PROGRAMMEZ !
Le magazine du développement
PROgrammez !
www.programmez.com
mensuel n°133 - septembre 2010

Agile et productif !

- Adopter le meilleur des méthodes agiles
- Productivité du développeur : un code plus propre et de qualité
- Nos experts témoignent
- Reportage : pas d'heures supplémentaires, ni retard sur les projets !

Silverlight
Fonctions audio, comment ça marche ?

Dreamweaver CS5
Toujours plus puissant pour le web !

Google Maps
La géolocalisation sous Android

SGBD
Le futur des bases de données

Sécurité
Microsoft étend sa méthodologie SDL

Windows
Synchronisation avec le framework Sync

Outils
Utiliser les Private Build

Code
Blinder votre code avec SOLID

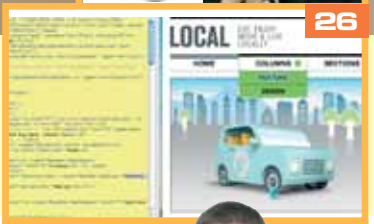
Smartphone
Vos jeux pour Windows Phone 7 avec XNA

M 04319 - 133 - F: 5,95 €
Printed in France - Imprimé en France - BELGIQUE 6,45 €
SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 €
Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

DÉVELOPPEZ 10 FOIS PLUS VITE
WINDEV
Tout est inclus
Gère le cycle complet de développement.
Support Technique gratuit
Elu «Langage le plus productif du marché»
VERSION EXPRESS GRATUITE
Téléchargez-la !
Fournisseur Officiel de la Préparation Olympique
WINDEV®
ENVIRONNEMENT PROFESSIONNEL DE DÉVELOPPEMENT (AGL)
Créez des applications pour Windows, Internet, Mobile et SaaS
Java, .Net, PHP, J2EE, Webservice, XML, Ajax, Linux, Android, Client riche, Base de Données...
AGL N°1 EN FRANCE
Demandez le dossier + DVD gratuit
555 NOUVEAUTÉS
PC SOFT 25e ANNÉE Au service des développeurs
WINDEV 15
www.pcsoft.fr
Dossier gratuit 200 pages sur simple demande. Tél: 04.67.032.032 info@pcsoft.fr
document non contractuel. * 15 requêtes gratuites sur la version en cours de commercialisation, seule la communication est à votre charge. Logiciel professionnel

PROgrammez! Le magazine du développement
numéro 133 - septembre 2010

sommaire\\



éditO

Le café n'est plus ce qu'il était

Surprise ! Alors que le mois d'août se passait calmement, hormis la polémique autour de la lettre de Google et Verizon sur comment faire sauter la neutralité du web sans le dire, voici que vendredi 13, la nouvelle se répandait partout : Oracle attaque en justice Google pour violation de brevets liés à Java dans le système d'exploitation mobile Android. Des débats sans fin ont alors déferlé sur les forums. Oracle perd-t-il ses nerfs ? L'éditeur qui cherchait à être un ami de l'open source va-t-il ruiner ses chances d'être en bons termes avec le Libre ? Bien évidemment, cette affaire ne peut se résumer à une simple bisbille. Au contraire, cette offensive est une volonté d'Oracle de protéger Java (depuis le rachat de Sun) et d'en retirer les dividendes financiers…

Pour Oracle, Android viole plusieurs brevets et copyrights Java aujourd'hui détenus par l'éditeur depuis le rachat de Sun. L'un des litiges vient de la machine virtuelle Dalvik. Celle-ci recompile le code Java en bytecode, non pas Java, comme dans une machine virtuelle Java classique, mais en bytecode Dalvik pour être exécuté sur des terminaux Android. Pour Oracle, Dalvik est clairement un concurrent à Java dans le sens où Dalvik prend du code Java et le recompile à sa sauce. Dalvik n'est pas donc une machine Java au sens strict du terme. Pour Google, Dalvik était un moyen d'éviter les licences et les spécifications Java (pour être certifié) tout en éloignant les brevets logiciels liés à ce langage. Mais pour Oracle, Dalvik viole délibérément ces deux éléments que Google cherchait à éviter… Le serpent qui se mord la queue en quelque sorte. Notons que le code Java n'est pas exécuté tel quel sur Android, il est transformé. Une des questions est de savoir si les concepteurs de Dalvik sont partis d'une feuille blanche pour écrire la compatibilité Java ou s'ils avaient une connaissance du code de la machine virtuelle Java de Sun.

Pourquoi maintenant ? Pourquoi pas Sun ? Analystes et journalistes spécialisés pointent du doigt le fait que Sun puis Oracle attendaient une masse critique d' Android sur le marché pour pouvoir réclamer leur licence. James Gosling, récemment parti d' Oracle, écrit sur son blog sa non-surprise que dès les discussions de rachats, des réunions abordèrent la questions des licences et brevets entre Sun et Google, à la grande joie des avocats d'Oracle… Cependant, reste à Oracle à prouver les violations et à apporter des éléments concrets.

Précisons un point important. Java est open source et sous licence GPL. Cependant, l'utilisation de Java est libre et gratuite, pour un éditeur, dans un fork, uniquement si les spécifications sont respectées et implémentées à 100 %. Dans le cas contraire, il faut passer à la caisse. Et en ce sens, Dalvik ne respecte pas les spécifications à 100 %.

Impossible à l'heure où nous écrivons de savoir comment l'affaire va se dérouler, ni l'impact sur la communauté des développeurs, le marché Android et les éditeurs. C'est au minimum une ombre portée à Android avec une incertitude qui peut faire hésiter développeurs et entreprises. Par contre, Oracle risque de nouveau, après les hésitations et stratégies confuses autour de Solaris / Open Solaris, MySQL, OpenOffice, de subir les réactions négatives des communautés open source, des développeurs et de favoriser des forks.

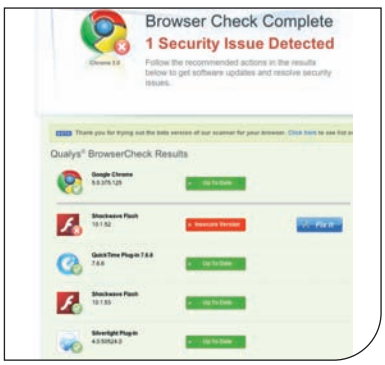
■ François **Tonic**
Rédacteur en chef
ftonic@programmez.com

Le magazine du développement
PROgrammez!
www.programmez.com

Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com.
Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky.
Rédacteur en Chef : François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazue, J.C. Vasselon, S. Belkhayat-Fuchs. **Experts :** R. Baduel, J.B. Cazeaux, L. Carbonneaux, X. Warzee, M. Szablowski, E. Mignot, P. Ognibene, A. Kolawa, S. Belkhayat-Fuchs, E. Stewart, A. Haouari, F. Bellahoene, G. Rouchon, J. Sautret, F. Pedro, M. Denié, J. Dollon, C. Faydi, T. Laurent, D. Deraedt, J. De Oliveira, F. Bellahcene.
Illustration couverture : © iStockPhoto.com/4x6
© iStockPhoto.com/YanC
Publicité : Régie publicitaire, K-Now sarl. Pour la publicité uniquement : Tél. : 01 41 77 16 03 - diff@programmez.com.
Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique. **Directeur de la publication :** J-C Vaudecrane

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10
Tél. : 01 55 56 70 55
abonnements.programmez @groupe-gli.com
Fax : 01 40 03 97 79 - 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 : 49 € - Etudiant : 39 € - CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € Dom : 66,82 € - Autres pays : nous consulter. **PDF :** 30 € (Monde Entier) souscription exclusivement sur www.programmez.com

■ **RIM** a dévoilé un SDK Java pour son nouveau Blackberry 6, la dernière évolution de son système d'exploitation mobile. Les deux éléments donnent accès pour les développeurs au moteur webkit, à la prise en charge partielle de HTML5, et proposent de nouvelles interfaces, des fonctions natives, ou encore une meilleure géolocalisation.



■ Comment tester la sécurité des navigateurs internet ? **Qualys** propose un service gratuit pour le faire. La plupart des attaques à base de codes malveillants ciblent en priorité les failles de sécurité au sein des navigateurs Web modernes et de leurs plug-ins. Mais peu d'utilisateurs sont conscients des risques auxquels les exposent leurs navigateurs Web chaque fois qu'ils font des achats en ligne, qu'ils jouent en ligne avec un navigateur ou qu'ils s'informent sur Internet. Site : https://browsercheck.qualys.com/

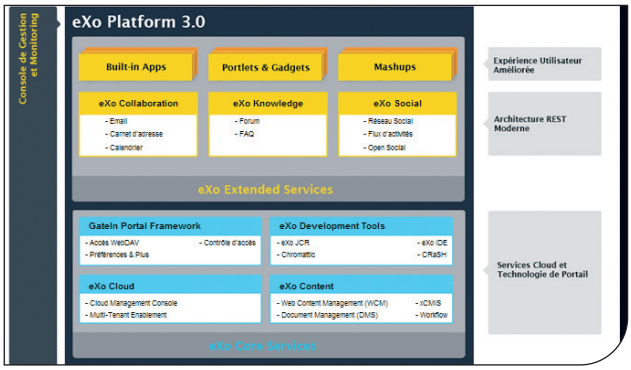
■ **Google** met fin à son service de communication tout intégré : Google Wave. Trop en avance, trop complexe, trop compliqué à utiliser, les causes ne sont pas clairement annoncées. C'est un échec pour Google mais le code sera ouvert et des fonctions de Wave devraient se retrouver dans d'autres services de l'éditeur.

■ **Nvidia** dévoile Parallel Nsight pour Visual Studio. L'outil permet de développer des applications CUDA C/C++ ou DirectCompute sur le GPU en employant les outils et techniques qui leur sont familiers pour le CPU. Parallel Nsight fournit aussi les outils d'analyse procurant aux développeurs les informations requises pour atteindre les plus hauts niveaux de performance applicative GPGPU.

Langage

Java dans les nuages : les grandes manœuvres

Le cloud computing serait-il le nouvel El Dorado des langages de programmation ? La réponse est un triple oui. Car aujourd'hui les principaux fournisseurs de plates-formes de cloud (PaaS) cherchent à être interopérables avec les langages. Java est l'un des plus courtisés depuis quelques mois. Windows Azure possède des SDK dédiés, VMware avec le rachat de SpringSource possède une pépite : un framework Java populaire et performant sur l'approche JEE. Ce n'est donc pas un hasard si Google et Salesforce.com proposent avec Spring un support de Java permettant dans les prochains mois de pouvoir migrer les applications Java / JEE sur leur cloud. Attention tout de même aux compatibilités et au niveau d'API requis, car c'est le point le plus sensible. Et sur lequel, les fournisseurs se montrent discrets.



Fin juillet, c'est au tour de l'éditeur français eXo d'annoncer un support étendu de Java via un programme « early adopter Programm » pour eXo Platform 3.0. Ce programme a été conçu pour donner accès aux entreprises utilisant Java à un ensemble de formations et de ressources techniques dont elles ont besoin pour développer des applications Java riches reposant sur eXo Platform 3.0.

Cette plate-forme, future offre phare de l'éditeur qui sera disponible d'ici la fin de l'année, a été entièrement repensée et bâtie sur le portail Gateln, co-développé par eXo et Red Hat. *« Nous avons déjà énormément de feedback positif sur eXo Platform 3.0 même à ce stade précoce du produit et cela valide l'investissement lourd que nous avons réalisé pour proposer la meilleure expérience utilisateur possible pour les entreprises Java. Ce programme donne tous les outils et la puissance nécessaire aux développeurs pour créer facilement des applications Java riches et facilite le passage en production de celles-ci »*, déclare Benjamin Mestrallet, PDG et fondateur d'eXo.

Evolution

Mozilla veut changer les onglets des navigateurs

Mozilla a dévoilé un nouveau projet dédié à l'interface et particulièrement à la navigation par onglet : Tabcandy. L'objectif est de faciliter la navigation quand l'utilisateur a un grand nombre d'onglets ouverts : 10, 15, 20. Il pourrait les faire défiler (les noms dans la barre onglet), ou encore avoir une visualisation par type : communication, à lire, etc. qui peut rassembler plusieurs onglets. On peut même les réorganiser et quand on affiche une « catégorie d'onglet », seuls ceux qui sont liés s'affichent alors dans le navigateur. Pourquoi pas, l'idée est assez séduisante mais est-ce réellement la meilleure interface de navigation pour les onglets ? Une chose est sûre, l'auteur du projet dit clairement qu'il s'agit d'un croisement entre Exposé et Spaces, deux fonctions de MacOS X. Il est envisagé de l'intégrer dans le futur Firefox 4 mais le projet n'en est qu'à ses débuts et son intégration demeure (très) incertaine. Le tout est développé en langages web : html, css, javascript.

Les animations sont assurées par css.

Sites :

http://azarask.in/projects/tabcandy/

http://www.azarask.in/blog/post/tabcandy/

Rejoignez la communauté de développeurs Nokia,

et créez des applications
pour les millions de Smartphones Nokia
à travers le monde !

Découvrez les nouveaux SDK Nokia

Et distribuez vos applications sur Ovi Store

auprès de millions d'utilisateurs Nokia.

Rendez-vous sur
www.forum.nokia.com/develop

NOKIA
Connecting People*

NOKIA
OVI

■ **Intel** a dévoilé une nouvelle approche pour les futurs prochains processeurs : Silicon Photonics. Il s'agit de l'utilisation de la lumière en lieu et place des électrons transportant les données. Le fondeur a en effet mis au point un prototype de recherche qui constitue la première liaison optique de données à base de silicium avec lasers intégrés. Celle-ci peut acheminer les données sur de plus longues distances et largement plus rapidement que le cuivre aujourd'hui, soit jusqu'à 50 gigabits de données par seconde. Il s'agit uniquement d'un projet de recherche, la disponibilité de ce type de puces prendra des années car il faut passer au stade industriel.

■ **Debian** se prépare au lancement de sa prochaine version majeure : Debian 6.0. Début août, l'équipe a annoncé le gel de la version, prélude à sa sortie. Elle se base sur un noyau 2.6.32, KDE 4.4.5, Gnome 2.3, les dernières versions stables d'Apache, PHP, MySQL, etc. Les développeurs s'occupent de tester et de déboguer la v6.

■ **Gnome** termine la mise au point de la v3 de son interface Gnome. Elle devrait être disponible fin septembre et les premières distributions l'incluant pourraient apparaître dès octobre ou novembre. Gnome 3 s'articule autour de Gnome-shell qui constitue une nouvelle approche de l'espace de travail. Gnome-zeitgest doit assurer une meilleure gestion des fichiers. Gnome veut ainsi rattraper le retard pris sur KDE 4.x.

Solution

Oracle renforce sa solution Business Intelligence

C'est en juillet dernier que Oracle a lancé mondialement sa nouvelle offre dédiée à la business intelligence : Oracle Business Intelligence. Avec un marché en explosion et des concurrents très actifs, Oracle ne veut pas se laisser distancer par IBM, SAP ou Microsoft. Les axes pour Oracle BI 11g sont très nombreux. L'un des objectifs est de proposer une intégration la plus large possible des sources de données pour pouvoir les traiter, les analyser et les présenter. Cela passe par une refonte profonde de l'interface et de la manière de présenter et de consommer la BI. Oracle voit BI 11g comme le sang qui irrigue l'ensemble du système d'informations. Et l'objectif est clairement de devenir le n°1 du marché.

Des qualificatifs ont été régulièrement cités par les représentants de l'éditeur : intégré, packagé, complet. Car BI 11g est une étape supplémentaire dans l'unification de l'architecture logicielle Oracle dans Fusion Middleware. Et BI 11g intègre de nouveaux frameworks comme le Action Framework. Le but ici est d'être le plus ouvert possible avec les autres outils et logiciels Oracle ce qui n'était pas forcément le cas précédemment.

Et cette nouvelle offre possède une administration renforcée pour avoir une vue d'ensemble de sa BI. La mobilité n'a pas été oubliée avec l'apparition d'un client iPad, et la partie mobile peut collaborer avec la couche entreprise par exemple pour demander des modifications, des rajouts (comme une nouvelle langue). Sur la personnalité et l'adaptation des rapports, le nouveau Publisher est assez pratique et surtout tout est découpé en couches permettant une modification rapide du contenu ou de la couche de présentation, par exemple pour rajouter une colonne à un tableau ou encore pour localiser un rapport.

Un autre mot d'ordre fut l'intégration de BI 11g avec les environnements SAP dont les utilisateurs sont aussi ceux d'Oracle. Notons aussi que Sun n'est pas présent, hormis pour la partie stockage et, étonnamment, si l'intégration avec Microsoft Office est à plusieurs reprises citée, rien du côté OpenOffice, indice que la suite open source n'est pas à niveau ou très présente sur les données stratégiques et les analyses en entreprise ?

L'éditeur met en avant plusieurs grosses nouveautés telles que la consultation et l'analyse des sources relationnelles, OLAP (R-OLAP et M-OLAP) et XML. Une interface et ergonomie plus simple pour l'utilisateur final, des tableaux de bord plus complets, plus dynamiques, une plus large ouverture vers les sources de données, compatibilité et intégration fine avec Microsoft Office, un nouvel environnement de création de rapport entièrement en mode web…

agenda \\

SEPTEMBRE

• Du 09 au 10 septembre 2010, Paris – Porte de Versailles. Le premier événement européen sur la recherche et développement dans le domaine du jeu vidéo : **Future Game On.** http://www.futuregameon.com

• Du 10 au 12 septembre, Parc des expositions de Paris – Porte de Versailles **Festival du Jeu Vidéo 2010.** http://www.festivaldujeuvideo.com

• Le 22 septembre 2010, Séminaires **Business Intelligence en Libre-service** avec Microsoft SQL Server 2008 R2 et PowerPivot Microsoft France - Centre de Conférences 41 Quai du Président Roosevelt - 92130 Issy Les Moulineaux http://www.finelog.fr

• Du 30 septembre au 1er octobre 2010, Paris la 3e édition de l'**Open World Forum** rassemblera les décideurs mondiaux du numérique ouvert, sous le signe de l'innovation Libre et Open Source. http://www.openworldforum.org

• du 29 septembre au 3 novembre, **Microsoft Days**. 7 villes : 30/09 : Aix - 6 et 7/10 : Issy, 12/10 : Lyon, 14/10 : Toulouse, 19/10 : Strasbourg, 21/10 : Lille, 3/11 : Nantes.

OCTOBRE

• Le 19 octobre, Bois Colombes, **Innovate 2010** **Rational Software Conference** : conférence annuelle de IBM Rational sur le développement logiciel, l'agilité, la gestion de projets avec les solutions Rational. http://www.-05.ibm.com/fr/events/innovate_2010/

ETRANGER

• Du 19 au 21 septembre 2010, San Francisco, Californie, **JavaOne 2010**, l'événement Java de l'année avec toutes les nouveautés, le futur du java, etc. http://www.oracle.com/us/javaonedeve-lop/062264.html

• Du 27 septembre 2010 au 29 septembre 2010, San Diego Californie, **iPhone/iPad DevCon 2010**. Evénement majeur pour les développeurs iPhone et iPad http://www.iphonedevcon.com

LES CONTRÔLES LES PLUS RAPIDES...

Chez Infragistics, nous nous assurons que nos contrôles pour .Net vous permettent de créer les meilleures interfaces utilisateur possible. C'est pourquoi nous avons testé et retesté nos Data Grids afin de nous assurer qu'elles sont les plus rapides sur le marché et que nos Data Charts surpassent tout ce que vous avez expérimenté. Utilisez nos contrôles et vous obtiendrez non seulement le temps de téléchargement le plus rapide mais aussi des applications qui sont toujours attrayantes. Rapide et belle... C' est une « Killer app ». Essayez les vous-même sur **infragistics.com/wow**.

Infragistics®
KILLER APPS. NO EXCUSES.

Infragistics Ventes France N° Vert 0800 667 307
Infragistics Europe Ventes +44 (0) 800 298 9055
twitter.com/infragistics

Imagine Cup 2010 : les objectifs millénaires

Le concours Imagine Cup organisé par Microsoft depuis maintenant 7 ans rassemble chaque année des milliers d'étudiants. Depuis 3 ans, les étudiants doivent répondre aux objectifs du millénaire fixés par l'ONU (supprimer la pauvreté et la faim, promouvoir l'égalité des sexes, etc). Le thème étant : « Imaginez un monde où la technologie résout les problèmes les plus coriaces ».


Fig.2

Cette année, la compétition a accueilli plus de 325 000 compétiteurs venant de 100 pays différents. Lors de la finale à Varsovie, 113 équipes étaient présentes soit plus de 400 étudiants venant de 70 pays différents.

La compétition tourne autour de trois catégories principales : Software Design (conception de logiciel), Embedded Development (Développement embarqué) et Game Design (conception de jeu). A celles-ci se rajoutent le challenge IT et la catégorie Digital Media. Les équipes ont en plus la possibilité de se qualifier pour les awards, des récompenses précises touchant des domaines particuliers tels que : Windows Phone 7, l'Interopérabilité, Internet Explorer 8, projection en 2020, les interfaces tactiles, les applications web. Au total, 240 000 dollars récompensent les meilleures équipes pour toutes les catégories et awards. Les étudiants apportent une fraîcheur, des projets riches souvent novateurs et intéressants. La plupart sont de futurs ingénieurs. Ils sont là pour raconter une histoire, pour mener un projet jusqu'au bout et améliorer le monde. Ils connaissent particulièrement bien leur environnement technologique et jouent avec pour mettre en pratique leurs idées, leur vision d'un monde meilleur. On se retrouve face à des étudiants venant de pays et cultures différents, souriant et plaisantant ensemble. Au-delà de la compétition, c'est un échange humain et culturel. Les étudiants en sortent grandis. Ils se démènent pour gagner, mais les relations demeurent très saines et chacun trouve un intérêt dans le projet des "adversaires".

La France avait trois projets en lice dans les catégories maîtresse. **Babies Angel** de *SUPINFO Bordeaux* en Software Design, **GERAS** de *l'ECE Paris* en Embedded Development et **Gears Studio** de *3IL Rodez* en Game Design. L'équipe à la base de **Babies Angel** [Fig.1] est constituée de trois étudiants dont deux terminant leur dernière année. Leur travail a porté sur la surveillance des bébés et cible les jeunes parents. 30 000 bébés meurent de l'apnée du nouveau né chaque jour. Ils ont donc mis un système de surveillance et d'alerte basé sur du matériel certifié pour garantir la meilleure sécurité possible. Un tapis couvre le fond du berceau de l'enfant et détecte un arrêt respiratoire. A ce moment là une alerte visuelle et auditive se déclenche, un signal est envoyé au cloud qui transmet une notification Push aux téléphones mobiles des parents. Une caméra IP permet, depuis Internet, de surveiller l'enfant en temps réel. A ce système de prévention s'ajoute une application web et mobile servant de carnet médical de l'enfant pour suivre les allergies et la médicamentation. Un écosystème autour des produits pour bébés a aussi été mise en place permettant de détecter plus facilement les allergies de l'enfant sur un produit particulier, ou gérer les stocks des consommables.

Ils ont mis l'accent sur l'interopérabilité avec une optimisation de la plate-forme web en Javascript et Ajax pour avoir un système léger et vraiment multiplateforme (ce qui, pour eux, ne pouvait se faire avec du Silverlight ou Flash). Côté mobile, c'est actuellement une application Windows Mobile mais ils prévoient de toucher d'autres smartphones. Malgré leur défaite, ils croient réellement dans leur projet et souhaitent le continuer en vue d'une commercialisation. A suivre !. **Gears Studio** [Fig.2] a créé un jeu d'aventure en XNA autour de trois personnages ayant des aptitudes complémentaires. Ce groupe d'aventuriers évolue au sein d'une île paradisiaque qui a vu son quotidien transformé par des machines pollueuses. La mission du joueur est d'accomplir des quêtes en lien avec les objectifs du millénaire pour ramener le calme et redonner sa beauté à l'île. L'équipe a fourni un formidable travail sur le moteur du jeu. Avec la mise en place d'une ambiance très particulière et une vision communautaire grâce à un éditeur avancé. Leur premier trailer, à voir sur : http://www.youtube. com/watch?v=9Hc-GTILKI8. Le jeu a bien évolué depuis et un nouveau trailer doit bientôt sortir.

La problématique abordée par les membres du projet **GERAS** [Fig.3] concerne la chute des personnes âgées. Tout le monde, dans son entourage proche, a ou sera confronté à ce drame et les solutions pour y pallier ne sont pas encore au point (la plupart étant des bracelets plutôt intrusifs). Pour cela, ils ont posé des dalles au sol détectant des masses


Fig.1

Fig.3

ainsi qu'un capteur de présence dans la pièce. Si le capteur de présence ne détecte plus personne et que par contre une masse importante est signalée au sol alors l'alerte est donnée. La personne peut alors infirmer l'alerte par ordre vocal si c'est juste un faux-positif.
Ils ont vraiment travaillé sur l'aspect « invisible » et non intrusif de la technologie. Une personne âgée ne doit pas avoir à configurer un système ou à apprendre à interagir avec. Le principal problème auquel ils ont du faire face est la limitation des coûts. Mais avec un système basé sur de nombreuses petites dalles, le prix du capteur devient faible. L'installation peut être prévue directement lors de la construction de nouvelles infrastructures médicales ou se poser après sur une surface existante. Sur du moyen terme, ils prévoient d'ailleurs la possibilité de l'implémenter dans du parquet flottant. La France remporte une médaille dans deux des cinq catégories principales (voir les podiums).

Nos coups de coeur

L'équipe **Skeek** de Thailande propose un projet pour aider les étudiants malentendants à suivre un cursus normalement grâce à un système de réalité augmentée. L'équipe a utilisé des algorithmes de reconnaissances vocales, permis la traduction de textes en langage des signes et utilisé de la reconnaissance faciale pour permettre à l'étudiant de visualiser qui lui parle. Gros coup de cœur pour l'équipe **TFZR** de Serbie qui utilise le Neural Impulse Actuator d'OCZ et son API pour permettre aux personnes paralysées ou ne pouvant parler de renouer avec la communication. Une application WPF basée sur une architecture modulaire « faite maison » leur permet donc d'écrire des sms, d'envoyer des emails, d'utiliser de la synthèse vocale, de naviguer sur internet, et bien d'autres possibilités. Tout cela grâce à un capteur (à la base pour joueurs) branché sur le crâne détectant les « ondes cérébrales ». Le concept est plutôt simple mais l'idée excellente a payé. C'est une solution peu coûteuse pouvant changer la vie de personnes se retrouvant isolées.

OneBeep a mis en place une solution intéressante pour l'éducation dans les pays en voie de développement. En se basant sur le déploiement des OLPC (les PC à 100$), ils se sont questionnés sur la qualité du réseau pour faire parvenir des contenus pédagogiques. Le seul réseau assez répandu est celui de la radio AM/FM. Ils ont donc mis en place une solution de conversion d'un contenu numérique vers un contenu audio. Les OLPC récupèrent l'enregistrement audio et le reconvertissent dans le format d'origine. Côté développement embarqué, **SmarterMe** est un "smarter meter", un compteur intelligent. Il mesure en temps réel la consommation des éléments électriques au sein d'une maison en affichant précisément de quels appareils proviennent les "fuites d'énergie". Le système les reconnaît grâce à leur fréquence sur le réseau électrique et affiche les données adaptées, les graphes de consommations, le nombre d'appareils connectés, etc sur un écran tactile. Autre coup de cœur, le projet **eyeSight** en embarqué a été mené jusqu'en finale de la compétition par une seule personne, Kevin Pfister du Royaume-Uni. Il a voulu aider les personnes mal voyantes à mieux évoluer à l'extérieur. Pour cela, il a modifié une paire de lunettes classiques en ajoutant une webcam et des écouteurs. Le principal système de ce concept se base sur la reconnaissance faciale. Les images des contacts facebook du porteur sont récupérées et, de cette manière, le visage des personnes rencontrées est envoyé, grâce à une application mobile, sur un serveur Azure, puis analysé. Leur nom est ensuite annoncé grâce à l'oreillette. La rapidité d'exécution n'est pas encore idéale mais rares ont été les équipes ayant pensé aux malvoyants. Dans un avenir orienté vers le tout-tactile, des projets d'accessibilité tel que celui là ont de l'avenir.

Les podiums

- **Software Design**
  1re Place : Skeek, *Thaïlande*
  2e Place : TFZR Team, *Serbie*
  3e Place : OneBeep, *Nouvelle Zelande*
- **Embedded Development**
  1re Place : SmarterME, *Taïwan*
  2e Place : MCPU, *Russie*
  3e Place : GERAS, *France*
- **Game Design**
  1re Place : By Implication, *Philippines*
  2e Place : NomNom Productions, *Belgique*
  3e Place : Gears Studio, *France*
- **IT Challenge**
  1re Place : WeiQiu Wen, *Chine*
  2e Place : Miklos Cari Sivila, *Bolivie*
  3e Place : Zhengbin Hu, *Singapour*
- **Digital Media**
  1re Place : Mirror Vita, *Taïwan*
  2e Place : Dreaming Spirits, *Arabie Saoudite*
  3e Place : Woolgathering, *Singapour*

Objectif : New York

L'Imagine Cup est vraiment une expérience formidable. Le niveau en finale mondiale est très élevé et au-delà des compétences techniques, il faut surtout pouvoir rendre son projet esthétiquement agréable et savoir le vendre. Les juges sont friands des histoires autour du projet, l'Imagine Cup est là aussi pour amener de l'espoir, du rêve, grâce aux technologies ! Les catégories IT Challenge et Digital Media, ainsi que dans une moindre mesure Embedded Development et Game Design sont relativement négligées par les étudiants et offrent pourtant d'excellentes possibilités de qualifications avec des thèmes toujours très intéressants. L'embarqué devient d'ailleurs de plus en plus attirant avec du matériel toujours plus puissant et les possibilités offertes par le Cloud. On appréciera l'importance des mentors qui font un excellent travail de coaching et de support pour les équipes. Les deux mentors français ont su s'investir pour leurs équipes, leur faire reprendre leur présentation, leur permettre de décompresser, les accompagner jusqu'à la finale !
Remerciements particuliers à Vanessa Arnauld de Microsoft France et à son équipe qui soutiennent activement les équipes françaises et apportent une dynamique à la compétition nationale.
A l'année prochaine à New York !

■ Jean-Christophe **Vasselon**

Effets sonores en direct avec Silverlight

Silverlight est surtout connu pour la possibilité de créer des interfaces utilisateur attrayantes grâce à la couche WPF incluse dans le moteur. Mais Silverlight, c'est avant tout une mini CLR .NET et la capacité à effectuer des traitements avec toute la performance qu'apporte un code compilé. Depuis quelque temps je travaille sur un projet personnel qui implique beaucoup de génération de son en quasi-temps réel.

L'application en question, bien que réduite en termes de fonctionnalités, devrait vous offrir, lors de son utilisation, quelques moments de détente et/ou de fou rire : le sujet du jour sera la création d'une boîte à effets sonores (écho, réverbération, transposition de la voix).

Quelles options pour générer des sons en Silverlight ?

Quand on envisage d'inclure des effets sonores dans son application Silverlight, la réponse est bien évidemment le contrôle MediaElement. Ce contrôle permet de jouer du contenu vidéo ou audio dans votre application. Son utilisation classique est de renseigner sa propriété Source avec une URL pointant vers le contenu audio ou vidéo :

```
<MediaElement x:Name=»medSound» Source=»/Test.asx»/>
```

Ceci est particulièrement adapté dans deux cas :

- Le contenu est statique et stocké sur le serveur Web
- Le contenu est du « streaming » généré à la volée par le serveur

Dans le cas où l'on souhaite générer à la volée sur le client le son, ce modèle n'est plus adapté. En effet dans notre exemple d'application, il est hors de question d'envoyer des échantillons audio capturés en local pour traitement sur le serveur avant de les rejouer. Il serait préférable d'utiliser la puissance de calcul du client local. C'est exactement ce que nous allons faire. Ceci est possible en utilisant la méthode SetSource de MediaElement. Cette méthode prend en argument un objet de type MediaStreamSource qui fournit toutes les informations nécessaires ainsi que les échantillons pour générer un signal audio.Nous allons donc créer notre propre classe qui héritera de MediaStreamSource. Cette classe doit implémenter plusieurs méthodes dont les plus importantes sont OpenMediaAsync() et GetSampleAsync (`MediaStreamType mediaStreamType`).

La première fournit toutes les informations sur le flux : type de flux (audio ou vidéo), opérations possibles sur le flux (ex : Seek) et caractéristiques du flux (fréquence d'échantillonnage, nombre de bits par échantillon, nombre de canaux audio ...). Nous fournissons ces informations en appelant la méthode :

```
ReportOpenMediaCompleted(IDictionary<MediaSourceAttributesKeys, string>
mediaStreamAttributes, IEnumerable<MediaStreamDescription>
availableMediaStreams)
```

La deuxième méthode quant à elle, sera appelée régulièrement par le contrôle MediaElement pour obtenir les échantillons sonores à reproduire.

Implémentation de MediaStreamSource

La première étape est donc de créer une classe qui dérive de MediaStreamSource :

```
public class AudioGeneratorStreamSource : MediaStreamSource
{
...
}
```

Ensuite, nous implémentons la méthode OpenMediaAsync. Il nous faut tout d'abord créer une collection d'attributs décrivant les caractéristiques de la source :

```
protected override void OpenMediaAsync()
{
 Dictionary<MediaSourceAttributesKeys,string> _sourceAttributes = new
   Dictionary<MediaSourceAttributesKeys,string>();
 _sourceAttributes.Add(MediaSourceAttributesKeys.Duration, «0»);
 _sourceAttributes.Add(MediaSourceAttributesKeys.CanSeek, «false»);
... (...)
}
```

Ensuite il faut également renvoyer une collection de descriptions de flux : ces descripteurs (MediaStreamDescription) précisent le type de flux (audio ou vidéo) ainsi que les informations de décodage (associées ici à l'attribut CodecPrivateData). Il est à noter que ces informations de décodage sont passées sous la forme d'une chaîne de caractères et codées en hexadécimal.

Création de la collection de descripteurs de flux :

```
string _hexWaveFormatExString =
     «0100» + //PCM
     «0200» + // 2 Channel
     «22560000» + // 22050 Samples / sec
     «88580100» + //22050*4 byte rate
     «0200» + //Block align
     «1000» + // 16 Bits per samples
     «0000»;
List<MediaStreamDescription> _streamDescriptions = new
List<MediaStreamDescription>();
Dictionary<MediaStreamAttributeKeys,string> _streamAttributes = new
   Dictionary<MediaStreamAttributeKeys,string>();
_streamAttributes.Add(MediaStreamAttributeKeys.CodecPrivateData,
   _hexWaveFormatExString);
_mediaStreamDescription = new MediaStreamDescription(Media
StreamType.Audio,
   _streamAttributes);
_streamDescriptions.Add(_mediaStreamDescription);
```

On peut noter qu'ici j'ai sélectionné une fréquence d'échantillonnage de 22 050 Hz, des échantillons sur 16 bits et une sortie en stéréo.

Remarque : vous pouvez être amenés à modifier ces paramètres du fait que certaines cartes audio peuvent ne pas supporter tous les modes de reproduction. Pour faire cela, il faut juste comprendre que la valeur en hexadécimal est codée en «little endian». C'est-à-dire l'octet de poids faible en premier. Ex : «88580100» = 0x15888 et «22560000» = 0x5622

Pour finir l'implémentation de la méthode OpenMediaAsync, il reste juste à appeler la méthode ReportOpenMediaCompleted :

```
ReportOpenMediaCompleted(_sourceAttributes, _streamDescriptions);
```

La dernière étape dans la création de notre MediaStreamSource est l'implémentation de la méthode GetSamplesAsync. Dans cette méthode nous devons récupérer les échantillons à reproduire et construire un objet de type MediaStreamSample qui contiendra :

- Un Stream qui contient les échantillons ainsi que l'offset et la longueur des données dans le Stream
- Une description du flux (objet de type MediaStreamDescription que nous avons déjà initialisé dans la méthode OpenMediaAsync)
- Un «timestamp» qui précise le moment auquel les échantillons doivent être lus (il s'agit de la position depuis le début exprimée en centaines de nanosecondes)
- Une collection de MediaSampleAttributeKeys (vide)

Ensuite, il suffit d'appeler ReportGetSampleCompleted en lui passant le MediaStreamSample :

```
protected override void GetSampleAsync(MediaStreamType media
StreamType)
{
 Stream _stmSamples;

   (...)

 Dictionary<MediaSampleAttributeKeys, string> _sampleAttributes = new
   Dictionary<MediaSampleAttributeKeys, string>();
 MediaStreamSample _sample = new
   MediaStreamSample(_mediaStreamDescription,_stmSamples,0,_stm
Samples.Length,
   _currentTimeStamp,_sampleAttributes);
 _currentTimeStamp += _stmSamples.Length * 2500000L / 22050;
 this.ReportGetSampleCompleted(_sample);
}
```

Il ne nous reste plus qu'à récupérer le Stream contenant les échantillons ... Ici nous allons utiliser une classe intermédiaire (Audio16BufferStream) qui hérite de Stream et aura la charge de créer des «paquets» d'échantillons. Elle permet d'écrire les échantillons un par un, mais de les regrouper au final en paquets. Ceci est important, car si nous ne le faisions pas, le nombre d'échantillons fournis au MediaElement lors de l'appel de GetSampleAsync serait trop petit. Cela aurait pour conséquence l'accumulation des échantillons et aucun son en sortie ! Pour créer les «paquets», Audio16BufferStream stocke les échantillons dans des MemoryStream de taille fixe. Une fois un des MemoryStream plein, celui-ci est empilé dans une Queue. En parallèle, Audio16BufferStream expose une méthode permettant de dépiler les MemoryStream. C'est cette méthode qui sera appelée dans GetSampleAsync. Il ne reste donc plus qu'à mettre le code manquant dans GetSampleAsync :

```
protected override void GetSampleAsync(MediaStreamType media
StreamType)
{
 Stream _stmSamples;
 while ((_stmSamples = _bufferStream.GetSamples()) == null)
 {
   Thread.Sleep(5);
 }

 (...)

 this.ReportGetSampleCompleted(_sample);
}
```

NB : *ici _bufferStream est un objet de type Audio16BufferStream qui est initialisé dans le constructeur de notre classe.*

Utilisation du MediaStreamSource

Pour pouvoir utiliser notre MediaStreamSource, il suffit d'en créer une instance, d'appeler la méthode SetSource puis la méthode Play du MediaElement. Bien sûr, si l'on s'arrête là, cela n'a aucun intérêt puisqu'aucun son n'est produit : nous devons maintenant écrire des échantillons dans le Buffer (Audio16BufferStream).

