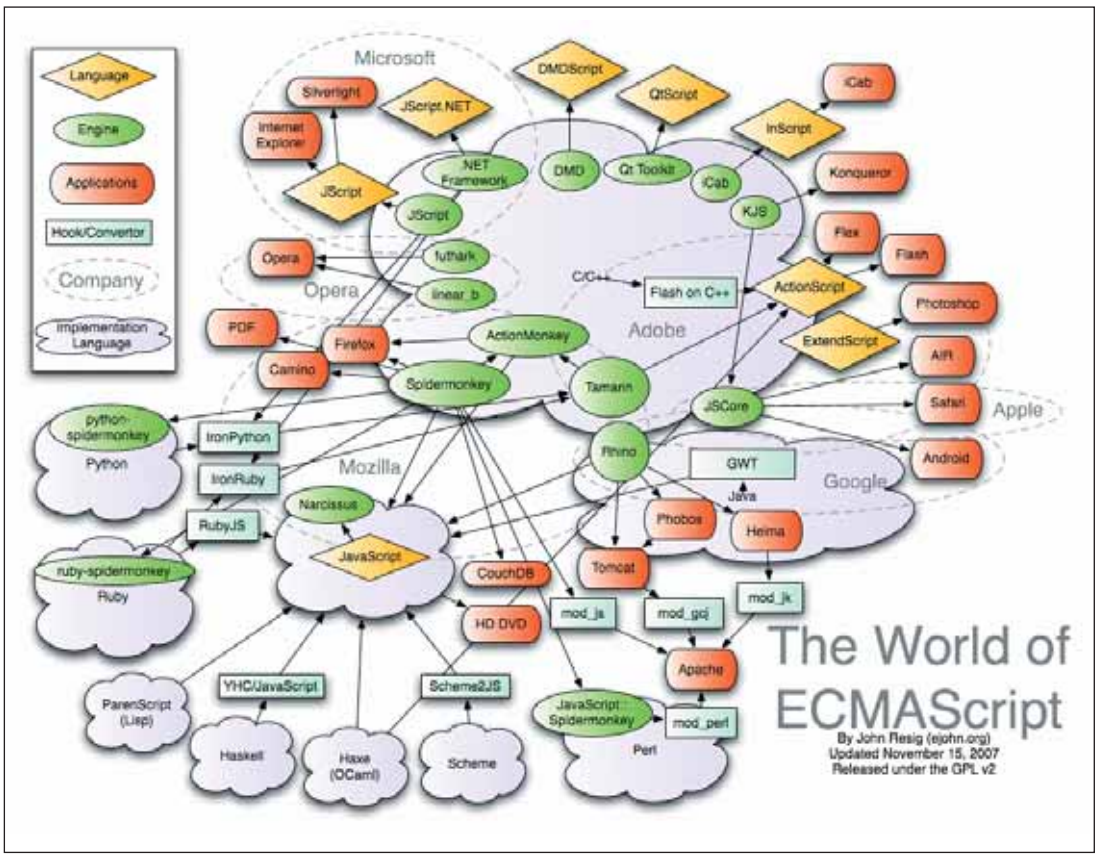
Maintenant vous êtes prêts à vous lancer dans de nouveaux types d'applications, testez cela, le potentiel est énorme et pas seulement restreint à la géographie ! Avec l'intégration de Virtual Earth et peut être la possibilité d'utiliser des cartes dans Reporting Services la solution devient complète, et à mettre dans toutes les mains.

Christian ROBERT - *Winwise*
Consultant - Formateur - MVP System Server - SQL Server
Email : christian.robert@winwise.fr
Blog : http://blogs.codes-sources.com/christian/
Site : http://www.winwise.fr

Le futur de Javascript

Java, Javascript, Jscript, ActionScript... les raisons de faire des confusions ne manquent pas au royaume de Javascript. La raison de ce désordre apparent est simple : depuis 1995, une foultitude d'acteurs enthousiastes y a apporté ses innovations pour donner peu à peu naissance à un enfant monstrueux, incompréhensible et fascinant, décrit par John Resig (créateur de la célèbre bibliothèque Ajax Jquery) sous la forme d'un schéma déjà célèbre, " The World of EcmaScript ".

Pour y voir plus clair, il faut s'intéresser à la base : ECMAScript, un langage de type script, développé dans le cadre des travaux de l'organisme de standardisation qu'est l'ECMA. Il s'agit donc d'un standard dont les spécifications sont implémentées dans de nombreux langages de script. Cela signifie en clair qu'ECMAScript est censé être le socle commun sur lequel s'appuient, entre autres, Javascript et ActionScript. Et c'est là que les choses se compliquent un peu...



L'univers d'Ecmascript selon John Resig

Plusieurs implémentations : Javascript, Jscript, ActionScript,...

Si vous codez un peu pour le Web, vous avez déjà probablement fait la désagréable expérience de ce superbe code qui fonctionne à merveille sur un navigateur et pas du tout sur un autre. Il y a de multiples raisons à cela, mais l'une d'entre elles est certainement que les navigateurs en question n'utilisent pas les mêmes langages. Ainsi l'implémentation d'Ecmascript utilisée par Internet Explorer se nomme Jscript et est différente du Javascript de Firefox ou Safari. Pire : au sein du même langage, des différences de version peuvent poser problème. On rencontre ainsi communément, des versions de Javascript allant de la 1.5 à la 1.8 dans les navigateurs modernes.

ActionScript, le langage de développement de Flash est également une implémentation d'Ecmascript. De ce fait, les syntaxes d'ActionScript et de Javascript tendent à converger au fur et à mesure de la sortie de nouvelles versions, ce qui, cette fois, est plutôt une bonne nouvelle pour les développeurs Web !

Les navigateurs et leurs moteurs d'exécution

Les choses ne seraient finalement pas si compliquées si les problèmes s'arrêtaient aux compatibilités entre les langages de script. Mais, même au sein du même langage d'une même version, le développeur peut encore avoir des surprises. En effet, chaque navigateur Web embarque un moteur d'exécution qui lui est propre. Ils ont parfois un petit nom (Jscore pour Safari, SpiderMonkey pour Firefox, KJS pour Konqueror), et toujours des particularités : performances, bugs, fonctionnalités supplémentaires. Autant d'éléments sources de casse-tête, mais aussi d'opportunités pour le développement Web, comme lorsque Firefox propose des " supercookies ", non limités en taille pour stocker des données de manière persistante sur le poste client.

Le contexte : HTML et ses évolutions

Javascript ne servirait pas à grand-chose si on ne pouvait pas manipuler la structure d'une page HTML avec. Or, à l'heure actuelle, les travaux du W3C et du WHATWG sur HTML5 influencent grandement les moteurs de rendu HTML des navigateurs Web... et par là même les possibilités de Javascript. Ainsi, la balise <canvas> permettant de réaliser des graphiques modifiables en Javascript est interprétée sous Safari, Opera et Firefox, alors qu'elle ne figure dans aucune version officielle d'HTML !

Ce qui va changer dans le langage

Une brève histoire d'Ecmascript

En décembre 1995, Netscape et Sun annoncent la sortie de Javascript implanté dès mars 1996 dans Netscape Navigator 2.0. En pleine guerre des navigateurs, Microsoft met alors au point Jscript pour Internet Explorer 3.0, sorti en août de la même année. En réponse, Netscape soumet Javascript à l'ECMA pour standardisation, afin d'asseoir sa légitimité : en juin 1997, ECMAscript est né.

Malgré une version 3 d'ECMAscript fin 1999 comblant un certain nombre de lacunes du langage (gestion des exceptions par exemple), Javascript souffre d'une mauvaise réputation : compatibilité Netscape/Internet Explorer, sécurité, langage peu structuré... Il ne reviendra en grâce qu'avec l'avènement d'AJAX et des RIA à partir de 2004.

Dès lors, on s'intéresse à nouveau à ECMAscript et l'on reparle des évolutions à donner au langage : le 22 octobre 2007, le groupe de travail de l'ECMA traitant de la question sort un livre blanc sur ECMAscript 4 (dont Javascript 2.0 sera une implémentation) et prévoit une validation des spécifications pour octobre 2008.

Enfin un vrai langage ?

ECMAscript 4 se montre résolument ambitieux et compte visiblement en finir avec sa réputation de sous-langage. Il introduit pour cela des spécifications attendues depuis longtemps :

- l'apparition des notions de packages et de namespaces qui permettront aux développeurs de rendre leur code plus aisément modulaire ;
- la possibilité de recourir à de véritables types de données, ce qui constitue pour le coup une petite révolution : Javascript 1.5 ne disposait que de types de données implicites au nombre de 10... et au comportement parfois déroutant !
- des outils pour coder de manière plus claire et plus concise : itérateurs, générateurs...
- un design pensé pour les performances. En effet, si les moteurs d'execution javascript ont progressé ces dernières années, la grande permissivité du langage posait un certain nombre de limites aux améliorations de performances : nous devrions voir apparaître des gains spectaculaires en la matière à partir de 2009.

Un vrai langage objet ?

Une des principales surprises d'Ecmascript 4 est sans aucun doute l'apparition de caractéristiques d'un langage orienté objet classique... assez inattendues pour un langage à prototypes. Petit rappel : dans un langage à prototypes, on ne distingue pas classe et instance de classe (les objets), tout est instance. Il est malgré tout possible de faire " hériter " une instance d'une autre en définissant un objet père comme prototype de la fonction constructeur fille, comme dans l'exemple suivant :

```
function Papa() { //mon " instance-classe " Papa
  this.parler = function() { //une méthode de mon objet
    alert('papa !');
  }
};

function Fille(){ //mon " instance-classe " Fille
};

Fille.prototype = new Papa(); //le constructeur de Fille " hérite " de Papa
par prototype

var kevina = new Fille(); // " j'instancie " maintenant une fille... en réalité,
cette instanciation n'est qu'un clonage
kevina.parler(); // son père lui a appris à dire papa !
}
```

Il sera maintenant possible de réaliser un mécanisme équivalent de manière plus courante pour les développeurs habitués aux langages à classes :

```
class Papa { //j'ai cette fois-ci une vraie classe Papa
   function parler() {
        alert('papa !')
 }
};

class Fille extends Papa{ //et une vraie classe Fille héritant de Papa
};

var kevina = new Fille(); //j'instancie maintenant réellement une fille
kevina.parler(); //elle dit encore papa !
```

Cette nouvelle possibilité n'apporterait pas beaucoup si elle n'était accompagnée d'un certain nombre de mécanismes qui composent un bon langage à classes : attributs de propriété, propriétés virtuelles, reflexion, propriétés statiques, interfaces... Vous n'aurez plus aucune excuse pour ne pas architecturer vos développements !

Les conséquences pour le développeur

Maintenant que nous en savons un peu plus sur ce qu'apportera la prochaine mouture d'Ecmascript, essayons de voir ce que cela va changer dans nos développements web. Subissant au quotidien les affres de la loi de Murphy, tout développeur va commencer par se demander quels problèmes de compatibilité vont engendrer cette nouvelle version et les changements majeurs qu'elle apporte. Réponse de **Brendan Eich** de chez Mozilla, créateur de Javascript : *" aucun "* ! Réponse de Chris Wilson, responsable d'Internet Explorer chez Microsoft : *" des tas "* ! La guerre des navigateurs reprendrait-elle de plus belle ?

Brendan Eich, le créateur de javascript

A la lecture des documents produits par l'ECMA, il semble néanmoins qu'on puisse être rassuré de ce point de vue : on constate un souci constant de compatibilité ascendante (que les moteurs ES4 puissent interpréter du code ES3) dans la mise en œuvre des nouvelles caractéristiques du langage. Au total, 12 véritables incompatibilités ont été relevées à ce jour et ne concernent que des aspects bien particuliers du langage : l'immense majorité des scripts actuels devrait donc tourner sur les prochaines générations de moteur Ecmascript.

Passons aux points positifs. Les efforts considérables portés sur les possibilités de bonne architecture des développements (classes, packages, namespaces), associés aux améliorations de performances liés au nouveau design, vont faire de Javascript un langage omnipotent. Les calculs complexes (permettant par exemple la 3D), tout comme les

développements " lourds " (que ce soit côté client ou serveur via un projet comme Rhino) lui deviendront accessibles... en un mot : Javascript aura tout pour être le langage incontournable de demain !
Malheureusement, la plupart des développeurs Web ne devraient pas bénéficier de ces avantages avant quelques années : il faudra en effet que les navigateurs intégrant les moteurs de nouvelle génération fassent leur apparition (Firefox 4, Internet Explorer 8 ?), puis que ces derniers aient une part de marché suffisante pour que les développements basés sur les nouvelles fonctionnalités d'Ecmascript soient rentables.

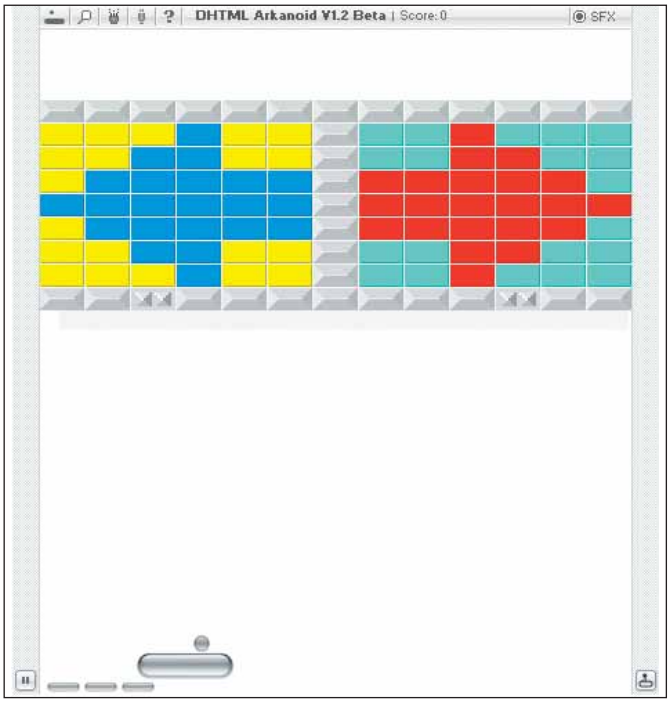
Javascript pour tout et partout

Javascript dépasse le navigateur

Le temps où Javascript était cantonné au navigateur est définitivement révolu. Pourtant, dès le départ, Sun et Netscape avaient eu l'ambition de l'utiliser en tant que langage serveur sur iPlanet ou SunOne... tout comme Jscript peut être utilisé sur Microsoft IIS, voire directement sur une plate-forme Windows via WHS ! Ces utilisations restent néanmoins anecdotiques par rapport à ce que nous promet le futur.
Les Rich Desktop Applications (RDA, parfois inclues dans la grande famille des RIA, Rich Internet Application) sont emblématiques de cet avenir. En essayant de réconcilier le meilleur du Web et du desktop, elles ouvrent la porte à de nouvelles applications : plus riches, plus fonctionnelles, fonctionnant indifféremment en live ou en mode déconnecté... autant de promesses qui constituent déjà un champ de bataille pour Mozilla (XUL), Adobe (AIR), Microsoft (WPF). Malgré l'esprit radicalement différent de chacune de ces plates-formes, Javascript y est toujours utilisable, voire incontournable, en tant que langage de script. De même, Javascript est d'ores et déjà le langage de développement universel pour la création de widgets desktop, qu'il s'agisse de Yahoo!, Google, MacOS ou Windows Vista.
Avec l'arrivée d'Ecmascript4, l'ambition de faire de Javascript un langage serveur reparaît également. Rhino est ainsi un moteur écrit en Java destiné à exécuter des scripts Javascript sur un serveur. Les projets Helma et Phobos utilisent par exemple Rhino pour offrir aux développeurs web le même langage côté client et serveur.

Javascript dépasse le PC

Dans son article " What is web 2.0 ", Tim O'Reilly prédisait déjà que les applications de demain devraient " se libérer du PC ". C'est ce que Javascript a commencé à véritablement faire cette année grâce à l'incontournable iPhone qui a pris le parti du tout DHTML pour ses applications. L'incontestable réussite de Safari et de son support Javascript a d'ailleurs remis un coup de projecteur sur les moteurs Javascript des navigateurs pour mobiles.
Android, le projet de plate-forme mobile ouverte de Google a fait le choix d'un navigateur basé sur Webkit (qui sert également de base à Safari) et aura lui aussi une bonne prise en charge de Javascript. Un nombre grandissant de navigateurs pour mobiles comporte désormais des moteurs Javascript dignes de ce nom.
Sans être exhaustif, on peut citer : Blazer 4 (PalmOS), Internet Explorer Mobile (Windows Mobile), Netfront 3.5, Opera Mobile et Opera Mini, Minimo 0.2 (de la fondation Mozilla), Openwave " Mercury Edition ", le navigateur du S60 de chez Nokia et WebUI de chez Motorola (l'un et l'autre basés sur webkit)... bref, n'importe quel mobile haut de gamme moderne devrait maintenant savoir exécuter une application Web massivement enrichie par Javascript.



DHTML arkanoid, un jeu 100% JS, fonctionne sur l'iPhone !

Mieux : les développeurs de navigateurs proposent déjà des produits sur tout un panel d'appareils en dehors du monde habituel du PC.
Les consoles de jeux (Wii, DS, PSP, PS3), le fameux Kindle ou encore certaines télévisions sont aujourd'hui à la portée de notre cher petit langage...

Javascript sans limite

Javascript est probablement le langage contre lequel les développeurs ont le plus pesté au monde... au point d'avoir pendant une époque milité pour sa disparition ! Mais l'avènement d'AJAX, à travers des applications telles que Gmail, nous a brusquement ouvert les yeux : nul salut pour le Web riche sans Javascript. Tout s'est alors inversé. Plutôt que de dresser la liste des raisons pour lesquelles il fallait se débarrasser de Javascript, chacun s'est mis à imaginer comment faire avec Javascript tel qu'il était. Les librairies comme Prototype ou Dojo nous ont par exemple montré comment simuler des classes et des namespaces avec ECMAScript3, Google Gears et Firefox 3 comment stocker des données de manière persistante côté client, Safari et sa balise canvas comment envisager de nouvelles manières de réaliser des fonctions graphiques avancées... Adobe et Mozilla se sont même lancés sur la piste de la compilation Just In Time, avec le moteur d'exécution Tamarin qui promet de booster les performances tant décriées du langage.

La roue a donc bel et bien tourné pour le langage honni des années 2000. Au point que la révolution Ecmascript4 et l'invasion de Javascript sur des supports chaque jour plus nombreux font maintenant dire à certains qu'il pourrait être le nouveau langage de référence. Comme le furent C et Java en leur temps... étonnant, non ?

Jean-Baptiste Boisseau
Eutech SSII : http://www.eutech-ssii.com/
Le Web 2.0 c'est pas du buzz : http://blog.eutech-ssii.com/

À la découverte de KDE 4 !

KDE 4 est disponible depuis quelques semaines. Les développeurs sont parvenus au mois de janvier à boucler le long travail de développement. Ce cycle décrit comme intensif par les contributeurs du projet a porté ses fruits. Voyons ensemble les nouveautés et la partie développement.

KDE 4 a été en grande majorité réécrit pour bénéficier d'un système d'affichage plus performant et moins gourmand en mémoire. Pour y parvenir, les développeurs ont notamment utilisé Qt 4, la librairie graphique de Trolltech. Les éléments principaux Kicker, KDesktop, SuperKaramba ont également fusionné au sein de Plasma, le fer de lance du nouveau KDE. Parmi les avancées, ont peut également remarquer la présence d'un outil de recherche instantanée sur le poste de travail, ce qui n'est pas sans rappeler Windows Vista et Mac OS X. Les nombreux effets graphiques viennent confirmer cette volonté d'offrir une ergonomie de haut niveau aux utilisateurs. Au delà de l'interface, KDE intègre désormais une application d'intégration du matériel appelée Solid. Celle-ci tente d'uniformiser la prise en charge des périphériques via une interface unique. Enfin, Phonon marque le pas quant à l'intégration des média au sein des systèmes Linux. Nous allons traiter dans le détail les principales nouveautés à disposition du développeur. Nous allons notamment nous intéresser à des cas concrets qui pourront peut être déclencher chez vous des inspirations…

Le bureau Plasma

L'application de gestion de bureau utilisée depuis KDE 1 disposait d'une architecture rigide ne permettant le développement d'applet qu'en utilisant le langage C++. Avec le temps, la nécessité de repenser le bureau est apparue de plus en plus évidente pour l'équipe de développement. Au même moment, les utilisateurs font clairement savoir qu'ils souhaitent une interface plus ergonomique et plus esthétique. C'est pour cette raison que le projet Slicker a vu le jour afin d'apporter une alternative dynamique et évolutive à Kicker qui était imposé avec KDE.

Plasma est le projet majeur de KDE 4. Celui-ci se base en partie sur le projet Slicker et remplace définitivement Kicker, KDesktop et SuperKaramba. Ce dernier est l'application qui permet l'affichage de widget sur le bureau. KDesktop était quant à lui le gestionnaire de fond d'écran. Désormais, Plasma sera la seule interface pour les développeurs. Les Applets pourront être programmées avec de nombreux langages, dont C/C++, Java, Python et Ruby.

Développer un widget

Pour cet exemple, nous allons reprendre le tutorial officiel du wiki. Il consiste en la création d'un widget qui va simplement afficher une image, une icône et un peu de texte. Pour commencer, nous allons créer le fichier .desktop. Celui ci définit la façon dont Plasma doit lancer le widget ainsi que quelques propriétés, comme son nom et les informations sur son auteur.

plasma-applet-tutorial1.desktop

```
[Desktop Entry]
Encoding=UTF-8
Name=Tutorial 1
Comment=Plasma Tutorial 1
Type=Service
ServiceTypes=Plasma/Applet

X-KDE-Library=plasma_applet_tutorial1
X-KDE-PluginInfo-Author=Bas Grolleman
X-KDE-PluginInfo-Email=bgrolleman@emendo-it.nl
X-KDE-PluginInfo-Name=plasma_applet_tutorial1
X-KDE-PluginInfo-Version=0.1
X-KDE-PluginInfo-Website=http://plasma.kde.org/
X-KDE-PluginInfo-Category=Examples
X-KDE-PluginInfo-Depends=
X-KDE-PluginInfo-License=GPL
X-KDE-PluginInfo-EnabledByDefault=true
```

Nous pouvons à présent commencer à programmer notre widget. Tout d'abord, nous allons créer le fichier d'en-tête. C'est dans celui-ci qu'est définie notre classe. Ici, il n'y a rien de particulier. Nous définissons notre classe qui dérive de la classe Applet de la librairie Plasma. Nous retrouvons en revanche nos deux éléments m_svg, m_icon qui sont respectivement l'image et l'icône de notre application. Le texte sera affiché dynamiquement.

plasma-tutorial1.h

```
#ifndef Tutorial1_HEADER
#define Tutorial1_HEADER

#include <KIcon>

#include <Plasma/Applet>
#include <Plasma/Svg>

class QSizeF;

// Definition de l'applet Plasma
class PlasmaTutorial1 : public Plasma::Applet
{
    Q_OBJECT
    public:
        // Simples constructeur et destructeur
        PlasmaTutorial1(QObject *parent, const QVariantList &args);
        ~PlasmaTutorial1();

        // La fonction paintInterface affiche l'applet à l'ecran
        void paintInterface(QPainter *painter,
                const QStyleOptionGraphicsItem *option,
                const QRect& contentsRect);
        void setContentSize(const QSizeF& size);
        QSizeF contentSizeHint() const;

    private:
        Plasma::Svg m_svg;
        KIcon m_icon;
    QSizeF m_size;
};

// Permet de lier notre applet au fichier .desktop
K_EXPORT_PLASMA_APPLET(tutorial1, PlasmaTutorial1)
#endif
```

Nous pouvons écrire la partie fonctionnelle du code. Le constructeur initialise les attributs de la classe et notamment la taille de l'applet (200x200 pixels). La fonction la plus importante est PaintInterface. C'est celle-ci qui va, à proprement parler, afficher le contenu du widget. Il s'agit de code Qt. Les habitués de l'API ne seront donc pas dépaysés.

plasma-tutorial1.cpp

```
#include "plasma-tutorial1.h"
#include <QPainter>
#include <QFontMetrics>
#include <QSizeF>

#include <plasma/svg.h>
#include <plasma/theme.h>

PlasmaTutorial1::PlasmaTutorial1(QObject *parent, const QVariantList &args)
    : Plasma::Applet(parent, args),
    m_svg("widgets/background", this),
    m_size(200,200),
    m_icon("document")
{
    // active l'affichage du fond par défaut
    setDrawStandardBackground(true);
    m_svg.setContentType(Plasma::Svg::SingleImage);


    // Démonstration de l'utilisation de la fonction setFailedToLaunch
    // Permet la gestion des erreurs
    if (m_icon.isNull()) {
        setFailedToLaunch(true, "Impossible de charger l'icone");
    }
}

PlasmaTutorial1::~PlasmaTutorial1()
{
    if (failedToLaunch()) {
        // Le code de nettoyage avant de quitter
      // ...
    } else {
        // Sauvegarde des paramètres
      // ...
    }
}

void PlasmaTutorial1::setContentSize(const QSizeF& size)
{
    m_size = size;
}

QSizeF PlasmaTutorial1::contentSizeHint() const
{
    //Si vous souhaitez changer la taille de l'applet, mettez à jour m_size
et appelez la fonction updateGeometry()
    return m_size;
}
```

```
void PlasmaTutorial1::paintInterface(QPainter *p,
        const QStyleOptionGraphicsItem *option, const QRect &contents
Rect)
{
    p->setRenderHint(QPainter::SmoothPixmapTransform);
    p->setRenderHint(QPainter::Antialiasing);

    // Now we draw the applet, starting with our svg
    m_svg.resize((int)contentsRect.width(), (int)contentsRect.height());
    m_svg.paint(p, (int)contentsRect.left(), (int)contentsRect.top());

    // We place the icon and text
    p->drawPixmap(20, 0, m_icon.pixmap((int)contentsRect.width() - 40));
    p->save();
    p->setPen(Qt::white);
    p->drawText(contentsRect,
                Qt::AlignBottom | Qt::AlignHCenter,
                "Hello Plasmoid!");
    p->restore();
}

#include "plasma-tutorial1.moc"
```

Enfin, nous allons compiler notre projet à l'aide de l'outil cmake. Voici le script de compilation. (Vous pourrez le réutiliser pour tout vos widgets. La syntaxe, bien que particulière de ce fichier, reste simple et accessible).

```
# Nom du projet
project(plasma-tutorial1)

# Recherche des librairies requises
find_package(KDE4 REQUIRED)
include(KDE4Defaults)
find_package(Plasma REQUIRED)

add_definitions (${QT_DEFINITIONS} ${KDE4_DEFINITIONS})
include_directories(
   ${CMAKE_SOURCE_DIR}
   ${CMAKE_BINARY_DIR}
   ${KDE4_INCLUDES}
   )

# Ajout du code source
set(tutorial1_SRCS plasma-tutorial1.cpp)

# Définition des emplacements
kde4_add_plugin(plasma_applet_tutorial1 ${tutorial1_SRCS})
target_link_libraries(plasma_applet_tutorial1
                      ${PLASMA_LIBS} ${KDE4_KDEUI_LIBS})

install(TARGETS plasma_applet_tutorial1
        DESTINATION ${PLUGIN_INSTALL_DIR})

install(FILES plasma-applet-tutorial1.desktop
        DESTINATION ${SERVICES_INSTALL_DIR})
```

Nous pouvons tester l'applet. Il nous suffit pour l'installer de copier les fichiers au bon endroit :

```
cp ./lib/plasma_applet_tutorial1.so $KDEDIR/lib
cp ./plasma-applet-tutorial1.desktop $KDEDIR/share/kde4/services/
```

et de lancer kbuildsycoca. Pour tester l'applet, vous pouvez utiliser le lanceur prévu à cet effet :

```
plasmoidviewer applet_name
```

Afin que le navigateur d'applet puisse le reconnaître, vous devez redémarrer votre session afin de forcer le chargement de celui-ci.