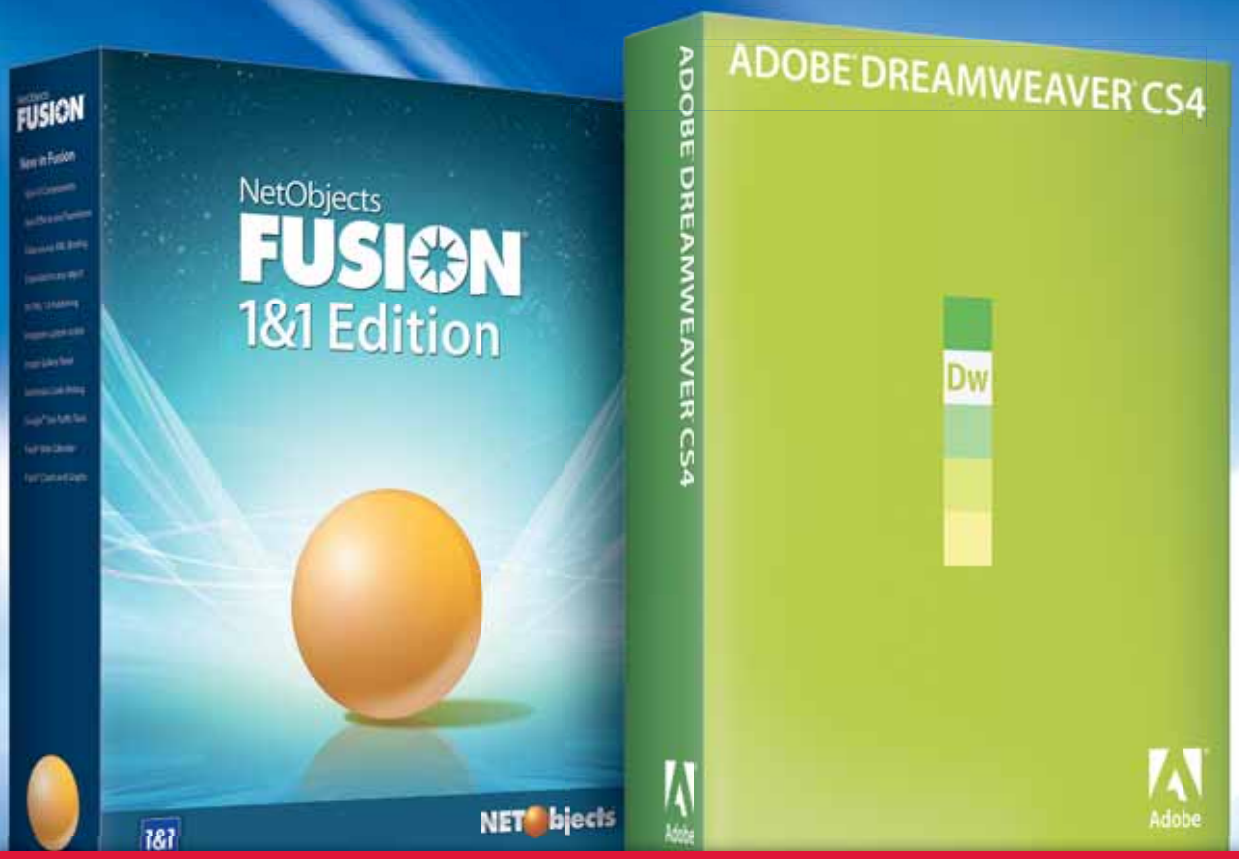
Capture des échantillons en provenance du microphone

Pour pouvoir appliquer des effets sonores, il nous faut donc tout d'abord une source. Nous allons utiliser la possibilité (depuis Silverlight 4) de capturer les échantillons en provenance d'un périphérique audio (microphone). Ceci est possible en implémentant une classe qui dérive de la classe AudioSink. Notre classe implémentera pour l'essentiel la méthode «OnSamples» qui est appelée à chaque fois que des échantillons sont disponibles. Pour rester assez générique, j'ai décidé d'implémenter une classe AudioSink qui écrit dans un Stream (le Stream est passé en paramètre du constructeur). Son implémentation est alors très simple :

```
public class MyAudioSink : AudioSink
{
 public MyAudioSink(Stream outputStream)
 {
   if (outputStream == null)
       throw new ArgumentException(«Output stream must not be null»);
   if (!outputStream.CanWrite)
       throw new ArgumentException(«Output stream must be writeable»);
   OutPut = outputStream;
 }
 public Stream OutPut { get; private set; }

 (...)

 protected override void OnSamples(long sampleTimeInHundred
Nanoseconds, long
     sampleDurationInHundredNanoseconds, byte
[] sampleData)
 {
   OutPut.Write(sampleData, 0, sampleData.Length);
 }
}
```

Les nouveaux packs hébergement 1&1 ré

VOTRE SITE DE



LOGICIELS GRATUITS

INCLUS DANS LES NOUVEAUX PACKS HÉBERGEMENT 1&1*

NetObjects Fusion® 1&1 Edition est un logiciel de création qui vous permet de réaliser votre site ou de l'adapter afin qu'il puisse être consulté depuis n'importe quel appareil mobile. Cette version optimisée de NetObjects Fusion® 11 vous offre des modèles de mise en page et des designs exclusifs spécialement adaptés au Web mobile.

Référence absolue en matière de conception de site Internet, **Adobe® Dreamweaver® CS4** vous apporte des fonctionnalités avancées pour prévisualiser et tester vos pages dans un environnement mobile. Grâce à Device Central, vous adaptez le code HTML et les applications flash de votre site à tous types de terminaux mobiles.

* Logiciels offerts pour toute nouvelle commande d'un pack hébergement (voir pages suivantes) et disponibles en téléchargement à partir de l'Espace Client 1&1.

MEMBRE DE united internet

0970 808 911 Appel non surtaxé

volutionnent votre présence Web

VIENT MOBILE

De plus en plus d'utilisateurs consultent le Web à partir de leurs appareils mobiles, mais de nombreux sites ne sont pas encore adaptés à l'affichage sur écrans de petite taille.

Désormais, nous vous fournissons les logiciels pour convertir votre site au Web mobile.

- ✓ **Votre site compatible avec les navigateurs mobiles**
- ✓ **Interface intuitive et simple d'utilisation**
- ✓ **Modèles de mise en page exclusifs**

1&1

www.1and1.fr

Pour pouvoir utiliser notre AudioSink, il va tout d'abord falloir capturer le périphérique d'entrée audio, le configurer et enfin le «connecter» à notre AudioSink. La première étape nécessite d'accéder aux périphériques audio. Ceci n'est possible que sur une action utilisateur (ex : appui sur un bouton) et nécessite une élévation des privilèges (l'utilisateur devra accorder l'accès). Pour ce faire on appelle la méthode RequestDeviceAccess :

```
private void btnStart_Click(object sender, RoutedEventArgs e)
{
 if (CaptureDeviceConfiguration.RequestDeviceAccess())
 {
 //ici on a accès
 (...)
```

Lors de l'exécution, l'utilisateur pourra accepter ou refuser l'accès : [Fig.1]
Une fois l'accès autorisé, on crée un objet de type CaptureSource et on le configure. CaptureSource capture les périphériques vidéo et audio par défaut. Afin de ne pas capturer la vidéo on mettra à null la propriété VideoCaptureDevice. Par ailleurs, nous devons configurer notre périphérique Audio (nombre d'échantillons par seconde, résolution). Cela se fait en parcourant les différents formats supportés, puis en sélectionnant celui qui convient :

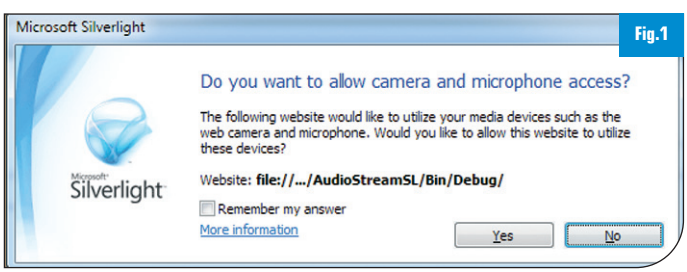
```
_captureSource = new CaptureSource();
_captureSource.AudioCaptureDevice.AudioFrameSize = 50;
_captureSource.VideoCaptureDevice = null;
foreach (AudioFormat _format in _captureSource.AudioCapture
Device.SupportedFormats)
{
 if ((_format.BitsPerSample == 16) &&
   (_format.Channels == 1) && (_format.SamplesPerSecond == 22050))
 {
  _captureSource.AudioCaptureDevice.DesiredFormat = _format;
 }
}
```

Il ne reste plus qu'à connecter notre AudioSink et la CaptureSource. Cela se fait très simplement :

```
_audioSink.CaptureSource = _captureSource;
```

Nous pouvons déjà tester simplement en construisant un AudioSink en lui passant notre buffer précédemment créé. Si tout se passe bien, le son du Micro sera reproduit en accéléré sur la sortie (en effet, on capture en mono et on joue en stéréo). Le code devient donc :

```
Audio16BufferStream _bufferStream = new Audio16BufferStream();
AudioGeneratorStreamSource _streamSource = new
AudioGeneratorStreamSource(_bufferStream);
_audioSink = new MyAudioSink(_bufferStream);
```




```
_audioSink.CaptureSource = _captureSource;
```

Pour démarrer la capture il faut appeler la méthode Start de CaptureSource :

```
_captureSource.Start();
```

Pipeline de traitement audio

Bien qu'il soit possible d'appliquer une multitude de traitements (exemple : filtres passe bande, trémolo, distorsion ...) je me suis limité à 4 effets sonores qui peuvent être appliqués séparément ou simultanément. Ces effets sont les suivants :

- Modulation par une forme d'onde (sinusoïdale) : cet effet, quand il est correctement réglé, permet de truquer la voix.
- Réverbération : le principe est de réinjecter des échantillons avec un décalage (typiquement le décalage est de quelques centaines de millisecondes).
- Écho : le principe est le même que pour la réverbération mais le décalage est plus important (ici on peut aller jusqu'à une ou deux secondes).
- Balance stéréo modulée par forme d'onde. Celui-ci est très agaçant : il fait passer progressivement le son de droite à gauche puis de gauche à droite.

Le principe mis en œuvre est un traitement en «pipeline» : chaque étape de traitement reçoit des échantillons de l'étape précédente, les traite, puis les passe à l'étape suivante.
Concrètement, les différents traitements sont implémentés sous la forme de Stream. Ainsi le Sink Audio écrit dans un Stream, chargé d'appliquer une modulation. Ce dernier écrit dans un Stream appliquant une réverbération et ainsi de suite jusqu'au BufferStream.

NB : *Les différents Stream utilisés à chaque étape n'ont pas besoin d'implémenter l'ensemble des fonctionnalités de la classe Stream. En effet, ils ne sont utilisés qu'en écriture et n'ont pas de nécessité d'être «Seekable».*

Implémentation des différents Stream

Tout d'abord, lors des traitements, nous avons fréquemment besoin de lire des données sur un buffer cyclique (exemple : on doit moduler avec une forme d'onde stockée dans un tableau : une fois en bout de tableau il faut revenir au début. De même, nous avons besoin de garder un historique des échantillons pour les traitements de réverbération et d'écho : il est plus simple de les stocker dans un Buffer «tournant». Cette fonctionnalité est fournie par un Stream spécial : le CyclingStream. Il prend en paramètre de son constructeur un Stream et ensuite il effectue les opérations de lecture en prenant soin de «rembobiner» le Stream interne quand il est arrivé au bout :

```
public class CyclingStream : Stream
{
 public Stream BaseStream { get; set; }


 public CyclingStream(Stream _stream)
 {
   BaseStream = _stream;
   if (!BaseStream.CanSeek)
         throw new ArgumentException(«The stream must be seekable»);
 }

 (...)
```

CHOISISSEZ VOTRE PACK HÉBERGEMENT

À MOITIÉ PRIX

Non seulement 1&1 vous fournit gratuitement les meilleurs logiciels pour rendre votre site mobile, mais en plus vous bénéficiez de -50 % pendant 6 mois sur nos nouveaux packs d'hébergement !

6 MOIS À -50 % LOGICIEL OFFERT !*

1&1 PACK CONFORT

- 2 domaines au choix INCLUS
- NOUVEAU : 100 Go d'espace disque
- Trafic ILLIMITÉ
- 100 comptes email
- 5 bases de données MySQL (100 Mo)
- Outils de création de site : éditeurs Web, blog, album photo, e-Boutique Start
- PHP5, PHP6 (bêta), Perl, Python, Ruby, C, tâches cron
- Outils de communication : formulaire, chat
- 1&1 Référencement
- 1&1 WebStat
- NOUVEAU : Logiciel offert* NetObjects Fusion® 1&1 Edition

~~4,99 €~~ HT/mois (5,97 € TTC/mois)

2,49 € HT/mois (2,98 € TTC/mois)*

1&1 PACK PRO

- 3 domaines au choix INCLUS
- NOUVEAU : 250 Go d'espace disque
- Trafic ILLIMITÉ
- 500 comptes email
- 20 bases de données MySQL (100 Mo)
- Outils de création de site : éditeurs Web, blog, album photo, e-Boutique Start
- PHP5, PHP6 (bêta), Perl, Python, Ruby, C, tâches cron
- Outils de communication : formulaire, chat, newsletter, RSS, listes de discussion
- 1&1 Référencement
- 1&1 WebStat
- Google Adwords® : 50 € offerts
- NOUVEAU : Logiciel offert* NetObjects Fusion® 1&1 Edition ou Adobe® Dreamweaver® CS4 (au choix)

~~9,99 €~~ HT/mois (11,95 € TTC/mois)

4,99 € HT/mois (5,97 € TTC/mois)*

1&1 PACK PREMIUM

- 4 domaines au choix INCLUS
- NOUVEAU : 500 Go d'espace disque
- Trafic ILLIMITÉ
- 1000 comptes email
- 50 bases de données MySQL (100 Mo)
- Outils de création de site : éditeurs Web, blog, album photo, e-Boutique Start
- PHP5, PHP6 (bêta), Perl, Python, Ruby, C, tâches cron
- Outils de communication : formulaire, chat, newsletter, RSS, listes de discussion
- 1&1 Référencement
- 1&1 WebStat
- Google Adwords® : 75 € offerts
- Certificat SSL dédié INCLUS
- NOUVEAU : Logiciel offert* NetObjects Fusion® 1&1 Edition ou Adobe® Dreamweaver® CS4 (au choix)

~~19,99 €~~ HT/mois (23,91 € TTC/mois)

9,99 € HT/mois (11,95 € TTC/mois)*

Votre domaine à prix sensationnel :

le .fr à 4,99 € HT/an (5,97 € TTC/an),

le .eu à 0,99 € HT/an (1,18 € TTC/an)* !

* Offre « 6 mois à -50 % » soumise à un engagement de 12 mois. Frais de mise en service : 5,97 € TTC (Pack Confort) ou 11,95 € TTC (Pack Pro, Pack Premium). A l'issue des 6 premiers mois, les produits concernés sont aux prix habituels (Pack Confort à partir de 5,97 € TTC/mois, Pack Pro à partir de 11,95 € TTC/mois, Pack Premium à partir de 23,91 € TTC/mois). Logiciel offert pour toute nouvelle commande d'un pack hébergement et disponible en téléchargement à partir de l'Espace Client 1&1. Offre domaine applicable la première année uniquement au lieu du prix habituel de 6,99 € HT/an (8,36 € TTC). Conditions détaillées sur www.1and1.fr. Offres sans engagement également disponibles.

MEMBRE DE united internet

Appel non surtaxé **0970 808 911**

www.1and1.fr

1&1

```
 public override int Read(byte[] buffer, int offset, int count)
 {
   lock (this)
   {
     int _bytesRemaining = count;
     while (_bytesRemaining != 0)
     {
       int _totalRead = BaseStream.Read(buffer, offset, _bytes
Remaining);
       offset += _totalRead;
       _bytesRemaining -= _totalRead;
       if (_bytesRemaining != 0) BaseStream.Seek(0, SeekOrigin.
Begin);
     }
     return count;
   }
 }

(...)

}
```

Les Streams effectuant une composition

Tous les traitements décrits (modulation, réverbération…) sont en fait des opérateurs composant deux sources de données. Par exemple la modulation reçoit, en première source de données les échantillons sonores et en deuxième des échantillons décrivant une forme d'onde. Ils sont donc tous dérivés d'une classe de base qui va gérer l'initialisation, la création d'un BinaryWriter pour écrire sur la sortie et d'un BinaryReader pour lire le second flux. Cette classe exposera également certaines propriétés telles que IsEnabled qui gère l'activation / désactivation du traitement :

```
public abstract class Audio16CompositionStream : Stream
{
 protected BinaryWriter _writer;
 protected BinaryReader _secondReader;

 public Audio16CompositionStream(Stream secondaryInput,  Stream
outputStream)
 {
   if (secondaryInput == null)
     throw new ArgumentException(«Secondary input stream must
not be null»);
   if (outputStream == null)
     throw new ArgumentException(«Output stream must not be
null»);
   if (!secondaryInput.CanRead)
     throw new ArgumentException(«Input stream must be rea-
dable»);
   if (!outputStream.CanWrite)
     throw new ArgumentException(«Output stream must be wri-
teable»);

   OutPut = outputStream;
   SecondInput = secondaryInput;
   _writer = new BinaryWriter(OutPut);
   _secondReader = new BinaryReader(SecondInput);
 }

 public Stream SecondInput { get; protected set; }
 public Stream OutPut { get; protected set; }
 public bool IsEnabled { get; set; }

(...)
}
```

À partir de cette classe, dérivent 3 classes :

- Audio16MixStream
- Audio16ModulationStream
- Audio16StereoBalancerStream (en fait cette classe dérive de Audio16ModulationStream)

Audio16MixStream permet de mélanger deux flux : c'est cette classe qui est utilisée pour la réverbération et l'écho. Elle prend en paramètre du constructeur (également exposé en tant que propriété) un pourcentage entre 0 et 100 :

```
public Audio16MixStream(Stream secondaryInput, int percentage,
Stream outputStream) : base(secondaryInput,outputStream)
{
    this.Percentage = percentage;
}


public int Percentage { get { return _percentage; } set
(...)
```

Ensuite, son implémentation est relativement simple et réside entièrement dans la méthode Write. On distingue 3 cas :

- Si IsEnabled est false ou le pourcentage est à zéro, on n'effectue aucun traitement particulier (on passe les échantillons à l'étape de traitement suivante).
- Si le pourcentage est de 100, on ignore les échantillons en entrée (on passe ceux provenant de la seconde source).
- Dans le cas «normal» on mélange les deux sources.

```
public override void Write(byte[] buffer, int offset, int count)
{
 lock (this)
 {
   if ((Percentage == 0)||!IsEnabled)
   {
     OutPut.Write(buffer, offset, count);
     return;
   }
   if (Percentage == 100)
   {
     byte[] _secondBuffer = new byte[buffer.Length];
     SecondInput.Read(_secondBuffer, 0, (count - offset));
     OutPut.Write(_secondBuffer, 0, _secondBuffer.Length);
     return;
   }
   MemoryStream _buffer = new MemoryStream(buffer, offset, count);
   BinaryReader _reader = new BinaryReader(_buffer);
   int _totalSamples = count / 2;
```

```
    while (_totalSamples != 0)
    {
      Int16 _sample = (Int16)((_reader.ReadInt16() * _invertPercentage +
        _secondReader.ReadInt16() * _percentage) / 100);
      _writer.Write(_sample);
      _totalSamples--;
    }
  }
}
```

Pour la modulation (Audio16ModulationStream), il y a pour l'essentiel trois méthodes implémentées. Le constructeur reçoit deux paramètres supplémentaires : le niveau et la vitesse de modulation (également exposés en tant que propriétés) :

```
public Audio16ModulationStream(Stream secondaryInput, int modula
tionLevel, Stream outputStream, int modulationSpeed) : base
(secondaryInput,outputStream)
```

La méthode principale est toujours la méthode Write : elle va pour l'essentiel implémenter la vitesse de modulation en déterminant :

- Pour une vitesse positive, le nombre d'échantillons de la source secondaire à ignorer (sauter).
- Pour une vitesse négative, le nombre d'échantillons en entrée à lire avant de charger un échantillon de la source secondaire.

```
public override void Write(byte[] buffer, int offset, int count)
{
  MemoryStream _buffer = new MemoryStream(buffer, offset, count);
  BinaryReader _reader = new BinaryReader(_buffer);
  int _totalSamples = count / 2;
  lock (this)
  {
    while (_totalSamples != 0)
    {
      if (_sampleToSkip == 0)
      {
        _sampleToSkip = ModulationSpeed;
        _lastModulationSample = _secondReader.ReadInt16();
      }
      else
        if (_sampleToSkip > 0)
        {
          while (_sampleToSkip != 0)
          {
            _sampleToSkip--;
            _lastModulationSample = _secondReader.ReadInt16();
          }
          _sampleToSkip = ModulationSpeed;
        }
        else
        {
          _sampleToSkip++;
        }
      Int16 _sample = _reader.ReadInt16();
      ProcessSample(_sample);
      _totalSamples--;
    }
  }
}
```

Enfin le «véritable» traitement est effectué dans la méthode ProcessSamples :

```
protected virtual void ProcessSample(Int16 sample)
{
  if (IsEnabled)
  {
    sample = (Int16)((sample * _lastModulationSample) / Modulation
Level);
  }
  _writer.Write(sample);
}
```

Le troisième Stream (Audio16StereoBalancerStream) héritant de Audio16ModulationStream ne fait que surcharger la méthode ProcessSample de manière à générer deux échantillons (gauche et droite) à partir d'un seul en le répartissant en fonction de la valeur de la modulation :

```
protected override void ProcessSample(Int16 sample)
{
  Int16 leftSample = sample;
  Int16 rightSample = sample;
  if (IsEnabled)
  {
    leftSample = (Int16)((sample * _lastModulationSample) / Modulation
Level);
    rightSample = (Int16)((sample * (ModulationLevel - _lastModulationSample)) /
      ModulationLevel);
  }
  _writer.Write(leftSample);
  _writer.Write(rightSample);
}
```

Le «BackupStream»

Il s'agit en fait d'un Stream très simple dont la fonction est de faire une copie des échantillons en entrée dans un flux secondaire (en fait il sert à historiser dans le buffer utilisé pour l'écho et la réverbération). L'implémentation est faite, comme à l'habitude, dans la méthode Write :

```
public override void Write(byte[] buffer, int offset, int count)
{
    _MainStream.Write(buffer, offset, count);
    _BackupStream.Write(buffer, offset, count);
}
```

Assemblage du pipeline

Nous avons maintenant toutes les briques nécessaires pour générer des effets sonores. Il reste à les assembler pour construire notre pipeline de traitement. Nous avons d'abord besoin de plusieurs CyclingStream. En effet il nous faut :

- Deux Stream pointant vers le buffer echo/réverb pour accéder aux échantillons précédents (un pour la réverbération, un pour l'écho).
- Un Stream pointant vers le buffer pour sauvegarder les échantillons (à l'aide du BackupStream).
- Deux Stream pointant vers une forme d'onde sinusoïdale (un pour la modulation, un pour le balancer stéréo).

Ces Streams sont créés dans le constructeur d'une classe (AudioEffectGenerator) qui va nous permettre de contrôler les différents paramètres des effets :

```
public AudioEffectGenerator()
{
  _rawBuffer = new byte[BUFFER_SIZE]; // init Echo / Reverb Buffer
  Array.Clear(_rawBuffer, 0, BUFFER_SIZE);

  InitParameters();
  _modulationLevel = 50;

  _storeStream = new CyclingStream(new MemoryStream(_rawBuffer));
  _EchoStream =  new CyclingStream(new MemoryStream(_rawBuffer));
  _ReverberationStream =  new CyclingStream(new MemoryStream(_rawBuffer));
  _ModulationStream = new CyclingStream(new
MemoryStream(BuildModulationWaveTable(ModulationType.Sinus,
_modulationLevel)));
  _BalancerStream = new CyclingStream(new
MemoryStream(BuildModulationWaveTable(ModulationType.Sinus,50)));

  (...)

}
```

L'assemblage du pipeline est ensuite très simple, il suffit de passer les bons paramètres aux différents constructeurs.
Petit détail, on commence la composition depuis la sortie du pipeline en remontant vers l'entrée :

```
_OutputStream = new Audio16BufferStream(); //Output Buffer
_balancerMixer  =  new  Audio16StereoBalancerStream(_Balancer
Stream, 100, _OutputStream,     -275); //Stereo Balancer
_IntermediaryOutputStream  =  new  BackupStream(_balancerMixer,
 _storeStream);
   // Backup for reverb & echo
_EchoMixer = new Audio16MixStream(_EchoStream, 30, _Intermediary
OutputStream); //Echo

_ReverberationMixer = new Audio16MixStream(_ReverberationStream,
 30, _EchoMixer);   //Reverb
_ModulationMixer  =  new  Audio16ModulationStream(_Modulation
Stream,   100,_ReverberationMixer , 0); //Modulation
```

Notre entrée sera ici le Stream de modulation et la sortie, le Buffer. C'est d'ailleurs ce qui est exposé par les propriétés Input et Ouput de la classe AudioEffectGenerator :

```
public Stream Input { get { return this._ModulationMixer; } }
public Audio16BufferStream Output { get { return this._Output
Stream; } }
```

Il est alors possible de «câbler» notre pipeline à notre AudioSink et MediaStreamSource en lieu et place du BufferStream que nous avions mis au début :

```
_effectGenerator = new AudioEffectGenerator();
AudioGeneratorStreamSource _streamSource = new
   AudioGeneratorStreamSource(_effectGenerator.Output);
_audioSink = new MyAudioSink(_effectGenerator.Input);
```

Arrivé à ce point, aucun effet n'est actif ou correctement paramétré, mais le son du microphone est reproduit sur la sortie.

Contrôle des effets

Pour pouvoir contrôler les effets sonores, la classe AudioEffectGenerator expose des propriétés qui peuvent directement être «bindées» depuis un fichier XAML : une sorte de viewmodel ou tout au moins une façade «bindable». Il aurait également été possible de faire de l'assemblage dynamique en laissant l'utilisateur choisir ses composants, mais la complexité aurait été tout autre et le binding plus difficile. Les propriétés exposées sont les suivantes :

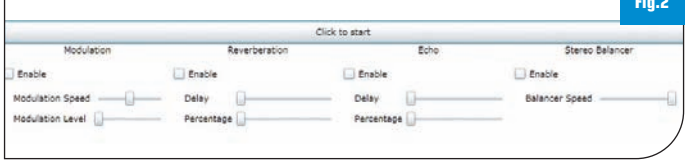
```
public bool IsEchoEnabled
public bool IsReverberationEnabled
public bool IsModulationEnabled
public bool IsBalancerEnabled
public int EchoDelay
public int EchoPercentage
public int ReverberationDelay
public int ReverberationPercentage
public int ModulationSpeed
public int ModulationLevel
public int BalancerSpeed
```

Bien évidemment, la modification d'une de ces propriétés impacte les propriétés des Stream associés. Pour pouvoir contrôler ces paramètres j'ai créé un petit contrôle Silverlight très simple : EffectBox. Tout y est défini directement dans le XAML.
Il est composé d'une grille avec 4 colonnes, chaque colonne correspondant à un des traitements (Modulation, Reverbération, Echo, Stéréo Balancer). Chaque traitement est représenté par un StackPanel vertical qui contient une CheckBox pour activer / désactiver l'effet et un (ou des) Sliders pour contrôler les paramètres. Il n'y a plus qu'à déclarer le contrôle dans la page principale et à affecter son datacontext :

```
<Grid x:Name=»LayoutRoot»>
  <Grid.RowDefinitions>
    <RowDefinition Height=»30»/>
    <RowDefinition/>
  </Grid.RowDefinitions>
    <MediaElement x:Name=»medSound»/>
  <Button x:Name=»btnStart» Content=»Click to start» Click=»btn
Start_Click»/>
  <loc:EffectBox x:Name=»ctlEffect» Grid.Row=»1»/>
</Grid>
```

Au final, on obtient cela : [Fig.2]

Fig.2



N'hésitez pas à tester, à expérimenter de nouveaux effets, tout le code source est sur codeplex (http://echobox.codeplex.com) et si vous créez un effet sympa vous pouvez toujours me l'envoyer. :-)

■ Roch **Baduel**
MVP BizTalk - Directeur du pôle BizTalk – MCNEXT

MODEL2CODE

Agile Development in action!

CERTAINS AGL VOUS PROMETTENT DE DÉVELOPPER 10 FOIS PLUS VITE?

ET SI VOUS VOUS LAISSIEZ... DIRIGER PAR LES MODÈLES ?

NE DÉVELOPPEZ PLUS, MODÉLISEZ !

Passez à la vitesse du Model Driven!

Maquettez vos IHMs en HTML ou MXML et modélisez simplement vos processus métier sous forme de diagrammes UML, nos générateurs les transforment instantanément en application Spring, Java EE, Flex ou Improve Foundations, prête à être déployée !

Nouveau ! Plugin CRUD Booster

Créez en quelques minutes vos prototypes et accélérez considérablement vos projets applicatifs. Générez une application CRUD fonctionnelle (y compris les IHMs) à partir d'un simple diagramme de classe !

VERSIONS D'ÉVALUATION GRATUITES & TUTORIAUX :

WWW.MODEL2CODE.COM

Model2Code propose des ateliers agiles et collaboratifs de génération d'applications web et riches, combinant les meilleures technologies Model Driven du marché : l'outil de modélisation UML Magicdraw® associé au moteur de génération par transformation de modèles BLU AGE®.

magicdraw™ BLU AGE AGILE MODEL TRANSFORMATION

BLU AGE Software
Immeuble le Gabriel Voisin - 79, rue Jean-Jacques Rousseau 92158 Suresnes cedex FRANCE.
Tel +33 1 56 05 60 91 Fax +33 1 56 05 88 01

www.model2code.com

Quel futur pour les bases de données ?

Le marché des bases de données, après une période de très forte activité, ralentit depuis quelques mois, du moins en termes d'annonces, de nouveautés. Or, les versions les plus récentes dévoilent des axes techniques très intéressants et quelques pistes pour les 12, 18 et 24 mois à venir sont déjà là. Encore faut-il les voir car elles sont parfois très bien cachées.

Les bases de données évoluent constamment. Aujourd'hui, il pourrait être plus « sexy » de parler Business Intelligence que du cœur produit en lui-même. Car finalement, la BI s'appuie sur la donnée, donc les bases de données. Les pistes d'améliorations, d'évolutions sont nombreuses : partionnement, sécurité, gestion des données chaudes et froides, support optimal de XML, données géospatiales, etc.

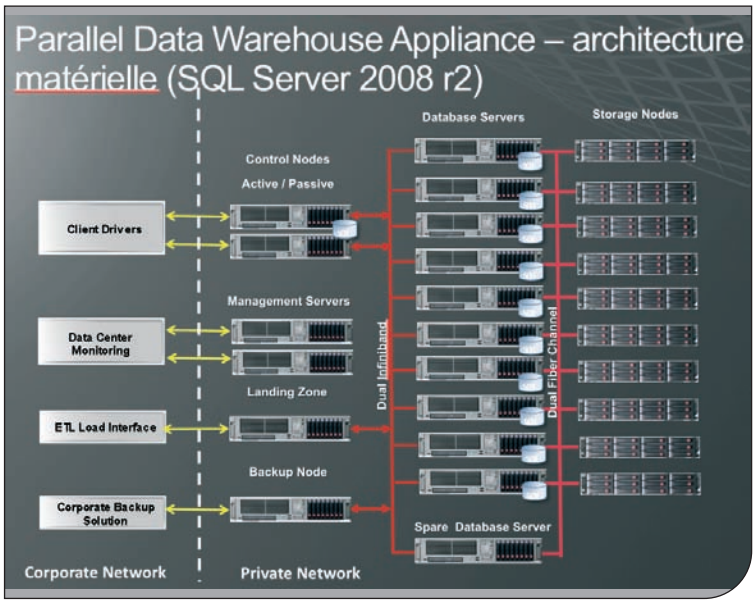
Les bases de données dites « en mémoire » (In-memory)

Les SGBD en mémoire ne constituent pas une réelle nouveauté car ce principe de fonctionnement est connu depuis longtemps, mais l'idée revient à la mode, notamment avec Sybase ASE 15. La base de données en mémoire est un principe très simple : la base est chargée en mémoire vive ainsi que les données adéquates, par contre, le serveur SGBD reste sur un support physique. Et beaucoup de fonctions s'exécutent directement en mémoire, supprimant ainsi les temps de latences dus aux Entrées / Sorties du matériel physique et évitant les temps d'accès des disques durs. Comme l'indique Sybase, ces bases sont avant tout destinées aux applications et données hautement transactionnelles et critiques, nécessitant des performances optimales ou l'obligation de traiter d'énormes masses documentaires.

Cependant, plusieurs limites s'imposent aux bases in-memory. Tout d'abord, il faut tenir compte du temps pour monter et démarrer services et données dans la mémoire. Par contre, il est difficile de pouvoir répliquer en temps réel la base in-memory sans dégrader les performances et si la mémoire se corrompt ou tombe, on perd tout. De la qualité de mémoire installée peut dépendre la fiabilité du in-memory. Et on peut créer, par code la mise en mémoire. Exemple dans ASE 15.5 d'une création de deux « zones » (ou devices) dans le imdb_cache :

```
1> disk init name = 'imdb_datadev1', physname = 'imdb_cache',
2> size = '250M', type = 'inmemory
3> go
```

```
1> disk init name = 'imdb_logdev1', physname = 'imdb_cache',
2> size = '50M', type = 'inmemory
3> go
```



Cet exemple montre un in-memory depuis le cache. Si on montait une base directement en in-memory, on aurait plutôt :

```
1> create inmemory database imdb
2> on imdb_datadev1 = '250M'
3> log on imdb_logdev1 = '50M'
4> with durability = no_recovery
5> go
```

Et il est possible de créer des bases temporaires en mémoire, par exemple, pour traiter un pic ponctuel de statistiques ou des données spatiales. Ce marché des SGBD en mémoire est assez large aujourd'hui : Sybase, IBM (SolidDB), Oracle. MySQL possède une fonction très limitée en mémoire aussi.

Le in-memory n'est pas l'approche choisie par d'autres acteurs du marché comme InterSystems pour sa base Caché. L'éditeur essaye de trouver une approche hybride qui ne soit ni du tout in-memory ni du tout disque dur, en utilisant au mieux le cache. Cela permet de bonnes performances, une sécurité des données et de la disponibilité que l'on ne peut garantir en in-memory. De même, l'éditeur ne croit pas réellement à l'usage du SSD…

Le cloud computing, ou l'avenir prédit

Ira, ira pas ? Aujourd'hui, tous les éditeurs de SGBD se posent la question : faut-il aller sur le cloud. Même si le marché reste très minoritaire, le cloud devient un axe stratégique pour les fournisseurs de bases de données. Microsoft a engagé fortement la démarche avec SQL Azure, moteur relationnel, qui se veut le pendant cloud computing de SQL Server 2008. Cependant, ces portages comportent plusieurs problèmes : quels niveaux fonctionnels avoir ? Tout va dépendre de la couche cloud que l'éditeur va cibler. Sur une plate-forme PaaS, un SGBD cloud sera privé des mécanismes de trop bas niveaux, ceux qui s'appuient et utilisent des mécanismes systèmes et matériels, le PaaS masque la couche

Advanced database technology for breakthrough applications

Laissez vos applications s'envoler...