La librairie Phonon

Au tout début de KDE, il n'existait pas de librairies intégrées à KDE permettant le développement d'applications audio et multimédia. Avec l'arrivée de aRts dans KDE 2, un premier pas en avant est effectué. Par la suite, l'environnement de bureau se dote d'un serveur sonore autorisant notamment le multiplexage audio, ainsi que d'une API multimédia. Avec le temps aRts a évolué mais celle-ci ne dispose toujours pas de support pour la vidéo et il est nécessaire d'utiliser des librairies comme Xine, au détriment de la portabilité.

Phonon est la solution qui permet de franchir le cap en offrant à KDE 4 un framework purement multimédia accessible aux développeurs. A présent, il n'est plus nécessaire de réinventer la roue. Phonon met à disposition les fonctionnalités nécessaires pour ne plus avoir à coder son propre moteur sonore ou vidéo. Les développeurs s'affranchissent ainsi de la mécanique souvent complexe et difficile à maintenir des solutions s'appuyant sur des API spécifiques.

Phonon est portable et est prévu pour avoir une simplicité et une stabilité garanties sur toute la durée de vie de KDE 4. Phonon sera disponible pour la plate-forme Windows et Trolltech utilisera Phonon dans Qt 4.4 pour fournir des fonctionnalités multimédia multi-plates-formes. Autant dire tout de suite que Phonon sera incontournable dans les développements multimédia. A titre d'exemple, le code suivant effectue la lecture d'un fichier OGG en seulement 4 lignes de C++. Auparavant, aRts nécessitait une trentaine de lignes…

```
media = new MediaObject(this);
connect(media, SIGNAL(finished()), SLOT(slotFinished());
media->setCurrentSource("/home/username/music/filename.ogg");
media->play();
```

Decibel

Aujourd'hui, les utilisateurs sont en interface avec de nombreuses applications de communication. Les concepteurs de Decibel sont partis du constat suivant : il n'existe pas d'outil permettant d'intégrer pleinement ces logiciels et protocoles à l'environnement de bureau. L'objectif de Decibel est donc concret : offrir un pont de communication pour les différentes technologies de communication.

L'utilisation de Decibel permet ainsi d'apporter une intégration sans égal des applications telles que MSN, AIM ou encore Skype à l'environnement KDE.

Désormais, il sera possible de placer tous les contacts au même endroit, ou encore de factoriser les actions comme se mettre hors ligne

ou encore changer son statut. Bien sûr, Decibel permet également de centraliser les informations de connexion : login et mot de passe.
Cependant Decibel ne se limite pas au grand public. Les intégrateurs de systèmes et les revendeurs de matériels pourront se baser sur cette solution pour fournir des pilotes standard aux utilisateurs Linux, qu'ils utilisent KDE ou Gnome. Les développeurs aussi peuvent tirer parti de ce framework, notamment pour les applications qui utilisent un protocole de communication temps réel : allant du simple texte jusqu'à la téléphonie en passant par la voix sur IP.

Gérez le matériel avec Solid

Solid est un nouveau framework permettant une gestion innovante des périphériques et du matériel de l'ordinateur. Celui-ci offre un service de découverte qui permet la détection et l'utilisation du matériel sans considération d'architecture de matériel et de système d'exploitation. Comme nous allons le voir, il s'agit d'un framework simple à mettre en œuvre.
Notre premier programme sera un outil en ligne de commande permettant d'afficher la liste du matériel à l'écran. Cela s'écrit en seulement deux lignes de code. Tout d'abord, nous récupérons la liste via le gestionnaire Device, puis nous affichons le nom de chaque périphérique.

```
foreach( Solid::Device device, Solid::Device::allDevices() )
{
    kDebug() << device.udi();
}
```

L'appel de la méthode Device.udi() retourne la valeur de l'identifiant unique du périphérique au type QString. Même si vous avez des périphériques qui sont identiques, le UDI de chacun d'entre eux est garanti pour être unique.
Afin d'effectuer une recherche sur un périphérique particulier, nous allons mettre en oeuvre un filtre fourni par le framework. Le filtrage s'opère selon les fonctionnalités, sous périphériques ou autres paramètres du matériel. Ici, nous allons filtrer les matériels audio.

```
// Headers de Qt
#include <QList>

//Headers de Solid
#include <solid/devicenotifier.h>
#include <solid/device.h>

//Headers de Kde
#include <kcomponentdata.h>
#include <kcmdlineargs.h>
#include <klocale.h>
#include <kdebug.h>

#include <iostream>

using namespace std;

int main(int args, char **argv)
{
    KComponentData componentData("tutorial1");
    foreach (Solid::Device device, Solid::Device::allDevices())
    {
        kDebug() << device.udi().toLatin1().constData();
    }
    return 0;
}

#include "tutorial1.moc"
```

Maintenant que l'on connaît les différentes méthodes pour récupérer un périphérique, nous pouvons nous intéresser à son utilisation. Nous allons réaliser une application qui récupère le premier processeur et affiche sa fréquence. Nous ajouterons un test qui ici est inutile mais qui permet de montrer comment on peut vérifier la présence d'une fonctionnalité sur le matériel.

```
Solid::DeviceNotifier *notifier = Solid::DeviceNotifier::instance();

//récupère la liste des processeurs
QList<Solid::Device> list = Solid::Device::listFromType(Solid::DeviceInterface::Processor, QString());

//sélectionne le premier
Solid::Device device = list[0];
if(device.is<Solid::Processor>() ) kDebug() << "Il y a un processeur!";
else kDebug() << "Le périphérique n'est pas un processeur";

Solid::Processor *processor = device.as<Solid::Processor>();
kDebug() << "Vitesse du processeur :" << processor->maxSpeed();
```

Kross, moteur de plug-in

KDE fournit de nombreux frameworks. Kross en est un autre et il est utilisé pour le développement de scripts. Kross augmente ainsi la productivité de création de modules pour les applications KDE. Ce framework supporte actuellement les langages Python, Ruby et JavaScript et ne nécessite donc pas de recompilation ni de procédures complexes pour faire évoluer les applications. Ce coup de baguette magique se base sur une interface transparente pour le développeur qui utilise des fonctionnalités avancées de Qt 4. Des travaux sont en cours pour supporter d'autres langages tels que Java et Falcon.
Les interpréteurs de scripts sont des plug-in qui sont chargés à la demande lors de l'exécution par le framework Kross. L'application n'a donc pas besoin de connaître le fonctionnement du langage de script utilisé. Le développeur de l'application ne s'occupe donc pas des détails de l'implémentation et se contente de fournir un système d'intégration de module à ses utilisateurs. C'est Kross qui assure le bon déroulement de l'exécution du module au sein de l'interpréteur embarqué. Bien que Kross soit d'une grande aide et que son fonctionnement soit très intuitif, il est nécessaire de savoir développer en Qt. L'exemple suivant met en œuvre un QObject et y connecte un signal. Le tout est codé en Python. Kross est certainement le framework le moins accessible de KDE :

```
#!/usr/bin/env kross
import Kross
```

```
m = Kross.module("myobjectmod")
m.name = "MyObjectModuleName"
obj1 = m.create("OtherObjectName")
def myCallbackFunc(args):
    print "The parent of obj1 changed"
obj1.connect("parentChanged()",myCallbackFunc)
obj1.setParent(m)
print "%s %s" % (obj1.name,obj1.parent().name)
```

Les scripts dans KWord

KWord dispose de son propre moteur de script embarqué. Celui-ci permet le développement de macros afin d'automatiser les tâches que l'on peut avoir à faire de manière quotidienne pour les grands utilisateurs de bureautique.

Le moteur de script se décompose en deux parties. La première est celle qui fournit l'interpréteur et la seconde, celle qui définit les objets sur lesquels les scripts peuvent interagir. La première partie contient notamment la classe KWScriptingPart qui implémente l'intégration de l'interpréteur en tant que plug-in dans KWord. Son rôle est primordial. Ensuite, il y a la classe Scripting::Module qui permet l'accès aux fonctionnalités de KWord depuis les scripts. Dans la seconde partie, on retrouve les classes FrameSet et Frame qui contiennent ce qui est affiché à l'écran, tandis que la classe TextDocument représente le document proprement dit.

L'exemple suivant ajoute du texte à la fin du document :

```
import KWord
doc = KWord.mainFrameSet().document()
cursor = doc.rootFrame().lastCursorPosition()
cursor.insertHtml("<b>Some text</b>")
```

Voici un autre exemple Python qui affiche l'URL et quelques autres méta-informations sur le document.

```
import KWord
print KWord.document().url()
print KWord.document().documentInfoTitle()
print KWord.document().documentInfoAuthorName()
```

Utilisez le système d'impression

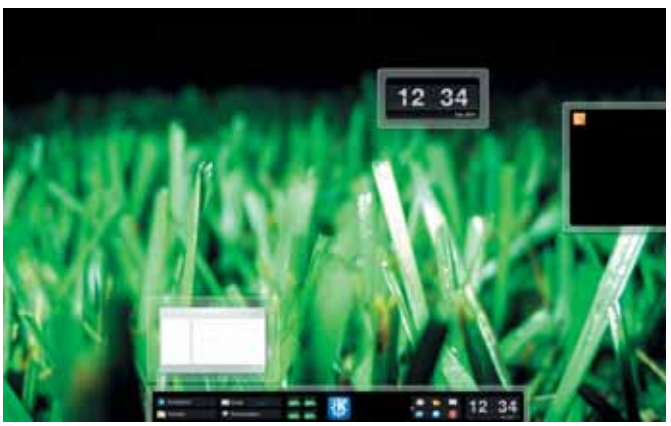
Pour finir, nous allons nous intéresser à une API déjà présente dans KDE 3 qui permet de communiquer avec l'imprimante. Cette API du nom de KPrinter se veut simple d'utilisation, mais son objectif principal est ailleurs, puisque celle-ci est portable et indépendante du matériel. Ici, nous allons écrire un exemple en langage C qui imprime une simple chaîne de caractères :

```
#include <kprinter.h>
#include <qpainter.h>
#include <kapplication.h>
#include <kaboutdata.h>
#include <kmessagebox.h>
#include <kcmdlineargs.h>

int main(int argc, char *argv[])
{
  KAboutData aboutData( "test", "test", "1.0", "test",
     KAboutData::License_GPL, "(c) 2006" );
  KCmdLineArgs::init( argc, argv, &aboutData );
  KApplication app;

  KPrinter job;
  job.setFullPage( true );
  if ( job.setup() )
  {
    QPainter painter;
    painter.begin( &job );
    painter.drawText(100,100,"Hello World");
    painter.end();
  }
}
```

Il ne reste plus qu'à compiler le source et lancer l'application. Vous devriez voir s'imprimer le texte "Hello World" sur l'imprimante par défaut configurée dans KDE. La librairie KPrinter offre de nombreuses fonctions, notamment en ce qui concerne les graphismes. Nous vous invitons à vous reporter à la documentation du projet.

Quel bilan ?

Depuis quelques années, nous le savions déjà : les systèmes Linux disposent d'un noyau abouti, fiable et performant et désormais il fallait que l'effort de la communauté open source se porte sur les environnements de bureaux. KDE était déjà bien avancé dans cette démarche et



la version 4 n'est là que pour le confirmer. Les systèmes Linux peuvent disposer d'une belle interface graphique avec une ergonomie irréprochable. KDE 4, non content de cette avancée améliore également ses performances ainsi que l'interopérabilité avec les autres systèmes.

Loïc Guillois

CMake

Si vous développez en environnement Linux, vous devriez être familier de l'outil de compilation Make. Sous KDE, le choix a été fait par les développeurs, d'utiliser en priorité l'outil CMake. Celui-ci est un outil open source qui permet la compilation cross plateformes et permet ainsi d'améliorer considérablement la portabilité des scripts provenant du projet KDE.

LANGAGE : C++
DIFFICULTÉ : ●●●●●○
APPLICATION : JEU
SOURCE : non

Une structure de données fascinante : programmer un labyrinthe en C++

Dans cet article, nous allons étudier la réalisation d'un labyrinthe de forme rectangulaire et nous proposerons un algorithme qui mettra en lumière la solution pour trouver un chemin qui permettra d'en sortir. Notre objectif ici est de créer un labyrinthe parfait, *le plus simple à générer et à résoudre pour un ordinateur. Un labyrinthe parfait est défini comme ayant un et un seul chemin de n'importe quel point vers n'importe quel autre point du labyrinthe. Cela signifie qu'il n'existe ni parties inaccessibles, ni chemins circulaires, ni zones ouvertes. D'apparence simple, ce problème requiert l'utilisation de plusieurs structures de données : tableaux, piles et listes. Notre but consiste à utiliser les classes de C++ pour définir plusieurs structures hybrides.*

Lorsque l'on étudie un langage de programmation, on rencontre souvent des structures de données telles que les piles, les listes chaînées et les arbres, entre autres. Selon le paradigme, les structures de données assument une position de squelette dans le programme, tandis que les algorithmes ne figurent qu'une manière de manipuler les données. Malheureusement, dans la plupart des documents traitant du sujet, on introduit ces structures de façon simpliste, avec des exemples d'une effroyable trivialité. Ici, nous vous proposons d'étudier un exemple intéressant bien que légèrement complexe : la réalisation d'un labyrinthe et la découverte de sa sortie. Considérez la situation suivante.

On commence avec une maille composée de plusieurs lignes et colonnes de cellules rectangulaires (Figure 1). En cassant certains murs de certaines cellules, on peut construire un labyrinthe (Figure 2). Reste à déterminer quels murs détruire, le moment où le dessin du labyrinthe sera terminé, ou encore s'il existe bien un chemin entre la cellule d'entrée et celle de sortie.

1 Donner de la structure à notre programme

Pour répondre à ces questions, nous allons doter notre programme de trois types de structures. En premier, nous trouvons le tableau. Il s'agit de la structure la plus simple. Elle consiste en un rectangle de cellules composé d'un certain nombre de lignes et de colonnes. Si le tableau n'est pas numérique, on ne peut pas faire grand-chose, à part lire ou écrire ses entrées. Il se montre néanmoins assez commode pour représenter des données sous forme compacte, voire visuellement esthétique. Dans le cadre de cet article, nous l'utilisons pour figurer notre labyrinthe à l'intérieur de la classe de même nom. Nous attirons à ce sujet votre attention sur la variable Puzzle, située dans le code des listings 5 et 6. Ici, on triche un peu; Puzzle est en fait un tableau linéaire en mémoire, ce que nous pouvons nous permettre puisque nous avons opté pour un labyrinthe de forme rectangulaire. Dans ce contexte, on fournit aux utilisateurs de la classe suscitée un opérateur d'indexation de la forme Puzzle(i,j). Si celui-ci indique tout simplement la cellule placée dans la ligne i et la colonne j, il faut savoir qu'en mémoire, pour accéder à cette cellule, on se déplace de *i * nombre total de colonnes + j espaces* dans la mémoire.

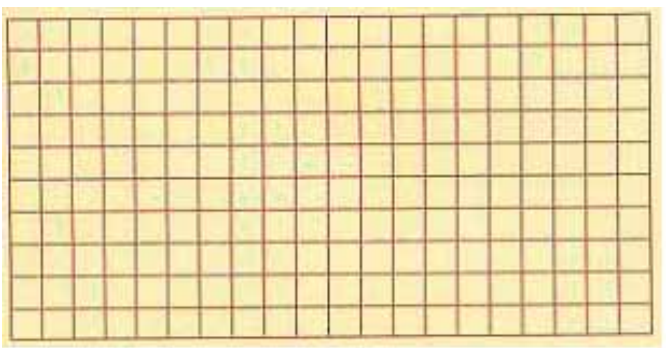


Fig.1

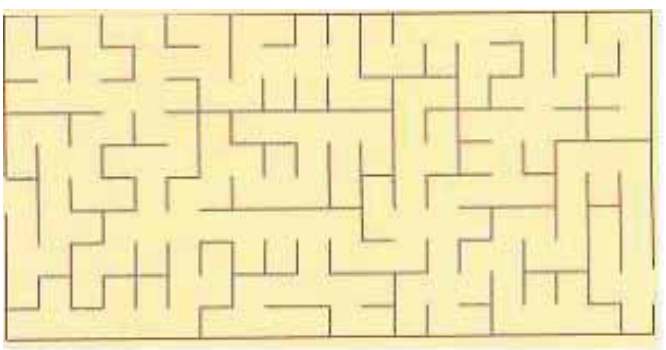


Fig.2

Deuxième structure utilisée ici, voici la liste ("doublement liée", dans notre cas). Elle se compose de plusieurs noeuds, chacun contenant un élément de données indépendant de toute implémentation, ainsi que deux pointeurs (dont les valeurs correspondent aux adresses des noeuds suivants et précédents). Par convention, la "tête" d'une liste est le noeud n'ayant pas de noeud précédent. Et la "queue" celui qui n'a pas de noeud suivant. On peut réaliser plusieurs opérations sur les listes, la plus courante étant l'insertion (au début, au milieu ou à la fin) de nouveaux éléments. A ce titre, observons le code du listing 5. On y implémente la structure de liste pour la classe Chambre. Ici, la fonction Glue fournit la seule opération qui nous intéresse : elle réalise l'addition de deux listes contenant les chambres données. Pour ce faire, on cherche la tête dans une liste et la queue dans une autre. Puis on "colle" simple-

ment les deux noeuds en question l'un sur l'autre. On l'aura compris, par rapport au tableau, la liste offre l'avantage indéniable de pouvoir grandir. Inutile, donc, de connaître sa taille par avance.
Passons maintenant à une troisième structure, la pile. Il s'agit d'un cas spécial de liste, que l'on peut imaginer comme un tas d'objets. Elle n'a qu'un seul pointeur et, de fait, on ne peut pas accéder librement à ses éléments puisque l'on ne peut extraire que celui qui se trouve en surface. On résume ces propriétés en appelant la pile une structure "Dernier Entré, Premier Sorti" ou abréviation anglaise **LIFO** (**L**ast **I**nput **F**irst **O**utput). Dans une pile, les seules opérations possibles sont *empiler un nouvel élément* (fonction push()) et *retirer* (si possible) *le dernier empilé* (fonction pop()). Cette propriété rend précisément la pile intéressante dans notre cas. En effet, perdus dans le labyrinthe, nous aurons toujours la possibilité de revenir en arrière. Du coup, si l'on stocke notre trajectoire dans la pile, le dernier élément empilé correspondra toujours à une solution de repli. On implémente cette structure dans les listings 3 et 4. Les données de notre structure sont les coordonnées de la cellule (chambre) ou l'on se situe.

2 Coder les ouvertures et les fermetures

La création du labyrinthe se fait avec une classe qui accepte le nombre de lignes et de colonnes. Elle alloue un tableau du type *lignes* * *colonnes* = *chambres*. Les données de chaque chambre reviennent principalement à deux entiers. L'un spécifie l'ensemble auquel elle appartient et l'autre l'état de ses portes. À l'exception des bords, toutes les cellules possèdent quatre portes (bas, droite, haut et gauche). Mais comme une porte se trouve partagée par deux chambres juxtaposées, on considèrera que chaque chambre ne gère que les portes Bas et Droite (parce qu'en bas et à droite des cellules, sur nos schémas) qu'elle contient. Evidemment, les bords font exception. On représente l'état des portes (ouvertes, fermées et, soyons fous, fermées à clé) par deux bits avec la convention suivante :

```
00 :  la porte est fermée
01 :  la porte est ouverte
11  :  la porte est fermée à clé
```

Par ailleurs, on décidera que les deux premiers bits (à droite dans l'écriture d'un nombre binaire) symbolisent l'état de la porte Bas et les deux suivants (en partant vers la gauche pour l'écriture d'un nombre binaire) celui de la porte Droite. Ainsi, voici à quoi correspondent certaines valeurs décimales :

```
0 =  00 00
_ les portes Bas et Droite fermées

7  =  01 11
_ la porte Bas fermée à clé, porte Droite ouverte
```

Tous les états possibles sont énumérés dans le type DoorStates du listing 5. Afin de s'en souvenir facilement, on utilise D pour (D)own, C pour (C)losed, L pour (L)ocked, O pour (O)pen. Avec cette convention, il s'avère assez facile d'écrire les routines autorisées à changer l'état des portes. Pour ce faire, il suffit d'altérer bit à bit leur condition actuelle à l'aide des opérateurs | et &. Les différentes fonctions sont implémentées dans les listings 5 et 6. A titre d'illustration, considérez l'implémentation de la fonction LockRight(). Celle-ci ne fait qu'appliquer l'opérateur disjonctif " | " bit à bit. Par exemple, sachant que 12 (décimal) = 11 00 (binaire), la fonction ferme à clé la porte Droite et ne touche pas à celle du Bas.

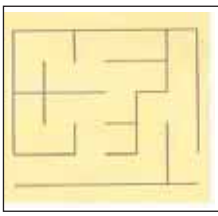


Fig.3

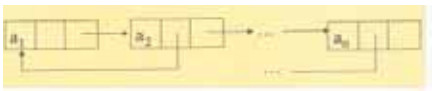


Fig.4

3 Disposer les ouvertures et les fermetures

Cette première étape définie, nous voulons maintenant que notre labyrinthe soit parfait ou, autrement dit, qu'il n'y ait qu'un et un seul chemin entre deux chambres quelconques. D'un point de vue technique, il s'agit donc de représenter l'ensemble des trajectoires possibles par un arbre. D'un point de vue pratique, la règle consistera à ne pas créer une porte entre deux pièces si celles-ci peuvent déjà se rejoindre par un autre chemin. Voici par exemple quel pourrait être l'énoncé de la création d'un labyrinthe : " marquer chaque chambre de façon unique, avec un identificateur. Tant que le labyrinthe n'est pas fini, se placer – au hasard – sur une chambre. Si on y trouve une porte fermée, regarder l'identificateur de la chambre voisine. Si les identificateurs sont différents entre les deux chambres, ouvrir leur porte commune et unifier leur identificateur ainsi que leurs listes. Sinon fermer la porte à clé (car signifie que les deux chambres se trouvent déjà connectées ; ouvrir cette porte formerait un îlot). On considère le labyrinthe comme fini dès lors que toutes les chambres ont le même identificateur ".
En général, la complexité d'un labyrinthe peut se mesurer à son arbre. Plus celui-ci est "balancé" (un arbre est dit "balancé" si chacune de ses branches a environ la même taille), plus le labyrinthe devient complexe. À noter qu'au lieu de prendre au hasard des cellules, on peut également progresser aléatoirement dans le labyrinthe.
L'algorithme est implémenté dans le constructeur de la classe Labyrinthe. Ce constructeur accepte les dimensions comme entrée. On commence la construction en choisissant au hasard quelques cellules. Cette opération nous permet d'introduire quelques couloirs de façon à rendre difficile la recherche du bon chemin. Puis, on insère une dernière boucle réalisant la même tâche à la fin. Celle-ci sert juste à vérifier que toutes les chambres se trouvent bien connectées.

4 Et maintenant ... sortir !

Considérons un robot virtuel qui, placé dans une chambre, doit parcourir le tableau pour y dénicher la sortie. Celui-ci ne peut réaliser que quatre types de mouvements : haut, bas, droite et gauche. Si l'on part du principe qu'il se trouve à la position (i, j), voici comment les exprimer :

```
Droite : on passe à  (i + 1, j)

Gauche : on passe à  (i – 1, j)

Bas : on passe à (i, j + 1)
```

```
Haut : on passe à (i, j – 1)
```

Ensuite, il convient d'appliquer la classique règle de la main droite (ou gauche), laquelle consiste à toujours suivre d'une main le mur du labyrinthe. Celle-ci marche très bien dès lors que l'on part d'un bord du labyrinthe. Mais si notre robot se trouve jeté à n'importe quel endroit du dédale, il risque fort de tourner en rond parce qu'il n'a pas choisi la bonne main ! On comprend ici tout l'intérêt de stocker la trajectoire effectuée dans la pile : cela nous permet de revenir en arrière à la recherche d'une autre bifurcation plutôt que de reprendre indéfiniment le même chemin dans le même sens. L'algorithme pour sortir du labyrinthe prend l'allure suivante :

```
Mettre dans la pile (push) la position de départ
Tant que l'on ne trouve pas la sortie
Evaluer l'ouverture des portes de la chambre pour chaque porte ouverte
    Si la porte n'a pas encore été franchie
        Avancer
        Stocker (push) la nouvelle  position
    Sinon on est dans une impasse
        Revenir en arrière (pop)
```

On l'aura deviné, dans l'implémentation, nous nous devons de suivre un ordre strict quant à la recherche des portes ouvertes à partir de celle par laquelle nous sommes entrés dans la chambre soit dans le sens des aiguilles d'une montre (cas de la main gauche), soit dans le sens inverse (cas de la main droite). L'algorithme final se trouve implémenté dans le listing 8. Pour compiler et lancer le programme il suffira par exemple de taper à partir du Shell :

```
$ make

$ rodrimaze 40 60 8 &
```

Vous obtiendrez alors quelque chose de similaire à la figure 6. A noter que le code source présenté permet d'implémenter des labyrinthes plus complexes, comme ceux en forme de pentagones, d'hexagones, etc. En fait, seul l'affichage marque surtout la différence.

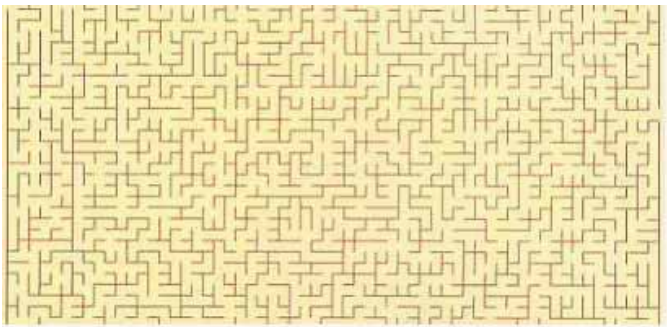


Fig.5

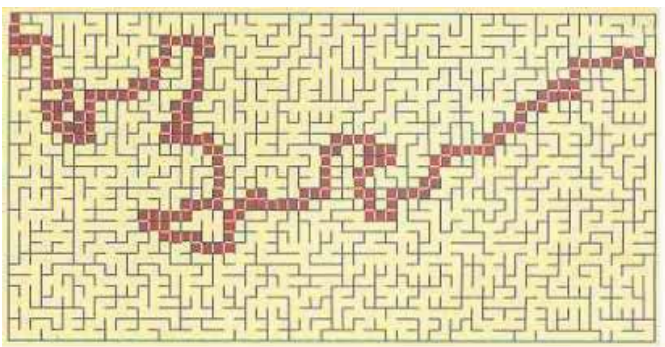


Fig.6

5 Visualiser le labyrinthe

Sous Shell, il suffit de taper :

```
:$  make –f  Show

:$  make 30 60 10 &
```

Ces commandes vont dessiner un labyrinthe de trente lignes sur soixante colonnes avec des cellules ayant dix pixels de côté (neuf, en réalité, puisqu'un pixel sert au dessin du mur). Vous obtiendrez quelque chose ressemblant à la figure 5.