Avec InterSystems Caché, faites décoller vos applications. Elles bénéficieront immédiatement de performances hors du commun, deviendront massivement scalable et ne nécessiteront plus d'administration fastidieuse.
InterSystems **Caché**® base de données post-relationnelle, mais aussi serveur d'application, framework Ajax, ... se fonde sur une technologie Objet avancée qui permet de construire beaucoup plus facilement des applications XML, Web Services, AJAX, Java et .NET.
InterSystems Caché est aussi une base SQL jusqu'à 5 fois plus rapide que les bases de données relationnelles classiques en accès SQL et bien plus en accès Objet !
Grâce à son Architecture de Données Unifiée unique, Caché élimine le besoin de mapping objet-relationnel, réduit les temps de développement, et facilite l'évolution et la maintenance de votre application.
Déployé sur plus de 100.000 systèmes de par le monde pour des applications de 2 à 50.000 utilisateurs, Caché est disponible sur toutes les plates-formes majeures du marché.

Nouveau: L'intégration de Caché avec JAVA – JNI permet littéralement d'exploser les performances et d'offrir enfin aux programmeurs JAVA un moteur sans compromis et digne de leurs réalisations.
Depuis plus de 30 ans InterSystems vous apporte des technologies avancées qui vous permettent de construire des applications qui font la différence.

InterSystems

Téléchargez votre version gratuite complète InterSystems Caché - sans limite de temps: InterSystems.fr/avancee

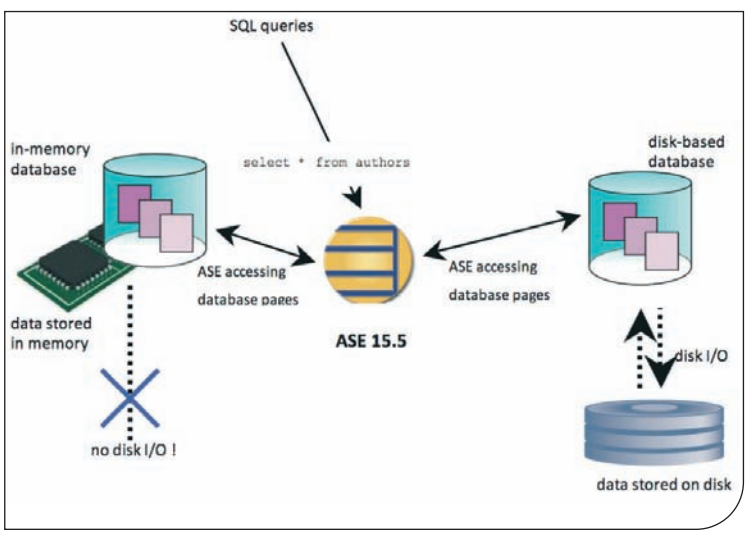
infrastructure. Par exemple, T-SQL n'est pas totalement supporté, la commande Use non plus, ainsi que le mirroring de base de données, les options serveur, la couche SSIS absente. Ces lacunes sont normales du fait de l'absence d'interaction directe avec le matériel. Par contre, SQL Azure va au fur et à mesure compléter ses fonctions logiques.

D'une manière générale, deux catégorie de base de données, ou plutôt de moteur de base de données sont disponibles : relationnel et non relationnel. Dans le non relationnel, on retrouve le stockage de Google App Engine, Azure Storage. Les deux s'articulent principalement autour du blob, avec des capacités de plusieurs Go. Nous sommes là dans le monde en dehors du relationnel qui paraît trop limité pour le cloud. C'est pour cela que l'on retrouve des bases techniques comme NoSQL qui privilégie le transactionnel, les partionnements massifs, la disponibilité. Et surtout on ne retrouve plus la même structure et organisation des données que dans un SGBDR classique. Pour les promoteurs de NoSQL, cette approche permet une montée en charge de la base, une réplication plus rapide et plus simple.

MySQL : le risque des forks

On pourrait titrer « rien ne va plus dans le monde MySQL », « panique à bord ». Sans aller aussi loin, il faut avouer que la situation de MySQL, racheté par Sun, lui-même racheté par Oracle il y a un an, va de mal en pis. On se perd dans la stratégie, parfois peu claire, d'Oracle sur la partie open source de Sun. Déjà tancé par les anciens de MySQL, et notamment par son créateur qui a lancé MariaDB, ou encore les pétitions pour sauver MySQL, la situation apparaît très confuse d'autant qu'il semble bien se confirmer une plus grande prudence des entreprises envers MySQL. C'est dans ce contexte délicat que début juillet, une nouvelle structure a été dévoilée : SkySQL. Et c'est Monty (le fondateur de MySQL) lui-même qui l'a annoncé. L'objectif de SkySQL est de construire et de proposer des services, du support autour de MySQL, en concurrence directe avec les offres Oracle. Oracle tente depuis 9 mois de rassurer la communauté et les utilisateurs de la base de données. Malgré les efforts d'explication, rien n'est fait.

Pourtant, Oracle cherche à rassurer sur l'état du dauphin, notamment depuis la conférence MySQL (avril 2010) et la présentation de Edward Screven (Oracle). Edward a passé en revue l'engagement d'Oracle dans le monde Linux, open source : support mondial, édition communautaire (gratuit et en licence gpl).



Cette période d'incertitude peut profiter aux SGBD concurrents tels que Ingres ou PostgreSQL. Le premier vante sa technologie tandis que le second peut s'appuyer sur sa robustesse d'exécution, reconnue.

Au-delà de SQL ?

Une des questions que l'on se pose régulièrement concerne l'avenir de SQL. En a-t-il un ? Les avis diffèrent : oui pour certains et malheureusement non pour d'autres. L'héritage et le patrimoine SQL sont tels qu'il est impossible d'envisager la mort de SQL avant une dizaine d'années. Le requêtage « natif » aux langages est sans doute l'évolution normale et nécessaire, à l'instar d'un Linq (Microsoft). Mais ces alternatives tardent à sortir et à s'étoffer. Pour passer au-delà de SQL, une approche homogène par rapport aux modèles de développement est nécessaire avec le risque, réel, de voir un requêtage natif, par langage ou SGBD. Cela permettrait aussi de redonner une place plus importance aux développeurs par rapport aux DBA. Car cet éloignement du développeur de la base, de la donnée demeure un problème.

Une autre question survient immanquablement : comment analyser, traiter des masses d'informations toujours plus importantes ? La parallélisation apparaît aujourd'hui la solution idéale. Encore faut-il que les éditeurs de SGBD fournissent un modèle de développement ayant un bon niveau d'abstraction afin d'éviter une programmation trop complexe. Mais là aussi se pose la question de l'abstraction du modèle par rapport aux langages et aux SGBD. Plusieurs projets existent avec plus ou moins de parallélisation : VectorWise (Ingres), Ct (Intel), PLinq (Microsoft).

VectorWise utilise deux fondamentaux : la vectorisation et les colonnes orientées pour le traitement des données. Ces deux techniques doivent permettre un traitement de données plus rapide pour un moindre coût (en ressources, puissance et finances). Cette approche est particulièrement intéressante pour le stockage et l'exécution des requêtes. Pour pouvoir réaliser un tel changement, Ingres a revu en profondeur l'architecture du moteur de stockage en adoptant une nouvelle architecture de traitement des requêtes pour pouvoir tirer parti aussi bien des grandes quantités de mémoires vives et de processeurs dernières générations.

IBM prend une autre voie avec la technologie Apache Hadoop. Hadoop est une technologie pour utiliser une grosse volumétrie de donnée en mode distribué. Elle peut supporter plusieurs petas de données et des centaines de nœuds serveurs. La combinaison DB2–Hadoop permet de distribuer les données. La technologie n'est pas encore disponible mais l'idée est plaisante.

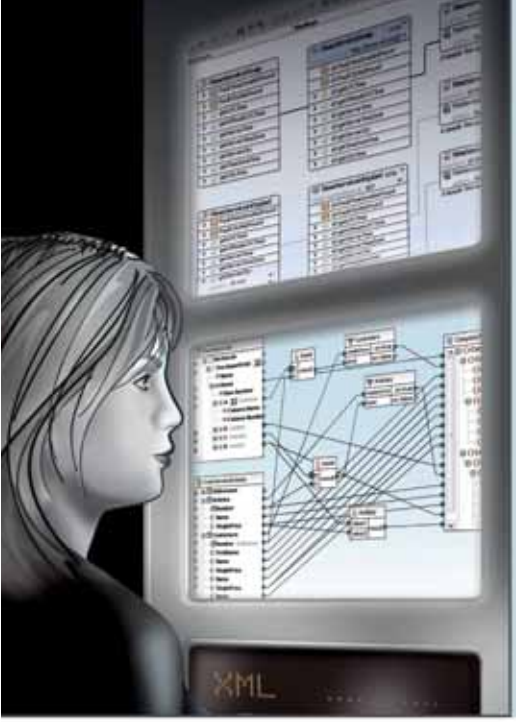
En complément à Hadoop, IBM a sorti DB2 pureScale il y a 10 mois. Cette technique vise à assurer une haute disponibilité sans coupure et une montée en charge illimitée.

Cette technologie tire parti du matériel tout en fournissant les mécanismes logiciels nécessaires. Il suffit alors de rajouter des serveurs pour accroître les capacités de DB2.

Cette technologie concerne les volumes critiques et les entreprises ayant des besoins conséquents, et elle est un concurrent à la solution matérielle – logicielle Oracle Exadata.

Merci à Eric Soares (directeur général Ingres France), Francis Arnaudies (spécialiste IT / SGBD IBM France), Lionel Billon (Microsoft France), Olivier Caudron (senior sales engineer, InterSystems), François Guérin (responsable avant vente, Sybase).

■ François **Tonic**



Alliez des technologies patrimoniales abordables à la série complète d'outils d'intégration de données d'Altova®

Voyez comment le MissionKit® d'Altova, la suite intégrée d'outils XML, de mappage de données et de bases de données vous permet d'exploiter pleinement les technologies existantes et vos investissements en logiciels professionnels tout en intégrant des technologies modernes, et ce sans vous ruiner.

Nouveautés dans la version 2010:

- Version 64 bits
- Prise en charge de l'édition et du mappage WSDL 2.0
- Nombreuses améliorations dans le mappage de données
- Optimisations de la création, du mappage et de la transformation des données XBRL
- Edition et conversion JSON
- Prise en charge de IBM® iSeries® 6.1
- Intégration de Visual Studio® 2010

Le MissionKit d'Altova comprend plusieurs outils intelligents pour l'intégration de données :

MapForce® – Outil graphique de mappage, de transformation & de conversion de données

- Conversion de données par glisser-déposer avec transformation instantanée et gén. de code
- Prise en charge du mappage XML, DBs, EDI, Excel® 2007+, XBRL, fichiers texte, services Web

XMLSpy® – Outil d'édition XML et de services Web

- Editeur XML avec forte intégration de bases de données
- Outil de services Web, convertisseur JSON <> XML

DatabaseSpy® – Outil de requête, de conception et de comparaison multi-bases de données

- Prise en charge de toutes les principales bases de données relationnelles et de la traduction ente les différents types de BD
- Editeur SQL, concepteur graphique de bases de données et éditeur de contenu

Téléchargement gratuit!

Testez avant d'acheter avec une version gratuite et entièrement fonctionnelle de 30 jours disponible sur **www.altova.com**.

Dreamweaver CS5 : le couteau suisse du web !

La Creative Suite 5 Web Premium est une solution de création de contenu pour les standards du web et pour la Flash Platform. Parmi les onze outils desktop que compte la suite, trois vont particulièrement retenir notre attention. D'une part Dreamweaver, pour la partie HTML et standards du web, et d'autre part Flash Builder et Flash Catalyst, pour la partie Flash Platform.

Dreamweaver CS5 a pour objectif de s'adapter à l'évolution des pratiques du web à travers une meilleure prise en charge des sites dynamiques, et plus particulièrement des Content Management Systems comme Wordpress, Drupal, ou encore Joomla. Ces CMS, qui présentent de nombreux avantages en termes de gestion de contenu, ont l'inconvénient de compliquer le processus d'édition des pages web du site par les éditeurs HTML. En effet, ces derniers ont typiquement pour objectif d'éditer des fichiers HTML et CSS alors que, dans le cas d'un CMS, ces fichiers HTML n'existent pas à proprement parler mais sont générés dynamiquement à l'exécution côté serveur d'un ensemble de fichiers PHP qui communiquent avec une base de données [Fig.1].

Bien entendu, l'objectif de l'édition de sites basés sur les CMS ne concerne pas tant le contenu des pages, puisque celui-ci est créé via une interface d'administration dédiée qui alimente la base de données, que leur mise en forme à travers l'édition des templates, qui sont principalement basés sur des feuilles de styles CSS. Pour éditer ces templates de manière WYSIWYG, il est donc nécessaire de réellement exécuter les scripts côté serveur pour afficher les pages concernées comme elles le sont dans le contexte d'un navigateur.

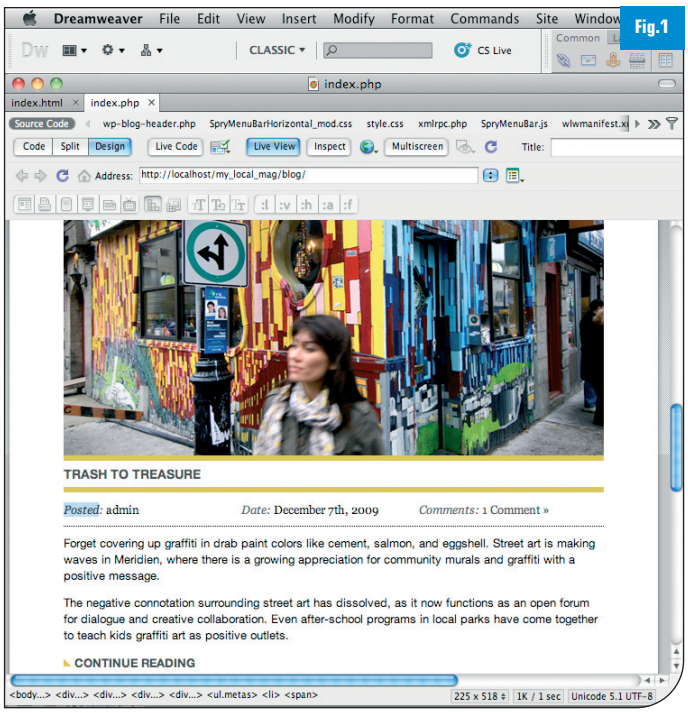


Fig.1

Analyse des pages web

Depuis la version CS4, Dreamweaver est capable d'analyser une page et d'afficher les fichiers liés à cette page, comme les fichiers JavaScript, les feuilles de style CSS, ou autres scripts serveurs. La liste des fichiers associés est ainsi toujours présente dans l'interface et permet d'accéder très simplement au code externe, mais lié, à la page éditée. Avec cette version CS5, cette fonctionnalité est étendue aux pages dynamiques de manière récursive: c'est donc bien l'ensemble des fichiers liés à une page qui est détecté, y compris d'ailleurs si ces fichiers ne sont pas présents localement.

D'autre part, le mode d'affichage en direct ou "LiveView" peut commander l'exécution des fichiers de scripts situés sur un serveur local ou distant afin d'exécuter la page comme si elle s'exécutait dans un véritable navigateur. Et pour cause… Ce mode utilise le moteur de rendu Webkit -ce même moteur qui équipe les navigateurs tels que Safari ou Google Chrome- afin d'afficher la page directement dans l'interface du logiciel sans avoir à systématiquement lancer la page dans un navigateur externe pour avoir un aperçu du rendu final de sa page HTML [Fig.2].

Dreamweaver CS5 apporte une nouvelle dimension à ce mode de prévisualisation en faisant de lui un véritable petit navigateur, capable d'afficher n'importe quelle page web, dynamique ou non, quelle que soit sa complexité en termes d'imbrication (on y trouve d'ailleurs un champ URL, des boutons précédent/suivant et un bouton Home). Le premier avantage de ce navigateur intégré est de permettre de tester l'ensemble des pages d'un site et ses liens externes en une seule session, plutôt que d'avoir à prévisualiser les pages une à une. Il suffit de cliquer sur un lien en maintenant appuyée la touche Ctrl / Pomme pour naviguer vers le lien concerné. D'autre part, LiveView permet désormais à l'utilisateur d'inspecter les styles CSS (bouton Inspect) des éléments de la page simplement en les survolant, à la manière d'un FireBug. Le survol d'un élément permet de "surligner" celui-ci dans la page, ce qui en fait un excellent moyen d'étudier le Box Model, et de corriger les problèmes asso-

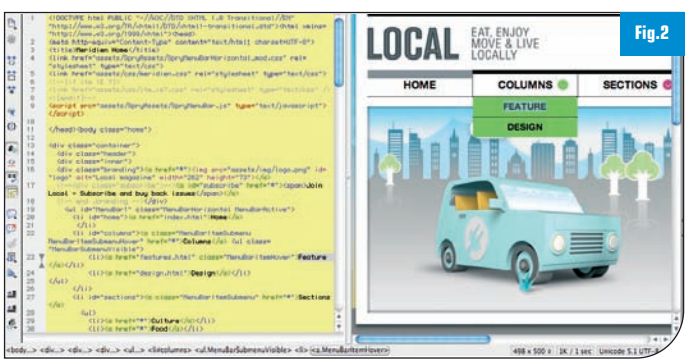


Fig.2

ciés. Après avoir sélectionné l'élément, il devient possible de modifier les styles qui y sont associés, voire de les désactiver définitivement ou temporairement [Fig.3].

Un éditeur de code de plus en plus complet

Du côté de l'édition de code PHP, l'auto-complétion est désormais capable de reconnaître les classes personnelles, ainsi que n'importe quel code spécifique au site. Ainsi, par exemple, dans un site Wordpress, le développeur bénéficiera de l'auto-complétion pour l'ensemble des classes Wordpress en plus de ses propres classes et des classes de base de PHP (pour lesquelles Dreamweaver affichera la documentation directement dans l'éditeur) [Fig.4].

La configuration d'un site été largement simplifiée. Pour créer un site, il suffit de donner un nom et un répertoire local. Si pour l'utilisation de certaines fonctions Dreamweaver a besoin d'informations complémentaires, il nous les demandera le moment venu. L'interface permet également de définir plusieurs configurations de serveurs, en distinguant les serveurs de production et les serveurs de test.

Les services Live très présents

L'une des principales nouveautés de Dreamweaver CS5 se trouve non pas dans le logiciel lui même, mais dans votre navigateur! En effet, la Creative Suite 5 introduit de nouveaux services web, les CS Live Services. Parmi ces services, BrowserLab est sans aucun doute celui qui séduira le plus rapidement les développeurs Web. En effet, les différences d'intéropérabilité et d'implémentation des standards du web entre les OS, les navigateurs et leurs différentes versions, obligent les développeurs web a constamment s'assurer que leurs pages sont correctement affichées dans ces différents contextes. Cette étape est particulièrement fastidieuse et nécessite de disposer de tous ces contextes à sa portée. BrowserLab nous propose de faire ce travail à notre place à la demande, côté serveur [Fig.5].

Accessible depuis n'importe quel navigateur, ce service exécute un rendu d'une page web sur une liste de navigateurs de systèmes d'exploitation, prend un instantané de ces rendus, et nous les affiche. Il nous permet ensuite de comparer ces images de différentes manières, côte à côte ou de manière superposée, afin de mettre en évidence rapidement les différences, si subtiles soient-elles. Il nous permet également d'afficher des règles et des guides, et même un double curseur de souris afin d'être le plus précis possible dans nos comparaisons. La liste des navigateurs et systèmes d'exploitation est entièrement configurable. Il dispose également d'un certain nombre d'option utiles, notamment la possibilité de spécifier un délai entre le rendu de la page et la capture de l'image, afin de laisser le temps aux contenus côté client (Javascript, Flash) de s'exécuter.

Enfin, et bien que ces futurs standards du web ne soient pas encore finalisés, le support du HTML5 / CSS3 n'est pas en reste. En effet, simultanément à la sortie de Dreamweaver CS5, Adobe a publié sous la forme d'une Technology Preview une extension dédiée à ces nouveaux langages. Le Dreamweaver CS5 HTML5 Pack ajoute l'auto-complétion pour les nouveaux tags et attributs HTML5, et les sélecteurs et styles CSS3. L'extension ajoute également un nouveau panneau "multiscreen" qui permet d'avoir un aperçu de sa page affichée dans différentes résolutions, l'idée étant de profiter du Media Query CSS3 pour adapter l'apparence de la page à son contexte. Ce sujet (Dreamweaver et le HTML5) fera l'objet d'un article dédié dans une prochaine édition.

Flash Builder 4 et le développement de RIA

Pour la première fois, la CS5 web intègre l'ensemble des produits de la famille flash pour couvrir l'ensemble des types de contenu associés à ce format, tous profils confondus. Flash Professional demeure l'outil de référence pour l'exploitation de flash avec des compétences créatives. Mais nous allons plus particulièrement nous concentrer sur Flash Builder 4, l'IDE pour le développement d'applications flash [Fig.6]. Flash Builder 4, qui est en réalité la nouvelle version de Flex Builder, fait pour la première fois son entrée dans la Creative Suite. Historiquement, cet environnement de développement basé sur Eclipse a d'abord eu pour vocation d'aider les développeurs à créer des applications Flash basées sur le framework Flex. Cependant, il est progressivement devenu l'éditeur de référence pour l'édition de code AS3 d'une manière générale. Ainsi, utilisé conjointement à Flash Pro, il permet en effet de disposer d'un environnement dédié au développement, et de mieux séparer le travail du graphiste de celui du programmeur.

La CS5 a donc tenu compte de cette utilisation en facilitant la collaboration entre les deux logiciels. Flash Builder est donc désormais capable de créer des projets Flash Professional. Ces projets, contrairement aux simples projets ActionScript, ont besoin d'une référence au fichier de projet Flash Pro (FLA ou XFL) afin de permettre aux deux logiciels de communiquer. Notons qu'il reste possible d'utiliser Flash Pro conjointement à Flash Builder dans le cadre

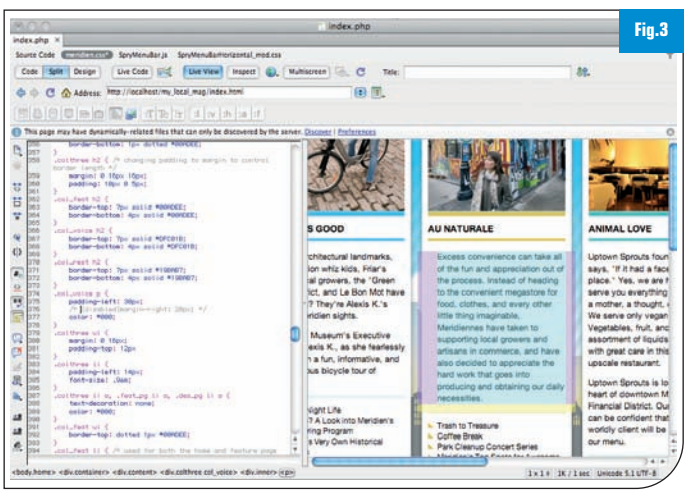


Fig.3

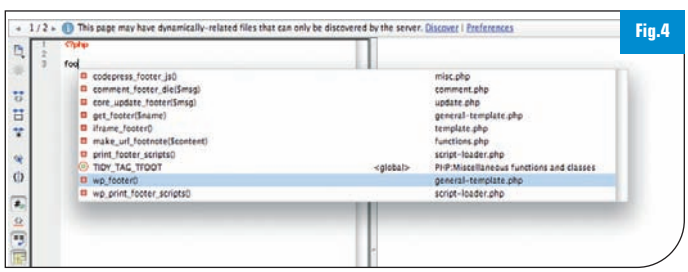


Fig.4

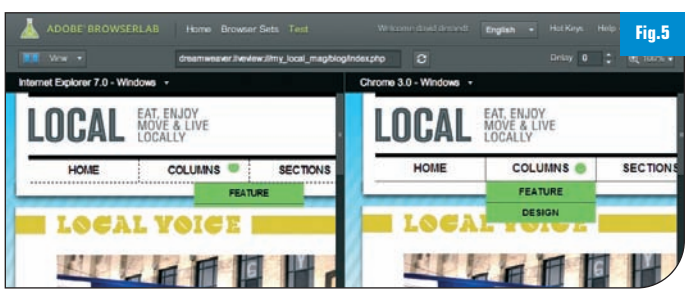


Fig.5

du développement d'un projet Flex. Dans ce cas, un développeur qui aura créé un projet Flex dans Flash Builder pourra placer dans son application un composant ou un conteneur Flex créé dans Flash Pro. Au-delà de cette collaboration avec Flash Pro, l'environnement de Flash Builder s'est considérablement enrichi. Le package explorer, qui remplace le navigateur, nous permet comme son nom l'indique de naviguer dans ses sources sous la forme de paquetages plutôt que de fichiers, et permet l'inspection directe du contenu des fichiers de bibliothèques liées à notre projet. Le panneau de View Call Hierarchy permet quant à lui de repérer où dans notre projet une méthode est appelée afin de trouver facilement ses dépendances.

Coté éditeur, l'aide de code s'est enrichie d'un panneau ASdoc qui nous permet d'obtenir de précieuses informations sur les éléments proposés par l'auto-complétion. De nombreux outils permettent de générer du code afin de gagner en productivité: getter/setters, listeners d'événements déclarés en attributs MXML, proxy pour les services distants (après leur introspection à distance via un panneau dédié), de value objects, formulaires master/details, ... [Fig.7]

Les tests unitaires sont également à l'honneur puisque le framework FlexUnit4 est directement intégré: le framework est automatiquement ajouté aux projets à la demande, et les cases, les suites de test, et même le runner sont générés pour nous à partir de la classe à tester: il ne reste donc plus qu'à se consacrer au contenu test en lui-même. Autre ajout important dans cette version 4: le panneau network monitor permet d'inspecter en temps réel les données échangées entre client et serveur. Une fois cette fonctionnalité activée, il est possible de lire le contenu d'un message afin de constater si oui ou non les données que l'on imagine envoyer de part et d'autre sont réellement envoyées, ce qui est fondamental pour le débogage d'applications client/serveur. En parlant de débogage, il faut aussi noter l'apparition de points d'arrêts conditionnels et de watchpoints.

La sortie de Flash Builder 4 est concomitante avec celle de Flex 4. Le SDK et le framework se trouvent profondément modifiés. Cette mise à jour du framework a un objectif principal: améliorer la créativité dans le développement d'applications Flex. Les RIA créés sous Flex 3 avaient tendance à être relativement austères du point de vue graphique notamment. Le framework flex 4 introduit un tout nouveau jeu de composants, "spark", qui sépare les responsabilités entre comportement du composant et aspect visuel (les composants mx sont toujours présents mais dépréciés). Ainsi, les skins de composants sont désormais déclarés dans des classes MXML totalement séparées du composant auxquels ils se rattachent, et s'utilisent par composition. L'extrait de code suivant montre l'utilisation d'un skin spark pour personnaliser l'apparence d'un bouton :

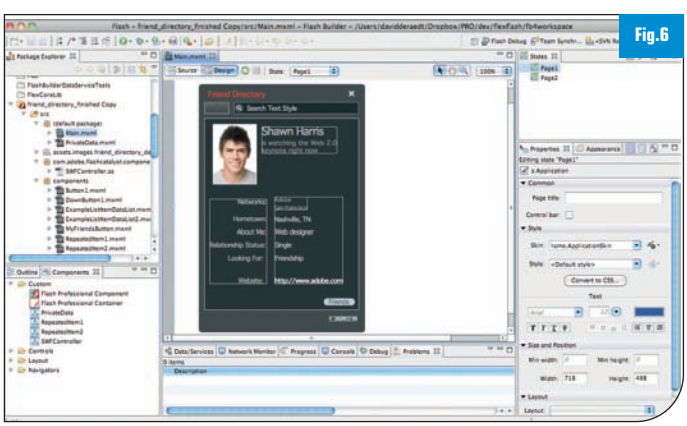


Fig.6

```
<s:Button x="10" y="10" label="Button" skinClass="components
.MyFriendsButton" />
```

Ces classes de skins définissent l'aspect visuel du composant au sens large. Tout d'abord, l'aspect graphique est déclaré grâce aux balises MXML Graphics, primitives de dessins très proches de la syntaxe du FXG (le nouveau format de description d'images vectorielles d'Adobe).

Faciliter la collaboration designer / développeur avec Flash Catalyst 1.0 et le skinning Flex4

Avec ce nouveau mécanisme de skinning de composants spark Flex4, il devient donc potentiellement possible de créer des applications Flex qui soient extrêmement riches graphiquement. Cependant, la création de classes de skins MXML peut représenter un travail titanesque si on attend d'un développeur qu'il les rédige lui même. De plus, une bonne séparation des responsabilités voudrait que ces skins soient réalisés par des créatifs plutôt que par des programmeurs [Fig.8].

Flash Catalyst répond précisément à cette problématique. Il permet à un designer de partir d'une simple image Photoshop, Illustrator, ou FXG, et de générer automatiquement les classes MXML de skins à partir de cette image, sans une ligne de code. Les fichiers générés pourront alors être passés à un développeur qui se chargera de programmer le comportement de l'application sans se préoccuper de son aspect visuel.

Le designer, lui, n'a pas forcément conscience qu'il génère des classes de skins MXML. De son point de vue, il dessine simplement une application dans son logiciel graphique préféré, puis ouvre le fichier de cette image dans Flash Catalyst afin d'effectuer un travail de description de l'application. Pour ce faire, il va reprendre chacun des éléments de son image et indiquer son rôle grâce au panneau *Convertir en Composant*. Par exemple, après avoir sélectionné une partie de l'image qui représente un bouton, l'utilisateur vient sélectionner le champ "bouton" dans le menu déroulant du panneau. Il peut alors définir les différents états d'affichage du composant pour les personnaliser graphiquement. La même opération sera répétée pour les cases à cocher, champs de textes, liste déroulantes, et autres éléments qui constituent l'application.

Au fur et à mesure de son travail de description, un projet Flex 4 est généré en coulisse, avec ses composants, et leurs skins. Parfois, certains éléments de l'image originale n'auront pas vocation à être transformés en composant. Dans ce cas, l'utilisateur choisira la commande "optimiser" qui permettra au choix de garder les informations vectorielles ou d'aplatir l'image afin d'en faire un élément de la bibliothèque du projet.

Une fois qu'un composant est créé, l'utilisateur peut rentrer dans ce composant en double-cliquant sur celui-ci, afin de l'éditer. Au delà de la modification graphique de l'élément, il est alors possible de définir

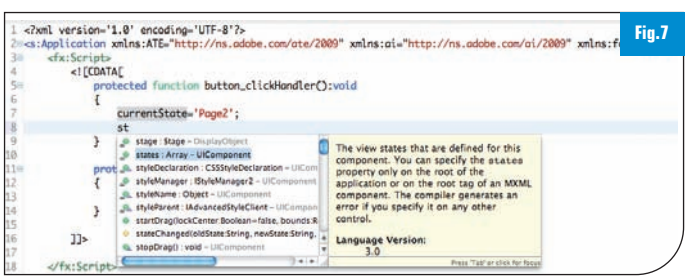


Fig.7

les états de ce composant. Par exemple, pour un bouton, il sera possible de dire que, à son survol (état "rollover"), il change de couleur. Pour des composants personnalisés, il est aussi possible de créer ses propres états.
Il est aussi possible de spécifier des transitions spécifiques entre les états d'un composant (zoom, déplacement, fondus, etc). Par exemple, je peux vouloir faire en sorte que, si un élément disparaît d'un composant dans un certain état, cette disparition soit progressive. Pour ce faire, on utilisera la TimeLine de FlashCatalyst, qui nous permettra, comme son nom l'indique, de bien contrôler l'évolution de ces transitions dans le temps.
Certaines formes d'interactions minimales peuvent être ajoutées à l'application depuis Flash Catalyst. Celles-ci sont principalement liées à la navigation. Elles permettent par exemple de faire en sorte que le clic sur un bouton nous amène vers une autre section de l'application, lise une vidéo, ou encore déplace un élément, toujours sans écrire une ligne de code.
Même si elles restent simples, ces interactions permettront au designer de produire une ébauche d'application, certes non totalement fonctionnelle, mais tout à fait utilisable par un tiers. L'application résultante pourra en effet être facilement distribuée pour le web (sous la forme d'un fichier SWF) ou comme application native multiplateforme (via le runtime AIR). Flash Catalyst est donc particulièrement adapté au prototypage car totalement autonome dans le cadre d'un concept ou d'une maquette.
Si au contraire l'application a pour vocation d'être entièrement développée, le résultat du travail effectué dans Flash Catalyst devra être passé à un développeur Flex utilisateur de Flash Builder. Dans ce cas, l'utilisateur de Flash Catalyst pourra soit exporter l'ensemble du code source de l'application généré sous la forme d'un fichier FXP (projet d'application Flash Builder), soit exporter sa bibliothèque de composants et de skins sous la forme d'un fichier FXPL (projet de bibliothèque Flash Builder).
Dans un cas comme dans l'autre, le développeur Flex retrouvera donc dans Flash Builder l'ensemble des éléments créés dans Flash Catalyst, que ce soit dans le panneau component, dans la liste des skins dans le panneau de styles CSS, ou via l'auto-complétion de l'éditeur de code.

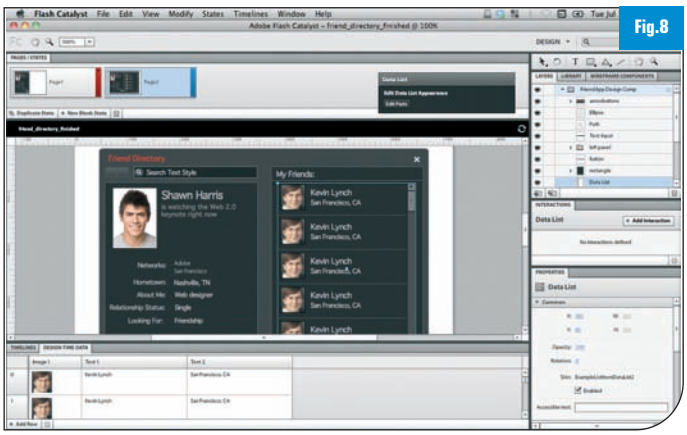


Fig.8

■ David **Deraedt**, *Consultant Web chez Adobe*

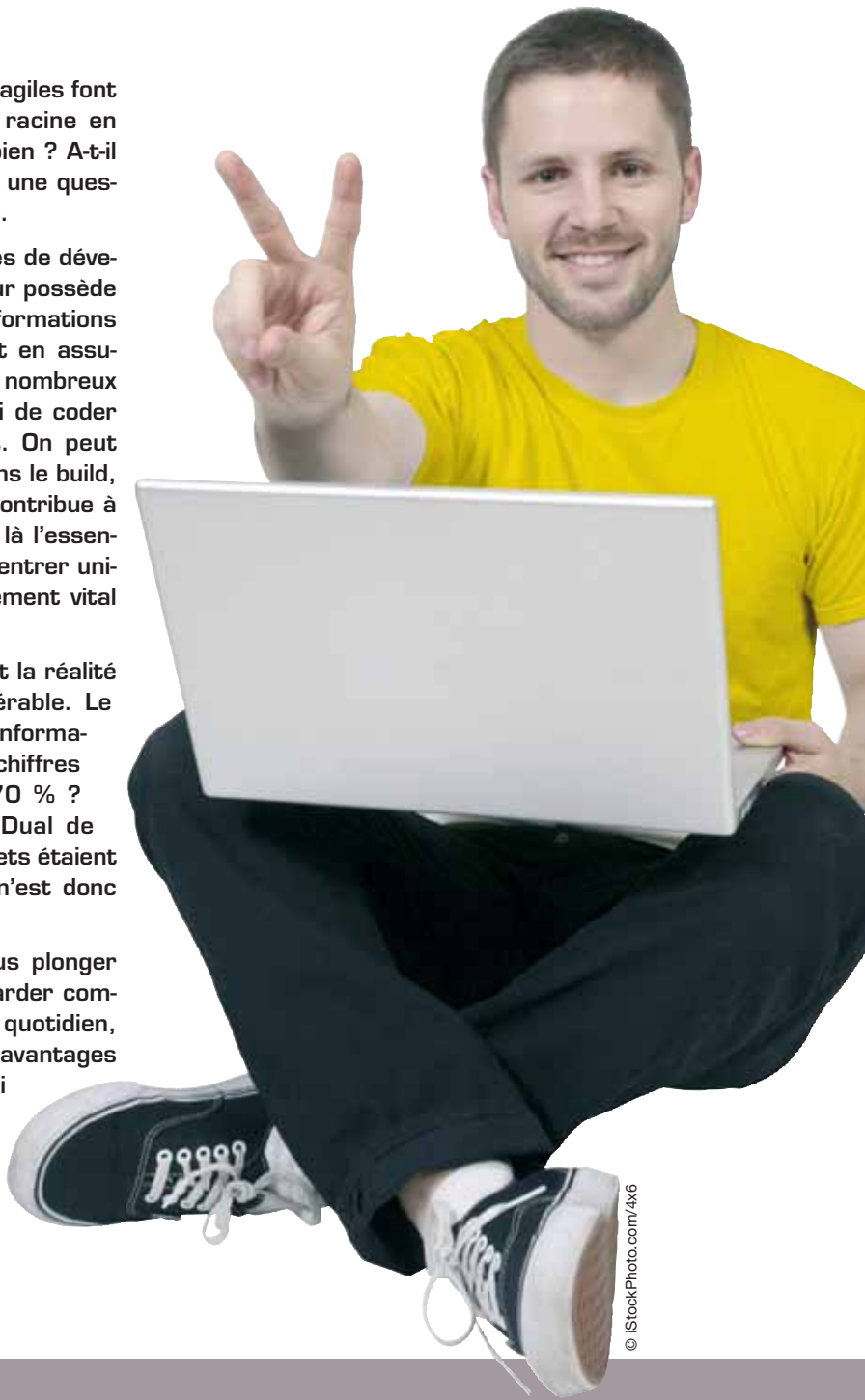
Agile et productif !

Cela fait des années que les méthodologies agiles font parler d'elles. Elles commencent à prendre racine en entreprise, SSII. Mais le développeur le vit-il bien ? A-t-il envie de ressembler à un acrobate ? Tout est une question de dosage et d'adapter la méthode choisie.

Comme toute chose, les méthodes et pratiques de développement évoluent. Aujourd'hui, le développeur possède un arsenal de bonnes pratiques, d'outils, de formations pour mieux développer, plus rapidement, tout en assurant la qualité du code et du produit final. Les nombreux frameworks, librairies, API évitent aujourd'hui de coder de zéro de nombreuses couches techniques. On peut autoriser beaucoup de choses, notamment dans le build, les tests et même le déploiement. Tout cela contribue à être plus productif au quotidien. N'est-ce pas là l'essentiel finalement ? Se simplifier la vie et se concentrer uniquement sur le code, le développement réellement vital et stratégique ?

Malheureusement, entre les bonnes paroles et la réalité du terrain, il existe un fossé parfois considérable. Le retard ou l'échec pur et simple d'un projet informatique restent encore bien trop fréquents. Les chiffres sont difficiles à cerner : 50 %, 60 % ou 70 % ? Cependant, comme nous l'a indiqué Eric K'Dual de Neoxia, il y a à peine 3 ans, seuls 5 % des projets étaient agiles, aujourd'hui c'est 50 % ! Tout espoir n'est donc pas perdu !

Dans ce dossier de rentrée, nous allons nous plonger dans l'agilité, les méthodologies agiles et regarder comment elles peuvent aider le développeur au quotidien, dans son travail. Rien n'est trivial mais les avantages sont indéniables ! Mais il faut savoir aussi adapter et prendre ce qui est utile dans l'agilité. Appliquer "bêtement" une méthode agile de A à Z ne présente aucun intérêt. Nous verrons ce qu'il faut entendre par productivité du développeur et comment vous pouvez y parvenir…



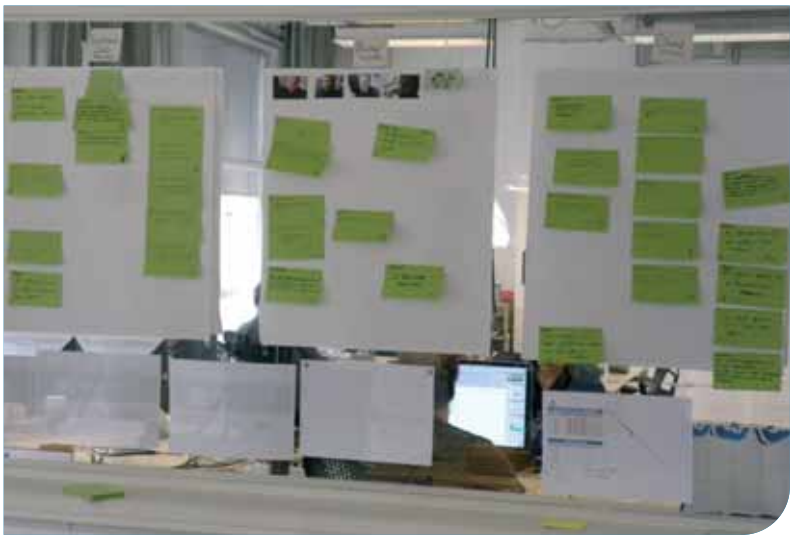
© iStockPhoto.com/4x6

François **Tonic**

L'agilité vue par les développeurs : "une journée en enfer"

Pierre et Louis sont développeurs agiles. Non, ils ne sont pas nés comme ça, mais leur société a su s'adapter et innover pour gagner en compétitivité et proposer à leurs équipes des méthodes de travail plaisantes et enrichissantes pour les collaborateurs.

Par la force des choses d'abord, et ensuite par le constat d'une réelle efficacité dans leurs tâches quotidiennes, Pierre et Louis ne reviendraient pour rien au monde à leurs anciennes méthodologies lourdes et complexes. Mieux, au fil des semaines, ils proposent eux-mêmes des améliorations aux méthodes agiles classiques, dérivant vers une méthodologie qui colle de mieux en mieux au contexte de leur société. Nous allons suivre 3 périodes clés de leur cycle de développement.



Un exemple de Scrum Board. D.R.

LA JOURNÉE DE DÉVELOPPEMENT

Daily meeting

Un des points cruciaux de la matinée est le daily meeting. Pendant 15 minutes (précises!) les membres de l'équipe vont parler de leur avancement depuis le meeting de la veille. Le rituel est simple. Les développeurs indiquent sur quelles tâches ils ont travaillé, pendant combien de temps, s'ils sont bloqués par un problème (c'est le moment de demander de l'aide!) et enfin estiment en gros combien de temps de travail il reste à effectuer pour compléter la tâche. Pierre, comme ses collègues, sait bien que pour pouvoir passer ses post-it (représentant une tâche) dans la colonne "fait" du tableau, il doit avoir rempli plusieurs conditions : que la tâche soit finie, testée et envoyée dans le gestionnaire de sources (Subversion ici par exemple).

Depuis quelques sprints, l'équipe a décidé que lorsqu'une User Story est terminée, le Product Owner peut la valider. Ce sera donc lui qui passera le post-it de la story de la colonne "A valider" à la colonne "Validé". Cette validation au fil de l'eau permet de revenir plus rapidement sur une User Story au lieu d'attendre la fin du sprint pour se replonger dedans en cas de problèmes détectés lors de la validation.

Ensuite, Pierre va choisir les post-it qu'il aura à faire durant la journée. Évidemment son choix est dicté par l'ordre de priorité des tâches et la possibilité de les paralléliser. Le but étant de tout faire pour terminer en premier les User Stories qui ont le plus de valeur ("Valeur métier") pour le Product Owner. Dans cette optique Pierre va prendre les tâches d'une User Story déjà commencée, afin de ne pas avoir non plus 3 ou 4 User Stories ouvertes en même temps.

Après la réunion, Pierre observe le scrum master mettre à jour la courbe d'avancement en comptant les points des post-it des User Stories qui sont passées dans la colonne "Validé". Il voit que le sprint a pris un peu de retard et qu'il va falloir s'appliquer davantage pour se recaler sur la droite théorique d'avancement.

Le développement

Après avoir pris ses post-it, Pierre se remet dans le contexte de la première tâche. Même si la User Story avait été discutée et expliquée lors de la séance de cotation, il est nécessaire de s'assurer de la bonne compréhension en relisant le backlog du projet. La partie qui l'intéresse particulièrement est celle récapitulant les tests d'acceptation. En effet, y sont détaillés les exemples concrets (avec des jeux de données) de ce que son code doit faire.

Si la tâche de développement est complexe, le post-it est souvent accompagné d'une tâche de conception en équipe. Pierre va donc se réunir, avec le nombre de personnes et pendant le temps défini lors du découpage de la User Story en tâches. Si les choix sont déterminants pour l'application, par exemple le choix d'un framework, il est néces-

saire de convier le "responsable technique" de l'application (l'architecte ou le leader technique) afin de prendre les décisions qui s'inscrivent dans l'architecture globale de l'application. Pendant la réunion, Pierre prend des notes dans le wiki de l'application sur les choix techniques qui sont faits.

Louis, lui, commence sa journée par réparer le build continu. Pas question de laisser le build cassé plus longtemps, ici c'est la responsabilité commune du code qui prévaut. Même si la plupart du temps le build est immédiatement réparé par celui qui l'a cassé, lorsque le build de nuit (avec tous les tests d'interfaces qui prennent trop de temps pour être lancés à chaque commit tout au long de la journée) est en échec, c'est la première personne de l'équipe qui s'en aperçoit qui prend l'initiative de réparer les tests en erreur. Une fois les choix techniques arrêtés, Pierre va se concentrer sur l'aspect métier. C'est le moment de décrire le comportement de l'application, mais cette fois-ci avec du code. C'est ici qu'intervient l'écriture des tests en amont.

L'option du choix

Pour cela, le développeur agile a deux possibilités : écrire des tests unitaires (TU) ou écrire des tests d'un peu plus haut niveau (TDR). Ces méthodes ne sont évidemment pas exclusives et il est même préférable de les combiner. La limite d'utilisation de l'un ou de l'autre étant très fine il revient souvent au développeur de faire un choix pour tester un aspect particulier. Bien sûr, ce choix est également dicté par la stratégie de tests mise en place sur le projet. Par exemple si les spécifications ont déjà été rédigées dans l'outil de TDR, il est beaucoup plus pratique et rapide de partir de cette base pour créer les tests.

Pierre choisit d'utiliser son outil de TDR favori (fitnesse, greenpepper) car il compte bien sur son Product Owner pour enrichir a posteriori les jeux de tests qu'il vient de créer. En effet la grande force du TDR est de permettre d'insérer des jeux de tests dans une page du wiki de l'application. Cette page de spécifications sera "exécutée" par l'outil de TDR pour passer les jeux de tests. L'édition de cette page se révélant des plus simples, le Product Owner qui n'est pas programmeur pourra ajouter à sa guise les données les plus tordues auxquelles il n'avait pas pensé dans un premier temps.

Les premières lignes de code sont donc destinées à l'écriture des tests (classes jUnit pour les TU, fixtures pour les tests TDR) et du squelette du code (les signatures des méthodes, les classes des nouveaux objets).

Une des bases du développement agile est l'amélioration continue du code. C'est pourquoi Pierre en étudiant la classe qu'il va devoir modifier, remarque que des avertissements sont remontés par l'outil de mesure de qualité du code de son IDE. Un paramètre de méthode non utilisé ? Une exception non loggée ? Une absence de tests sur une classe sensible ? Si les avertissements sont simples à corriger, Pierre apporte tout de suite la correction. Il peut sur le champ lancer les tests et s'assurer que ses modifications n'ont pas fait régresser l'application.

Lorsque Pierre voit des remaniements de fond qui seront plus longs à mettre en place, comme par exemple la montée de version d'une librairie tierce, il va ajouter une ligne dans le backlog technique. Ce backlog tenu par les développeurs est dépilé de temps en temps lorsque le développeur a du temps pour s'y consacrer, ou lorsque la tâche est suffisamment conséquente et importante pour être ajoutée dans le backlog du projet. Ainsi un post-it dédié avec des points de complexité sera créé dans le prochain sprint.

Dans le processus d'amélioration continue, il est aussi important de surveiller la liste des actions correctives qui ont été prises en compte lors des dernières rétrospectives. En effet, les développeurs agiles cherchent non seulement à améliorer leur code mais aussi à progresser sur leur méthode de travail afin d'être le plus efficace possible. Ainsi Pierre peut se remémorer les bonnes pratiques à suivre, en relisant dans le wiki ce qui avait été statué lors des dernières rétrospectives.

Une fois son travail commencé, Louis se rend compte que tous les cas de figure n'ont pas été abordés par le Product Owner dans le backlog. Afin de savoir comment doit réagir l'application dans des conditions particulières, Louis doit aller questionner directement le Product Owner. C'est là qu'on voit toute l'importance d'avoir un Product Owner présent ou joignable rapidement afin de prendre les bonnes décisions pour ne pas passer du temps à développer quelque chose qui n'aura pas le bon comportement. Bien sûr avant de commencer l'implémentation, Louis commence par coder les tests induits par les nouvelles spécifications.

Encore des tests

Après quelques heures de travail, Pierre a finalement ses nouveaux tests qui passent au "vert". L'équipe fonctionnant dans un mode agile, un build continu est mis en place. Pierre s'assure donc que tous les tests préexistants continuent à passer. Cela signifiera qu'il a rempli le contrat défini par le Product Owner sans faire de régression sur le reste de l'application.

Il veut maintenant partager son code avec le reste de l'équipe. Pour cela, il s'assure dans un premier temps que ses collègues n'ont pas eux-mêmes déjà envoyé leur travail dans le gestionnaire de sources. Si tel est le cas, Pierre récupère leur travail et va s'assurer que l'ensemble des tests fonctionnent toujours.

Louis a moins de chance, alors qu'il récupérait le travail de ses collègues, il s'aperçoit qu'il a de gros conflits sur une classe sur laquelle ils étaient plusieurs à travailler en même temps. Comme il veut s'assurer de ne pas supprimer le travail de son collègue tout en conservant le sien, il va demander à celui-ci de venir l'aider à

"merger" les sources. Une fois le code finalisé, Louis s'apprête à envoyer son travail. Mais comme le sort s'acharne sur lui, il se rend compte qu'un autre développeur a envoyé son travail avant lui, et en plus vient de casser le build. Là, pas question de tout de même envoyer ses sources dans le gestionnaire de sources. De toute façon les tests ne passeraient pas chez lui non plus. Louis est frustré mais il sait qu'un peu de patience est le prix à payer pour garantir une application qui fonctionne en permanence dans une chaîne de production bien huilée. Alors il offre son aide au développeur qui a commis une erreur afin de l'aider à corriger le build défaillant.

Afin d'éviter des envois simultanés et donc le risque de casser le build, Pierre récupère la petite peluche qui symbolise le droit de commit. Une fois les sources envoyés, il replacera la peluche dans sa boîte.

Avant de commencer une nouvelle tâche, Pierre s'assure que personne n'a besoin d'aide.

La prochaine tâche est un peu particulière : Pierre doit programmer en binôme avec Louis, qui vient de rejoindre l'équipe, afin de le faire monter en compétences sur une partie de l'application. Car travailler en mode agile, c'est aussi être capable de pouvoir intervenir sur n'importe quelle partie du code, même s'il existe toujours des leaders sur tel ou tel domaine fonctionnel ou technique. Pierre a l'habitude de binômer pour transmettre ses connaissances mais aussi pour aider un collègue bloqué par un bug ou des parties de code qu'il a écrit pas suffisamment claires.

Revue de Sprint

Les cérémonies de fin de sprint commencent. Une journée bien remplie en perspective !

Et il y a du monde : l'équipe est là au complet bien sûr, et il y a aussi le business et le marketing.

Ça démarre par la revue de sprint. Le Scrum Master présente l'état du sprint. Ce coup-ci, l'équipe a fait fort, 37 points de vélocité, soit 5 de mieux que le sprint précédent. Après 5 sprints, l'équipe avait stabilisé une vélocité de 32 points. Cela fait plaisir à l'équipe mais aussi au Product Owner qui trouve bien sûr son intérêt dans l'avancement et la réussite de son projet. Louis nous fait une démonstration des différentes user stories en « live », ainsi que la correction des 3 anomalies trouvées lors de la validation du sprint précédent.

La présentation du sprint et la démo n'auront pas duré plus de 25 minutes chrono !

Les tests fonctionnels sont tous bons et le Product Owner a validé les 37 points produits. Cette fois-ci en plus, l'intégration continue est allée jusqu'à la mise en pré-production de l'application, sur laquelle tourne maintenant la démo.

Rétrospective

La cérémonie de rétrospective commence. Chacun expose les événements importants qui ont eu lieu lors de ce sprint. Pierre a identifié le fait d'avoir travaillé sur une nouvelle user story d'un thème qu'il ne connaissait pas sur laquelle le support du Product Owner a été plus que bénéfique.

Ensuite, c'est le rituel : qu'est-ce qui s'est bien passé, qu'est-ce que l'on peut améliorer, que doit-on arrêter ? Chacun y va de ses post-it, et pour ce sprint c'est au tour de Louis de récolter les post-it de tout le monde, de les présenter à l'équipe et de les regrouper par thèmes sur le tableau.

Puis chacun se lève et vote en mettant des points (de 1 à 5) sur chaque item qui lui semble le plus important. Pierre mets 3 sur « continuer les workshops avec le Product Owner sur nouvelle user story » et 2 sur « améliorer la qualité du code ». Louis compte les points : Il en ressort que l'équipe insiste sur les mêmes points que Pierre. Ils en déduisent des actions à entreprendre dans le but d'améliorer encore la production sur le prochain sprint. Après 2 heures de discussions, la réunion est terminée.

Sprint planning

Le Product Owner présente les user stories qu'il souhaite voir implémentées dans ce sprint. Il ne s'agit pas de faire un workshop complet par user story. Ayant la chance d'avoir l'équipe business sur place, l'équipe peut poser ses questions au moment de démarrer la première tâche de la story.

Le but est de présenter l'ergonomie, le contexte de la user story, la façon dont elle s'insère dans l'application et les exigences décrites dans green pepper (le wiki où sont consignées les spécifications). À partir de là, les développeurs peuvent évaluer la charge en point de complexité, en fonction de leur expérience sur le projet, des similitudes avec les user stories précédentes, des POC (Proof of concept) évalués précédemment sur ce sujet et bien sûr aussi de toutes les expériences passées.

Pour la cotation, les développeurs utilisent le planning poker. Pour chaque user story, chacun propose une carte de « poids », de 1 à 13 maximum suivant la suite de Fibonacci (0,1/2, 1,2, 3, 5,8,13). S'il y a désaccord, l'équipe re-discute de la user story en soulevant les problèmes ou en questionnant le Product Owner pour lever des doutes. Après un second vote, s'il y a accord, on passe à la story suivante. S'il y a toujours désaccord et que le poids semble dépasser les 13, alors la user story est redécoupée ou n'est pas acceptée dans ce sprint (elle est reportée à une prochaine itération). Sans doute un POC aiderait à mieux l'appréhender. Il est aussi possible que la story ne soit pas assez aboutie et que l'équipe refuse de la traiter sans avoir plus d'informations.

Au total, le Product Owner demande un effort de 37 points identiques au sprint précédent. Toute l'équipe est d'accord pour jouer le jeu et s'engage sur cette vélocité attendue.

Les User Stories sont ensuite découpées en tâches auxquelles un nombre d'heures est affecté. Bien sûr il faut intégrer le temps d'implémentation des tests sur chaque tâche. Les tâches de moins d'une heure sont donc très peu courantes !

L'équipe est prête à repartir pour un nouveau sprint et une nouvelle journée !

■ Jean-Baptiste **Cazeaux**
■ Laurent **Carbonneaux**
Consultants Valtech Technology

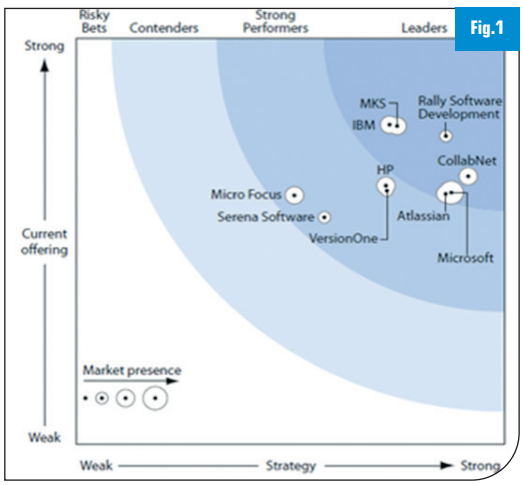
Le développeur au centre de l'agilité

Entre un contexte économique de plus en plus contraignant, imposant des délais plus courts avec toujours plus d'exigences , et une course à l'innovation introduisant de plus en plus vite de nouvelles technologies (mobilité, cloud computing, Green IT, …), les équipes de développement adoptent de plus en plus des pratiques agiles dont les plus connues sont celles issues de XP et Scrum.

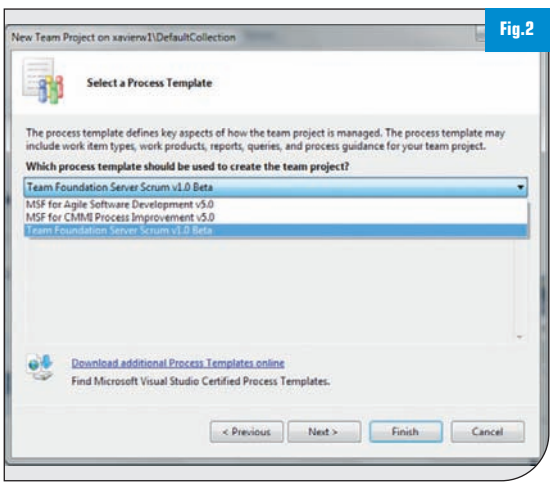
Dans ce contexte, de plus en plus de développeurs, pas toujours « agiliste » dans l'âme, doivent trouver de nouveaux repères, se former, et continuer à livrer leurs projets comme avant ! Malgré la promesse de l'agilité où le taux de réussite des projets est de 80 % (à comparer avec les statistiques de projets en général montrant un taux de réussite de 33 % environ), une organisation doit veiller à piloter avec attention sa transition vers l'agilité sous peine de voir ses développeurs désorientés.

Les valeurs fondatrices de l'agilité

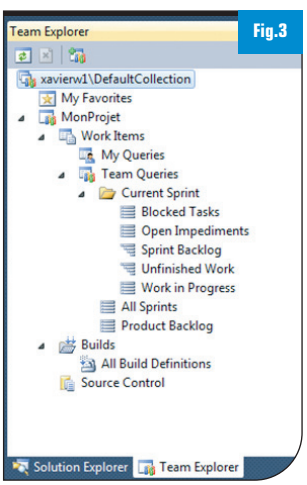
Les valeurs d'XP, engagement, courage et de Scrum, transparence, adaptation, ne conviennent pas toujours d'emblée à des équipes de développement parfois organisées de façon très hiérarchique et habituées à exécuter les décisions d'un chef de projet, unique interlocuteur d'une MOA. Les esprits doivent être préparés à tous les niveaux : MOA, équipe/MOE et développeur. Une démarche d'accompagnement est souvent nécessaire d'autant plus que les pratiques agiles remettent en cause la spécialisation des développeurs (codeur, testeur, concepteur, …). En effet, il est tout à fait possible par exemple qu'un « ancien testeur » code plus régulièrement (en mode « pair programming ») par exemple dans une équipe « agile ».
Le changement peut être bien perçu (augmentation des compétences opérationnelles) ou rejeté (perte de mon expertise). Un coach en gestion du changement peut être alors très utile pour aider les développeurs à suivre un vrai processus de « deuil » .
Les individus et leurs interactions priment par rapport aux processus et aux outils avec les pratiques agiles .
Une fois, le développeur convaincu et motivé par l'agilité, il demeure encore de nombreux écueils liés à la communication, au comportement,… Des approches basées sur la **P.N.L.** (Programmation Neuro Linguistique) peuvent alors aider des équipes à mieux interagir .
Reste encore à proposer au développeur des outils soutenant ces pratiques lui permettant de s'exprimer pleinement. L'un des meilleurs moyens est de proposer une formation adressant l'usage d'outils modernes combiné aux pratiques logicielles (type XP) et Scrum pour la gestion de projets. Ken Schwaber a d'ailleurs créé le cours « Professional Scrum Developer » allant dans ce sens.
Ce cours aborde les sujets comme le TDD, l'intégration continue, la définition de la notion de « Done », des tests d'acceptation avec un angle de vue pratique au travers de la prise en main d'outils. Sur 5 jours, les développeurs en formation travaillent en Scrum avec un backlog produit, plusieurs sprints, des rétrospectives avec en fin de semaine une mise en production de la version améliorée d'un logiciel. En effet, il s'agit de partir d'un logiciel existant (en Java ou .NET) qu'il faut enrichir. Chaque sprint permettra de prendre conscience de l'ensemble des pratiques et outils à maîtriser et donner ainsi envie aux participants d'aller plus loin dans chaque pratique.



Source Forrester Research, Inc



Création d'un projet avec le template Scrum v1.0 Beta



Team Explorer

Ce type de cours est un bon moyen pour le développeur de rapidement être opérationnel dans le monde .NET ou Java, les cibles principales de ces cours. Seule la version .NET de ce cours est donnée actuellement et d'ailleurs Microsoft met aussi un site d'information à ce sujet : « Professional Scrum Developer Program ». Le cours .NET est basé sur les outils Visual Studio 2010 et Team Foundation Server (TFS) 2010.

Microsoft et plusieurs de ses partenaires, ont investi afin de proposer un environnement intégré puissant au développeur agile et à une équipe agile avec Visual Studio 2010 et TFS 2010.

Effectivement, avec l'arrivée du template Scrum pour TFS 2010, le support de Scrum, la méthode agile la plus utilisée au monde devient native. Notons qu'avant même ce nouveau template, Forrester positionnait déjà TFS 2008 parmi les solutions leaders du marché des outils supportant Scrum : [Fig.1].

Du Scrum dans les outils de développement

Le template pur Scrum tient compte d'une réalité importante du marché : Scrum est de loin la méthode de gestion de projets la plus adoptée et loin devant les autres comme DSDM ou FDM. Reste qu'un nombre non négligeable de projets ne suit encore aucune méthode précisément [Fig.2].

Le template Scrum est aligné directement sur les notions Scrum comme la notion de Sprint (et non plus d'itération) :[Fig.3].

On peut aussi noter que la notion d' « Impediment » apparaît aussi directement dans les « Team Queries ».

Ce nouveau template « Scrum » pour TFS est destiné à être particulièrement optimisé pour les projets Scrum (Team Foundation Server Scrum v1.0 Beta). Ce template est aussi disponible sur MSDN (Process Templates and Tools).

De façon plus détaillée, voici les différences entre les templates MSF Agile v5.0 et TFS Scrum v1.0 concernant les types de "Work Item" (Work Item Types ou WITs) :

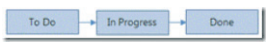
TFS Scrum v1.0 WITs	MSF Agile v5.0 WITs
Product Backlog Item	User Story
Bug	Bug
Task	Task
Impediment	Issue
Test Case	Test Case
Shared Steps	Shared Steps
Sprint	Itération

Les types de "Work Item" avec le template TFS Scrum correspondent bien à la terminologie Scrum.

Les états et transitions entre états des « Work Items » correspondent eux aussi à Scrum :

Product Backlog Item et Bug Work Items :

Task Work Item :



Impediment Work Item :

Les états et la terminologie collent maintenant mieux à Scrum.

De nouveaux rapports sont maintenant proposés pour afficher les métriques Scrum :

- Release Burndown - Ce graphique représente la quantité totale de points restant à faire dans la release, au fil des sprints. Il permet d'avoir une vue sur l'avancement de la release [Fig.4].
- Vélocité – Ce graphique représente le montant du travail effectué (en "story points") par une équipe et par sprint [Fig.5].
- Sprint Burndown – Ce graphique représente le travail restant à faire (en heures) par une équipe sur un sprint [Fig.6].

Ce nouveau template Scrum doit permettre à chacun d'appliquer Scrum directement et simplement avec TFS.

De plus, grâce au mécanisme de template de TFS, il est toujours possible d'adapter le template Scrum à vos besoins particuliers (remontée d'indicateurs supplémentaires démontrant la conformité à des contraintes d'assurance qualité spécifiques, etc.).

Avec ce template purement Scrum, un certain nombre de partenaires proposent des addins puissants comme Pyxis avec Urban Turtle et une vue de type Post-It des tâches en cours (Task Board) offrant ainsi une vue plus intuitive pour des praticiens de l'agilité qui ne souhaitent pas être bridés par un outil.

Dans le cas précis d'Urban Turtle, les différentes vues offertes permettront à une équipe de développement agile de partager et de collaborer, et ce

Fig.4

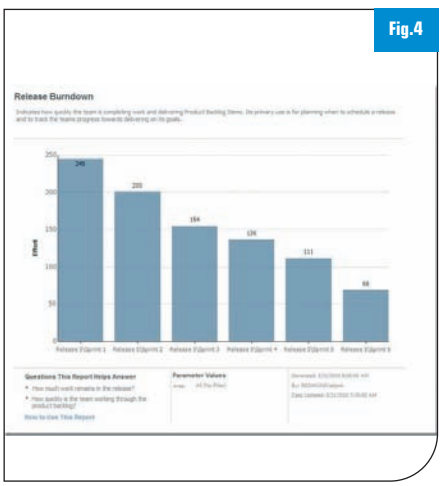


Fig.5

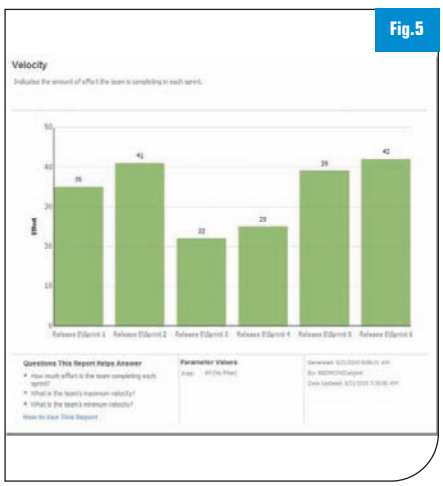
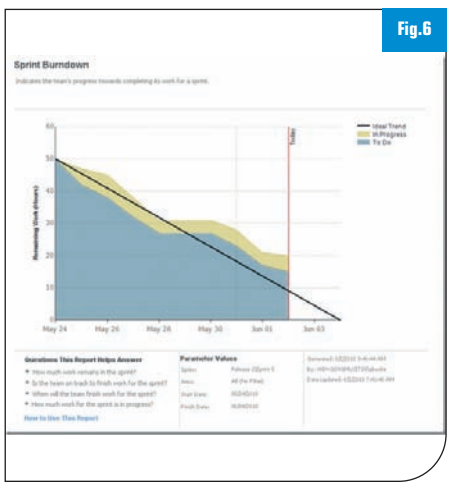


Fig.6



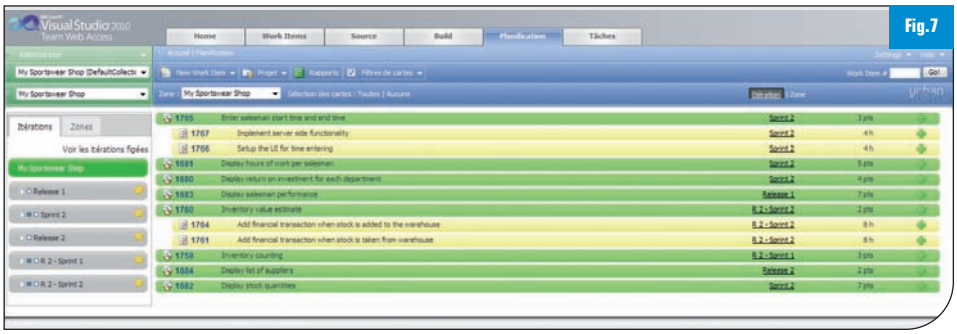


Figure 4 Urban Turtle : vue Sprints

Figure 5 Urban Turtle : vue TaskBoard

durant les différentes phases du sprint. Visualiser le contenu du travail en cours, préciser certains scénarios avec le gestionnaire de produit ou bien encore communiquer sur son avancement dès l'archivage d'une modification du code, les équipes agiles tendent à simplifier au maximum les efforts à fournir à ces tâches de gestion de projet [Fig.7].

Une nouvelle génération d'agilistes

Si les plus conservateurs des agilistes peuvent tenir un discours préconisant l'utilisation de cartes en papier, apposées sur les murs et accessibles rapidement, la mise en place d'un outil de gestion de changement comme Team Foundation Server ne manque pas d'arguments.

En effet, Team Foundation Server offre la traçabilité entre les différents éléments du projet (les scénarios, les tâches, le code, les tests, les versions compilées…).

Cette traçabilité et les liens créés entre les différents artefacts permettront d'analyser sprint après sprint les efforts et la progression de l'équipe dans son apprentissage de la méthode mais également dans son rôle principal de créateur de produit logiciel [Fig.8].

D'autre part, même si la co-localisation des membres d'une équipe est plus que fortement recommandée par le framework Scrum, il faut admettre que certains projets nécessitent de voir collaborer des équipes ou des équipiers distants. Les limites de l'approche sans outil logiciel (le mur, le papier et les petites mains de développeurs bougeant et mettant à jour ces différentes cartes) sont alors atteintes rapidement. Il convient alors de préférer un système virtualisé qui permettra à chacune des parties prenantes au projet de consulter, dans les mêmes conditions, les mêmes informations.

Cette égalité face à l'accès à l'information est primordiale lorsqu'une équipe est répartie géographiquement. Dans cette situation, certaines personnes pourront faire entendre qu'il s'agit pour eux d'une détérioration de leur environnement (en général, lorsque la répartition de l'équipe n'est pas équilibrée).

On pourra les rassurer en leur présentant les autres avantages d'un système électronique et notamment les diverses améliorations ergonomiques d'un tel système.

En effet, comme une base de données, Team Foundation Server et les outils qui l'accompagnent permettront à l'utilisateur de filtrer, regrouper et trier les différents éléments de travail.

La consultation de l'historique permet également de consulter l'évolution de l'évaluation d'un scénario ou d'une tâche limitant certains malentendus ou en facilitant la compréhension.

Ce détour par Urban Turtle montre que le secteur est très dynamique dans l'outillage pour combiner richesse fonctionnelle et savoir en même temps être le moins intrusif possible dans les pratiques. Ce dynamisme se retrouve au niveau communautaire. Ainsi en France, le développeur agile peut compter sur une communauté active avec les associations « Agile-France » (ex XP-France), le « French Scrum User Group » (FrenchSUG), les événements comme « Agile France Conférence », l'AgileTour, et les événements du FrenchSUG. Il s'agit d'autant d'occasions d'échanger entre praticiens de tous horizons.

En conclusion, le développeur agile par son engagement et ses résultats devient l'un des actifs les plus précieux d'une organisation ayant bien opéré sa transition et son institutionnalisation des pratiques agiles.

Donc pour éviter un passage à Scrum un peu « flaccid », comme dirait Martin Fowler, aucune hésitation, suivre des formations combinant pratiques modernes d'ingénierie logicielle comme le TDD, l'intégration continue, les tests d'acception, la définition du « Done », utilisation des outils supportant ces pratiques et bien évidement une mise en situation dans un projet piloté avec Scrum.

■ Xavier **Warzee**
Architecte, Microsoft France, Certified Scrum Master & Président du « French Scrum User Group »

■ Mathieu **Szablowski**
développeur agile chez Pyxis (http://pyxis-tech.com), Certified Scrum Master

egilia®

LEARNING

LE SPÉCIALISTE DE LA
FORMATION CERTIFIANTE
EN INFORMATIQUE
ET MANAGEMENT

Faire de vos succès
notre réussite

www.egilia.com

CONTACTEZ NOS CONSEILLERS FORMATION

N°National 0 800 800 900

APPEL GRATUIT DEPUIS UN POSTE FIXE

ANVERS . LIEGE . PARIS . LYON . LILLE . AIX-EN-PROVENCE .
STRASBOURG . RENNES . BRUXELLES
TOULOUSE . BORDEAUX . GENEVE . LAUSANNE . ZURICH .

Le développement agile Java : vive Scrum !

Le développement agile vise à offrir une alternative aux cycles de développement traditionnels dits « en cascade » ou « en V ». Les principes en sont assez simples, mais leur mise en place représente souvent de véritables défis pour les équipes.

L'approche agile, au sens du manifeste agile [1], ne sous-entend pas d'accointances technologiques. Pour autant, les équipes de développement logiciel sont la plupart du temps liées à une technologie, et Java représente une part importante des compétences recherchées aujourd'hui [2]. Les « équipes Java » qui souhaitent tirer profit d'une approche agile sont rapidement amenées à se demander comment mettre en pratique les tests automatiques dans cet environnement.

Pourquoi un intérêt tout particulier pour les tests automatiques ?

Lorsqu'une équipe découvre l'agilité, elle commence la plupart du temps par mettre en place un cadre de gestion – type Scrum – et elle commence à vivre sur un rythme itératif avec des cycles le plus souvent de 2 ou 3 semaines visant à maximiser l'apprentissage et à favoriser l'amélioration continue. La plupart des équipes héritières d'un processus de développement en cascade vont commencer leur aventure agile en reproduisant la cascade au sein de chaque itération. Cascade dont le goulet d'étranglement est souvent le test manuel en fin d'itération. Cette façon de réaliser du logiciel atteint très vite une limite. En effet, après quelques itérations, l'équipe se retrouve souvent dans ce qui semble une impasse : comment continuer à ajouter des fonctionnalités alors que les tests de non-régression manuels occupent tout le temps disponible de l'itération ?
Une voix assez commune pour se sortir de cette ornière est alors d'apprendre à automatiser ces tests.

Mais peut-on automatiser tout type de test ?