Rodrigue S. Mompelat

Docteur en Sciences Cognitives-Intelligence Artificielle de l'EHESS. Il a enseigné les mathématiques et l'informatique, et effectué des recherches dans les Universités de Paris III Sorbonne Nouvelle, Paris V René Descartes, Cergy-Pontoise et à Copenhagen Business School, l'Université de Aalborg et à RISØ National Laboratory au Danemark.

Blitz++ et la méta-programmation C++

C++ est un langage plein de surprises. Une fonctionnalité "imprévue", la méta-programmation, permet à la librairie Blitz++ de concurrencer Fortran sur le plan des performances brutes en matière de calculs mathématiques. Partons à sa découverte !

Plein de qualités et plein de défauts, c'est ce qui fait son charme, C++ est parfois critiqué pour ne pas offrir l'équivalent de Fortran en termes de performances brutes pour les calculs mathématiques. Il est vrai que l'orientation-objet présente un coût à l'exécution en contrepartie des capacités d'abstraction et de réutilisation du code qu'elle apporte. La surcharge des opérateurs a un coût en contrepartie de l'élégance et de la clarté du code. Enfin C++ lui-même, basé sémantique de valeur, implique la copie d'objets et la réduction de performance qui va avec, en attendant que C++0X, le standard à venir, ne modifie cette donne en introduisant la sémantique de déplacement. C++ c'est un monde à part entière et un monde entièrement à part. A l'origine le langage n'a jamais été conçu pour supporter la méta-programmation, domaine essentiellement réservé aux langages fonctionnels. Ainsi le vieux Lisp, le premier d'entre eux, la supportait déjà. Toutefois, quand C++ s'est vu doté des templates, des programmeurs astucieux les ont détournés de leur usage premier. La méta-programmation C++ était née, et un gain significatif de performances avec elle. Nous allons voir brièvement en quoi celle-ci consiste, puis nous apprécierons ce qu'elle peut apporter en découvrant Blitz++.

1 Notions de meta-programmation en C++

Prenons un exemple usé jusqu'à la corde, le calcul d'une factorielle. Voici un code C++ classique qui effectue le calcul et affiche le résultat :

```
#include <iostream>

using namespace std;

int factorielle(int n  /* n supposé >= 1) {
   if(n == 1)
        return n;
   else
        return n*factorielle(n-1);
}

int main(int argc, char* argv[]) {
   cout << factorielle(6) << endl;
   return 0;
}
```

Qu'est-ce qui ne va pas avec ce code ? A priori rien. Pourtant les programmeurs C++ ont une légère (et souvent compréhensible et justifiée) tendance à l'obsession de la performance.
C'est lorsque l'on regarde le code produit par le compilateur, ici VC 8.0, que l'on commence à penser. Dans la fonction main, le compilateur appelle notre fonction factorielle ainsi:

```
push      6
call      factorielle (4110C8h)
add       esp,4
```

Le paramètre (6) est poussé sur la pile, la fonction est appelée, puis le pointeur de pile est réajusté. C'est classique et, somme toute, raisonnable. Si l'on regarde la fonction factorielle elle-même c'est un peu plus inquiétant :

```
int factorielle(int n)
{
00401000  push   esi
   if(n == 1)
        return n;
   else
        return n*factorielle(n-1);
00401001  mov    esi,dword ptr [esp+8]
00401005  lea    eax,[esi-1]
00401008  cmp    eax,1
0040100B  jne    factorielle+12h (401012h)
0040100D  imul   eax,esi
00401010  pop    esi
}
00401011  ret
   if(n == 1)
        return n;
   else
        return n*factorielle(n-1);
00401012  push   eax
00401013  call   factorielle (401000h)
00401018  imul   eax,esi
0040101B  add    esp,4
0040101E  pop    esi
}
0040101F  ret
```

Il n'y a pas besoin d'être un expert en assembleur pour deviner que ce code n'est pas très performant. Notre fonction étant récursive, de nombreuses instructions de pile sont exécutées. De plus l'instruction imul n'est pas forcément ce qu'il y a de plus rapide pour faire une multiplication. Il existe des techniques pour écrire le code source différemment et obtenir une amélioration considérable à l'exécution. Pour cela nous renvoyons le lecteur à l'article "Les Algorithmes Récursifs" paru dans Programmez! 95. Des options de compilations bien choisies amélioreront encore les choses, et c'est tout. Mais les jusqu'au-boutistes de l'optimisation C++ voient plus loin et diraient, à propos de notre exemple: "puisque la valeur (6) passée à la fonction est connue au moment de la

compilation, pourquoi ne pas trouver le moyen pour que la factorielle (6) soit, elle aussi, calculée au moment de la compilation ? Si on y parvient, le temps d'exécution de la fonction factorielle sera nul!" Le premier à avoir eu cette idée est, semble-t-il, un certain Irwin Unruh qui a écrit un code tel que le compilateur émettait, sous forme de messages d'erreur, la liste des nombres premiers qu'il calculait lors de la compilation. Le code était tout rempli de templates et ce sont ces templates qui sont à la base de la meta-programmation C++. Le calcul de notre factorielle ne déroge pas à la règle, voici son code:

```
#include <iostream>
using namespace std;

template <unsigned int n>
struct Factorielle {
   enum { valeur = n * Factorielle<n - 1>::valeur };
};

template <>
struct Factorielle<1> {
   enum { valeur = 1 };
};

int main(int argc, char* argv[]) {
   cout << Factorielle<6>::valeur << endl;
   return 0;
}
```

Ce code est étonnant. Plus de fonction factorielle! A la place, des structures de données. Que fait le compilateur quand il s'attaque à notre code ? Il commence à ne rien faire d'autre que de prendre note de l'existence des templates (en simplifiant un peu :). Puis il arrive dans la fonction main où l'instanciation du template factorielle est demandée. Le terme instanciation ne doit d'ailleurs pas tromper. Ici et en vertu de la politique minimaliste d'un compilateur, aucun code n'est généré. Seule est définie, finalement, une valeur sous la forme d'une constante de compilation. Ensuite le compilateur voit que cette valeur doit être multipliée à son alter ego définie par le même template, mais instanciée pour la valeur n-1. Comme le compilateur sait calculer n-1, il va répéter l'opération. Et ainsi de suite. Nous effectuons ainsi notre calcul à la compilation par le moyen d'une instanciation récursive de templates. Tout calcul récursif doit terminer. Ici on termine le calcul par la spécialisation de notre template pour la valeur 1. Quand le compilateur arrive au bout de la récursion, il conserve la valeur constante finale pour l'impression avec cout. Que reste-t-il d'autre finalement ? Rien! Comme en témoigne un examen du code machine assembleur produit par cette compilation:

```
push      2D0h
call      dword ptr [; la plomberie de cout]
```

2D0h c'est 720 en décimal, soit la valeur de 6! Comme prévu. Cette valeur n'est pas calculée à l'exécution. Ici elle est directement poussée sur la pile à l'attention de cout, toute comme elle le serait en tant que résultat du calcul dans le programme classique. Avec nos templates, le coût du calcul est bel et bien nul à l'exécution. Mais bien sûr il se répercute sur le temps de compilation :)

2 Bien plus qu'une curiosité

Notre exemple peut sembler de peu d'utilité en pratique. C'est effectivement le cas. Mais il montre quelques principes de base qui, exploités avec imagination et habileté, permettent de faire des choses étonnantes. Nous avons par exemple les fameuses listes de types d'Andrei Alexandrescu. Le point de départ est très simple :

```
struct NullType {};

template <class T1, class T2>
struct TypeList {
    typedef T1 Head;
    typedef T2 Tail;
};
```

Et si nous voulons une liste de type [double, int, double] nous écrivons :

```
typedef TypeList<double, TypeList<int, TypeList<double, NullType> > > MaList;
```

Cette dernière écriture pouvant être raccourcie par l'usage de macros. Voulons-nous calculer le nombre d'éléments d'une liste de types, autrement dit sa longueur ? Voici le code, toujours basé sur les "templates récursifs":

```
template <class TList> struct Length;
template <> struct Length<NullType> {
   enum { valeur = 0 };
};

template <class T, class U>
struct Length< TypeList<T, U> > {
   enum { valeur = 1 + Length<U>::valeur };
};
```

Et on pourrait afficher ainsi la longueur de MaList, définie plus haut:

```
std::cout << Length<MaList>::valeur /* vaut 3 */ << std::endl;
```

De la même manière Andrei a écrit toute une quantité de templates pour la manipulation des listes de types. Nous renvoyons le lecteur à son célèbre ouvrage, cité en annexe.

3 Une analogie avec la programmation fonctionnelle.

Programmez! 104 vous a présenté Haskell, un langage fonctionnel pur. Avec ce langage, notre fonction factorielle peut s'écrire comme ceci :

```
factorielle 1 = 1
factorielle valeur = valeur * factorielle (valeur-1)
```

L'analogie avec notre code à templates est frappante. C'est même plus qu'une analogie. En programmation fonctionnelle, il n'y a pas de différence entre code et données. Par exemple en Lisp tout est liste, même les fonctions, donc tout est structure de données. Or comme nous l'avons remarqué plus haut, notre calcul de factorielle à base de templates n'est constitué que de structure de données. Là encore ces remarques sont un point de départ pour aller très loin. Le lecteur intéressé pourra consulter un article (cité en annexe) de Gabriel Dos Reis, un des meilleurs spécialistes français de C++. Dans cet article l'auteur

montre comment on peut traduire un programme écrit en Scheme (langage fonctionnel) en un méta-programme C++.

4 Une nouvelle panacée ?

La méta-programmation C++ n'est certainement pas une panacée. Les avis à son sujet sont partagés. Certains trouvent cela génial, d'autres trouvent cela atroce. Qui a raison ? Sans doute tout le monde :) Il est clair que cette approche permet d'optimiser fortement le code C++. Il est également clair que cette approche est difficile, réservée à des spécialistes des templates C++, et que le moindre code non trivial est absolument illisible :) Ce n'est, de l'humble avis de votre serviteur, pas une bonne idée que vouloir faire de la méta-programmation à tout prix. Par contre c'est une bonne idée d'utiliser des librairies telles que Boost et Blitz++ dont la méta-programmation est une pierre angulaire. Intéressons nous maintenant à Blitz++.

5 Introduction à Blitz++

Blitz++ se télécharge gratuitement à http://www.oonumerics.org/blitz/. Pour l'utiliser, vous devez disposer d'un compilateur C++ digne de ce nom et récent. Un vieux Visual C++ 6.0 ou un vieux Borland C++ ne font pas le poids. J'ai testé Blitz++ sous Visual Studio 2005 sans problème notable. Bien suivre toutefois les indications du fichier ReadMe fourni avec Blitz++. Gcc et Intel++ sont parfaits. D'après leurs auteurs (benchmarks à l'appui) Blitz++ concurrence, voire dépasse Fortran sur le plan de la performance. En outre, C++ apporte son expressivité. Voici un exemple, tiré de la documentation qui montre que Blitz++ n'a pas son pareil pour manipuler des tableaux :

```
#include <blitz/array.h>
using namespace blitz;

int main() {
    Array<int,2> A(3,3), B(3,3), C(3,3);

    A =
        1, 0, 0,
        2, 2, 2,
        1, 0, 0;

    B =
        0, 0, 7,
        0, 8, 0,
        9, 9, 9;

    C = A + B;

    cout << "A = " << A << endl
        << "B = " << B << endl
        << "C = " << C << endl;

    return 0;
}
```

La première ligne dans *main* construit, au moyen d'instanciation de templates, trois tableaux d'entiers à deux dimensions et de taille 3x3. Le contenu des deux premiers tableaux est ensuite initialisé. On remarque avec quelle facilité, grâce à la surchage de l'opérateur virgule. Ensuite on obtient dans le troisième tableau le résultat de la somme des deux premiers, puis le résultat est imprimé.

6 Un exemple de calcul matriciel

La documentation de Blitz++ ne mentionne pas toutes les possibilités offertes. Cela viendra sans doute. Pour l'instant, il faut étudier les exemples de code et aussi les en-têtes de la librairie, sans quoi il ne saute pas aux yeux, par exemple, qu'il est possible de calculer la rotation d'un vecteur en multipliant celui-ci par une matrice de rotation. Voici un exemple qui fait tourner le vecteur unitaire de l'axe des x d'un angle de Pi/3 (60 degrés) autour de l'axe vertical :

```
#include <blitz/tinymat.h>
#include <blitz/tinyvec.h>
using namespace blitz;

int main() {
    // Le vecteur à faire tourner:
    TinyVector<double, 3> vec;
    vec = 1.0, 0,0 ;
    cout << "Vecteur: " << vec << endl << endl;
    // La matrice de rotation
    TinyMatrix<double,3,3> matrot;
    matrot =
        0.5, -0.866, 0.0,
        0.866, 0.5, 0.0,
        0.0,    0.0,        1.0;

    cout << "Matrice: " << matrot << endl << endl;
    // Effectuer la rotation
    TinyVector<double, 3>result = product(matrot, vec);
    // Et afficher le résultat
    cout << "Vecteur apres rotation: " << result << endl;

    return 0;
}
```

Ce code est concis, clair et très direct en comparaison de ce qu'il faudrait écrire, puis éventuellement optimiser, sans une librairie comme Blitz++, Avec elle, le calcul vectoriel, matriciel et même tensoriel (cf. la documentation) est un jeu d'enfant à mettre en œuvre, pour un résultat performant. A meta-bientôt pour de nouvelles aventures en C++ :)

Quelques bonnes lectures

C++ Template Metaprogramming
David Abrahams, Aleksey Gurkovoy - Pearson Education
C++ Templates, The Complete Guide
David Vandevoorde, Nicolai M. Josuttis - Addison-Wesley Professional
C++ Gems
Stanley B. Lippman - Cambridge University Press
Modern C++ Design
Andrei Alexandrescu - Addison Wesley Pearson Education
Metaprogramming in C++
Gabriel Dos Reis
http://www.cmla.ens-cachan.fr/Utilisateurs/dosreis/C++/talks/meta-programming-in-cxx.pdf

■ **Frédéric Mazué** - *fmazue@programmez.com*

Maîtriser la nouvelle API Windows Contacts de Vista

Windows Vista présente une nouvelle API en remplacement du carnet d'adresses (WAB) des systèmes précédents. Cette API est d'une approche ardue et mal documentée. Programmez! a débroussaillé le terrain pour vous.