Cette question est rarement liée à la possibilité technique d'automatisation. En effet dans le monde numérique tout est techniquement possible. Cette question du « pouvoir » cache une question de l'ordre du « devoir » : devrait-on automatiser tous les tests ? Un premier axe étudié communément repose sur le coût de l'automatisation. Un raisonnement court terme amène les équipes à repousser l'écriture de tests qu'elles ne savent pas écrire. En effet, le temps passé à apprendre à écrire un test est du temps qui leur semble perdu. D'autre part, il arrive qu'une technologie donnée ne dispose pas sur étagère de l'outillage de tests automatisés. Il faut alors envisager de créer cet outillage, ce que les équipes débutantes rechignent à faire. Ce sont souvent les mêmes qui se retrouvent bloquées après plusieurs itérations. Dans le cas de cette équipe hypothétiquement bloquée, elle serait sans doute dans une situation beaucoup plus confortable si elle disposait d'un harnais de tests qui la rassureraient sur le fait qu'elle n'ait pas détruit les incréments logiciels précédents. Pour cela, une condition nécessaire, c'est-à-dire minimum, est de disposer de tests automatiques qui décrivent le logiciel souhaité par le client : ce sont les tests qui décrivent que l'équipe construit le bon logiciel.

Fig.1

```
@Test public void
containsLogo() {
        assertThat( homePageContent, hasImageHtmlTag( "logo.png" ));
}
```

Fig.2

GreenPepperized ☑ - Configure
Execute for [refactoring]
Rights: 2 Wrongs: 0 Errors: 0

Rule for	Delivery service		
recipient	available	send	actually sent ?
joe@example.com	yes	try send	yes
jack@example.com	no	try send	no

Et dans l'environnement Java, cet outillage existe-t-il ?

La bonne nouvelle avec Java c'est que l'outillage de test existe et est très mature. Que vous soyez sur des architectures web ou non, toutes les couches logicielles peuvent aujourd'hui être testées/décrites/spécifiées les unes indépendamment des autres. Par mature, nous entendons ici que l'utilisateur de cette technologie a le choix. Un large choix consiste finalement à reconnaître que chacun a une façon de penser personnelle et que la grande diversité d'outils de test lui permettra de choisir ceux qui lui semblent les plus naturels. De façon très anecdotique nous pouvons citer l'exemple de Mockito. L'auteur de Mockito nous explique les raisons qui l'ont poussé à écrire une nouvelle librairie pour écrire des mocks [3]. Si vous partagez son avis sur EasyMock, Mockito vous semblera sans doute plus naturel.
Aujourd'hui, il est possible de couvrir par des tests automatiques toutes les couches logicielles d'un système [4] sans avoir à investir dans la création d'une librairie de tests spécifique ; la librairie qu'il vous faut existe probable-

ment déjà. Si ces notions sont nouvelles pour vous, une façon simple de faire le grand saut consiste à rechercher sur le web par exemple « TDD Spring » et vous aurez accès à plusieurs articles qui vous aideront à vous lancer dans l'écriture de tests automatiques dans le cadre de l'utilisation de Spring.

Show me the code ! ;)

Effectivement un test automatique est un programme que l'on écrit comme tous les autres programmes avec un langage ; en l'occurrence ici un code en Java [Fig.1].

Lorsque l'on choisit d'écrire un test, on a pratiquement toujours plusieurs choix d'outils ou de conteneur pour ce test. Par exemple [Fig.1], l'équipe a choisi d'écrire ce test avec Junit [5] dont l'intention est de spécifier qu'un logo doit être présent dans la page d'accueil. Le choix de l'outil dépend la plupart du temps de la personne avec qui il faut avoir une discussion pour pouvoir l'écrire. Le code ci-dessus pourrait ne pas mettre à l'aise une personne assez éloignée de la technique. D'autres outils à vocation plus collaborative ont donc vu le jour. Certains comme GreenPepper [6] se basent sur un wiki et donnent une vision moins technique des tests [Fig.2]. Du choix de l'outil peut dépendre notre capacité à collaborer avec les personnes nécessaires à l'écriture du test. Ce besoin de collaboration devient crucial lorsque les tests sont écrits en premier, c'est-à-dire avant que le programme lui-même n'existe.

Ecrire les tests avant le programme à tester ?

Oui, c'est là que les tests automatiques décuplent leur valeur.

En effet, considérant comme un minimum les tests qui décrivent que nous avons construit le bon logiciel, cela semble une bonne idée de les connaître avant de commencer. L'avantage de cette approche est de rendre non équivoque la spécification, car les exemples contenus dans le test pourront explorer non seulement les cas nominaux d'utilisation mais également les scénarios exceptionnels.

Comment faire dans le cas de la reprise d'un code existant sans test ?

Le développement piloté par les tests (TDD) a un impact important sur l'architecture d'une application. Il favorise l'émergence de zones de responsabilité distinctes et découplées, propres à favoriser l'évolutivité d'un système. Les codes, souvent anciens, développés sans tests sont également souvent les moins aptes à évoluer et à livrer de nouvelles fonctionnalités. Lorsque la meilleure idée de l'équipe est de conserver ces programmes, leur reprise de contrôle passe par l'écriture de tests systèmes englobant totalement la zone sur laquelle on souhaite intervenir avant d'entreprendre une ré-écriture de la zone.

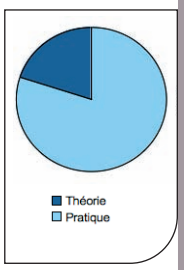
Cela ne semble pas trivial, existe-t-il des formations spécifiques ?

Ne nous trompons pas d'objectif : ce qui est non trivial c'est le développement logiciel. Les pratiques d'ingénierie agiles ne sont pas faciles à maîtriser, mais avec de l'aide et de la discipline, les gains sont vite visibles. Le programme Professional Scrum Developer (PSD) de scrum.org est un stage de cinq jours pour les développeurs d'une équipe Scrum. Le stage apprendra aux équipes à répondre à des demandes de fonctionnalité en construisant des incréments logiciels prêts pour la production en utilisant les pratiques d'ingénierie modernes. Pyxis est partenaire de ce programme et propose des formations dans les technologies Java ou .Net [7].

Objectifs

Scrum sera mis en place et étudié via des discussions, démonstration et exercices pratiques. Les participants apprendront à utiliser Scrum correctement en étant aidés par l'instructeur. En particulier, les sujets suivants seront étudiés :

- La construction d'équipe performante
- Le travail sur un code pré-existant
- Définition de critères de qualité, de critère d'acceptation
- Définition de « terminé »
- Build automatisé
- Correction d'anomalies
- Vérification de l'élimination des anomalies via des tests automatiques
- Planification de releases et d'itérations
- Estimations
- Création et gestion d'un sprint backlog
- Tenir efficacement une revue de sprint
- Améliorer son processus de développement grâce aux rétrospectives
- Éviter la dette technique grâce à l'architecture émergente
- Le développement piloté par les tests comme outil de conception
- Intégration continue
- Tests de toutes les couches logicielles
- Agir sur les dysfonctions d'équipe



Audience

Ce cours s'adresse à tout membre d'une équipe de développement – architecte, programmeur, testeur, etc. Nous encourageons les équipes à venir vivre ce stage ensemble. Les personnes venant seules intégreront une équipe. Les participants devront s'auto-organiser pour former des équipes multidisciplinaires. Ces équipes ont en effet besoin d'un ensemble spécifique de compétences pour adresser le cas d'étude proposé aux participants. Les Product Owner, ScrumMaster et autres contreparties sont également les bienvenus. Néanmoins, gardez en tête que tous les participants seront censés se comporter comme des membres d'équipes à part entière et participer aux engagements collectifs de leur équipe.

■ Eric **Mignot**
Pyxis Technologies

[1] http://www.agilemanifesto.org
[2] http://www.indeed.com/jobtrends?q=java&l=
[3] http://monkeyisland.pl/2008/01/14/mockito/
[4] http://www.natpryce.com/articles/000772.html
[5] http://www.junit.org
[6] http://www.greenpeppersoftware.com
[7] http://www.pyxis-tech.com/en/services/ formation/inscription/

L'intégration continue : le pilier du développeur

L'intégration continue est une pratique de plus en plus adoptée dans les projets informatiques. Mélange de processus et d'outillage, elle se situe à l'intersection de la plupart des bonnes pratiques d'ingénierie logicielle – qu'il s'agisse du domaine du test, de la documentation, du déploiement, de la gestion de configuration logicielle, ou de la livraison.

Dans cet article, nous nous attacherons dans un premier temps à établir la valeur et les cas d'utilisation de l'intégration continue. Nous ferons un détour du côté des méthodes agiles, puis nous aborderons les moyens de mettre en place une intégration continue selon le contexte technique.

Un peu d'historique

L'intégration continue est fortement mise en avant depuis l'introduction de la méthode XP (Extreme Programming), dont elle est une pratique à part entière. Cependant, le concept d'une validation fréquente et automatisée d'un produit logiciel est lui-même ancien ; il a été mis en œuvre dans le domaine militaire et aéronautique dès le début des années 80.

Un modèle en strates

Nous nous écarterons volontairement des définitions classiques de l'intégration continue, comme celle donnée par Martin Fowler par exemple (1).

Nous préférons présenter l'intégration continue comme un modèle comportant de multiples strates : [Fig.1]

Ces strates sont organisées de la plus simple à la plus complexe à mettre en œuvre. Elles ne sont pas toujours toutes nécessaires, ni même toujours possibles, selon la technologie utilisée.

En fait, le lecteur avisé aura déjà remarqué que ces différentes pratiques ne sont pas spécifiques à l'intégration continue. Il s'agit de tâches courantes du développement logiciel, qu'il est parfaitement possible d'accomplir manuellement. L'intégration continue consiste simplement à répéter fréquemment et automatiquement ces tâches.

Compilation

Compiler le logiciel fréquemment permet de vérifier… que le code se **compile** ! Ce constat paraît trivial, mais il ne l'est pas toujours, surtout avec des équipes distribuées travaillant sur les mêmes portions de code. Il n'est pas rare de rencontrer des équipes, dans un projet géré avec un cycle en cascade, qui intègrent le code à une étape précise du projet. En général, juste avant la phase de test, ce qui est une façon presque infaillible pour rencontrer des conflits dus aux changements effectués. Cette simple étape de compilation de l'intégralité du code pré-suppose que tous les

Le modèle en strate de l'intégration continue

Fig.1

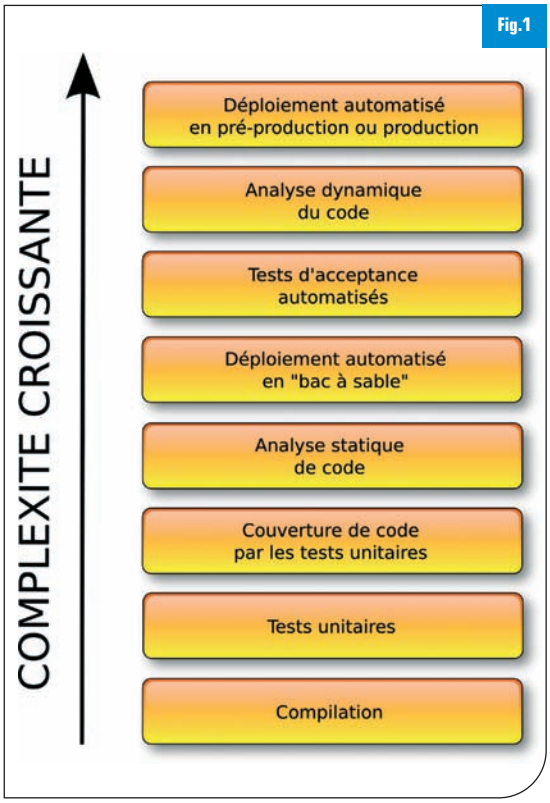
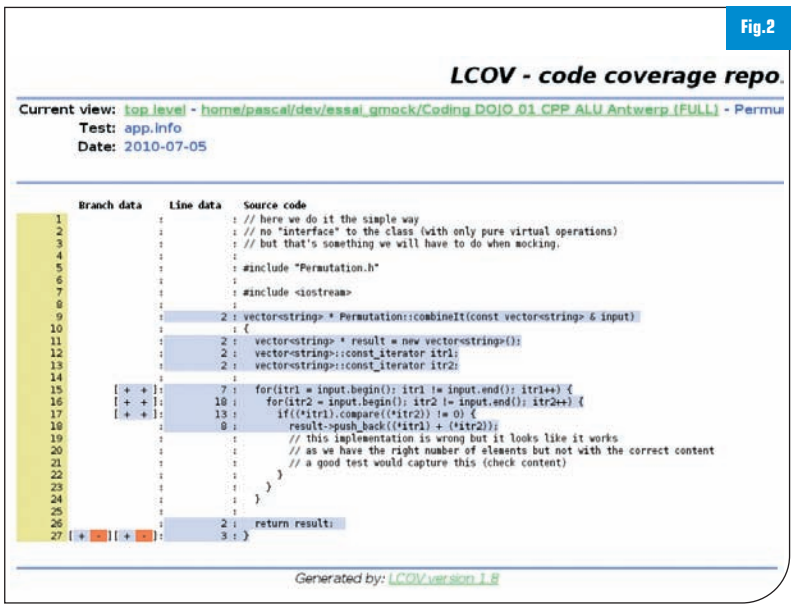


Fig.2



Exemple de couverture de code C++ avec gcc, gcov et lcov.

(1) Voir http://martinfowler.com/articles/continuousIntegration.html

membres d'une ou plusieurs équipes reversent leurs modifications dans une base commune versionnée, qui sera un gestionnaire de configuration logicielle, comme Clearcase, Subversion, Git, Mercurial, Team System, ou autre (2).
En intégration continue, on utilise toujours la version la plus récente des sources.

Test unitaire et couverture de code

Écrire des tests unitaires permet à un programmeur de prouver que son code est correct. Il ne permet pas de prouver que le comportement du système complet est correct.
Cependant, il s'agit d'une étape indispensable pour assurer la qualité intrinsèque du code, (qualité non perçue par le client, car en règle générale un test unitaire ne peut pas être relié directement à une exigence).
Il est essentiel que les tests unitaires soient joués le plus fréquemment possible ; dans l'idéal, chaque fois qu'une modification est faite dans la base de code.
Des tests unitaires sans mesure de couverture de code n'ont que peu de valeur. On peut en effet écrire des centaines voir des milliers de tests, sans pour autant explorer toutes les branches du code.
Lors de la mise en place d'une intégration continue, il est donc indispensable d'activer simultanément les tests et la couverture de code [Fig.2].

Analyse statique de code

Très souvent, l'outillage permettant d'analyser le code est le premier mis en place dans une intégration continue. Pourtant, il apporte une valeur inférieure aux tests unitaires. En effet, détecter que le code est mal formaté, mal documenté, ou encore ne respecte pas des conventions de codage, n'apporte pas une garantie ni de fonctionnalité ni de stabilité du produit (3). Tout au plus pourra-t-on assurer que le code sera plus facile à maintenir.

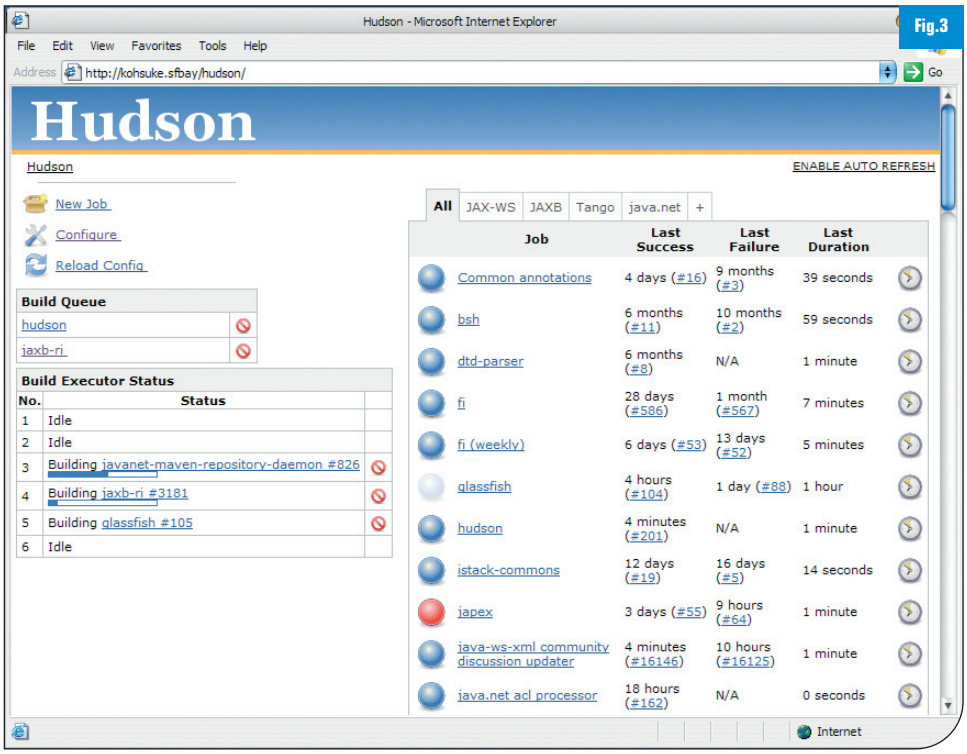


Fig.3

Copie d'écran de Hudson

Tests fonctionnels et déploiement automatisés

Il n'est pas toujours facile de définir ce qu'est un test fonctionnel (parfois appelé « test de système, ou de sous-système »). On peut cependant affirmer qu'un test fonctionnel suppose une orchestration d'appels à des fonctions élémentaires du système, souvent à travers une interface graphique (4).
Un test fonctionnel suppose que le système doit être déployé, au moins dans un environnement « bac à sable ». L'automatisation de ce déploiement nécessite une industrialisation de l'application, et donc un niveau de maturité supplémentaire.

Analyse dynamique du code

Depuis l'intégration continue, on est amené à lancer différentes catégories de tests. Ces tests traversent différents chemins d'exécution du code. Avec certaines technologies (en particulier les langages à bytecode comme java et c#), on peut, grâce à l'injection d'aspects (5), tracer les appels faits entre différentes couches du système. Il s'agit d'un aspect complémentaire de l'analyse statique de code, qui permet par exemple de détecter si une couche présentation fait un appel direct à une couche de persistance de donnée.

Déploiement automatisé en production

Le but ultime de l'intégration continue est de tout automatiser… Y compris le passage en production ! Il s'agit bien sûr d'un niveau de maturité

(2) Les spécialistes de la GCL nous pardonnerons de mettre dans le même sac des gestionnaires de révision, comme cvs, et des gestionnaires de configuration plus complets, comme Clearcase.

(3) On pourrait objecter qu'une analyse de code statique sur du code C++, par exemple, peut détecter des bugs qui feraient planter le logiciel. Cependant, si le logiciel est complètement testé avec des tests unitaires (en appliquant TDD), alors ce bug est très certainement déjà capturé avant même d'arriver au stade de l'analyse de code.

(4) A noter que dans le monde de l'embarqué et des Télécoms, les acteurs d'un test « fonctionnel » sont souvent des acteurs informatiques. La frontière entre test unitaire et test de sous-système est parfois difficile à établir.

(5) Par exemple avec AspectJ.

exceptionnel. La plupart des projets n'en sont pas à ce stade, d'autant plus que le déploiement en intégration ou en production est souvent délégué à des équipes disjointes.

Liaison avec l'agilité

Arrivé à ce point de l'article, le lecteur se demande peut-être quel lien on pourrait trouver entre l'intégration continue et les méthodes agiles. Et le fait est qu'on peut parfaitement exécuter toutes ces tâches manuellement, ou de manière automatique, dans le cadre d'un projet géré avec un cycle en V ou en cascade. Cependant, le lien existe : dans les méthodes agiles, le concept *d'itération courte* est essentiel. À chaque fin d'itération, l'équipe projet doit livrer un produit incomplet, mais fonctionnel. C'est-à-dire correctement testé. Il devient donc essentiel d'accomplir toutes les tâches d'ingénierie très fréquemment et avec un coût minimal, dans le but de maintenir la dette technique (6) à un niveau très bas. L'intégration continue permet de remplir cette condition.

implémentation

L'intégration continue peut être mise en place à peu près avec n'importe quelle technologie.

Cependant, certaines sont mieux loties que d'autres ! [Fig.3]

Dans le monde Java, l'outillage est largement disponible avec des outils de test et de couverture (junit, cobertura, emma), d'analyse statique (pmd, javancss, checkstyle, findbugs), de test fonctionnel ou TDR (Fit/fitnesse ; Robot Framework, Selenium, Soap UI). Pour automatiser l'appel à l'ensemble de ces outils, Hudson fait parfaitement l'affaire, mais de nombreuses autres solutions existent. Et on peut toujours ajouter Sonar, qui offre des vues très graphiques sur les métriques générées par les outils évoqués plus haut [Fig.4].

Dans le monde .net, la solution TFS de Microsoft intègre la plupart des outils nécessaires. Il est également possible d'utiliser des portages d'outils Java. Pour l'analyse du code, on pourra citer FXCop, Simian, Ncover, Ndepend, et d'autres.

Les langages de script, comme Ruby ou Python, sont également accompagnés de nombreux outils intéressants.

Par contre, dans le monde C/C++, qui est aujourd'hui principalement celui de l'embarqué, l'outillage existe, mais est assez disparate.

Selon le compilateur utilisé, la couverture de code sera faite avec gcov ou tcov ou Visual C++, les rapports graphiques seront disponibles ou il faudra les générer manuellement.

Les outils d'analyse statique de code Open Source sont rares et peu puissants, si on les compare avec leur homologue du monde Java (7).

Enfin, des produits ayant un historique de plusieurs dizaines d'années et des millions de lignes de code peuvent parfois mettre plus d'une journée à se compiler, ce qui compromet sérieusement l'idée de développeurs « commitant » plusieurs fois par jour.

Fig.4

Copie d'écran de Sonar (tirée du site nemo)

Conclusion

L'intégration continue est indispensable dans un projet agile, afin d'assurer la qualité de livraisons fréquentes. Mais elle a également une grande valeur pour des projets non agiles, basés sur des cycles plus longs, afin d'éviter des intégrations douloureuses en fin de projet.

Il est possible de l'utiliser quelle que soit la technologie employée. La mise en place de l'outillage est relativement peu coûteuse. C'est plutôt l'ensemble des pratiques issues d'Extreme Programming qui sont plus longues à mettre en place dans un projet. Par exemple, l'introduction du Pair Programming et de TDD donnera toute leur valeur aux tests unitaires.

L'intégration continue produira alors des mesures de métriques pertinentes.

■ Pascal **Ognibene**
Directeur technique
Valtech Technology
pascal.ognibene@valtech.fr

(6) La dette technique représente l'ensemble des tests manquants, des problèmes de style de code, de manque d'industrialisation, qu'on peut retrouver dans un projet informatique. Tout comme un emprunt, la dette devient de plus en plus lourde à résorber avec le temps, car les intérêts s'accumulent ! Souvent, la seule solution est alors de casser complètement l'existant pour le reconstruire correctement.

(7) Par exemple Flawfinder ou les variantes de Lint. Mais dans les outils commerciaux, Coverity est très efficace.

Améliorer la production du développement logiciel : les équations de l'efficacité

■ Adam **Kolawa**, *CEO de Parasoft* - Traduit de l'anglais par Gilbert **Vidal**

1 POURQUOI LA PRODUCTIVITÉ ?

Dans l'économie actuelle, le développement logiciel représente un budget important pour beaucoup de sociétés. Les retards de productivité dans ce domaine peuvent avoir un impact important sur les capacités concurrentielles, et même la survie de l'entreprise. La plupart des organisations qui font du développement logiciel ne sont pas optimisées. Le « C-level » considère toujours le développement logiciel comme un centre de coût avec un processus mal compris. Mais, la demande en logiciel augmente, notamment dans les systèmes embarqués. Sans efficacité, il devient difficile de profiter des opportunités, de manière rentable.

Cela nécessite plus d'outils. Il faut des processus générant de la qualité pour produire le logiciel de façon cohérente et efficace. A l'instar des constructeurs automobiles, les entreprises de développement ont accès peu ou prou aux mêmes outils et technologies. Les entreprises qui se sont dotées des processus de travail sont parmi les plus performantes, avec un coût de production moindre et sont les mieux préparées à la concurrence.

2 COMMENT CALCULER LA PRODUCTIVITÉ D'UN DÉVELOPPEMENT LOGICIEL ?

Un indicateur simplifié de productivité consiste à diviser le nombre de lignes de code produit par une organisation, par l'effectif (développement et contrôle qualité) travaillant sur ce code. En bref, je ferai référence à cet indicateur par LDC (ligne de code) et par individu. En fait, il représente l'horizon du développeur: quel volume de code chaque développeur peut appréhender. Cette mesure est la meilleure représentation de la productivité réelle de l'organisation. Si ce nombre est grand, cela signifie que vous avez un nombre relativement restreint de personnes travaillant sur le code. C'est-à-dire que vous faites beaucoup avec peu de ressources.

10 à 20 000 lignes de code par développeur est la norme actuelle dans l'industrie. C'est basé sur une moyenne générale d'un million de lignes de code pour un programme type, avec 50 à 80 développeurs contribuant au projet.

Evolution temporelle du LDC par individu

On me demande souvent: « vous référez-vous au nombre de lignes de code par individu et par mois ? par semaine ? ou à autre chose ?». Ma réponse est que l'intervalle temporel n'a aucune importance. Car le nombre de lignes de code d'un logiciel en évolution reste globalement constant. Chez les non-développeurs, on pense de façon erronée que la taille du code augmente continuellement au cours du développement. Mais en discutant avec les développeurs, vous découvrirez que le travail de développement (y compris l'ajout de nouvelles fonctionnalités) consiste principalement à la réécriture de code existant.

L'exception étant la phase de prototypage au début d'un tout nouveau projet, ce qui dure normalement quelques semaines. Ainsi, la compréhension du code par chaque développeur est un facteur critique pour la productivité des développements. Comme je le précise ci-après, la productivité de l'équipe s'accroît avec le volume de code que chaque membre appréhende réellement et peut effectivement modifier.

3 POURQUOI AJOUTER DES DÉVELOPPEURS N'EST PAS LA SOLUTION ?

Il est important de comprendre que le moyen le plus utilisé pour accélérer les développements d'un projet, c'est à dire augmenter la taille de l'équipe, a l'effet inverse, d'après cette équation. Cela semble illogique à première vue, mais si vous pensez à ce qui se passe quand de nouveaux développeurs rejoignent l'équipe, cela devient évident. Ajouter de nouveaux développeurs empêche au départ l'équipe de réaliser ses progrès habituels. En premier, les anciens membres de l'équipe doivent passer du temps à former et à communiquer avec les nouveaux. Plus grave, ces nouveaux membres ont une compréhension limitée du code. De ce fait, ils sont incapables de revoir le code existant (qui, comme je l'ai indiqué, est l'activi-

té principale du développement logiciel), ce qui limite les progrès de l'équipe. Ce manque de familiarité avec le code accroît le risque d'introduire de nouveaux défauts ou problèmes, ce qui entraînera un besoin ultérieur de réécriture en impactant négativement la productivité.
Dans la section suivante, nous examinerons le pourquoi de l'impact sur la productivité.
Plutôt que d'accroître la taille de l'équipe, la clé de l'accélération des développements est dans l'accroissement de l'efficacité du processus de développement: en d'autres termes, augmenter le LDC par individu. C'est une leçon qui est revenue à chaque fois dans d'autres industries. Quand on optimise les processus de production pour tirer avantage des dernières technologies, moins de personnes sont nécessaires sur la «ligne de production» car elles sont capables de faire plus - en moins de temps.

4 COMMENT ACCROÎTRE LA PRODUCTIVITÉ DU DÉVELOPPEMENT DE LOGICIEL

De par notre expérience depuis 21 ans dans le développement logiciel, nous avons retenu les équations suivantes :

- avoir un plan clair et un plan d'attaque = productivité
- améliorer la connaissance du code = productivité
- réduire les reprises sur un travail déjà effectué = productivité
- réduire le débugging = productivité

D'autres industries se sont battues contre les mêmes problèmes de productivité que ceux rencontrés par l'industrie du logiciel aujourd'hui, et ont rencontré des succès prodigieux en adoptant des approches similaires. En les appliquant dans notre industrie, nous pouvons amener la productivité des développements à des niveaux plus rentables et plus compétitifs.

Equation 1 : avoir un plan d'attaque clair et efficace

Plus le développeur passe de temps à se demander ce qu'il doit faire, moins il est productif pour aider l'organisation à atteindre ses objectifs. Pour garder les développeurs focalisés sur un travail productif, définissez les tâches de façon appropriée, installez un système de distribution des tâches, et assignez leur des priorités.

Définir des tâches appropriées

Dans le cadre du développement logiciel, les tâches sont souvent floues: elles sont définies de façon vague, ce qui rend difficile aux développeurs de décider quand les commencer et comment procéder.
Pour dépasser cette difficulté, il est essentiel d'avoir quelqu'un (un architecte ou un manager par exemple), avec une bonne appréhension conceptuelle du projet. Cette personne pourra prendre des besoins de haut niveau et les traduira en tâches élémentaires que le développeur pourra comprendre rapidement et implémenter avec succès.
Quand les développeurs reçoivent des tâches restreintes (par exemple qui peuvent être réalisées en moins d'une journée) a contrario de se trouver impliqués dans la participation à une tâche importante et floue, l'expérience a montré qu'ils deviennent bien plus productifs.
Comme ils comprennent exactement ou démarrer et quoi faire, ils sont bien plus efficaces: leurs objectifs sont atteignables et les progrès appréciables immédiatement.

> « Plutôt que d'accroître la taille de l'équipe, la clé de l'accélération des développements est dans l'accroissement de l'efficacité du processus de développement »

Disposer d'un système de distribution des tâches

Idéalement, les développeurs devraient tous avoir devant eux une liste des tâches par priorité, dans leur environnement naturel de développement. Cela diminue le nombre de questions sur ce qui est demandé et comment le faire.

Assigner les tâches de façon appropriée

Si une tâche est assignée à un développeur peu familier avec un code, le temps d'implémentation (avec les risques d'erreurs et de réécriture) sera beaucoup plus important que si la tâche est confiée à un développeur familier de cet environnement.

Equation 2 : augmenter la connaissance du code

La vitesse de programmation d'un développeur n'a rien à voir avec sa vitesse dactylographique. C'est totalement lié à la compréhension du code qu'il doit modifier. Quand un développeur reçoit une tâche, il doit comprendre ce que fait le code existant, comment modifier son comportement et prévoir l'impact des changements apportés. Cela nécessite la compréhension du code. En développant cette compréhension, vous pourrez accroître le nombre de tâches que vous pourrez confier à chaque développeur, et réduire le temps imparti à chaque tâche. Comment le faire ?
L'une des méthodes est la revue de code par un pair. Quand un développeur revoit le code développé par un autre, il en apprend la complexité et les connexions. Cela finit par accroître le volume de code que chaque développeur peut gérer, et la vitesse de réalisation des tâches. En plus d'augmenter les connaissances du relecteur, ce processus donne plus d'information au développeur qui reçoit un feedback d'un autre membre du groupe sur l'intégration de son code dans le projet global.
Une autre méthode consiste à faire des tests par régression. Si un développeur qui n'est pas familier avec du code doit le modifier, sa plus grande crainte est sans doute de casser

quelque chose qui fonctionne. Les tests par régression alertent immédiatement les développeurs de tout impact imprévu - par exemple si un changement affecte une fonctionnalité existante dans une autre partie de l'application. En plus d'enseigner aux développeurs les corrélations entre le code modifié et les autres parties du projet, cela accroît la productivité en donnant le courage aux développeurs d'améliorer et d'étendre le code sans être bloqués par la peur.

Etendre la connaissance du code de chaque développeur augmente la productivité en évitant les erreurs, qui entraînent de longues mises au point et des réécritures, qui sont détaillées dans les deux équations suivantes. Moins les développeurs sont familiarisés avec le code, plus ils feront des erreurs qui diminueront la productivité bien après l'implémentation du morceau de code considéré.

Equation 3 : Minimiser la mise au point

La mise au point est essentiellement le processus de réparation de ce qui a été mal implémenté. Plus ils passent de temps à régler les problèmes, moins les développeurs ont de temps à passer sur des tâches plus productives. L'une des façons de diminuer le temps passé en mise au point est de mettre en place une infrastructure qui évite aux défauts de rentrer dans la base du code. Elle devrait inclure une prévention automatique des défauts comme l'analyse statique, les tests unitaires etc.

La vraie valeur de la diminution des temps de mise au point grâce à la prévention des défauts, réside dans la réduction du besoin de réécriture. Sur le long terme, les pratiques de détection de catégories entières de défauts au fil des développements sont beaucoup plus efficaces que la chasse aux problèmes individuels, et leur correction en aval du processus de développement - ce qui est plus difficile, plus cher et plus long.

Equation 4 : minimiser la réécriture

Minimiser la mise au point est juste l'une des méthodes pour minimiser la réécriture. D'autres méthodes sont liées à la définition des besoins. Si les architectes et les managers ne comprennent pas les besoins dès le départ, et si à leur tour les développeurs ne les implémentent pas correctement, l'équipe finira par perdre un temps important à réécrire le code. Il est critique d'implémenter les besoins tels que les attend le client et tels que définis dans les spécifications. Afin de réaliser cette étape, vous devez vous assurer que chacune des parties comprend réellement les besoins avant de passer à l'étape suivante. Pour comprendre ce que cela implique, considérez cette analogie: le travail de développement est le même que celui d'un chantier de construction. En premier l'architecte trouve ce que souhaite le client et définit les besoins. A partir de là, comme les sous-traitants réalisent un bâtiment suivant les plans, les développeurs commencent à écrire le code.

Les erreurs proviennent de deux sources possibles :

- l'architecte ne comprend pas ce que souhaite le client, ce qui entraîne une mauvaise définition des besoins,
- le développeur ne comprend pas les spécifications et les implémente de façon incorrecte.

Les prototypes sont un moyen efficace d'éviter la réécriture liée aux incompréhensions entre le client et l'architecte. Un prototype donne une chance au client de revoir visuellement ce que les développeurs lui construisent et d'alerter l'équipe immédiatement si le projet semble partir dans la mauvaise direction.

Les incompréhensions client-architecte peuvent être évitées par une politique spécifiant que tout besoin doit donner lieu à un prototype pour vérifier la bonne implémentation des besoins. Cela force les développeurs à regarder les besoins sous deux angles différents - implémentation et validation - ce qui entraîne une compréhension intégrale de chaque demande.

Productivité = qualité

Il est surprenant de constater que l'impact de la mise en oeuvre de ces équations s'étend au delà de la productivité, dans le domaine de la qualité. Qualité et productivité sont liées inextricablement. Si vous suivez ces équations, la qualité de votre application augmentera car vous aurez mis en place un processus de développement plus résistant aux erreurs du fait :

- des développeurs qui comprennent mieux le code sur lequel ils travaillent et les tâches qu'ils doivent réaliser
- d'une infrastructure mise en place pour prévenir de nombreuses erreurs qui, sans cela, demanderaient de la mise au point et réécriture. ■

Reportage

Smartesting : agilité et l'esprit créatif

Editeur français spécialisé dans les outils de qualité logicielle, Smartesting fait travailler une trentaine de personnes dont le tiers de développeurs. Pour arriver à un niveau d'excellence dans le code des logiciels maison et dans la tenue des plannings de développement, les méthodes agiles ont été progressivement introduites pour aujourd'hui être l'alpha et l'oméga de l'organisation du développement interne.

Smartesting se concentre sur l'outillage des tests fonctionnels et l'industrialisation de ceux-ci depuis les besoins métiers jusqu'au cas de tests. Dans un système d'information de plus en plus complexe, ces tests sont vitaux pour assurer une bonne intégration des applications et garantir une non-régression à chaque nouveau développement. Le cœur stratégique de l'éditeur est le moteur de génération des tests.

La mise en œuvre d'un cycle de vie

Pour assurer le développement, l'intégration continue et la non-régression, tout en respectant les délais, l'équipe de développement de l'éditeur basé à Besançon met en œuvre des méthodes agiles mais aussi toute une batterie d'outils.

Pour le développement, un standard unique a été mis en place pour l'ensemble de l'équipe : Story Cards pour le planning, IntelliJ Idea pour l'édition de code, Junit et Fitnesse pour les tests, Hudson pour l'intégration continue et SVN pour la base de code.

Pour assurer la qualité et l'intégration continue, une ferme de 13 ordinateurs compile et teste l'application à chaque changement du code. Cela permet de valider le produit sur les différentes plateformes supportées ainsi que la connexion aux outils tiers. Les problèmes sont signalés aux développeurs par des lumières rouges. Ceux-ci doivent corriger le défaut le plus rapidement possible pour garder une version livrable.

Au sein de l'équipe, une approche XP a été mise en place. Le but est de développer rapidement et avec qualité, même si la paire de développeurs n'est pas homogène. Chaque membre du binôme peut apporter sa vision, son expertise. *« Avec le product manager, l'équipe est finalement très transverse »*, indique Olivier Albiez (expert agile). Aujourd'hui sur la partie Besançon, l'équipe de développeurs est d'environ une dizaine de personnes.

La salle peut se découper en trois zones : zone développeur, en cercle, une war room avec le planning, les fiches des développements et une zone de repos. L'organisation agile exige une très grande rigueur. Smartesting sort chaque année, en moyenne, 3 versions majeures et 8 versions mineures. *« Il faut avoir un focus et s'y tenir, sans se disperser »* explique André Dhondt (coach XP).

Ainsi pour la version prévue en septembre et en développement depuis plusieurs mois, les itérations se réalisent chaque semaine. Cependant, cette rigueur agile ne doit surtout pas tuer l'esprit créatif du développeur.

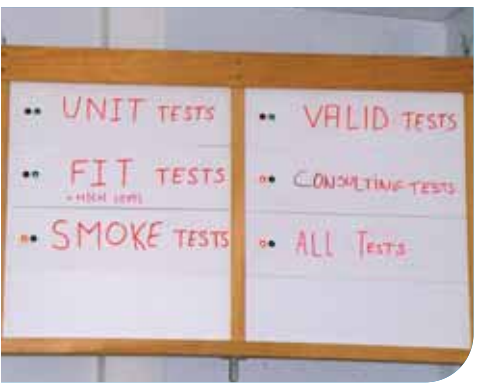
C'est pour cela que l'équipe a du temps de création, le vendredi, pour chercher, tester, développer une idée que le développeur pourra soumettre. C'est un aspect important dont il faut tenir compte dans tout projet agile : l'agilité impose un cadre, un mode de fonctionnement mais le product manager doit toujours veiller à garder la créativité. C'est ce qui fait la force d'un développeur et encore plus, d'une équipe. *« Lorsque nous avons mis en place Extreme Programming, ce fut un changement important (pour l'équipe). Aujourd'hui, nous cherchons le cœur des fonctions que l'on développe. C'est ce cœur qui apporte la véritable valeur »* poursuit André Dhondt.

La meilleure agilité est dans la pluralité agile.

Mais comme souvent quand on veut mettre en place une méthode agile, on ne prend pas une seule méthode mais plusieurs, l'objectif étant à chaque fois de prendre le plus pertinent pour le projet, l'équipe, le développeur. Prendre au pied de la lettre, une méthode agile n'a pas d'intérêt. Il faut savoir l'interpréter, la moduler. *« Nous nous sommes inspirés de différentes méthodes comme le Lean. Avec deux focus : aller à l'essentiel, apporter de la valeur. Ainsi, chaque jeudi, nous regardons tout ce qui s'est passé dans la semaine. On peut alors voir les tensions (et les résoudre) »,* explique Olivier Albiez.

Ce n'est donc pas un hasard si Smartesting utilise à Besançon aussi bien du Lean que du XP.

■ François **Tonic**



Le tableau agile : vive le papier

Le tableau de planification est découpé en trois rangées. Au-dessus on trouve la vision stratégique du produit, les features classés dans des enveloppes, au milieu, la vision tactique de l'itération courante, en bas, les activités créatives de l'équipe.

Les fiches blanches représentent les demandes du product managment, les rouges la non-qualité, les bleues les tâches administratives et les fiches jaunes les activités de recherche. Mais l'agilité ne doit pas faire oublier la technique, le code, même s'il faut délivrer le plus vite possible : *« l'équipe fait un point 5 à 6 fois par jour. Ces points concernent tout le monde et ils sont très courts. On arrête des fiches pour la journée »,* explique Olivier Albiez. Et son arrivée dans l'équipe a aussi permis de remettre à plat les horaires de travail. En général, les développeurs arrivent vers 8 – 9h, avec une pause déjeuner vers midi puis la journée s'arrête vers 17h. Et il n'y a quasiment aucun débord horaire. *« On ne peut pas lâcher sur les horaires »,* avertit Olivier. Ce n'est pas parce que les développeurs travailleront jusqu'à 20h, 22h que le code sera meilleur et que le projet avancera. Et Olivier a aussi modifié la manière de développer. Car contrairement à ce que l'on dit souvent, chez Smartesting, le code n'est pas commenté ! *« Le code est parfaitement lisible et se lit de lui-même »* nuance Olivier Albiez. C'est de cette clarté et de cette structure du code que découle aussi la qualité de celui-ci et le peu de bugs rencontrés.

L'agilité a permis de mettre en avant un objectif commun à l'ensemble de l'équipe. *« Un bon indicateur est le nombre de bugs et le temps que l'on dégage pour les résoudre. En 2009, nous étions à 25-28 bugs chez les clients, en 2010, nous en sommes à moins de 12 »,* précise Olivier Albiez. La correction de bugs constitue un travail à part entière qui nécessite de la réflexion et de les faire rentrer dans les processus d'agilité du projet.

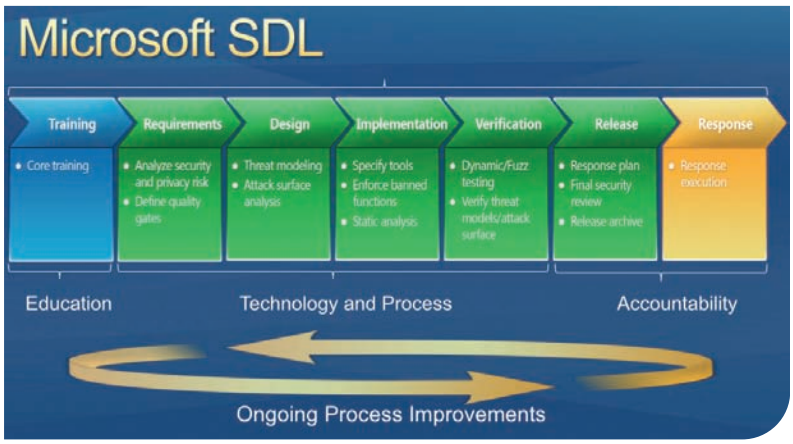
Eternel recommencement

L'agilité est un processus sans fin. En permanence, un build est réalisé automatiquement. Si un bug apparaît, il faut comprendre pourquoi et résoudre le problème. *« On se pose tout le temps des questions. Il faut tester de nouvelles approches. On regarde si cela fonctionne. Si ce n'est pas le cas, on revient en arrière. Il faut l'unanimité »,* précise André Dhondt. Et oui, l'agilité n'est pas figée. Il faut savoir l'adapter, la changer, l'améliorer.

Bien entendu, l'agilité ne supprime pas le stress, au contraire. Elle introduit le challenge quotidien : itération hebdomadaire, build permanent, horaires stricts. Ensuite, l'agilité fait partie des méthodes de management d'une équipe. Cela forme un tout et à Smartesting Besançon, on tente de trouver le cercle vertueux. Avec une certaine réussite !

SDL : la quête du développement sans faute

Initialement conçu pour épurer Windows et les logiciels maison de ses bugs, SDL s'est peu à peu transformé en méthode intégrée à une série d'outils standards, utilisable par les développeurs du monde entier.



L'initiative Trustworhty Computing est née en 2002. Cela fait donc déjà près de 8 ans que Microsoft affiche sa volonté de mieux sécuriser son environnement. Un véritable branle-bas de combat s'ensuivit qui, à l'époque, provoqua des remous dans le monde des systèmes d'exploitation : pour la première fois, une société commerciale d'envergure internationale voire mondiale allait suspendre, le temps qu'il fallait, la sortie de sa prochaine version, afin d'en améliorer la sécurité. Folie que tout cela ? Plutôt, instinct de survie. En effet, Microsoft n'aurait peut-être pas survécu à une nouvelle version si aucun effort n'avait été pratiqué en termes de sécurité dans le développement du logiciel. Internet, l'ouverture des systèmes d'information … il n'en a pas fallu beaucoup plus pour que l'on se rende compte que l'OS était devenu une véritable passoire face aux attaques, internes comme externes, qui se généralisaient et devenaient même de plus en plus « professionnelles ». Alors que faire ?

Etablir des procédures

La première d'entre elles a consisté tout naturellement à pousser toutes les équipes de développement vers des formations et puis rapidement à regarder et concevoir des modèles de vulnérabilité. Puis de réviser le code existant et rapidement engager des batteries de tests. Des tests qui entraînent la découverte de nouveaux défauts à corriger. Enfin de mettre en place les outils pour auditer. Cette nouvelle façon de procéder avant d'envisager la mise sur le marché d'une nouvelle version d'un logiciel s'avère rapidement payante et améliore notablement les problèmes de vulnérabilité dans le code. Très vite l'idée s'impose que seule l'intégration de la sécurité dans le cycle de développement d'un logiciel pourra améliorer la situation en diminuant la surface de risque. Le SDL, Secure Development Lifecycle est né. Nous sommes en 2004.

La santé sécuritaire en question

La firme de Seattle n'a cessé depuis d'améliorer sa « méthode » qui selon, Steven Lipner, Senior Director of Security Engineering Strategy, peut s'appliquer à n'importe quel logiciel qu'il soit ou non, d'origine microsoftienne. D'un point de vue pratique, c'est un cercle « vertueux » dont toutes les étapes permettent de s'assurer que le programme en cours de conception est bien sécurisé et ce, depuis sa conception. En phase finale, des tests récurrents permettent de remettre en question de façon quasi permanente la « santé » sécuritaire dudit logiciel et donc de l'améliorer dans le temps.

Pour commencer le cycle reproduit l'histoire qu'à vécu Microsoft : passage obligé à la formation en sécurité de toute l'équipe de développement. Cette étape dépassée, il reste 5 étapes à franchir dont la suivante est l'élaboration des requêtes en termes de sécurité. Là, on établit la liste des requêtes possibles en termes de sécurité et de confidentialité que requiert le projet. Ensuite, c'est le moment de désigner les experts sur lesquels l'on va s'appuyer pendant le cycle, un conseiller interne voire externe et le leader de l'équipe de développement avec qui il correspond entre autres. Des aides précieuses dont la première action sera de déterminer des critères de sécurité minimum. Une fois cela établi, il ne reste plus qu'à choisir des outils de suivi (bug, travail) puis tout de suite de spécifier les limites de qualité comme de bug à tolérer sans oublier l'inventaire de risques. Une fois ces bases établies, le SDL passe à l'étape suivante : la conception. Dès que les sous-tâches sont exécutées, les conseillers passent en revue les aspects Sécurité, Confidentialité et Chiffrement. En ce qui concerne la surface d'attaque potentielle, les couches de défense sont installées et les niveaux de privilège sont abaissés. Enfin les modèles de menaces : des menaces inventoriées via STRIDE.

L'étape suivante est l'implémentation,

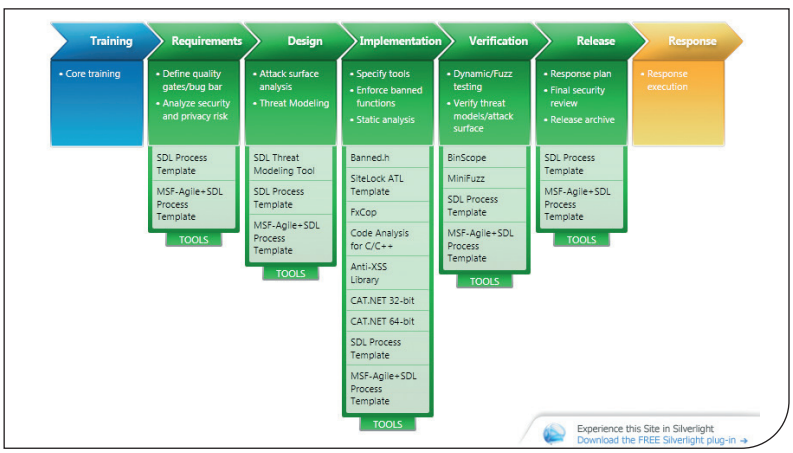
à savoir, connaître tous les outils, compilateurs, flags et options utilisés lors du développement. Puis là, reste à bannir toutes les fonctions et API non sécurisées et enfin procéder régulièrement à l'analyse statique du code généré.

Avant-dernière étape : la vérification. Ce sont essentiellement des tests de vérification des runtimes. Puis appliquer des procédures de recherche de vulnérabilités automatisées et systématisées, soit faire du fuzzing. Appliquer également des modèles de validation sur l'intégralité du code. Et pour terminer, pentester (réaliser des tests d'attaques) sur les composants critiques. Enfin le moment est venu de l'ultime étape : la release du programme. Le moment où l'on établit la documentation nécessaire en cas de problème. Le moment où également l'on passe en revue une dernière fois le programme sur les points de Sécurité et de Confidentialité. Sans oublier d'archiver toutes les données techniques pertinentes.

Arrivé au bout du cycle, l'on boucle sur la première étape pour maintenir un développement sécurisé dans le temps. Pour Microsoft, l'idée derrière tout cela reste d'avoir un maximum d'actions automatiques, comme utiliser des outils et des plates-formes automatisés. De ce fait, les formations se font également en ligne. Une automatisation qui n'amoindrit en rien l'efficacité de l'ensemble nous démontre Steven Lipner *« si l'on considère les vulnérabilités dans SQL Server, l'édition 2005 développée en s'appuyant sur les principes SDL, avait 91 % de moins de vulnérabilités que l'édition précédente développée de façon traditionnelle ».*

Des outils pour suivre les phases

Microsoft propose sa propre gamme d'outils pour aborder les différentes étapes proposées. Notamment celles d'attaque de surface et de fuzzing. En ce qui concerne la première, après avoir mesuré l'exposition potentielle d'un programme, pour pallier les faiblesses détectées, Microsoft propose non seulement des listes de binaires exécutables ou de services opérationnels existants mais également des listes d'interfaces réseau ou d'associations de fichiers. Pour alimenter l'outil de fuzzing, il propose aujourd'hui des fichiers mal formés pour tester les applications sous la forme de 100 000 itérations de tests avec en plus des jeux représentatifs de templates. Précisons que Microsoft utilise également les outils de fuzzing d'autres éditeurs avec lesquels il collabore comme Peach fuzzer et Codenomicon. En interne, Microsoft propose Threat Modeling Tool, Binscope, Minifuzz ou !Exploitable.



SDL, un standard en devenir ?

C'est une question que l'on peut se poser car Microsoft ne s'est pas contenté d'écrire un guide SDL applicable à ses produits et dans ses environnements (170 pages en version 5.0 aujourd'hui). Est également publié ce que l'on appelle le Simplified SDL. Comme son nom l'indique cela correspond à une édition simplifiée de l'original avec la particularité d'être adaptée à tous les environnements hormis celui de Microsoft. Selon Dave Ladd, Principal Security Program Manager Trustworthy Computing *« L'implémentation simplifiée du SDL guide les développeurs non « microsoftiens » pour les aider à adopter cette façon de faire. « Agnostique »* en termes de plate-forme, ce guide convient à toutes les tailles d'entreprise ». Mais Microsoft est loin d'être le seul à vouloir promouvoir sa façon de développer sécurisé tel Cisco avec le CSDL. Parmi eux également Adobe avec son SPLC, Secure Product Life-Cycle. Une façon pour l'éditeur de se manifester sur le terrain de la sécurité à un moment où le nombre de failles est tel que l'éditeur est montré constamment du doigt par tous les médias. Ce ne sont encore que les débuts pour SPLC face au mature SDL. Adobe a encore beaucoup à apprendre mais n'a pas encore misé comme Microsoft, il y a quelques années de cela, sur la sécurité en arrêtant du jour au lendemain la sortie de ses nouvelles versions afin de sécuriser l'actuelle. À suivre …

Automatisation du SDL avec Archer

Archer, racheté récemment par RSA est également un partenaire de Microsoft sur la partie SDL. Parmi les caractéristiques d'Archer, il propose des solutions pour l'entreprise dans les domaines de la Gouvernance, du Risque et de la Conformité, eGRC.

La société a, entre autres, créé une plate-forme qui permet d'organiser des processus automatisés, de contrôler les requêtes et d'exécuter des prioritisations d'enregistrements. Microsoft est au départ un client d'Archer car ses plates-formes permettent au géant de Seattle de respecter la conformité en vigueur. En insérant les prérogatives SDL au sein d'une plate-forme Archer, l'on crée une nouvelle source d'information qui s'additionne aux nombreuses d'ores et déjà proposées par l'éditeur. Grâce à cela, on obtient une plate-forme de gestion du code en cours de développement permettant d'appliquer SDL à la lettre sans rien oublier du processus.

■ Solange **Belkhayat-Fuchs**

Les builds privées avec TeamBuild 2010

Avec la version 2010 de Team Foundation Server, Microsoft a revu entièrement TeamBuild son outil de compilation : nouvelle architecture distribuée et nouveau moteur entièrement basé sur Workflow Foundation 4.0. Grâce à ces innovations, Microsoft propose de nouveaux types de builds dont la build privée.

Avant de voir en détail toutes les possibilités d'une build privée, faisons un bref retour sur la problématique à laquelle ce type de build répond et ce que proposaient les précédentes versions.

Lorsque l'on n'utilise pas de serveur de build, la compilation d'une application va se faire sur un poste de développeur et parfois selon le poste, on n'obtiendra pas le même résultat. En effet il pourrait arriver que, par exemple, les développeurs n'aient pas tous les mêmes versions des librairies tierces ou ne compilent pas l'application qui peut être composée de multiples composants dans le même ordre. Une solution pour éviter ce problème est de mettre en place un serveur de build avec une configuration et un script de build déterminés qui seront utilisés pour toutes compilations de l'application. Dans ce cas, une nouvelle problématique apparaît : comment le développeur peut-il générer une version de développement en s'assurant de suivre la même procédure que le serveur ? Dans les versions précédentes de TeamBuild, les scripts étaient basés sur MsBuild, il était donc possible de les lancer localement. Une propriété « IsDesktopBuild » permettait dans le script de faire la distinction entre une exécution sur un poste de développement et une exécution sur le serveur de build, certaines tâches, comme la pose d'un label par exemple, ne devant s'exécuter que sur le serveur. Si l'on considère que les tâches non exécutées sont spécifiques à TeamBuild, le problème de n'utiliser qu'un seul script est donc résolu. Il restait par contre encore le problème d'avoir des environnements identiques en termes de librairies tierces et d'applications tierces installées. Un développeur, pour ses besoins, pourrait au cours du développement utiliser une nouvelle librairie tierce qui ne serait pas installée sur le serveur de build. En local tout marcherait mais une fois ces modifications dans le contrôleur de source et compilées sur le serveur de build, la build sortirait en erreur. Afin de répondre à cette problématique, Microsoft propose dans TeamBuild 2010 les builds privées. Ce mécanisme permet aux développeurs de compiler leur code sur le serveur de build sans avoir à l'archiver avant !

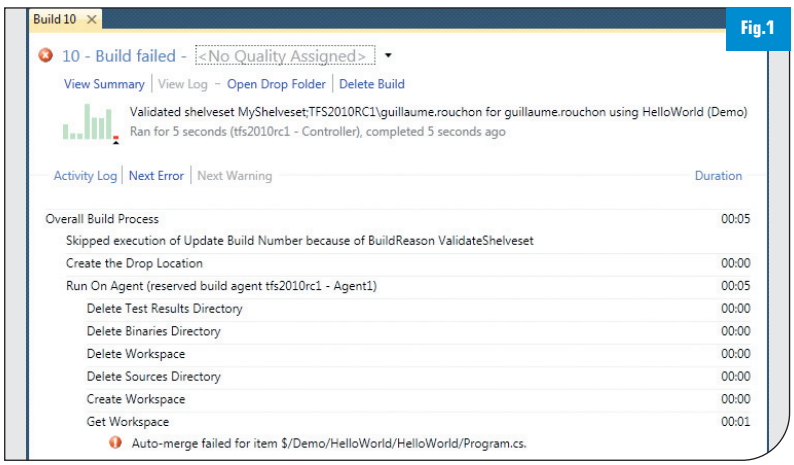


Fig.1

Les builds privées

Le but d'une build privée est, pour un développeur, de pouvoir valider les modifications qu'il a effectuées en utilisant le serveur de build, ceci afin d'utiliser le même processus et le même environnement que la build normale. Mais comment faire pour prendre en compte les modifications du développeur sans que celui-ci les intègre dans le contrôleur de sources ? Tout simplement en utilisant une des autres fonctionnalités du contrôleur de sources de TFS : les étagères. Lors du lancement manuel d'une build, quel que soit le type de build (manuel, intégration continue, ...), une nouvelle option est apparue avec TeamBuild 2010 permettant de spécifier si l'on veut compiler la dernière version dans le contrôleur de source ou la dernière version dans le contrôleur de source et le code se trouvant sur une étagère [Fig.1]. Le fait d'utiliser l'option permettant de spécifier une étagère fait que la build deviendra une build privée. Un développeur pourra choisir soit d'utiliser une étagère existante soit d'en créer une nouvelle. Dans ce dernier cas, l'interface de mise sur étagère sera affichée. Une fois l'étagère sélectionnée et la build lancée, lors de la synchronisation du workspace la dernière version des sources sera récupérée et fusionnée avec l'étagère spécifiée. Dans le cas où la fusion ne peut se faire automatiquement, la build sortira en erreur avec un message explicite [Fig.2]. Dans ce cas le développeur devra modifier le code de son étagère afin qu'il puisse être fusionné sans erreur avec la dernière version des sources.

Les builds privées permettant seulement la validation des modifications, il est important de pouvoir les distinguer facilement des autres types de build. Dans TeamBuild 2010, Microsoft a modifié l'interface graphique de suivi des builds et chaque type de build a maintenant sa propre icône [Fig.3]. Le résumé des builds privées

contient aussi un descriptif permettant de les reconnaitre facilement en affichant le type de build ainsi que le nom de l'étagère validée : « Validation de l'étagère MyShelveset ».

Comme pour les versions précédentes de TeamBuild, toutes les tâches de la build normale ne seront pas exécutées dans le cas d'une build privée :

- Aucun numéro de build ne sera généré ; les builds auront donc pour nom uniquement leur identifiant numérique.
- Aucun label ne sera apposé sur les sources.
- Il n'y aura pas d'indexation des sources ni de publication des symboles.
- La build ne sera ni associée à un changeset ni associée à des work items.
- Les binaires générés par la build ne seront pas déposés sur le partage réseau normal.

Toutes les autres tâches seront exécutées, notamment les tests unitaires, la publication de leurs résultats et les tests impactés.

Concernant les binaires générés, ils ne seront pas copiés dans le partage réseau normal (paramètre DropLocation) afin de ne pas les utiliser par erreur (par exemple, dans le cas où une application externe ferait des déploiements automatiques dès que de nouveaux fichiers sont déposés). Dans certains cas, il peut être utile d'avoir ces binaires afin d'exécuter d'autres tests comme de tester le déploiement de l'application via des MSI générés lors de la build. Le processus de build fourni par défaut par Microsoft permet de spécifier un partage réseau de dépose spécifique pour les build privées via le paramètre « PrivateDropLocation ». Si cette propriété n'est pas spécifiée, un message d'avertissement sera affiché dans le log de la build afin de préciser que les binaires ne seront pas disponibles. Afin de pouvoir simplifier le travail des développeurs souhaitant utiliser les builds privées pour valider leurs modifications uniquement via le processus de build et qui, si la build est réussie, vont intégrer leurs modifications dans le contrôleur de source, il est possible de spécifier au lancement de la build de faire un archivage automatique en cas de réussite [Fig.1]. Dans ce cas, la build change de type et devient un « Gated Check-in ». Attention, dans ce cas-là, la build n'étant plus privée, l'ensemble des tâches seront exécutées. De plus, une nouvelle tâche en fin de processus archivera les fichiers de l'étagère en cas de réussite en spécifiant le commentaire « ***NO_CI*** » afin de ne pas lancer inutilement une build de type intégration continue si une telle build existe.

Prise en compte de la build privée dans le workflow

Comme expliqué plus haut, certaines tâches ne sont pas exécutées dans le cas d'une build privée. Si vous souhaitez personnaliser votre processus de build en éditant le workflow associé, vous pouvez aussi créer des sections qui ne seront pas exécutées dans le cas d'une build privée. Pour cela Microsoft fournit une activité de workflow spécifique représentant une séquence d'activité sur laquelle on va pouvoir indiquer les types de build qui pourront exécuter la séquence : « InvokeForReason » [Fig.4]. La seule propriété de cette activité est « Reason » et peut prendre une ou plusieurs des valeurs suivantes :

- None
- All

Fig.2

Name	Build Definition
HelloWorld_20100430.1	HelloWorld
10	HelloWorld
9	HelloWorld
HelloWorld_20100429.6	HelloWorld
7	HelloWorld
HelloWorld_20100429.5	HelloWorld
HelloWorld_20100429.3	HelloWorld

- BatchedCI
- CheckInShelveset
- IndividualCI
- Manual
- Schedule
- ScheduleForce
- Triggered
- ValidateShelveset.

Une build privée aura pour type « ValidateShelveset », les autres valeurs représentant les autres types de build avec la valeur « Triggered » correspondant aux types « BatchedCI », « IndividualCI », « Manual », « Schedule », « ScheduleForce » et « CheckInShelveset ». C'est d'ailleurs cette activité qui est utilisée dans le workflow fourni par Microsoft.

Conclusion

Ce nouveau type de build est un gros ajout aux fonctionnalités fournies par TeamBuild 2010 ; elle devrait permettre de simplifier encore plus la validation et la génération des applications. Il est enfin possible de centraliser entièrement le processus de compilation, de versioning et de packaging, tout en fournissant une certaine souplesse aux développeurs pour valider leurs modifications en utilisant les outils et processus standard mis en place dans l'entreprise.

■ Guillaume **Rouchon**
Architecte .Net et expert ALM pour les Sogeti. Net Rangers
Blog : http://blog.quetza.net

Fig.4

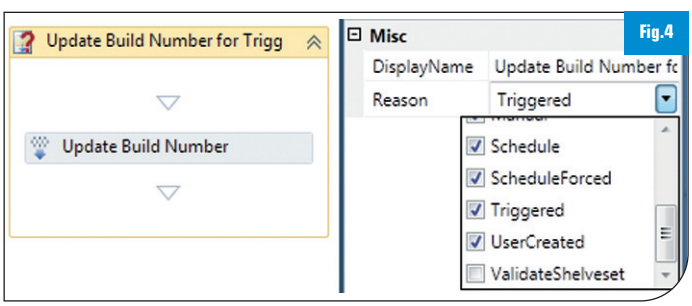
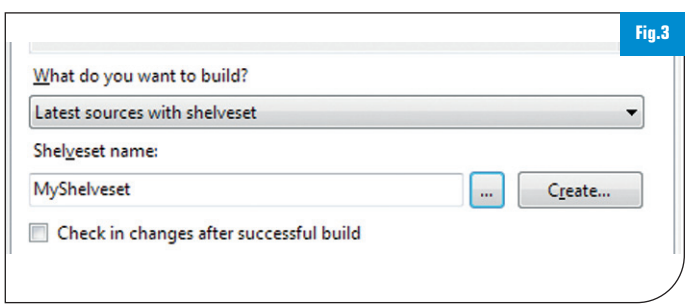


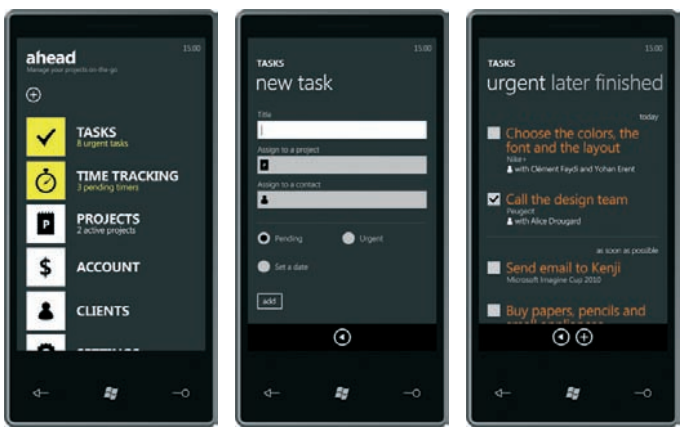
Fig.3



Ahead, retours d'expérience d'une collaboration designers/développeurs sur Windows Phone 7

Ahead est une application Windows Phone 7 permettant de gérer ses projets en mobilité. L'application s'articule autour de trois valeurs fondamentales, visées tout au long de la création du projet : Simplicité. Rapidité. Mobilité.

Cette application a initialement été conçue dans le cadre de l'édition 2010 du concours international Imagine Cup organisé par Microsoft. Très vite, l'équipe a pris conscience du potentiel de cette application et a décidé de poursuivre le développement quel que soit le résultat du concours. L'application n'a malheureusement pas reçu de prix mais une note assez élevée pour se positionner dans le top 10 mondial. L'application est découpée en cinq rubriques principales : « Tasks », « Time tracking », « Projects », « Account », « Clients ». Elle permet ainsi de gérer aisément ses tâches (et celles assignées à des collègues), son temps, ses dépenses et factures ainsi que son réseau de clients. Tout cela de n'importe où, sans avoir besoin de connexion internet.



L'équipe

L'équipe est née d'une collaboration entre deux développeurs (Julien Dollon & Bertrand Vergnault) et deux designers (Mickaël Denié et Clément Faydi), initiée lors d'une rencontre en Egypte au concours ImagineCup 2009. Suite au départ de Bertrand, l'équipe s'est dotée de deux développeurs : Jean-Christophe Vasselon et Thomas Laurent.

Une structure entrepreneuriale est en train de se former et il est à noter que le projet AHEAD a rejoint le programme Microsoft BizSpark Startup en ce début de Juin.

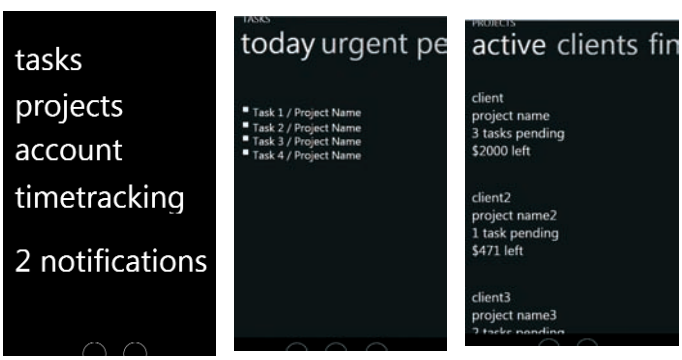
Windows Phone 7 et « Metro UI »

Microsoft a énormément travaillé pour reconstruire totalement son nouveau système d'exploitation mobile.

Au-delà des innovations technologiques conséquentes, l'équipe en charge du projet a également repensé l'expérience utilisateur du produit autour d'un nouveau langage graphique baptisé « Metro ». C'est donc en toute logique que l'application Ahead s'est construite autour de cette nouvelle vision, autant dans l'architecture que dans le design d'interface : beaucoup de « whitespace » pour respirer, des aplats de couleurs (pas de dégradés), une mise en avant de la typographie, pas d'artifices divers (ombres, reflets, reliefs…). Si vous souhaitez avoir plus de détails sur les guidelines de « Metro », le guide officiel « UI Design and Interaction » est disponible sur : http://download.microsoft.com/download/D/8/6/D869941E-455D-4882-A6B8-0DBCAA6AF2D4/UI Design and Interaction Guide for Windows Phone7 Series.pdf.

Conception et maquettage

Dès que le principal fut pensé sur papier, l'interface a été maquettée par les designers grâce à SketchFlow (voir images ci-dessous), un outil de prototypage rapide livré dans la suite Expression de la firme de Redmond. Pour Clément et Mickaël, cet outil était nouveau, mais ils ont très vite mesuré sa puissance : permettre à l'équipe entière de se rendre compte des erreurs d'architecture générale de l'application et de discuter de l'esthétique pour la suite. De plus, SketchFlow offre une vue panoramique sur la maquette grâce à SketchFlow Map. Cette étape a été essentielle dans la création du produit.



Développement

Étant donné que le développement de l'application Ahead est en Silverlight, l'équipe a travaillé constamment en vectoriel. Habitués à concevoir les interfaces sur Photoshop, les designers ont rapidement commencé à créer les différents écrans de l'interface sur Illustrator une fois le SketchFlow créé. Parallèlement, les développeurs ont travaillé sur une étude de faisabilité afin de vérifier si toutes les fonctionnalités prévues étaient possibles.

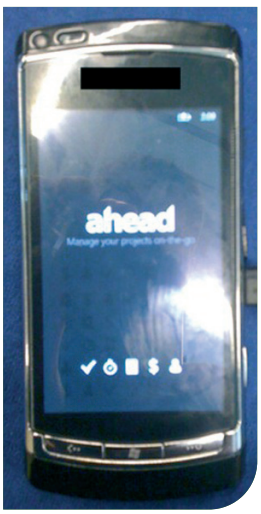
Dès lors que les fonctionnalités et la globalité du design de l'application ont été fixées, tous les écrans de l'application ont été livrés aux développeurs au format Blend. Le développement a alors démarré, les designers et développeurs travaillant main dans la main. Tous les jours, les designers révisaient l'avancée des développeurs afin que l'interface corresponde exactement aux maquettes. Après plusieurs allers-retours ainsi que de nombreux tests de débogage, l'application Ahead a vu enfin le jour.

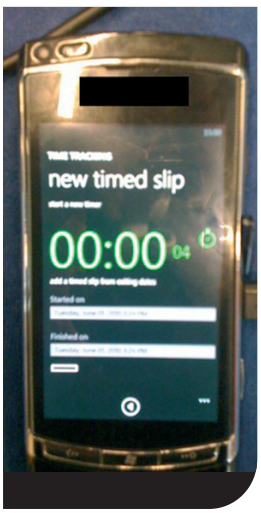
Gestion de projet

L'équipe a retenu la méthodologie SCRUM pour mener ce projet de manière Agile. Cela a été l'occasion d'utiliser la dernière mouture de Team Foundation Server et de son template «MSF for Agile», le travail du Product Owner, du Scrum Master et de l'équipe a été ainsi grandement facilité. Les produits de l'offre ALM de Microsoft, de Project Server, à TFS 2010 en passant par SharePoint 2010, SSRS ou tout simplement Excel 2010 ont permis une planification et un suivi performant.

Test sur un vrai téléphone

Pierre Cauchois (Microsoft) a donné l'occasion au projet Ahead d'utiliser un vrai téléphone. Après quelques petites adaptations à cause d'une version de l'OS différente, l'application a été totalement opérationnelle. L'émulateur permet certes de développer mais rien ne vaut un test grandeur nature. L'équipe croise les doigts pour faire partie des développeurs sélectionnés pour avoir un téléphone avant la sortie officielle et pouvoir être parmi les premiers à se positionner sur le Marketplace.





Utilisation de spécificités du téléphone

Depuis la dernière CTP de Windows Phone 7 (Avril), il est possible d'implémenter les **Tasks** qui permettent d'utiliser les fonctionnalités du téléphone comme « passer un appel » ou « écrire un texto ».
Pour commencer, il faut ajouter une référence vers **Microsoft.Phone.Tasks.dll** qui contient toutes les classes nécessaires. Si on prend l'exemple de la caméra, il faut créer une instance de type **CameraCaptureTask**, appeler sa méthode **Show** et abonner l'événement de la classe static **CameraCaptureTask**.

```
1.  internal void TakePhoto()
2.  {
3.    new CameraCaptureTask().Show();
4.    ChooserListener.ChooserCompleted += OnCompleted;
5.  }

1.  private BitmapImage currentImage = new BitmapImage();
2.  private void OnCompleted(object s, EventArgs e)
3.  {
4.    var task = (TaskEventArgs<PhotoResult>) e;
5.    if (task.TaskResult == TaskResult.OK)
6.    {
7.      var photo = task.Result.ChosenPhoto;
8.      currentImage.SetSource(photo);
9.    }
10.   ChooserListener.ChooserCompleted -= OnCompleted;
11. }
```

De la même manière on peut utiliser les classes suivantes:

- **EmailAddresseChooserTask** : Pour choisir une adresse email dans le répertoire du téléphone
- **WebBrowserTask** : Pour ouvrir un navigateur au lieu d'utiliser le contrôle WebBrowser
- **SmsComposeTask** : Pour créer un nouveau SMS
- **SavePhoneNumberTask** : Pour enregistrer un numéro de téléphone dans le répertoire
- **SearchTask** : Pour lancer une recherche
- **PhotoChooserTask** : Pour choisir une image dans le dossier Pictures du téléphone
- **SaveEmailAddressTask** : Pour sauvegarder un email dans le carnet d'adresses
- **PhoneNumberChooserTask** : Pour choisir un numéro de téléphone dans le répertoire
- **PhoneCallTask** : Pour démarrer un appel
- **MarketplaceLauncher** : Pour démarrer le Marketplace
- **MediaPlayerLauncher :** Pour démarrer la lecture de vidéo (ne fonctionne pour le moment qu'avec des flux HTTP)

Il manque encore certaines tâches comme la création d'un "Rendez-vous" dans Outlook ou l'ouverture de Bing Map. Les builds suivantes sont très attendues par les équipes de développeurs pour exploiter pleinement toutes les fonctionnalités.