L'API Windows Contacts est un mécanisme permettant de stocker et gérer les informations relatives à vos contacts. Un logiciel de messagerie comme Windows Mail utilise cette API pour son carnet d'adresses. L'avantage d'une telle API, découplée de tous logiciels, est qu'elle offre une interface commune à toute application souhaitant l'utiliser. Un autre avantage étant que les données sont sauvegardées au format XML. L'inconvénient dans le cas présent résulte de ce que cette API est entièrement écrite en COM. Microsoft ne semble donc pas tant que cela disposée à abandonner la technologie COM, contrairement à ce qui se dit parfois. La programmation d'un client COM en C ou C++ n'est pas chose facile dans l'absolu. Dans ce cas particulier, la difficulté est considérablement augmentée par une documentation pas du tout à la hauteur, contenant moult erreurs et des exemples de codes ne fonctionnant pas. Nous parlons de la MSDN de juillet 2007, dernière version disponible au moment de la rédaction de cet article.

1 Quelques rappels à propos de COM

COM, pour Component Objects Models est une architecture qui permet d'appeler des fonctions de composants logiciels. Un composant logiciel COM est un agrégat d'interfaces COM. Une interface COM est fondamentalement un pointeur sur une table de pointeurs de fonction. Pour le programmeur, une interface COM présente donc une forte analogie avec la vtable d'une classe C++. Toute interface COM dérive de l'interface racine IUnknown et de ce fait expose au moins trois fonctions :

Fonctions de IUnknown	Rôle
AddRef	Incrémente un compteur de référence.
QueryInterface	Obtient une autre interface de l'agrégat.
Release	Décrémente un compteur de référence.

Avec ce tableau, on comprend que quand le compteur de référence d'une interface tombe a zéro, les ressources système correspondantes sont libérées. Si un programmeur n'a pas à s'occuper de l'implémentation des interfaces, il doit au minimum appeler Release une fois, car quand une interface est obtenue, son compteur est automatiquement incrémenté. Peut-être le programmeur devra-t-il en outre appeler le tandem AddRef/Release autant que nécessaire, selon le travail effectué par l'application. Dès que l'on a une interface du composant, QueryInterface permet d'accéder, depuis n'importe quelle interface du composant, à n'importe quelle autre interface du composant. La question est alors: comment obtenir une première interface ? Par les API Windows CoCreateInstance ou CoGetClassObjet, qui vont chercher les informations nécessaires dans la base de registres dans laquelle tout composant COM est enregistré, puis créent une instance du composant en mémoire. Ceci à condition que COM soit initialisé, ce qui se fait au moyen de l'API Windows CoInitialize. Une application bien élevée libère dès que possible les ressources en appelant l'API symétrique CoUninitialize.

2 Quelques classes C++ pour manipuler COM

La gestion des compteurs de références COM peut vite tourner au cauchemar dans du code écrit dans un style C. Par contre il est clair qu'en C++, gérer les pointeurs d'interfaces COM au moyen de pointeurs intelligents C++, non seulement facilitera le travail, mais éclaircira considérablement le code. Si on ajoute à cela l'initialisation et la libération de COM, il est possible d'écrire entièrement un client COM en se basant sur l'idiome "acquisition de ressources par initialisation" de C++ Votre serviteur vous propose donc quelques classes à cet effet, dans l'en-tête COMUtils.h. En cas d'erreur, une levée d'exception assure la libération de toutes les ressources dans les destructeurs des classes. Le code de ces classes et celui des exemples a été écrit avec Visual Studio 2005. Il n'y a aucune difficulté à compiler avec un autre compilateur. Par contre il est évident qu'il est nécessaire, au préalable, de télécharger gratuitement la SDK de Windows Vista qui contient les en-têtes et les librairies relatives à l'API Windows Contacts.

3 Les interfaces COM de l'API Windows Contacts

Elles sont au nombre de 5, et décrites ci-dessus

Interface	Description
IContact	Gère un contact, c'est-à-dire une entrée dans le carnet d'adresses.
IContactCollection	Enumère tous les contacts connus par l'interface IContactManager.
IContactManager	Interface de haut niveau permettant d'accéder à tous les contacts, y compris le cas particulier de l'utilisateur courant.
IContactProperties	Gère les propriétés du contact, telles que nom, prénom, surnom, adresse de messagerie, etc.
IContactPropertyCollection	Enumère les valeurs d'une propriété. Par exemple tous les surnoms d'un contact.

4 Travailler avec l'interface IContactManager

Examinons notre premier exemple, ***ContactManager.cpp.*** Dès que l'interface *IContactManager* est obtenue, nous appelons sa fonction *Initialize* en lui passant un nom et numéro de version d'application quelconque. Ensuite nous appelons la fonction *GetMeContact.* Ainsi nous avons obtenu une interface décrivant un contact très particulier: l'utilisateur qui a lancé l'application. Nous avons là en quelque sorte le point d'entrée du carnet d'adresses. En effet, c'est par ce contact que

l'on peut accéder au répertoire de stockage. Normalement au moyen de la fonction *GetPath* de *IContact*. Mais dans ce cas particulier du contact utilisateur, *GetIDContact* retourne non seulement le GUID du contact mais aussi son Path, ce qui n'est pas annoncé par la documentation. Gare aux débordements de tampon, mais un lecteur de Programmez! averti en vaut deux :) Ensuite l'exemple énumère tous les contacts du carnet d'adresses, en affichant leurs noms et prénoms. Le code est trivial, hormis un point: la représentation de l'arborescence XML du contact. En effet, on ne donne pas celle-ci en dur, mais en juxtaposant des macros. Par exemple :

```
CONTACTPROP_PUB_L1_NAMECOLLECTION CONTACTPROP_PUB_L2_
NAME
```

qui une fois compilé est équivalent à :

```
L"ContactIDCollection/Name"
```

Toutes les macros sont documentées dans la documentation MSDN ou SDK, mais sont parfois fausses Ainsi CONTACTPROP_PUB_L1_NAMECOLLECTION est nommé CONTACTPROP_PUB_L1_CONTACTIDS dans la documentation. Comment faire alors ? Il faut consulter l'en-tête icontactproperties.h dans lequel ces macros sont définies. Signalons en passant que l'en-tête contact.h souvent cité dans la documentation n'existe pas. On doit comprendre icontact.h. Une dernière remarque à propos du premier exemple. La fonction GetString de IContactProperties, qui lit une propriété, est censée retourner un code d'erreur en cas d'échec. Elle le fait, mais sous la forme d'une erreur COM. Cela signifie que le code de retour ne peut être testé directement, mais seulement après un masque binaire effaçant les 16 bits de poids fort du code d'erreur. Le lecteur intéressé peut se reporter à l'en-tête winerror.h pour en savoir plus sur la constitution des codes d'erreur COM.

5 Créer un contact

Ce n'est pas une mince affaire car il faut déjouer les traquenards tendus par la documentation. Examinons le code de notre second exemple : ***DemoCreateContact.cpp.*** Cet exemple utilise le Contact Manager pour obtenir le répertoire de stockage, puis suit la démarche de l'exemple de la MSDN jusqu'à l'appel de la fonction *InitNew* de *IPersistStreamInit,* interface agrégée dans IContact. Au-delà, l'exemple de la MSDN ne fonctionne plus. La MSDN donne STG_WRITE comme flag à passer à l'API SHCreateStreamOnFileEx pour la création d'un flux pour l'interface *IPersistStream* du contact. Ceci ne fonctionne pas et aboutit à un code d'erreur abscons retourné par *IPersistStreamInit->Save*. Les flags corrects sont :

```
STGM_CREATE|STGM_READWRITE
```

Enfin, par exemple pour définir le nom d'un contact, la MSDN indique d'écrire cette valeur directement dans l'arborescence XML par IContactProperties->SetString, ce qui aboutit à un autre code d'erreur non moins abscons. En effet lors de l'initialisation du contact (appel à InitNew), seule une arborescence minimale est créée. Le programmeur doit d'abord créer, dans cette arborescence, un nœud relatif à la propriété qu'il veut définir. C'est ce que fait notre exemple par l'appel à la fonction *CreateArrayNode* de l'interface *IContactProperties.* Attention cette fonction a besoin d'un tampon pour travailler, et aucune information n'est donnée sur le rôle ni sur la taille de ce tampon. Apparemment le tampon ne contient que la description de l'arborescence, soit pour notre exemple : L"NameCollection/Name[1]". Nous en savons maintenant suffisamment pour manipuler l'API Windows Contacts avec profit. A bientôt pour d'autres visites guidées dans les profondeurs de Windows Vista.

Frédéric Mazué - *fmazue@programmez.com*

```
// ContactManager.cpp

#include <iostream>
#include <COMUtils.h> // cf. CD-Rom
// objbase.h est normalement déjà inclus par COMUtils.h
#include <objbase.h>
#include <icontact.h>
#include <icontactproperties.h>

int main() {
   try {
      HRESULT hr;
      // COM est initialisé ici,
      // *AVANT* la déclaration des pointeurs intelligents
      COMInitializer ci;
      SmartCOMPtr<IContactManager> cm;
      SmartCOMPtr<IContact> me, courant;
      SmartCOMPtr<IContactCollection> cc;

      hr = ::CoCreateInstance(CLSID_ContactManager,
           NULL,
           CLSCTX_INPROC_SERVER,
           IID_IContactManager,
           reinterpret_cast<void**>(&cm));
      COMHelper::TestOk(hr);
      hr = cm->Initialize(L"MyAPP", L"1.0");
      COMHelper::TestOk(hr);
      hr = cm->GetMeContact(&me);
      COMHelper::TestOk(hr);
      // ATTENTION ici MAX_PATH n'est PAS une
      // taille suffisante pour le buffer
      WCHAR buffer[MAX_PATH*2] = {0};
      DWORD size = ARRAYSIZE(buffer);
      DWORD required;
      hr = me->GetContactID(buffer, size, &required);
      COMHelper::TestOk(hr);
      std::cout << "ID du contact de l'utilisateur" << std::endl;
      std::cout << "Remarquez la presence du path!" << std::endl;
      std::wcout << buffer << std::endl;
      std::cout << "----------" << std::endl;
      std::cout << "Enumeration de tous les contacts" << std::endl;
      std::cout << "----------" << std::endl;

      // Obtenir tous les contacts par le biais
      // d'une instance de IContactCollection
      hr = cm->GetContactCollection(&cc);
      COMHelper::TestOk(hr);
```

```
// énumérer
WCHAR familyname[] = CONTACTPROP_PUB_L1_NAMECOLLECTION
    CONTACTPROP_PUB_L2_NAME L"[1]"
        CONTACTPROP_PUB_L3_FAMILYNAME;
WCHAR givenname[] = CONTACTPROP_PUB_L1_NAMECOLLECTION
    CONTACTPROP_PUB_L2_NAME L"[1]"
        CONTACTPROP_PUB_L3_GIVENNAME;
while(1) {
    hr = cc->Next();
    if(hr == S_FALSE)
        break;
    SmartCOMPtr<IContactProperties> icp;
    hr = cc->GetCurrent(&courant);
    COMHelper::TestOk(hr);
    hr = courant->QueryInterface(IID_IContactProperties,
        reinterpret_cast<void**>(&icp));
    COMHelper::TestOk(hr);

    // Obtenir et afficher le nom de famille du contact
    hr = icp->GetString(familyname, CGD_DEFAULT, buffer,
            ARRAYSIZE(buffer), NULL);
    bool nom = false; // pour affiner l'affichage
    switch(hr & 0x0000FFFF) {
        case ERROR_PATH_NOT_FOUND:
            //std::cout << "L'arborescence XML "
            // "n'existe pas" << std::endl;
            break;
        case S_FALSE:
            std::cout << "Le contact n'a pas "
                "de nom" << std::endl;
            break;
        case ERROR_INSUFFICIENT_BUFFER:
            std::cout << "Le nom du contact ne tient"
                " pas dans le buffer" << std::endl;
            break;
        case S_OK:
            std::wcout << buffer << ", ";
            nom = true;
            break;
        default:
            COMHelper::TestOk(hr);
            break;
    }

    // Obtenir et afficher le prénom du contact
    hr = icp->GetString(givenname, CGD_DEFAULT, buffer,
            ARRAYSIZE(buffer), NULL);
    switch(hr & 0x0000FFFF) {
        case ERROR_PATH_NOT_FOUND:
            //std::cout << "L'arborescence XML "
            //  "n'existe pas" << std::endl;
            if(nom)
                std::cout << std::endl;
            break;
        case S_FALSE:
            std::cout << "Le contact n'a pas "
                "de prenom" << std::endl;
            break;
        case ERROR_INSUFFICIENT_BUFFER:
            std::cout << "Le prenom du contact "
            "ne tient pas dans le buffer" << std::endl;
            break;
        case S_OK:
            std::wcout << buffer << std::endl;
            break;
        default:
            COMHelper::TestOk(hr);
            break;
        }
    }
  }
  catch(COMException ce) {
      std::cout << ce.raison() << std::endl;
      return EXIT_FAILURE;
  }
  return EXIT_SUCCESS;
}
```