Architecture MVVM sur Windows Phone 7

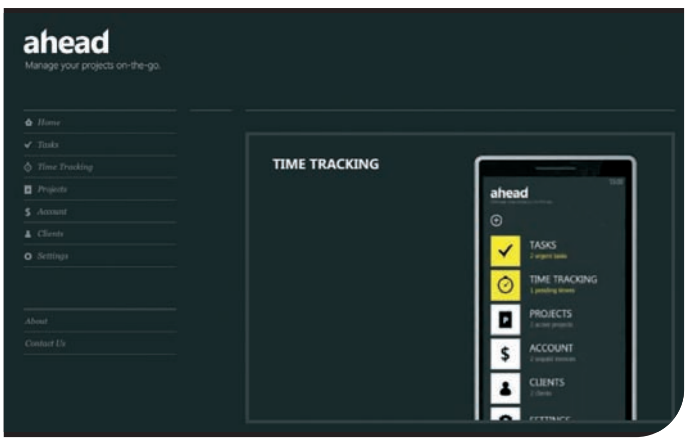
MVVM est l'architecture à favoriser dans les développements Silverlight et WPF. Cependant, il manque dans la version de Silverlight pour Windows Phone 7 les outils permettant d'exploiter cette architecture. Voici les classes indispensables que l'équipe AHEAD a réalisées et utilisées pour mettre MVVM en place sur leur projet :
DelegateCommand, pour pouvoir exécuter des actions dans les ViewModels.

```
1.  public class DelegateCommand : ICommand
2.  {
3.    private Predicate<object> canExecute;
4.    private Action<object> method;
5.    public event EventHandler CanExecuteChanged;
6.
7.    public DelegateCommand(Action<object> method)
8.      : this(method, null)
9.    {
10.   }
11.
12.   public DelegateCommand(Action<object> method, Predicate
    <object> canExecute)
13.   {
14.     method = method;
15.     canExecute = canExecute;
16.   }
17.
18.   public bool CanExecute(object parameter)
```

```
19.   {
20.     if (canExecute == null)
21.         return true;
22.
23.     return canExecute(parameter);
24.   }
25.
26.   public void Execute(object parameter)
27.   {
28.     method.Invoke(parameter);
29.   }
30. }
```

UIElementMouseLeft, pour pouvoir capter l'événement clic gauche sur un élément et l'associer à une action (DelegateCommand).

```
1.  public class UIElementMouseLeft
2.  {
3.    public static DependencyProperty MouseLeftProperty =
4.  DependencyProperty.RegisterAttached(
5.      «MouseLeft»,
6.      typeof(ICommand),
7.      typeof(UIElementMouseLeft),
8.      new PropertyMetadata(null, OnMouseLeft));
9.
10. private static void OnMouseLeft(DependencyObject obj, Dependency
    PropertyChangedEventArgs e)
11.   {
12.     UIElement elt = obj as UIElement;
13.     if ((e.NewValue != null))
14.       elt.MouseLeftButtonUp += new MouseButtonEventHandler
(button_MouseLeftButtonUp);
15. }
16.
17. static void button_MouseLeftButtonUp(object sender, Mouse
    ButtonEventArgs e)
18. {
19.     UIElement elt = sender as UIElement;
20.     ICommand command = elt.GetValue(MouseLeftProperty) as I
Command;
21.     command.Execute(e);
22. }
23.
```




```
24.   public static void SetMouseLeft(DependencyObject obj, I
Command value)
25.   {
26.     obj.SetValue(MouseLeftProperty, value);
27.   }
28.
29.   public static ICommand GetMouseLeft(DependencyObject obj)
30.   {
31.     return obj.GetValue(MouseLeftProperty) as ICommand;
32.   }
33. }
```

ViewModel, une classe abstraite qui implémente ce qui est nécessaire aux ViewModels.

```
1.  public abstract class ViewModel : INotifyPropertyChanged
2.  {
3.    public event PropertyChangedEventHandler PropertyChanged;
4.
5.    public void OnPropertyChanged(string propertyName)
6.    {
7.      if (PropertyChanged != null)
8.        PropertyChanged(this,  new  PropertyChangedEventArgs
    (propertyName));
9.    }
10.
11.   public event EventHandler<PageChangedEventArgs> PageChanged;
12.
13.   public void OnChangePage(string uri)
14.   {
15.     PageChanged(this, new PageChangedEventArgs(uri));
16.   }
17. }
```

Suite à ces étapes, nous pouvons facilement mettre en place un Locator enregistrant tous nos ViewsModels, les initialiser dans leur DataContext, capter des événements etc.…

Evolutions

Le prochain défi sera de mettre en place une architecture autour de Microsoft Prism de manière à avoir une application totalement composite. Ce qui permettra de développer beaucoup plus facilement de nouveaux modules mais permettra, au-delà, de mettre à disposition des versions multi-supports.

En effet, chaque module pourra s'ajouter sur l'application du téléphone mais pourra être potentiellement compatible avec une version Web ou tablet PC.

Le site web sera porté sur Azure pour faire face aux montées en charge et éviter de jongler avec différents serveurs. Cela se justifie par l'arrivée potentielle d'une version web de l'application et de la place centrale que prendra le site pour gérer les modules.

L'équipe prépare activement son arrivée sur le Marketplace. Le projet commence à se faire bien connaître au sein des sites spécialisés et des communautés Microsoft.

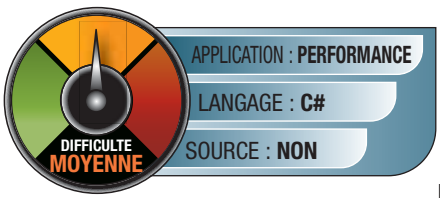
L'équipe AHEAD - http://www.ahead-app.com/ - http://twitter.com/ahead_app

■ Mickaël **Denié**, Julien **Dollon**, Clément **Faydi**, Jean-Christophe **Vasselon**, Thomas **Laurent.**

Accroissement de performances avec .NET Framework 4 Task Parallel Library et IMSL C#

La famille de Bibliothèques Numériques IMSL aide les développeurs à résoudre des problèmes complexes d'analyse numérique depuis 1970. Ce qui a commencé par une bibliothèque Fortran pour systèmes mainframe et supercalculateurs a évolué en même temps que l'industrie du logiciel. En passant par le support de C et Java, la bibliothèque IMSL C # est apparue en 2004 pour accompagner la communauté sans cesse croissante des développeurs .NET.

IMSL®
C# Numerical Library



Basée sur l'interface de programmation orientée objet développée pour la bibliothèque JMSL en Java, la première version de la bibliothèque IMSL C# a été bâtie sur .NET 1.1 et offre alors un large éventail de fonctionnalités mathématiques et statistiques. Bien que moins rapide que la version C, la version C # propose des performances tout à fait honorables pour développeurs .NET, sans aucun code natif empaqueté. Cependant, les développeurs dans le domaine de l'analyse numérique requièrent toujours plus de performance.

LES BONS OUTILS POUR LE MULTITHREAD

Avec la bibliothèque IMSL C 7.0, Visual Numerics a investi des efforts considérables pour ajouter des directives OpenMP à de nombreuses fonctions afin de profiter des architectures multi-cœur. En utilisant OpenMP pour améliorer la performance des algorithmes. Utiliser OpenMP dans ce but a été un choix naturel, car une grande partie de la logique reste la même (en effet, on n'a pas besoin de réécrire ou de réarchitecturer des codes robustes, existants et déjà pleinement testés), mais les boucles imbriquées pouvaient être analysées pour les calculs en mesure d'être décomposés sur des nœuds de calcul multiples. Microsoft a reconnu la facilité d'utilisation d'une approche de type OpenMP pour le Framework .NET et a commencé à travailler sur ce qu'on a alors appelé les « Extensions Parallèles ». Des versions bêta (« Community Technology Preview ») ont commencé à être réalisées en 2007. Ce travail a conduit à la Task Parallel Library (TPL), qui est intégrée dans le Framework .NET 4 et Visual Studio 2010. IMSL C# version 6.5 s'appuie sur la TPL pour paralléliser de multiples fonctions de la bibliothèque afin d'en améliorer les performances dans un environnement multi-cœur purement C#.

La série d'exemples qui suit, démontre l'impact sur les performances d'un code multithread (parallèle). La multiplication matricielle est une tâche qui se prête bien à l'illustration de l'écriture d'un code parallèle. L'algorithme naïf est une triple boucle relativement simple, et nous constatons que la parallélisation de la boucle la plus extérieure donne une amélioration significative des performances avec un effort minime. Dans cette série d'exemples, des matrices carrées seront utilisées pour des raisons de simplicité. La première méthode pour cette comparaison est la méthode de multiplication matricielle la plus simple. Cela peut être considéré comme une « approche naïve », et se présente sous la forme suivante :

```
private double[,] NaiveMultiply(double[,] a, double[,] b) {
  int n = a.GetLength(0);
  double[,] c = new double[n, n];

  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      c[i, j] = 0;
      for (int k = 0; k < n; k++)
        c[i, j] += a[i, k] * b[k, j];
    }
  }
  return c;
}
```

Ce code est acceptable pour des problèmes de taille raisonnable. Toutefois, pour des problèmes de taille importante, des algorithmes plus efficaces doivent être considérés. Ce qui suit est une méthode standard de « multiplication par blocs », méthode qui découpe le problème en petits morceaux.

```
private double[,] BlockMultiply(double[,] a, double[,] b)
{
  int n = a.GetLength(0);
  double[,] c = new double[n, n];
  int bBegin, bEnd, blockSize = 32;

    bBegin = 0;
  while (bBegin < n) {
        bEnd = bBegin + blockSize;
      bEnd = Math.Min(bEnd, n);

    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        for (int k = bBegin; k < bEnd; k++)
```

```
            c[i, j] += (a[i, k] * b[k, j]);
          }
        }
          bBegin = bEnd;
      }
      return c;
    }
```

La taille du bloc est configurable et est souvent une puissance de 2, choisie pour des raisons de taille du cache. En utilisant .NET toutefois, le matériel fait abstraction de ces préoccupations bas-niveau. Une valeur de 32 fonctionne bien et est utilisée dans ces exemples, bien qu'il n'y ait pas une forte dépendance sur cette valeur. Pour les problèmes de taille inférieure à 500x500, les méthodes « naïve » et « multiplication par blocs » donnent des temps d'exécution comparables. Pour les très petits problèmes, la méthode « multiplication par blocs » peut en fait être plus lente ; cependant, pour le problème de taille 1000 chronométré ci-dessous, la méthode « multiplication par blocs » révèle son efficacité.

L'USAGE DE .NET 4

Les deux algorithmes précédents s'exécutent en série, c'est-à-dire qu'ils s'exécutent sur un seul thread, sur un cœur unique, quel que soit le dimensionnement de l'architecture disponible. Avec la sortie du framework .NET 4, la TPL peut être utilisée pour paralléliser facilement ces deux algorithmes. Le concept est simple - remplacer la boucle « for » la plus extérieure par un appel à la nouvelle instruction « Parallel.For ». La syntaxe est différente parce qu'il s'agit davantage d'un appel à une méthode plutôt que d'un mot réservé ; mais pour ce cas, la correspondance entre les deux est évidente :

```
for (int i=0; i<n; i++) { }
```

devient

```
Parallel.For(0, n, (int i) => { });
```

La syntaxe lambda peut sembler étrange pour ceux qui ne sont pas habitués à ces expressions, mais les éléments sont très semblables à la déclaration standard « for ». L'entier « i » sera incrémenté de 0 (inclus) à n (exclus) et les instructions entre les accolades seront morcelées et exécutées en parallèle. Par défaut, le framework utilisera pour ce calcul tous les cœurs disponibles. Évidemment, il est des cas où un développeur peut souhaiter limiter explicitement le nombre de cœurs affectés, un paramètre optionnel peut être ajouté à cet effet. Ce paramètre est une instance d'un objet ParallelOptions qui est incorporé à l'appel de méthode. La classe ParallelOptions a une propriété MaxDegreeOfParallelism, qui par défaut est fixée à 1 (signifiant : utiliser tout ce qui est disponible »). Dans les tests ici, il est nécessaire de limiter le nombre de threads pour évaluer le rendement. En utilisant l'objet ParallelOptions en conjonction avec l'appel à la méthode Parallel.For, la manière d'effectuer la multiplication matricielle par la méthode « naïve » devient :

```
private double[,] ThreadedNaiveMultiply(double[,] a, double
[,] b, int nThreads)
{
  ParallelOptions po = new ParallelOptions();
      po.MaxDegreeOfParallelism = nThreads;
  int n = a.GetLength(0);
  double[,] c = new double[n, n];

  Parallel.For(0, n, po, (int i) =>
  {
    for (int j = 0; j < n; j++)
    {
      c[i, j] = 0;
      for (int k = 0; k < n; k++)
        c[i, j] += a[i, k] * b[k, j];
    }
  });
  return c;
}
```

La méthode « multiplication par blocs » peut être modifiée de manière similaire, en changeant le « for » de la boucle extérieure en « Parallel.For() ».

La bibliothèque IMSL C # version 6.5 inclut des méthodes qui utilisent la TPL en interne, la méthode Matrix.Multiply() en est un exemple. Cette méthode utilise un algorithme plus complexe pour la multiplication matricielle threadée, et les performances sont améliorées au-delà des exemples d'algorithmes présentés ci-dessus. L'utilisation de cette bibliothèque est simple ; le code permettant de multiplier deux matrices rectangulaires générales en utilisant un nombre donné de threads est tout simplement :

```
double[,] result = Imsl.Math.Matrix.Multiply(a, b, nThreads);
```

Les benchmarks ci-dessous ont été effectués sur un serveur Windows Server 2008 Enterprise SP2 64-bit avec Visual Studio 2010 RC. La machine est un biprocesseur Xeon 5140 dual-core à 2.33 GHz et 4Go de RAM.

	Méthode Naïve	Méthode par Blocs	IMSL C# 6.5
1 thread	17.84	11.45	7.56
2 threads	9.05	6.10	3.79
4 threads	4.52	3.06	1.90

Tableau 1. Temps (secondes) pour multiplier deux matrices 1000x1000.

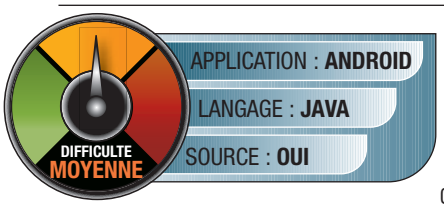
La vitesse de l'algorithme se révèle quasi-linéaire en fonction du nombre de cœurs utilisés. En d'autres termes, le test utilisant les quatre cœurs se montre presque quatre fois plus rapide que le test effectué par un seul thread. Notons que cet excellent gain a été obtenu en ne changeant littéralement qu'une ligne de code.

Dans cet exemple, seule la nouvelle méthode Parallel.For a été utilisée, et déjà des améliorations de performances significatives ont été observées sur un serveur multi-cœur. La TPL contient de nombreuses fonctionnalités supplémentaires, notamment une méthode Parallel.ForEach, une classe ParallelLoopState (pour contrôler les variables locales et partagées), une classe Interlocked (pour les opérations thread-safe atomiques telles que l'incrémentation d'un compteur), et bien plus encore. En continuant à mettre l'accent sur la productivité des programmeurs, la nouvelle TPL au sein du Framework .NET 4 offre d'excellentes prestations en termes de performances, sans pour autant introduire un système complexe de mise en œuvre.

■ Ed **Stewart**

La géolocalisation avec Android

Qui n'a jamais utilisé l'incontournable Google Maps ? Souvenez-vous de votre dernier voyage où vous l'avez sollicité pour trouver l'itinéraire vers votre hôtel, pour voir comment il est situé avec la fonctionnalité View Street, ou encore pour vérifier s'il y a une piscine sur son toit avec la vue satellite. Dans cet article, je vous propose de découvrir l'utilisation des services Google Maps au sein d'une application Android que nous allons construire au fur et à mesure. Laissez-vous emporter par l'univers du petit robot vert, vous ne serez pas au bout de vos surprises !



Avant de se lancer dans le code, je vous propose de valider les deux phases préliminaires : l'acquisition d'une clé pour l'utilisation de l'API Google Maps et la configuration de l'environnement de développement. Il n'est pas indispensable de disposer d'un terminal Android, l'émulateur suffit. En revanche, armez-vous de patience pour les deux phases préliminaires.

LA CLÉ TU TROUVERAS...

Afin de pouvoir utiliser l'API Google Maps, nous devons obtenir la bénédiction de son créateur, le tout puissant Google. Cette autorisation se matérialise par une clé sous la forme d'une chaîne de caractères, qui sera présentée d'une manière systématique lors de la communication avec les services web de l'API. Ainsi, Google pourra savoir par qui il est sollicité, et surtout distinguer les licences gratuites de celles qui sont payantes.

Pour les applications web, c'est le nom de domaine dans lequel l'application est hébergée qui servira d'identifiant. Pour une application Android qui ne dispose pas de nom de domaine, Google se base sur le certificat avec lequel l'application a été signée.

Le certificat ? Pas de panique pour ceux qui n'ont pas encore installé leur application « *HelloWord* » sur un terminal Android. En effet, toute application Android doit être signée pour pouvoir l'installer sur le terminal. Avec Eclipse, cela se résume à exporter un package apk signé. Notez que l'exécution de l'application en cours de développement sur l'émulateur n'échappe pas à la règle, le plugin ADT signe automatiquement le package apk avec un certificat de debug avant de l'installer sur l'émulateur.

Ce certificat de debug est généré d'une manière transparente avec le SDK Android et se trouve sous [*dossier-utilisateur*]*/.android/debug.keystore* avec un alias et un mot de passe respectivement *androiddebugkey* et *android*.

Pour plus d'informations sur la signature des applications, vous pouvez consulter ce lien : http://developer.android.com/guide

Revenons à l'acquisition de la clé, nous aurons besoin de l'empreinte numérique en MD5 du fichier debug.keystore. Pour récupérer cette empreinte, il faudrait utiliser la commande ci-dessous en utilisant android comme mot de passe de la keystore. Rendez-vous ensuite sur cette page afin de générer votre propre clé : http://code.google.com/android/maps-api-signup.html

```
$ keytool -list -keystore ~/.android/debug.keystore -storepass
android...
```

```
Certificate fingerprint (MD5):
            16:5E:41:46:2E:A8:2B:FF:1D:F5:51:7F:96:73:C8:28
```

Dans mon cas, la clé qui a été générée est *0GyBhuGyluoWTUA_8q UqCAHSqRNsAqBtm-4GGow*, la vôtre est certainement différente. Gardez-la de côté, elle vous sera certainement utile. Notez tout de même que cette clé sera remplacée par [*myGoogleMapsKey*] dans la suite de l'article.

Nous sommes maintenant en possession d'une clé permettant d'accéder aux services Google Maps. Mais attention ! elle n'est compatible qu'avec des applications signées par votre certificat de debug. Autrement dit, si vous voulez installer votre application sur un terminal, le package apk devrait être signé avec un autre certificat. Par conséquent, vous devez demander une autre clé qui remplacera la clé de debug.

TON ENVIRONNEMENT TU PRÉPARERAS...

Comme la majorité d'entre vous, j'utilise l'IDE Eclipse, couplé au plugin ADT dont l'installation a déjà été abordée dans un numéro précédent du magazine. Sachez qu'une installation minimale ne suffit pas pour pouvoir utiliser les fonctionnalités de Google Maps, l'Add-On « *Google APIs* » devrait en effet être installé. Pour cela, allez dans « *Window > Android SDK and AVD Manager* » puis sélectionnez à gauche « *Installed Packages* ».

Il faudrait ensuite vérifier que « *Google APIs* » figure bien dans la liste des packages. Si ce n'est pas le cas, ou si vous souhaitez installer sa dernière version, sélectionnez à gauche « *Available Packages* » et procédez à son installation en prenant soin d'inclure ses dépendances requises.

Dans mon cas, je suis passé au niveau 8 de l'API Android en instal-

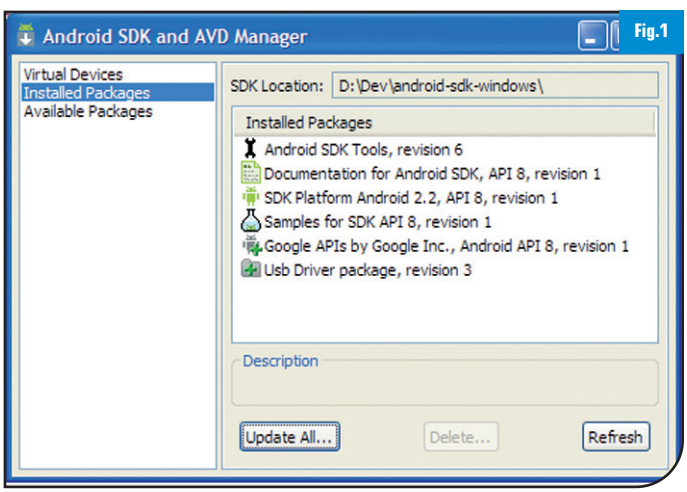


Fig.1

lant « *SDK Platform Android 2.2* » et son package « *Google APIs* » correspondant. [Fig.1]

Cet Add-On permet d'intégrer la librairie Google Maps aux projets Eclipse. Pour créer un nouveau projet, choisissez « *Android* » comme type de projet dans l'assistant de création de projet. Cochez ensuite « *Google APIs* » comme « *Build Target* ». Pour un projet existant, vous pouvez changer son « *Build Target* » dans les propriétés du projet. Une dernière configuration, celle de l'émulateur, est indispensable avant de commencer l'écriture du code. Pour pouvoir exécuter et déboguer dans l'émulateur une application utilisant l'Add-On « *Google APIs* », un AVD (Android Virtual Device) doit être créé et configuré avec le target « *Google APIs* ». Pour cela, allez dans « *Window > Android SDK and AVD Manager* » puis sélectionnez à gauche « *Virtual Devices* » et créez un nouvel AVD. Prenez soin de lui attribuer un nom significatif pour bien le distinguer. Dans mon cas je l'ai nommé android_*22_maps*. Avant de valider la création, assurez-vous de bien sélectionner le target « *Google APIs* ». [Fig.2]

PARIS TU SURVOLERAS...

Vous étiez patients pour les étapes préliminaires et vous ne serez pas déçus pour le reste. Désormais, tout est prêt pour entamer le code. Commençons par créer un nouveau projet Android avec le nom ProgrammezMap et une activité principale *ProgrammezMapActivity*. Le « *Build Target* » du projet devrait être « *Google APIs* ». [Fig.3]

Un projet de type « *HelloWord* » nous a été généré, mais sa configuration est encore insuffisante par rapport à nos besoins. Nous devons éditer le fichier *AndroidManifest.xml* en ajoutant la librairie Google Maps et en autorisant l'application à accéder au réseau Internet et aux données GPS.

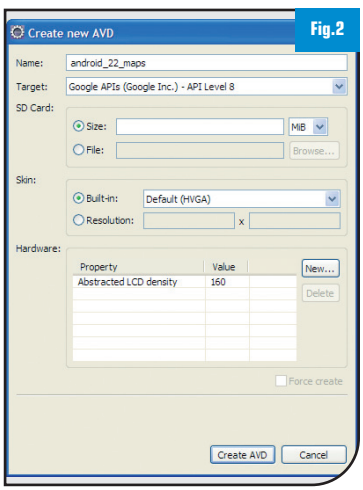
```
<?xml version=»1.0» encoding=»utf-8»?>
<manifest xmlns:android=»http://schemas.android.com/apk/res/android»
  package=»com.programmez.android.map» android:versionCode=»1»
  android:versionName=»1.0»>
  <application android:icon=»@drawable/icon»
android:label=»@string/app_name»>
    <uses-library android:name=»com.google.android.maps» />
    <activity android:name=».ProgrammezMapActivity»
      android:label=»@string/app_name»>
      <intent-filter>
        <action android:name=»android.intent.action.MAIN» />
        <category android:name=»android.intent.category.LAUNCHER» />
      </intent-filter>
    </activity>
  </application>
  <uses-permission android:name=»android.permission.INTERNET» />
  <uses-permission android:name=»android.permission.ACCESS_FINE
_LOCATION»/>
</manifest>
```

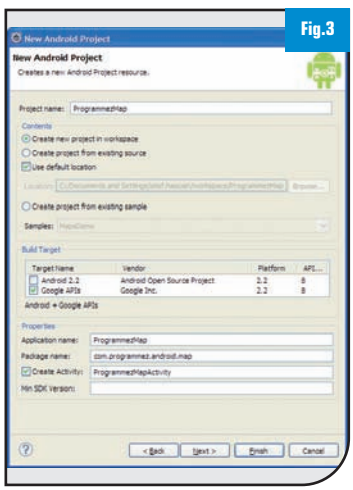
Ajoutons maintenant la carte en la paramétrant dans le fichier de layout *ProgrammezMap/res/layout/main.xml*. La librairie Google Maps d'Android nous offre une classe prête à l'emploi de type Vue qui permet de visualiser et d'explorer une carte. Son paramétrage requiert principalement la clé générée par Google au début de l'article.

```
<?xml version=»1.0» encoding=»utf-8»?>
<com.google.android.maps.MapView
  xmlns:android=»http://schemas.android.com/apk/res/android»
android:id=»@+id/mapview»
  android:layout_width=»fill_parent»  android:layout_height=
»fill_parent»
  android:apiKey=»[MyGoogleMapsKey]»
  android:clickable=»true» />
```

Il faudrait éditer le fichier *ProgrammezMapActivity.java* afin d'afficher la carte. La classe mère de notre activité devrait être *MapActivity* qui nous permettra d'interagir avec la *MapView* précédemment paramétrée. Notez que l'implémentation de la méthode abstraite *isRouteDisplayed()* est nécessaire.

```
package com.programmez.android.map;

import android.os.Bundle;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;

public class ProgrammezMapActivity extends MapActivity {

  private MapView mapView;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    initMapView();
    goToParis();
  }

  private void initMapView() {
    mapView = (MapView) findViewById(R.id.mapview);
    mapView.setBuiltInZoomControls(true);
  }

  private void goToParis() {
    GeoPoint parisPoint = new GeoPoint(48856668, 2350988);
    mapView.getController().animateTo(parisPoint);
    mapView.getController().setZoom(12);
  }
```





```
  @Override
  protected boolean isRouteDisplayed() {
    return false;
  }

}
```

Par souci de clarté, nous découpons par rôle des portions de code dans des méthodes privées. Dans la méthode *initMapView()* nous récupérons la *MapVue* configurée dans le fichier *main.xml* et nous lui ajoutons les contrôles de zoom.
La *MapView* délègue la navigation dans la carte à un contrôleur de type *MapController* qui lui est associé. Via ce contrôleur, nous pouvons centrer la carte en un point de type *GeoPoint* et effectuer un zoom de 1 à 21 niveaux.
Le niveau 1 étant le niveau le plus détaillé. Dans la méthode *goToParis()*, nous centrons la carte sur les coordonnées longitudinales de Paris avec un zoom de niveau 12. Vous pouvez remarquer que les coordonnées passées au constructeur du *GeoPoint* sont multipliées par 106. La classe *GeoPoint* gère en effet des entiers et exige la multiplication de la latitude et de la longitude par 1E6. Il est temps de lancer l'exécution de l'application avec « *Run As > Android Application* ». Notez que si vous démarrez le AVD pour la première fois, vous risquez d'attendre quelques minutes. Et voilà… vous survolez Paris ! [Fig.4]

TA POSITION TU AFFICHERAS…

La position GPS de l'émulateur est paramétrable via Eclipse. Dans la perspective « DDMS », puis dans la vue « Emulator Control », vous trouverez la zone « Location Controls » qui est dédiée à cela. Malheureusement, un bug bien connu et qui n'est pas encore résolu nous empêche d'utiliser cet utilitaire avec des options régionales paramétrées en français. Pour remédier à ce bug, il va falloir éditer le fichier eclipse.ini en ajoutant l'option -Duser.language=en et redémarrer ensuite Eclipse.
Comme indiqué dans [Fig.5], il est possible de définir la longitude et la latitude souhaitées. Dans mon cas, j'ai opté pour une longitude de 2.297971 et une latitude de 48.870545, des coordonnées qui pointent quelque part dans le 8e arrondissement de Paris. Une petite astuce pour récupérer les coordonnées d'un endroit : dans l'application web de Google Maps, il suffit de cliquer avec le bouton droit sur l'endroit choisi. L'option « Plus d'infos sur cet endroit » affichera les coordonnées de cet endroit dans la zone de recherche.
Notre émulateur possède maintenant une position GPS que nous allons marquer. Pour dessiner sur la carte, nous pouvons utiliser des objets de superposition personnalisables, appelés les « *Overlays* », caractérisés par une latitude et une longitude que nous pouvons marquer grâce à des images ou du texte.
La librairie Android pour Google Maps met à notre disposition un Overlay intitulé *MyLocationOverlay* dont les coordonnées sont liées à la position actuelle du GPS du terminal. Un simple ajout de cet Overlay à la MapView marquera cette position sur la carte.
Reprenons notre activité et ajoutons la méthode privée *addMyLocation()* pour dessiner le *MyLocationOverlay*.

```
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    initMapView();
    addMyLocation();
  }

  private void addMyLocation() {
    final MyLocationOverlay myLocationOverlay = new MyLocation
Overlay(this, mapView);
    mapView.getOverlays().add(myLocationOverlay);
    myLocationOverlay.enableMyLocation();
    myLocationOverlay.runOnFirstFix(new Runnable() {
      public void run() {
        mapView.getController().animateTo(
            myLocationOverlay.getMyLocation());
        mapView.getController().setZoom(16);
      }
    });
  }
```

Notez que la méthode *enableMyLocation()* de cet Overlay permet d'activer le suivi de la position GPS, et que la classe anonyme de type *Runnable* passée à la méthode *runOnFirstFix()* permet de déplacer la carte à la position GPS et de faire un zoom au niveau 16. Relancez l'exécution, et si vous avez utilisé les mêmes coordonnées que moi, vous verrez que vous n'êtes pas loin de l'arc de triomphe ! [Fig.6]

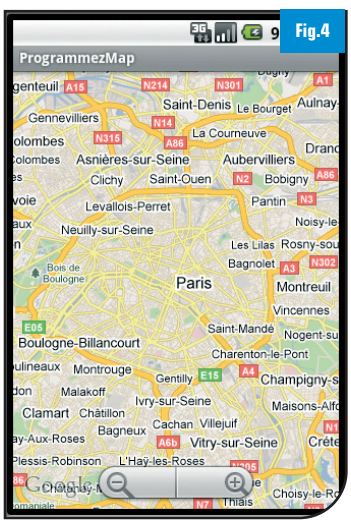


Fig.4

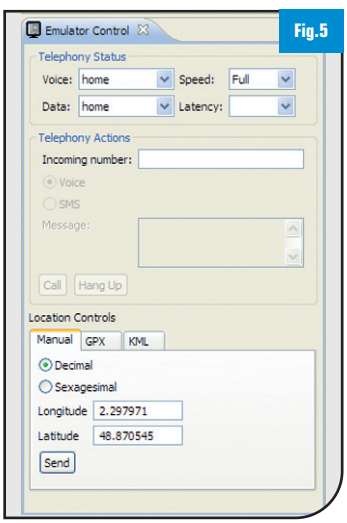


Fig.5

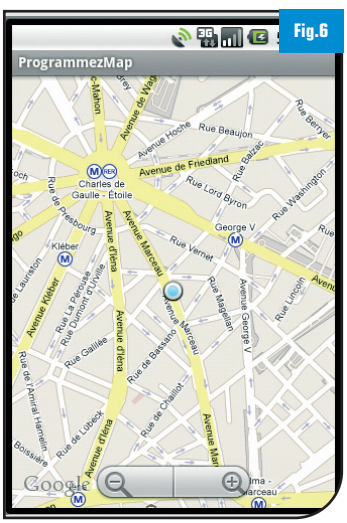


Fig.6

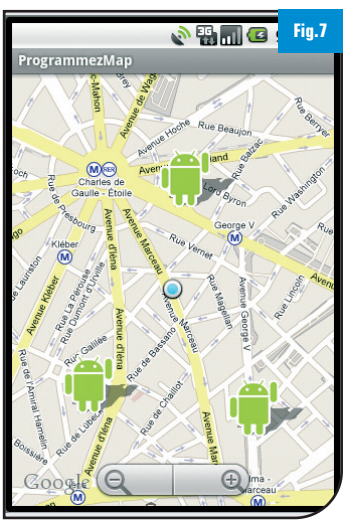


Fig.7

SUR LA CARTE TU DESSINERAS…

Dans une application géolocalisée, vous aurez probablement besoin de marquer des endroits sur la carte. Pour notre article, je vous propose de marquer la liste des points renvoyés par la méthode lookForRobots().

```
private List<GeoPoint> lookForRobots() {
  ArrayList<GeoPoint> robotLocations = new ArrayList<GeoPoint>();
  robotLocations.add(new GeoPoint(48866465, 2301400));
  robotLocations.add(new GeoPoint(48872887, 2298481));
  robotLocations.add(new GeoPoint(48867100, 2294383));
  return robotLocations;
}
```

Personnalisons notre propre Overlay pour marquer l'emplacement de ces points. La librairie Google Maps d'Android met à notre disposition la classe *ItemizedOverlay* qui permet de gérer un ensemble d'Overlays de même type. Pour personnaliser cette classe, nous devons l'étendre et implémenter ses deux méthodes : size() pour renvoyer le nombre d'Overlays gérés et *createItem(int i)* pour renvoyer l'*OverlayItem* à l'index *i*.
La classe *RobotsOverlay* décrite ci-dessous est une implémentation de *ItemizedOverlay*. Le constructeur de cette classe requiert un objet de type *Drawable* qui fera office de marqueur. La méthode *addRobot(GeoPoint point)* se chargera de l'ajout d'un nouveau point.

```
package com.programmez.android.map;

import java.util.ArrayList;

import android.graphics.drawable.Drawable;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;

public class RobotsOverlay extends ItemizedOverlay {

 ArrayList<OverlayItem> overlayList = new ArrayList<OverlayItem>();

 public RobotsOverlay(Drawable defaultMarker) {
   super(boundCenterBottom(defaultMarker));
 }

 @Override
 protected OverlayItem createItem(int i) {
   return overlayList.get(i);
 }

 @Override
 public int size() {
   return overlayList.size();
 }

 public void addRobot(GeoPoint geoPoint) {
   overlayList.add(new OverlayItem(geoPoint,»»,»»));
   populate();
 }

}
```

Revenons à l'activité *ProgrammezActivity* pour lui ajouter le marquage des points. Pour cela, ajoutons la méthode *drawGreenRobots()* tel qu'indiqué dans le code ci-dessous.

```
@Override
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   initMapView();
   addMyLocation();
   drawGreenRobots();
 }


 private void drawGreenRobots() {
   Drawable androidmarker = this.getResources().getDrawable(
       R.drawable.androidmarker);
   RobotsOverlay robotsOverlay = new RobotsOverlay(androidmarker);
   List<GeoPoint> robots = lookForRobots();
   for (GeoPoint geoPoint : robots) {
     robotsOverlay.addRobot(geoPoint);
   }
   mapView.getOverlays().add(robotsOverlay);
 }
```

Nous devons passer en paramètre du constructeur de *RobotsOverlay* un objet de type *Drawable*. Pour cela, j'ai choisi comme marqueur une image illustrant le petit robot faisant un coucou. Choisissez votre propre image devant avoir la taille d'une icône et mettez-la dans le répertoire *res/drawable*. Le nom du fichier de l'image que j'ai choisie est *androidmarker.png*. Ainsi, l'identifiant relatif à cette ressource est *R.drawable.androidmarker*.
Une fois notre objet de type *RobotsOverlay* instancié, nous lui indiquons les positions des robots récupérées depuis la méthode *lookForRobots()*. Nous l'ajoutons enfin à la *MapView*.
Relancez l'exécution de l'application. Si vous avez utilisé les mêmes valeurs que moi, vous verrez vos trois marqueurs éparpillés autour de la position GPS, dont un « aux… Champs-Élysées ! ». [Fig.7]

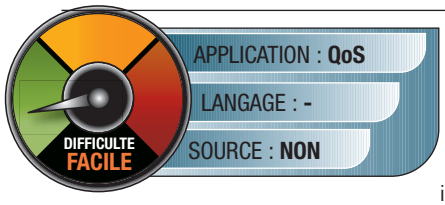
CONCLUSION

Pour aller plus loin dans votre découverte et pour apporter plus de richesse au contenu de votre future application géolocalisée, je vous invite vivement à regarder de plus près les différentes possibilités de dessin sur la carte via les Overlays. Je vous invite également à découvrir la technique de géocodage via la classe Geocoder qui permet de traduire des adresses en coordonnés longitudinales et vice versa. Cette technique est en effet utile pour situer une adresse sur la carte ou transformer une latitude et une longitude en une adresse textuelle. Maintenant… à vous de jouer !

■ Atef **Haouari**
Consultant Valtech Technology

Un code de qualité est un code S.O.L.I.D.(E) !

Rigidité, fragilité, immobilité… Vous rencontrez ces obstacles dans votre code au quotidien ? Vous rêvez de développer un code robuste, extensible et réutilisable. Un mirage ? Non ! Les principes S.O.L.I.D. ! Ces principes vous sont expliqués et leur mise en application vous facilitera la vie.