// DemoCreateContact.cpp

```
// DemoCreateContact.cpp :
#include <iostream>
#include <cstdlib> // pour _wsplitpath

#include <COMUtils.h> // cf. CD-Rom
// objbase.h est normalement déjà inclus par COMUtils.h
#include <objbase.h>
#include <icontact.h>
#include <icontactproperties.h>
#include <shlwapi.h>

#include <windef.h>   // pour MAX_PATH
#include <winternl.h> // pour ARRAYSIZE

#define FILENAME L"marie.aubin.contact"

#pragma warning(disable : 4996)

std::wstring GetRepository(const WCHAR* full) {
  WCHAR bdrive[_MAX_DRIVE];
  WCHAR bdir[_MAX_DIR];
  WCHAR bname[_MAX_FNAME];
  WCHAR bext[_MAX_EXT];
  _wsplitpath(full, bdrive, bdir, bname, bext);

  std::wstring result = bdrive;
  result += bdir;
  return result;
}

int main() {
```

```
std::cout << "Demo Create Contact" << std::endl;
try
{
    HRESULT hr;
    WCHAR buffer[MAX_PATH] = {0};
    DWORD size = ARRAYSIZE(buffer);
    DWORD required;

    // COM est initialisé ici,
    // *AVANT* la déclaration des pointeurs intelligents
    COMInitializer ci;
    SmartCOMPtr<IContact> me, ct;
    SmartCOMPtr<IContactProperties> cp;
    SmartCOMPtr<IContactManager> cm;
    SmartCOMPtr<IPersistStreamInit> psi;
    SmartCOMPtr<IStream> is;
    std::wstring fullname;
    hr = ::CoCreateInstance(CLSID_ContactManager,
        NULL,
        CLSCTX_INPROC_SERVER,
        IID_IContactManager,
        reinterpret_cast<void**>(&cm));
    COMHelper::TestOk(hr);
    hr = cm->Initialize(L"MyAPP", L"1.0");
    COMHelper::TestOk(hr);
    // Obtenir le repository des contacts
    // à partir du contact de l'utilisateur
    hr = cm->GetMeContact(&me);
    COMHelper::TestOk(hr);
    hr = me->GetPath(buffer, size, &required);
    COMHelper::TestOk(hr);
    fullname = GetRepository(buffer);
    // et maintenant constituer le nom
    // complet du fichier de contact à créer
    fullname += FILENAME;
    // Creer une instance de l'interface IContact
    // pour le nouveau contact.
    // Le contact n'a toutefois encore aucune réalité
    hr = ::CoCreateInstance(CLSID_Contact,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_IContact,
                reinterpret_cast<void **>(&ct));
    COMHelper::TestOk(hr);
    // Définition du contenu (données) du contact
    // D'abord, obtenir l'instance de IPersistStreamInit
    // agrégée à l'instance du contact
    hr = ct->QueryInterface(IID_IPersistStreamInit,
        reinterpret_cast<void **>(&psi));
    COMHelper::TestOk(hr);
    // Initialiser l'instance de IPersistStreamInit
    // ce qui donne "vie" au contact en créant une
    // arborescence XML minimum
    hr = psi->InitNew();
    COMHelper::TestOk(hr);
    // Dès ce stade, notre contact détient un ID
    hr = ct->GetContactID(buffer, size, &required);
    COMHelper::TestOk(hr);
    std::wcout << L"ID du contact en cours de creation: "
        << buffer << std::endl;
    // Obtenir l'interface propriétés du contact
    hr = ct->QueryInterface(IID_IContactProperties,
        reinterpret_cast<void **>(&cp));
    COMHelper::TestOk(hr);
    // Pour le fun Récupérer l'ID du contact, mais
    // cette fois depuis les propriétés
    WCHAR propid[] = CONTACTPROP_PUB_L1_CONTACTIDCOLLECTION
        CONTACTPROP_PUB_L2_CONTACTID L"[1]"
            CONTACTPROP_PUB_L3_VALUE;
    hr = cp->GetString(propid, CGD_DEFAULT, buffer,
                ARRAYSIZE(buffer), &required);
    COMHelper::TestOk(hr);
    std::wcout << buffer << std::endl;
    // Définir le nom de famille et le prénom du contact
    // Pour cela, d'abord déclarer un noeud dans l'arborescence XML
    WCHAR nodebuffer[2048];
    hr = cp->CreateArrayNode(CONTACTPROP_PUB_L1_NAMECOLLECTION,
        CGD_DEFAULT,
        TRUE,
        nodebuffer,
        ARRAYSIZE(nodebuffer),
        &required);
    COMHelper::TestOk(hr);
    // Puis définition du nom de famille
    WCHAR propname[] = CONTACTPROP_PUB_L1_NAMECOLLECTION
        CONTACTPROP_PUB_L2_NAME L"[1]"
            CONTACTPROP_PUB_L3_FAMILYNAME;
    hr = cp->SetString(propname, CGD_DEFAULT, L"Aubin");
    COMHelper::TestOk(hr);
    // et définition du prénom
    WCHAR propgivenname[] = CONTACTPROP_PUB_L1_NAMECOLLECTION
        CONTACTPROP_PUB_L2_NAME L"[1]"
            CONTACTPROP_PUB_L3_GIVENNAME;
    hr = cp->SetString(propgivenname, CGD_DEFAULT, L"Marie");
    COMHelper::TestOk(hr);
    // Obtenir une instance de IStream, à partir du nom de fichier
    hr = ::SHCreateStreamOnFileEx(fullname.c_str(),
        STGM_CREATE|STGM_READWRITE,
        FILE_ATTRIBUTE_NORMAL, FALSE, NULL, &is);
    COMHelper::TestOk(hr);
    // et enfin SAUVEGARDE effective
    hr = psi->Save(&(*is), TRUE);
    COMHelper::TestOk(hr);
    std::cout << "Creation du contact effectuee "
        " avec succes" << std::endl;
  }
  catch(COMException ce) {
    std::cout << ce.raison() << std::endl;
    return EXIT_FAILURE;
  }
  return EXIT_SUCCESS;
}
```

DEVELOPPEZ VOTRE SAVOIR-FAIRE

ABONNEMENT .NET

Programmez ! ***est le magazine du développement***

Langage et code, développement web, carrières et métier : Programmez, c'est votre outil de veille technologique.

Pour votre développement personnel et professionnel, abonnez-vous à Programmez !

PROGRAMMEZ

- **Abonnement 1 an au magazine : 45 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- **+ Les 5 n° HS .NET : 18 €** (Supplément à l'abonnement Programmez) (au lieu de 25 €, prix au numéro) *Tarif France métropolitaine*
- **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 57 €** *Tarif France métropolitaine*
- **Abonnement PDF 1 an : 30 €** *- Tarif unique*
  Inscription et paiement exclusivement en ligne : **www.programmez.com**
- **Abonnement Etudiant : 1 an au magazine : 39 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*

PROGRAMMEZ HORS SERIE .NET *Seulement*

- **Abonnement 1 an aux Hors Série : 5 numéros : 20 €** (au lieu de 25 € tarif au numéro) *Tarif France métropolitaine*

Abonnez-vous :

Le magazine mensuel

11 numéros par an : **45 €***

soit **3** Numéros GRATUITS

** Tarif france métropolitaine*

Abonnez-vous :

Les Hors-Série .Net

5 numéros par an : **20 €***

soit **1** Numéro GRATUIT

** Tarif france métropolitaine*

+ Abonnement INTÉGRAL

NOUVEAU

ACCÈS ILLIMITÉ aux ARCHIVES du MAGAZINE pour 1€ par mois ! *Prix de lancement*

Cette option est réservée aux abonnés pour 1 an au magazine, quel que soit le type d'abonnement (Éco, Numérique, Etudiant). Le prix de leur abonnement normal est majoré de 12 € (prix de lancement, identique pour toutes zones géographiques). Pendant la durée de leur abonnement, ils ont ainsi accès, en supplément, à tous les anciens numéros et articles /dossiers parus.

OUI, *je m'abonne*

Vous pouvez aussi vous abonner en ligne et trouver tous les tarifs www.programmez.com

PROGRAMMEZ

- ☐ **Abonnement 1 an au magazine : 45 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- ☐ **+ Les 5 n° HS .NET : 18 €** (Supplément à l'abonnement Programmez) (au lieu de 25 €, prix au numéro) *Tarif France métropolitaine*
- ☐ **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 57 €** *Tarif France métropolitaine*
- ☐ **Abonnement Etudiant : 1 an au magazine : 39 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*

PROGRAMMEZ HORS SERIE .NET NOUVELLE OFFRE

- ☐ **Abonnement 1 an aux Hors Série : 5 numéros : 20 €** *(au lieu de 25 € tarif au numéro) Tarif France métropolitaine*

☐ M. ☐ Mme ☐ Mlle Entreprise : ..... Fonction : .....

Nom : ..... Prénom : .....

Adresse : .....

Code postal : ..... Ville : .....

Tél : ..... E-mail : .....

☐ Je joins mon règlement par chèque à l'ordre de Programmez ! ☐ Je souhaite régler à réception de facture

A remplir et retourner sous enveloppe affranchie à :

Programmez ! - Service Abonnements - 22 rue René Boulanger - 75472 Paris Cedex 10.

abonnements.programmez@groupe-gli.com

Programmez! LE MAGAZINE DU DÉVELOPPEMENT

Offre limitée, valable jusqu'au 30 mars 2008

Le renvoi du présent bulletin implique pour le souscripteur l'acceptation pleine et entière de toutes les conditions de vente de cette offre.

Conformément à la loi Informatique et Libertés du 05/01/78, vous disposez d'un droit d'accès et de rectification aux données vous concernant.
Par notre intermédiaire, vous pouvez être amené à recevoir des propositions d'autres sociétés ou associations.
Si vous ne le souhaitez pas, il vous suffit de nous écrire en nous précisant toutes vos coordonnées.

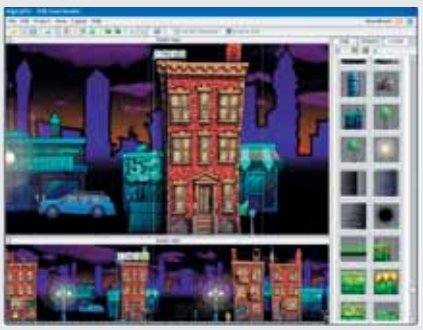
PROG 106

ACTUS

La course des consoles selon EA

Electronic Arts, à l'occasion de la publication de ses résultats de fin 2007, a diffusé des données sur les ventes globales de consoles en Europe, d'autant plus intéressantes que ces chiffres sont généralement bornés à chaque pays. Cette précieuse vue d'ensemble confirme ainsi que Nintendo, avec la DS (8,7 millions d'exemplaires) et la Wii (4,8 millions) garde un absolu leadership sur les modèles de salon comme les portables. Et le grand perdant de l'année 2007 est la Xbox 360, qui tombe à la dernière place avec moins de 2 millions d'unités, derrière Sony, dont les divers modèles (Play Station 2 en tête) cumulent près de 10 millions de pièces. Les prévisions 2008 de l'éditeur voient la Xbox poursuivre sa stagnation, tandis que la PlayStation 3 reviendrait quasiment à la hauteur de la Wii (respectivement 6 et 7 millions). Cependant, du côté des portables, l'écart resterait important entre PSP et DS, plus du simple au double (environ 3 millions contre plus de 6 millions).Ces informations et prospectives semblent tout à fait fiables et dignes de confiance, Electronic Arts étant par excellence l'éditeur multi-plate-forme, donc politiquement neutre.

Torque GB : objets améliorés



GarageGames vient de sortir une nouvelle version (la 1.7) de son Torque Game Builder. Rappelons que le logiciel est depuis plusieurs années une référence en matière de middleware de création de jeux à petit ou moyen budget. Le Game Builder combine programmation et gestion graphique, le tout spécialisé en 2D : sprites, tuiles, collisions, effets physiques, etc. Les améliorations de la nouvelle version portent surtout sur la gestion des objets des scènes, avec l'ajout d'un éditeur tout neuf, beaucoup plus complet, d'objets vectoriels (qui peuvent être intégrés en 3D). Au-delà du middleware général, les versions dédiées Xbox 360 et Wii intègrent aussi ces nouvelles fonctionnalités.www.garagegames.com/products/torque/tgb

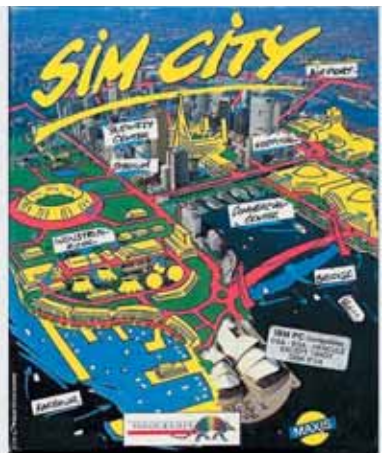
FaceTec : coucou, c'est moi !

Jouer avec sa propre tête comme personnage, est-ce un rêve ? A vous de voir. C'est en tout cas une fonctionnalité intéressante à offrir au joueur et à prévoir dès le développement. C'est ce que propose l'outil de reconnaissance graphique FaceTec : à partir d'une photo (webcam, scan, fichier classique, etc.), le logiciel crée un visage 3D qui peut s'intégrer instantanément au programme, animations (de base) comprises. Pour l'utilisateur, l'application est transparente (il suffit de charger la photo), et elle est traitée en quelques secondes pour devenir un des "objets" du jeu. Evidemment, FaceTec est aussi conçu pour s'adapter aux applications sur téléphones mobiles. www.genemation.com

Virtools Wii : interactivité directe

La version Wii de Virtools, attendue chez Dassault Systèmes depuis plusieurs mois, est enfin disponible. Le package Virtools Wii réunit tous les outils de prototypage et de production de jeux sur la console, en donnant accès à ses routines internes de comportement et au compilateur VSL. Depuis un PC, le développeur a ainsi accès à toutes les commandes des manettes Remote et peut les tester directement. Virtools Wii comprend aussi plus de 500 blocs de programmes prêts à être assemblés. Les modules de codes sont alors exécutables aussi bien sur la Wii que sur le PC, en cours de développement. Un mode " coopératif " entre plate-forme de développement et plate-forme de jeu qui est plus que précieux ! www.virtools.com

Sim City : ville ouverte !



C'est un événement historique : le jeu mythique de la fin des années 80, le fameux Sim City, devient Open Source ! Enfin, pour être exact, on l'appellera plutôt Micropolis dans cette version, le nom original étant la propriété d'Electronic Arts, qui continue à l'exploiter (Sim City Sociétés). On doit cette ouverture à Bill Simser et Don Hopkins, qui ont ré-écrit tout le code original (la toute première version était Mac) et l'ont mis aux normes C++ actuelles, avec une pleine compatibilité GNU/Linux. Le code a été intégré à Python, en utilisant le générateur d'interface SWIG. Les tuiles de base du jeu sont gérables par un TileEngine et un CellEngine, en modules Cairo intégrés ou indépendants. Les deux programmeurs n'ont cependant effectué aucune modification fondamentale sur la structure du jeu lui-même. Pour l'anecdote, il faut juste noter la suppression de la catastrophe " crash aérien " sur la ville, souvenir du 11 septembre oblige. Bill Simser insiste par ailleurs sur le caractère très particulier du code : " Il était en C, créé avant 1983. Il faut bien garder à l'esprit que le jeu devait alors tenir dans 640 Ko de mémoire vive. Les programmeurs d'alors ont donc employé des techniques très inventives pour contourner les problèmes, ce qui rend certaines portions du code plutôt mystérieuses et confuses. " Le code-source de " Micropolis " est donc désormais téléchargeable sur la page web de Don Hopkins (www.donhopkins.com/home/ micropolis), qui espère bien voir se créer une communauté de fanatiques pour retravailler le chef-d'œuvre des temps anciens. Sim City appartient sans doute à un genre moins populaire chez les programmeurs actuels que Quake, dont la communauté OpenArena est très active. La version 2D originale de Sim City autorise aussi moins d'innovations graphiques. Reste cependant, à inventer un nouveau jeu…

PAR JEAN-MICHEL MAMAN

Intelligence Artificielle : 5 stades d'évolution

Que va apporter l'intelligence artificielle des jeux de 2008 ? Gamasutra et les têtes pensantes de l'AI Game Dev. ont dévoilé leur vision des grandes tendances, qui s'organisent pour eux selon cinq axes :

Le sens coopératif : qu'il s'agisse de groupes d'ennemis ou d'une assistance pour le héros (un personnage secondaire auxiliaire comme le chien de Fable 2), la tendance va vers une IA qui fonctionne par interaction entre les personnages du jeu. Bref, moins de réflexion solitaire, mais des décisions concertées de groupes, apportant plus de profondeur au jeu. Cette prévision de tendance s'inspire clairement du remarquable travail de comportement déjà effectué sur le nouvel Half-Life Episode 2, d'ailleurs primé.

fear

halo3

stalker

L'autonomie : le contexte de jeu doit désormais laisser une forte liberté au joueur, ce qui vient d'être remarquablement démontré par S.T.A.L.K.E.R. et Crysis. Ce qui implique donc de construire une I.A immersive, qui doit pouvoir s'adapter à des événements imprévus. C'est l'obligation d'intégrer l'intelligence artificielle dans la structure même du jeu, environnementale, afin qu'elle puisse évoluer au-delà d'elle-même. Bref, si l'on peut dire, la décision descend de l'arbre ! Le prochain Grand Theft Auto 4, malgré son scénario ludique primaire, promet sur ce point une autonomie de décisions inégalée.

L'émergence : même dans les scénarii à déroulement presque linéaire, l'intelligence artificielle doit développer des comportements émergents, programmés d'avance selon la ligne du gameplay. Ainsi, dans Bioshock, le comportement biologique des " ennemis " évolue selon une progression pré-établie, que le joueur doit comprendre pour pouvoir en tirer parti. Dans Halo 3, l'apparition de groupes d'adversaires de types différents crée aussi de nouvelles attitudes de combat, gérées à l'avance. La tendance 2008 va ainsi vers une I.A. non linéaire à l'intérieur d'une histoire à étapes fixes.

La planification : la notion est intermédiaire entre l'autonomie et l'émergence. Le but est de construire des routines d'I.A. uniques, prêtes à l'emploi, bref, des bibliothèques de comportements selon les situations. Pour éviter tout schématisme, il faut plutôt parler de " briques " d'intelligence, extrêmement pointues et diversifiées. Au développeur de les assembler minutieusement, et selon une hiérarchie pertinente. Il y a ainsi gain de temps de développement, sans perte de finesse ou de variété de décisions. Ce procédé a fait dès 2006 le succès des monstres diaboliquement intelligents de F.E.A.R. Aujourd'hui, les bons middlewares d'IA intègrent donc toujours un module " planner " d'organisation et de hiérarchisation de ces briques de comportement : elles vont construire l'intelligence des plus grands jeux de 2008.

Le " danger " Lua : le langage Lua a peu à peu investi le domaine de l'I.A, comme le montre World of Warcraft (intelligence sommaire, mais rapide) ou encore Crysis (seules les portions les plus ardues de son IA sont en C++). Si cette tendance se prolonge, le problème que mettent en avant les développeurs de l'AI Game Dev est que l'interpréteur Lua, gérant aussi la plupart des paramètres logiques du jeu, risque d'être vite surchargé, en traitement comme en allocation mémoire. Avec en outre une prolifération de bugs, vu le côté " usine à gaz " d'une telle programmation… L'IA doit-elle pour autant rester dans le bastion de ses langages et routines spécifiques ? Ce n'est pas la tendance des grands développements actuels !

Bioshock

Crysis

Fable2

half2

WPF

- ***Difficulté : *****
- ***Editeur : Dunod***
- ***Auteur : collectif***
- ***Prix : 24,90 €***

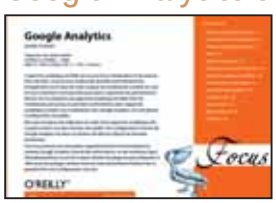
Enfin un bon ouvrage, en français, sur WPF, la nouvelle couche de présentation de .Net. Ce n'est pas un guide de référence mais une grande introduction à WPF : son architecture, son fonctionnement, sa mise en œuvre, ce qui est déjà bien. Avant tout didactique, les auteurs (provenant du labo .Net de Supinfo) proposent une approche claire, illustrée, sans chercher à être exhaustif dans le code. On retrouve l'ensemble des éléments composants WPF : les contrôles, la 2D / 3D, XBAP, le layout, les templates, la gestion des ressources, les styles, la personnalisation des contrôles. On termine par un exemple complet d'application.

Expression Web

- ***Difficulté : *****
- ***Editeur : Eni éditions***
- ***Auteur : Louise Villeneuve***
- ***Prix : 21,19 €***

Vous ne connaissez pas encore Expression Web, le nouvel IDE Microsoft pour le développement web en ASP.Net, HTML ? C'est l'occasion de combler vos lacunes. L'ouvrage se présente comme un guide du développement dans lequel l'auteur revient sur les fonctions, l'environnement, les modules de l'outil, très visuel, cela devrait aider à rapidement mettre en œuvre ces fonctions dans vos projets. Une bonne introduction.

Google Analytics décrypté



- ***Difficulté : *****
- ***Editeur : O'Reilly***
- ***Auteur : Justin Cutroni***
- ***Prix : 10 €***

Outil de référence pour le suivi de sites web, gratuit, Google Analytics possède une puissance fonctionnelle parfois peu connue. Ce focus O'Reilly décrypte les rouages de ce service. Après avoir expliqué comment fonctionne Google Analytics, l'auteur montre comment créer et configurer les profils qui stockent les données liées à un site Web particulier. Les filtres sont alors étudiés en détail. Ils séparent - par exemple - le trafic généré en interne de celui provoqué par les visiteurs d'un site. Les " objectifs " permettent de traquer les actions particulières des visiteurs. Les " entonnoirs " définissent les étapes éventuelles à franchir avant d'atteindre la page d'objectif. À partir de ces notions de base, l'auteur aborde les configurations courantes de Google Analytics, AdWords, etc. Edition PDF seulement.

L'intelligence collective ou les algorithmes qui font le web de demain

- ***Difficulté : ******
- ***Editeur : O'Reilly***
- ***Auteur : Toby Segaran***
- ***Prix : 42 €***

Les algorithmes d'exploitation de l'intelligence collective sont un passage obligé pour quiconque souhaite s'adapter au comportement des internautes et leur offrir des services utiles et efficaces. Ces algorithmes exploitent les comportements de groupe en collectant les données venant d'ensembles de personnes, puis en les combinant et en les analysant. L'auteur aborde la problématique des filtrages collaboratifs et des recommandations effectuées par les visiteurs d'un site, la formation et la découverte de groupes, l'indexation, la recherche. On y trouve aussi une section sur le filtrage, l'optimisation, la modélisation par des arbres décisionnels. Un livre qui deviendra rapidement une référence.

Premières applications web 2.0 avec Ajax et PHP

- ***Difficulté : ******
- ***Editeur : Eyrolles***
- ***Auteur : J-M Defrance***
- ***Prix : 39,90 €***

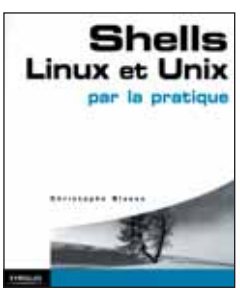
On parle depuis longtemps d'Ajax, de web 2, de PHP. Mais finalement, comment s'y prendre quand on est habitué à ASP ou à HTML ? On débute par une mise au point sur Ajax : de quoi il se compose, le modèle de développement, les +, les –… Les choses sérieuses débutent avec http et l'objet XMLHttpRequest, la base d'Ajax. Côté infrastructure, on aborde WAMP, Firefox, IE. Les outils de développement ne sont pas oubliés, même si l'auteur se limite à Dreamweaver. La partie codage explique comment interagissent Ajax et PHP (ensemble), avec ou sans paramètres (par exemple avec GET, POST). L'auteur n'oublie pas JSON, MySQL, ni les plug-in… Une parfaite introduction technique pour le développeur web.

Sharepoint server 2007 volume 2

- ***Difficulté : ******
- ***Editeur : Microsoft Press***
- ***Auteur : collectif***
- ***Prix : 49 €***

Dans ce second volume sur le très stratégique serveur Microsoft, les auteurs abordent la mise en œuvre de solutions et la gestion des informations. L'ouvrage repose sur une approche pratique, illustrée de nombreux exemples, ce guide contient des informations détaillées sur toutes les fonctionnalités, des conseils d'experts, des procédures et des astuces pratiques pour exploiter SharePoint Server 2007 dans les meilleures conditions. On y verra l'ensemble des possibilités comme les webpart d'Office, l'Office Forms ou encore les migrations.

Shells Linux et Unix par la pratique



- ***Difficulté : ******
- ***Editeur : Eyrolles***
- ***Auteur : Christophe Blaess***
- ***Prix : 39 €***

Le Shell est une des bases fondamentales des systèmes Unix. Mais encore faut-il les maîtriser pour l'exploiter au mieux. Qu'est-ce que le langage Shell ? Comment le mettre en œuvre ? Comment bien l'utiliser, le coder ? L'auteur tente de vous apporter ses lumières avec un apprentissage progressif. Plutôt orienté étudiant, chaque chapitre se termine par des exercices pratiques pour savoir si on a bien assimilé. Si vous voulez mieux exploiter votre Linux, c'est le moment de vous y mettre !


SERVEUR ECO D'AMEN
UNE TECHNOLOGIE
EN HARMONIE
AVEC L'ECOSYSTÈME.
D'ÉMISSION
-55%
DE CO2*
amen
IN WEB WE TRUST
A partir de
49 € HT/mois **
soit 58,60 € TTC/mois
NE CHOISISSEZ PLUS ENTRE
PERFORMANCE ET ÉCOLOGIE.
SERVEURS ECO D'AMEN :
DE 1 À 3 Go DE RAM,
55 % D'ÉMISSION DE CO2
EN MOINS PAR AN.
http://eco.amen.fr
AMD Athlon X2 3400+ ou BE-2350
De 2x1,8GHz à 2x2,1GHz
Interface Plesk 8.2 – 10 domaines
OS : Fedora Core 7, Ubuntu 6, Debian 4, Windows Server 2003
De 1 à 3 Go de RAM
Disque dur de 80 Go à 160 Go
Trafic mensuel : 1 To
1 à 2 adresses IP
Inclus : Reboot, Restore et Recovery
En option : Amen DataBackup 10 Go (Inclus sur ECO 3000)
AMD 64 Athlon X2
PARTENAIRE INDUSTRIEL
entrepreneurs. faites le choix de l'économie numérique
www.econumerique.pme.gouv.fr
NUMÉRO GRATUIT : 0800 740 935 ou www.amen.fr
NOMS DE DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT
* comparé à un serveur équivalent de la gamme contact - ** pour un engagement annuel - AMEN RCS PARIS : B 421 527 797 - IN WEB WE TRUST : Nous croyons au web. Conditions Générales de Vente sur le site www.amen.fr - REFLEXION FAITE