Après avoir vu de nombreux projets, une évidence s'impose : les principes du développement objet sont souvent mal appliqués ou ignorés par beaucoup de développeurs, limitant l'évolutivité et la maintenance des applications. Bien que les bases de la programmation orientée objet soient simples, il y a souvent des problèmes dans leur application entraînant des bugs, une mauvaise structuration, un code non maintenable, une exécution longue, une compréhension difficile.

Les principes S.O.L.I.D., indépendants du langage, sont des principes permettant un meilleur design des classes dans la programmation orientée objet. Correctement appliqués, ces principes vont vous permettre de réduire les erreurs de conception et de structure amenant aux problèmes précédents.

S.O.L.I.D se compose de 5 principes :

- Single Responsibility Principle (Principe de la responsabilité unique)
- Open-Closed Principle (Principe Ouvert-Fermé)
- Liskov Substitution Principle (Principe de substitution de Liskov)
- Interface Segregation Principle (Principe de ségrégation des interfaces)
- Dependency Inversion Principle (Principe d'inversion de dépendance)

Le but est donc d'avoir un code de qualité grâce à un ensemble de bonnes pratiques. Vous trouverez dans cet article les lignes directives pour améliorer vos développements.

PRINCIPE DE LA RESPONSABILITÉ UNIQUE

Selon Robert Martin, plusieurs responsabilités au sein d'une même classe sont couplées. Ainsi, toute modification d'une responsabilité peut impacter l'ensemble de la classe. De ce couplage naît une architecture fragile dont les modifications peuvent entraîner des dysfonctionnements. En des termes plus simples, une classe doit gérer une responsabilité unique, une responsabilité étant un regroupement de fonctionnalités ayant un sens commun.

On a souvent tendance à donner plusieurs responsabilités à une classe. De fait, si on souhaite modifier le comportement d'une responsabilité au sein d'une classe, le comportement de l'ensemble des responsabilités de la classe risque d'être impacté.

Utiliser le principe de la responsabilité unique permet d'éviter les défauts de design, de bénéficier d'une meilleure lisibilité du code et d'une meilleure cohésion, d'abaisser le couplage et de garantir l'encapsulation de l'information. Ce principe semble simple, en tout cas sur le papier. Toutefois dans la vraie vie des projets, c'est un des principes les plus difficiles à respecter.

Afin de bien appliquer ce principe, il faut au préalable faire une première phase d'analyse en définissant l'ensemble des méthodes nécessaires à la réalisation des fonctionnalités attendues. A ce stade, nous avons une liste de méthodes non organisées entre elles. A partir de cette liste, on va regrouper les méthodes qui répondent à un besoin commun au sein d'une même classe. Il ne faut pas hésiter à créer autant de classes que nécessaire, même si certaines méthodes se retrouvent seules.

Pour faire évoluer le code, on respecte ce principe. Une nouvelle méthode correspondant à un besoin déjà existant sera rajoutée à la classe correspondante. Par contre, s'il s'agit d'un nouveau besoin on devra créer une nouvelle classe.

Il est bien entendu possible d'appliquer ce principe à du code existant. Prenons comme exemple une classe « Modem » qui regroupe des méthodes de connectivité ainsi que des méthodes d'envoi de données [Fig.1]. Ceci est une violation du principe de responsabilité unique. La solution dans cet exemple est de créer deux classes « Connection » et « DataChannel » regroupées au sein de la classe « Modem » [Fig.2].

Suite à cette optimisation, chaque classe contient une seule responsabilité. Ainsi, une modification sur chacune des classes n'impactera plus les autres classes.

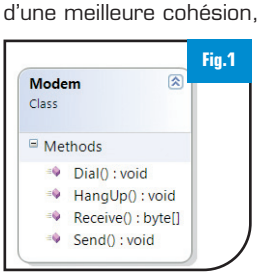
PRINCIPE OUVERT-FERMÉ

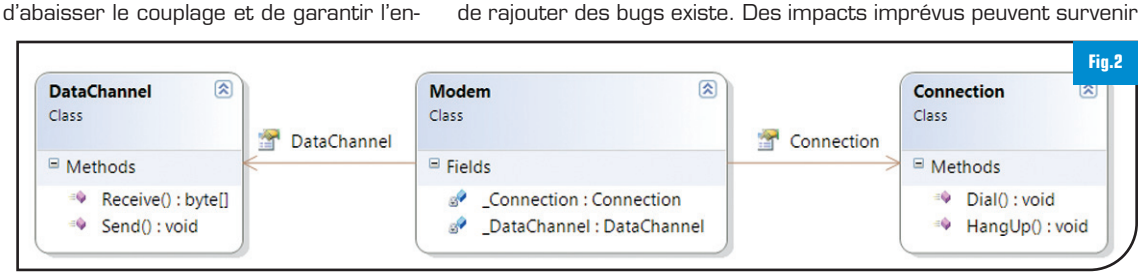
Selon Bertrand Meyer, les entités logicielles telles que les classes et les méthodes doivent être ouvertes pour l'extension, mais fermées à la modification.

C'est-à-dire que leur comportement peut être étendu ou bien complètement modifié si elles doivent remplir de nouveaux besoins (ouverte à l'extension) tout en s'assurant que leur code source ne peut pas être directement modifié, aucune modification n'est autorisée (fermé à la modification).

Après avoir lu cette définition, vous devez vous dire que cela semble contradictoire ! Comment une classe ou une méthode peut être à la fois ouverte pour extension et fermée à la modification ?!

Cela signifie simplement qu'une application doit être structurée de manière à pouvoir ajouter des fonctionnalités en touchant au minimum au code existant. À chaque modification de code, la possibilité de rajouter des bugs existe. Des impacts imprévus peuvent survenir





amenant à la modification du comportement de la classe et donc de l'application. L'implémentation devient fragile, difficilement maintenable et sujette aux régressions.
Prenons un exemple précis de non-respect du principe Ouvert-Fermé : afficher à l'écran deux formes graphiques, un carré et un cercle. L'implémentation comporte une seule classe nommée « Shape » avec une propriété « ShapeType » qui contient le type de formes graphiques voulues [Fig.3]. Le programme a une méthode « DrawAllShapes(…) » contenant le code nécessaire pour dessiner les différentes formes :

```
public void DrawAllShapes(List<Shape> shapeList)
{
  foreach (Shape obj in shapeList)
  {
    switch (obj.ShapeType)
    {
      case ShapeType.Square:
        DrawSquare(obj);
        break;

      case ShapeType.Circle:
        DrawCircle(obj);
        break;
    }
  }
}

public void DrawSquare(Shape obj) { /*Draw a square*/ }
public void DrawCircle(Shape obj) { /*Draw a circle*/}
```

Cette implémentation présente un défaut majeur. Si une nouvelle forme graphique doit être ajoutée, il est nécessaire de modifier la méthode « DrawAllShapes(…) ». Ceci alourdit le code, le rend illisible, sujet aux erreurs, difficile d'évolution, et il sera nécessaire de le re-tester dans son ensemble (surtout dans le cas où beaucoup de formes graphiques sont à gérer).
La solution est simple avec la programmation orientée objet : utiliser le principe de l'héritage.
Il suffit dans ce cas d'ajouter une classe abstraite qui contient la définition de la méthode « Draw() », mais qui ne contient pas d'implémentation. On crée ensuite une classe par forme graphique que l'on souhaite gérer. Les classes « Square » et « Circle » héritent des méthodes et des propriétés de la classe « Shape » [Fig.4]. Si maintenant on souhaite ajouter une forme graphique « Triangle », il suffit de créer une nouvelle classe qui hérite aussi de la classe « Shape » permettant d'éviter de faire des modifications dans le code existant.

```
public void DrawAllShapes(List<Shape> shapeList)
{
  foreach (Shape obj in shapeList)
  {
    obj.Draw();
  }
}
```

Le respect de ce principe permet donc de bénéficier des plus grands avantages de la programmation orientée objet, à savoir l'encapsulation, la réutilisation et la maintenabilité.

PRINCIPE DE SUBSTITUTION DE LISKOV

Selon Robert Martin, ce principe peut être défini de la manière suivante : les méthodes qui utilisent les objets d'une classe doivent pouvoir utiliser des objets dérivés de cette classe sans même le savoir. En d'autres termes, une classe de base doit être substituable par ses classes dérivées sans que cela nécessite des modifications dans la méthode qui utilise ces classes. Ce principe est complémentaire au principe Ouvert-Fermé que nous venons d'expliquer.
Il existe deux types de violation du principe de Liskov, une plutôt conceptuelle et une plutôt fonctionnelle.
La première violation consiste à déterminer le type concret d'un objet pour faire le traitement adapté. Ceci est déterminé par l'utilisation de « cast » ou bien directement par vérification du type de l'objet. Cette violation est simple à détecter, car elle constitue également une violation du principe Ouvert-Fermé.
Si nous reprenons l'exemple utilisé dans le principe Ouvert-Fermé, une violation du principe de Liskov serait la suivante :

```
public void PrintShapeArea(Shape obj)
{
  double area = 0;
  if (obj is Square)
    area = GetSquareArea((Square)obj);
  else if (obj is Circle)
    area = GetCircleArea((Circle)obj);
  Console.WriteLine(string.Format(«Shape Area: {0}», area));
}

private double GetSquareArea(Square obj)
{
  return obj.SideLength * obj.SideLength;
}

private double GetCircleArea(Circle obj)
{
  return Math.PI * obj.Radius * obj.Radius;
}
```

Dans cet exemple, la méthode « PrintShapeArea(…) » nécessite la connaissance du type réel de chaque objet passé en paramètre

Fig.3

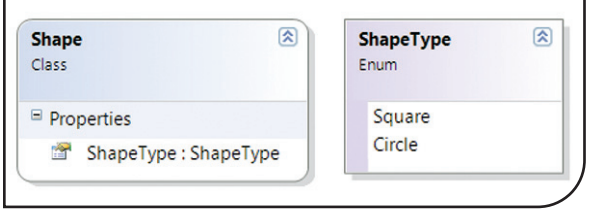


Fig.4

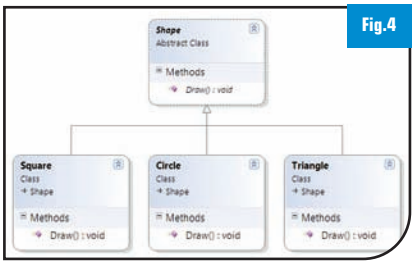
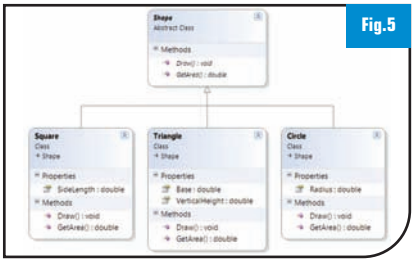


Fig.5



pour pouvoir appeler la fonction qui retourne la surface. Le traitement d'un nouveau type d'objet « Triangle » implique non seulement la modification de la méthode « PrintShapeArea(…) », mais aussi un renforcement du couplage entre la classe qui contient la méthode « PrintShapeArea(…) » et la classe « Triangle ». Pour éviter ce genre de violation, il faut donc minimiser l'utilisation des méthodes telles que « cast », « is », « as », « GetType », … qui permettent de travailler sur le type de l'objet.
Une bonne solution est d'appliquer un design qui respecte le principe Ouvert-Fermé, ce qui implique le rajout d'une méthode « GetArea() » à la classe de base « Shape » [Fig.5]. Tout nouveau type héritant de cette classe doit surcharger la nouvelle méthode avec le traitement adapté. La méthode « PrintShapeArea(…) » est maintenant en mesure de traiter tous les sous-types de manière transparente.

```
public void PrintShapeArea(Shape obj)
{
  Console.WriteLine(string.Format(«Shape Area: {0}», obj.GetArea()));
}
```

La deuxième violation de type fonctionnel est plus difficile à appréhender car elle n'est pas directement liée à une erreur de design, mais à une modification de comportement. Certains traitements nécessitent que les objets répondent à certaines spécifications et contraintes. Parfois, lorsque l'on dérive des objets qui ont des contraintes, il arrive qu'elles soient modifiées. Le résultat peut être différent de celui attendu, les objets se trouvant dans des états non-utilisables. À ce stade, on arrive à une difficulté de mise en œuvre des principes Ouvert-Fermé et de Liskov.
L'un pousse les développeurs à modifier les comportements dans les sous-types, alors que l'autre limite (voire interdit) ces mêmes modifications afin d'éviter les incohérences fonctionnelles. La solution, pour éviter les violations au principe de Liskov tout en respectant le principe Ouvert-Fermé, est l'utilisation du concept « Design By Contract » exposé par Bertrand Meyer. L'utilisation de ce concept dans les programmes consiste à établir des contraintes fonctionnelles dans l'utilisation des objets, ces contraintes définissant les conditions correctes d'utilisation d'un objet.
Il existe trois types de contrat :

- Les pré-conditions : tester l'état des objets avant le traitement (tester les paramètres)
- Les post-conditions : tester l'état des objets à la fin du traitement (tester les objets modifiés ou les résultats produits par le traitement)
- Les invariants : s'assurer que certains objets ne changent jamais pendant un traitement

Les modifications effectuées dans les sous-types sont donc encadrées par le biais de contrats « fonctionnels ». Cependant, il faut s'assurer que les fonctionnalités dans les sous-types respectent toujours les mêmes contraintes. Dans le cas où l'on souhaite utiliser le polymorphisme pour faire évoluer les fonctionnalités, les règles suivantes doivent être respectées :

- Aucune modification des invariants dans une classe dérivée
- Aucun renforcement des pré-conditions dans une classe dérivée
- Aucun affaiblissement des post-conditions dans une classe dérivée

Le principe de Liskov apparaît donc comme un complément du principe Ouvert-Fermé, il ajoute un aspect fonctionnel qui permet de s'assurer du bon fonctionnement des traitements.

PRINCIPE DE SÉGRÉGATION DES INTERFACES

Ce principe, également énoncé par Robert Martin, indique que les clients ne doivent pas être forcés de dépendre d'interfaces qu'ils n'utilisent pas. Une entité informatique (classe, service,…) est cliente d'une interface si elle utilise une classe qui implémente cette interface. Appliquer ce principe est assez simple. La méthodologie employée est similaire à celle du principe de responsabilité unique. La création de classes qui possèdent plusieurs responsabilités est souvent nécessaire lorsqu'on développe des services. Dans ce cas, l'application du principe de ségrégation des interfaces permet de limiter le couplage entre les différentes classes.
Prenons le cas d'une classe « BusinessObject » qui implémente plusieurs groupes de fonctionnalités (plusieurs responsabilités) comme par exemple la création, la mise à jour, la validation, la suppression [Fig.6].

```
public class ValidationManager
{
  public void ValidateBusinessObjects(List<IBusinessObject>
businessObjects)
  {
    if (businessObjects != null)
    {
      foreach (IBusinessObject bo in businessObjects)
      {
        bo.Validate();
      }
    }
  }
}
public class PersistManager
{
  public void InsertAll(List<IBusinessObject> businessObjects)
  {
    if (businessObjects != null)
    {
      foreach (IBusinessObject bo in businessObjects)
      {
        bo.Insert();
      }
    }
  }
}
```

On note ici l'utilisation de l'interface « IBusinessObject » comme type à la place de la classe « BusinessObject ». On appelle cela « Interface-Based Programming ». Cette pratique permet de programmer en utilisant des interfaces à la place de classes concrètes. Tous les objets qui implémentent l'interface peuvent être utilisés au sein des méthodes. Il n'y a pas de limitation par un certain type de classe, mais l'exemple précédent présente encore des défauts de conception. Dans cet exemple, chacun des clients n'utilise qu'un aspect de l'interface :

- La classe « ValidationManager » n'utilise que les méthodes liées à la validation
- La classe « PersistManager » n'utilise que les méthodes liées à la persistance

Une modification des méthodes liées à la Validation implique une modification de toute l'interface « IBusinessObject » et donc indirectement de la classe « PersistManager » bien qu'elle n'utilise pas ces méthodes ! Il faut donc trouver une solution pour que les différents clients accèdent et dépendent seulement des méthodes qu'ils utilisent. La première modification à effectuer pour améliorer le design est le regroupement des méthodes communes dans des interfaces spécialisées (de la même manière que pour le principe de responsabilité unique). L'interface « IBusinessObject » est séparée en deux interfaces spécialisées : Persistance et Validation, et va hériter ces interfaces [Fig.7]. Il est maintenant possible d'utiliser aussi bien les fonctionnalités de Persistance que de Validation de façon séparée, ou bien d'utiliser les deux en même temps. Une modification dans une de ces interfaces n'a plus d'incidence dans l'autre. Elles évoluent de manière indépendante.

Bien évidemment, il faut maintenant s'assurer que ces interfaces spécialisées, « IPersistable » et « IValidatable », sont utilisées à la place de l'interface initiale « IBusinessObject ».

```
public class ValidationManager
{
  public void ValidateBusinessObjects(List<IValidatable> valida
tableObjects)
  {
    if (validatableObjects != null)
    {
      foreach (IValidatable vo in validatableObjects)
      {
        vo.Validate();
      }
    }

  }
}
public class PersistManager
{
  public void InsertAll(List<IPersistable> persistableObjects)
  {
    if (persistableObjects != null)
    {
      foreach (IPersistable po in persistableObjects)
      {
        po.Insert();
      }
    }
  }
}
```

Une interface globale peut conduire à un couplage non voulu entre plusieurs clients qui devraient être isolés. Le principe de ségrégation des interfaces indique donc que plusieurs interfaces clients spécifiques sont plus appropriées qu'une grande interface globale. Ce principe s'applique de la même manière aux classes abstraites, car elles peuvent être considérées comme des interfaces.

PRINCIPE D'INVERSION DE DÉPENDANCE

Dans de nombreuses applications, les modules de haut niveau sont directement liés aux modules de bas niveau. Les modules de haut niveau portent souvent la logique complexe, alors que les modules de bas niveau gèrent les opérations basiques et primaires. Ceci peut poser des problèmes de dépendance entre les modules.

Une application type qui présente des problèmes de dépendance forte peut être désignée ainsi : la Couche Présentation est dépendante de la Couche Métier qui est dépendante de la Couche Accès aux Données [Fig.8].

Le changement d'un module de bas niveau engendre la modification de ceux de haut niveau, les modules de hauts niveaux ne peuvent donc pas être réutilisés au sein d'autres contextes techniques.

Le principe d'inversion de dépendance, toujours énoncé par Robert Martin, indique que les modules de haut niveau ne doivent pas dépendre des modules de bas niveau. Les deux doivent dépendre d'abstractions. Les abstractions ne doivent pas dépendre des détails. Les détails doivent dépendre des abstractions.

L'idée sous-jacente de ce concept est donc qu'une abstraction doit faire le lien entre les modules de haut et de bas niveau. Une abstraction est le plus souvent une interface, on peut également utiliser le concept « Factory Pattern ».

Ici la classe n'est dépendante d'aucune autre classe, les seules références à des objets sont typées via des interfaces :

```
public class ValidationManager
{
  private IValidatable _validatableObject;

  public ValidationManager(IValidatable validatableObject)
  {
  _validatableObject = validatableObject;
  }

  public void RunValidation()
  {
    _validatableObject.Validate();
  }
}
```

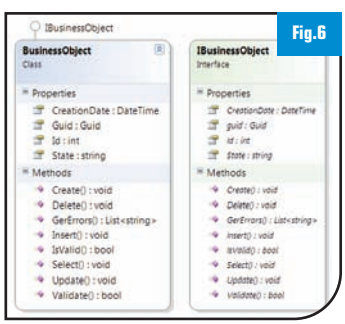


Fig.6

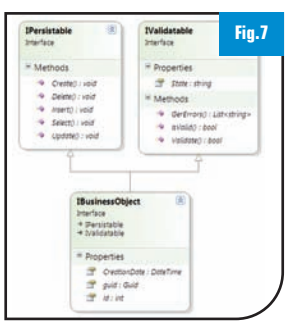


Fig.7

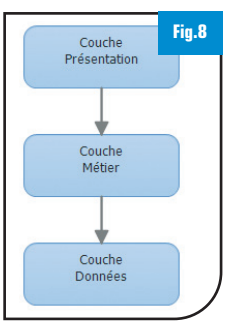


Fig.8

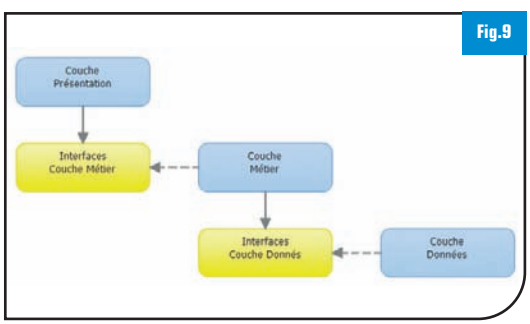


Fig.9

La classe « ValidationManagerFactory » se charge de résoudre les dépendances : elle associe un objet qui respecte l'interface « IValidatable » à un objet de type « ValidationManager ».

```
public class ValidationManagerFactory
{
  public ValidationManager GetValidationManager(IBusinessObject bo)
  {
    return new ValidationManager(bo);
  }
}
```

L'introduction d'interface permet donc d'inverser les dépendances et d'obtenir dans notre exemple des couches indépendantes [Fig.9].
Ce principe amène de la flexibilité dans les applications, les modules étant réutilisables. L'application de ce principe coupe donc totalement les liens pouvant exister entre deux modules mais complexifie le code en augmentant le niveau d'abstraction. Il faut donc savoir l'utiliser avec parcimonie.

CONCLUSION

S.O.L.I.D. est un groupe de principe qui va vous permettre d'aboutir à du code :

- compréhensible, car chaque classe sera spécialisée par responsabilité
- facilement maintenable, car le couplage entre les différentes entités est limité
- permettant l'intégration des évolutions de manière simple

Mais ces guides ne sont pas des règles figées dans leurs applications, elles doivent être mûrement réfléchies et adaptées à chaque contexte. L'intérêt de prendre en compte les principes S.O.L.I.D. dans les projets doit également être mis en relation avec la méthodologie de gestion de projet. Aujourd'hui les méthodes agiles comme SCRUM ou XP sont à la mode et remplacent de plus en plus les méthodologies classiques comme le modèle en cascade ou bien le modèle du cycle en V.
Les méthodes agiles reposent sur une multiplication de cycles ou d'itérations courtes dans le temps, elles favorisent la constante modification et la re-factorisation du design …ce qui est en fait très compatible avec S.O.L.I.D. !
Faire un projet en mode agile en respectant les principes S.O.L.I.D. évite l'apparition de problèmes liés au design. On se rend donc compte que ces principes de la programmation orientée objet sont plus que jamais d'actualité et aident à la réussite des projets Agile.

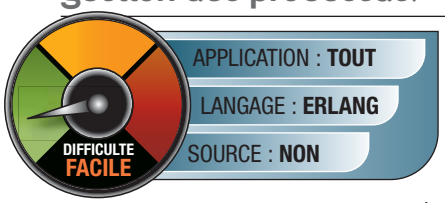
■ Jason **De Oliveira**, *Solutions Architect chez Winwise, 13 ans d'expérience dans le développement logiciel. Blog:* http://jasondeoliveira.com

■ Fathi **Bellahcene**, *Consultant/Formateur chez Winwise. 7 ans d'expérience dans le développement logiciel. Blog :* http://blogs.codes-sources.com/fathi

Introduction à Erlang

2e partie

Nous avions vu dans un premier article quels étaient les types de base de Erlang et comment écrire des fonctions et compiler un programme. Nous allons maintenant regarder de plus près une des caractéristiques les plus intéressantes du langage : la gestion des processus.



LES PROCESSUS

Les processus ou threads, sont un élément important dans Erlang. Ils sont très simples à créer, avec l'aide de la fonction spawn/3. Cette fonction prend en premier et deuxième paramètre les noms de module et de fonction à exécuter dans le nouveau processus. Le troisième paramètre est une liste des arguments à passer à la fonction à exécuter. La taille de la liste doit correspondre à l'arité de la fonction. spawn/3 retourne le PID du nouveau processus. Un processus s'exécute en tâche de fond et prend fin lorsque la fonction qui l'a créée se termine. La fonction self/0 retourne le PID du processus courant.

```
1> self().
<0.34.0>
2> spawn(io,format,["Bien le bonjour d'un nouveau processus~n
J'ai été créé par le processus ~p~n", [self()]]).
Bien le bonjour d'un nouveau processus
J'ai été créé par le processus <0.34.0>
<0.43.0>
```

Dans l'exemple ci-dessus, le PID du shell est <0.34.0>. Depuis le shell, un nouveau processus <0.43.0> est créé, le temps d'afficher un message. À noter que le self/0 passé en troisième paramètre du spawn/3 est évalué à l'appel du spawn, ce qui explique qu'il retourne le PID du shell et non du nouveau processus.

LE TYPE PID

Comme la gestion des processus est un élément essentiel dans Erlang, il existe un type dédié pour représenter les PID. L'utilisation des trois nombres que nous avons vue ci-dessus n'est qu'une représentation textuelle, qu'il n'est pas possible d'utiliser telle quelle :

```
1> A=<0.43.0>.
* 1: syntax error before: 0
```

Il existe une fonction qui permet de créer un PID depuis sa représentation textuelle, mais elle ne doit être utilisée que pour déboguer, son fonctionnement n'est pas garanti :

```
1> A=list_to_pid("<0.43.0>").
<0.43.0>
```

Il est possible de nommer des processus pour les manipuler plus facilement. On utilise pour cela la commande *register*, qui associe un nom (sous la forme d'un atome) à un PID.
register(Nom, Pid).

LES MESSAGES

Pour faire communiquer les processus entre eux, Erlang utilise un mécanisme de passage de message. L'opérateur ! est utilisé pour envoyer un message à un autre processus. Sa syntaxe est : Pid ! Message.
La valeur de retour de l'expression est le message. Un message peut être n'importe quel terme Erlang. On peut également utiliser le nom d'un processus nommé à la place du PID : Nom ! Message.
Pour recevoir un message, nous utilisons la commande receive, suivie d'une ou plusieurs clauses :

```
receive
MessageType1 -> Traitement1;
MessageType2 -> Traitement2;
…
MessageType3 -> Traitement3
after Millisecondes -> Traitement4
end.
```

Chaque processus dispose d'une queue des messages reçus. Lorsqu'il exécute une commande receive, la queue est parcourue pour trouver le premier message qui matche l'une des clauses du receive. Si un tel message est trouvé, il est retiré de la queue et le traitement correspondant est exécuté.
Si aucun message ne matche (ou si la queue est vide), alors receive bloque le processus jusqu'à ce qu'un message correspondant soit reçu. Le Traitement4 est exécuté si aucun message n'est reçu après le temps indiqué par la clause after. Cette clause after est optionnelle.

UN SERVEUR DE VALEUR

Nous en savons maintenant assez pour écrire notre premier programme fonctionnant sur le modèle du client/serveur. Nous pouvons utiliser le passage de message pour faire par exemple un serveur de valeur, capable de stocker une valeur et de l'afficher :

```
-module(procs).
-export([loop/1]).

loop(C) ->
receive
get ->
io:format("Valeur courante : ~p~n", [C]),
loop(C);
{set, I} ->
io:format("Nouvelle valeur: ~p~n", [I]),
loop(I);
quit ->
io:format("Bye~n", [])
end.
```

Le serveur est constitué d'une boucle récursive qui attend une instruction sous la forme d'un message. La valeur courante du serveur est stockée dans le paramètre de la boucle. Si le message reçu est *get,* le serveur affiche la valeur courante qui a été stockée auparavant sans la changer. Si le message reçu est un *tuple* {set, Valeur}, alors il affiche la valeur passée dans le deuxième élément du tuple et la stocke. Si le message reçu est *quit,* le serveur s'arrête. Essayons donc d'utiliser ce serveur, après l'avoir compilé :

```
2> register(serveur, spawn(procs, loop, [0])).
true
```

Nous utilisons un spawn pour exécuter la fonction procs:loop/1 dans un nouveau processus, en lui passant une valeur initiale de 0. Nous en profitons pour nommer ce nouveau processus afin de pouvoir lui envoyer des messages plus simplement.

```
3> serveur ! get.
Valeur courante : 0
get
```

Si nous envoyons un message get au serveur, il affiche la valeur courante, qui est celle passée à loop lors de la création du serveur.

```
4> serveur ! {set, 42}.
Nouvelle valeur: 42
{set,42}
5> serveur ! get.
Valeur courante : 42
get
```

Nous pouvons changer la valeur courante, et vérifier qu'elle est bien enregistrée par le serveur.

```
6> serveur ! quit.
Bye
quit
```

Si le serveur reçoit quit, il arrête la récursion et son processus se termine. Il n'est alors plus possible de lui envoyer des messages :

```
7> serveur ! get.
** exception error: bad argument
in operator  !/2
called as serveur ! get
```

GÉNÉRATEURS DE LISTE

Avant de continuer à jouer avec les processus, voyons quelques compléments utiles pour créer et manipuler les listes. Tout d'abord, lists:seq/2, qui crée une liste à partir de deux intervalles entiers :

```
1> L1 = lists:seq(1, 10).
[1,2,3,4,5,6,7,8,9,10]
```

Voyons ensuite une construction bien connue des programmeurs Python, le générateur de liste, qui utilise l'opérateur || :

```
2> L2 = [Number*Number || Number <- L1].
[1,4,9,16,25,36,49,64,81,100]
```

Cette expression doit se lire ainsi : pour chaque élément Number de la liste L1, construire une liste qui contient l'expression à gauche du ||, à savoir le carré de Number. Introduisons enfin lists:foreach/2, qui prend en premier paramètre une fonction d'arité 1 et en deuxième paramètre une liste. lists:foreach/2 va alors appeler la fonction d'arité 1 en lui passant successivement chaque élément de la liste. lists:foreach/2 retourne l'atome ok.
Pour la tester, définissons une fonction dite anonyme à l'aide de la construction fun() :

```
3> F = fun(N) -> io:format("~p~n", [N]) end.
#Fun<erl_eval.6.13229925>
```

fun() permet de retourner une fonction quelconque sans lui donner de nom. Nous stockons le résultat dans la variable F. F est donc lié à une fonction d'arité 1 qui se contente d'afficher sur une ligne le paramètre qui lui est passé. Nous pouvons maintenant passer cette fonction anonyme comme premier paramètre de lists:foreach/2 :

```
4> lists:foreach(F, L2).
1
4
9
16
25
36
49
64
81
100
ok
```

PROGRAMMATION MASSIVEMENT PARALLÈLE

Pour se convaincre de la capacité d'Erlang à gérer un grand nombre de processus, essayons ce petit test. Écrivons une fonction start_threads(N), qui crée N processus qui attendent un message 'die' pour se terminer, et retourne la liste des PID des processus créés :

```
start_threads(Total) ->
[spawn(?MODULE, thread, [N]) || N <- lists:seq(1, Total)].
```

Nous utilisons un générateur de liste pour retourner la liste des PID créés. ?MODULE est une macro qui contient automatiquement le nom du module courant. Il nous faut maintenant écrire la fonction ?MODULE:thread/1 qui est utilisée par le spawn pour créer un nouveau processus, et qui prend en paramètre un numéro attribué par start_threads/1 :

```
thread(N) ->
?DEBUG("~p, ~p starting", [N, self()]),
receive die ->
?DEBUG("~p, ~p dying", [N, self()]),
ok
end.
```

?DEBUG/2 est une macro que nous devons écrire qui nous permettra de vérifier que tout fonctionne correctement.

Nous écrivons maintenant une fonction kill_threads/1 qui prend en paramètre une liste de PID et envoie à chacun un message 'die' :

```
kill_threads(L) ->
Kill = fun(Pid) ->
Pid ! Die
end,
lists:foreach(Kill, L).
```

Nous utilisons une fonction anonyme qui prend un PID en entrée et envoie un message 'die', et nous l'appliquons à tous les PID de la liste passée en paramètre. Nous pouvons utiliser toutes ces fonctions pour écrire un module complet, en y ajoutant la macro ?DEBUG :

```
-module(procs2).
-export([thread/1, start_threads/1, kill_threads/1]).
-define(DEBUG(Format, Args),
 io:format("DEBUG [~p:~p]: "++Format++"~n", [?MODULE, ?LINE | Args])).

thread(N) ->
?DEBUG("~p, ~p starting", [N, self()]),
receive die ->
?DEBUG("~p, ~p dying", [N, self()]),
ok
end.

start_threads(Total) ->
[spawn(?MODULE, thread, [N]) || N <- lists:seq(1, Total)].

kill_threads(L) ->
Kill = fun(Pid) ->
Pid ! Die
end,
lists:foreach(Kill, L).
```

Compilons et testons ce module.
À ce point, L est lié à une liste de 5 PID, qui correspondent aux 5 processus créés. Nous pouvons maintenant envoyer un message à chacun de ces processus pour leur demander de se terminer :

```
3> procs2:kill_threads(L).
DEBUG [procs2:11]: 1, <0.42.0> dying
DEBUG [procs2:11]: 2, <0.43.0> dying
DEBUG [procs2:11]: 3, <0.44.0> dying
DEBUG [procs2:11]: 4, <0.45.0> dying
DEBUG [procs2:11]: 5, <0.46.0> dying
ok
```

Tout fonctionne correctement. Essayons maintenant de créer 100 000 processus :

```
4> procs2:start_threads(100000).

=ERROR REPORT==== 31-May-2010::11:45:58 ===
Too many processes

** exception error: a system limit has been reached
 in function  spawn/3
 called as spawn(procs2,thread,[32743])
 in call from procs2:'-start_threads/1-lc$^0/1-0-'/1
 in call from procs2:'-start_threads/1-lc$^0/1-0-'/1
*** ERROR: Shell process terminated! ***
```

Le programme s'est « planté » parce que nous avons essayé de créer trop de processus. La documentation de erl nous apprend que la limite par défaut est de 32 768, mais qu'il est possible de la changer à l'aide du flag +P (jusqu'à un maximum de 134 217 727, ce qui nous laisse de la marge). Profitons-en pour modifier la macro ?DEBUG afin d'enlever tous les messages de tests :

```
-define(DEBUG(Format, Args), ok).
```

Puis relançons notre machine virtuelle avec le flag :

```
% erl +P 200000
Erlang R13B03 (erts-5.7.4) [source] [64-bit] [smp:2:2] [rq:2]
[async-threads:0] [hipe] [kernel-
oll:false]
Eshell V5.7.4  (abort with ^G)
1>
```

Recompilons et tentons l'expérience à nouveau :

```
1> c(procs2).
{ok,procs2}
2> L = procs2:start_threads(100000).
[<0.42.0>,<0.43.0>,<0.44.0>,<0.45.0>,<0.46.0>,<0.47.0>,
<0.48.0>,<0.49.0>,<0.50.0>,<0.51.0>,<0.52.0>,<0.53.0>,
<0.54.0>,<0.55.0>,<0.56.0>,<0.57.0>,<0.58.0>,<0.59.0>,
<0.60.0>,<0.61.0>,<0.62.0>,<0.63.0>,<0.64.0>,<0.65.0>,
<0.66.0>,<0.67.0>,<0.68.0>,<0.69.0>,<0.70.0>|...]
```

Cette opération qui créé 100 000 processus et retourne les 100 000 PID correspondants prend moins d'une demi-seconde pour s'exécuter sur un simple laptop. Nous pouvons maintenant tester l'envoi de message :

```
3> procs2:kill_threads(L).
ok
```

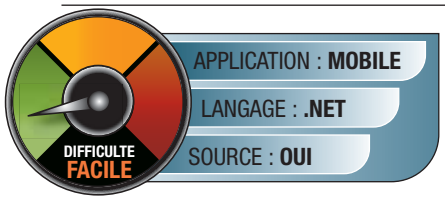
Cette fonction, qui envoie 100 000 messages aux processus créés ci-dessus pour les terminer s'exécute quasi instantanément.

CONCLUSION

Nous avons vu dans ce deuxième article l'une des forces d'Erlang : la possibilité de créer et de gérer un grand nombre de processus très simplement et sans nuire aux performances, et de les distribuer physiquement de façon transparente. Erlang offre bien d'autres aspects permettant d'écrire des programmes hautement performants et tolérants aux pannes, que vous pourrez découvrir en lisant le tutoriel ou en achetant l'un des ouvrages qui lui ont été consacrés (en français ou en anglais).

■ Jérôme **Sautret**
Directeur Technique, ProcessOne

Développer un jeu pour Windows Phone 7 avec XNA



Il y a encore trois/quatre ans, le développement sur mobile s'apparentait à un vrai parcours du combattant. Il fallait lutter pour grappiller le moindre octet et les configurations de téléphones étaient tellement disparates sur le marché qu'elles obligeaient à redévelopper plusieurs centaines de fois une même application. L'iPhone a déjà supprimé partiellement le problème puisqu'il peut être viable de ne cibler que ce support. Windows Phone 7 est par contre un système d'exploitation que différents constructeurs mobiles pourront déployer à condition de respecter des spécifications minimales pour tous les composants de leur téléphone. Un bon moyen de garantir la portabilité des applications et surtout une même expérience utilisateur : résolution unique (800 x 480), même système d'input, 256 Mo ram minimum, 8 Go de mémoire Flash minimum, un processeur / GPU puissant supportant DirectX 9. En résumé, une véritable machine de guerre !

XNA, QUOI D'AUTRE ?

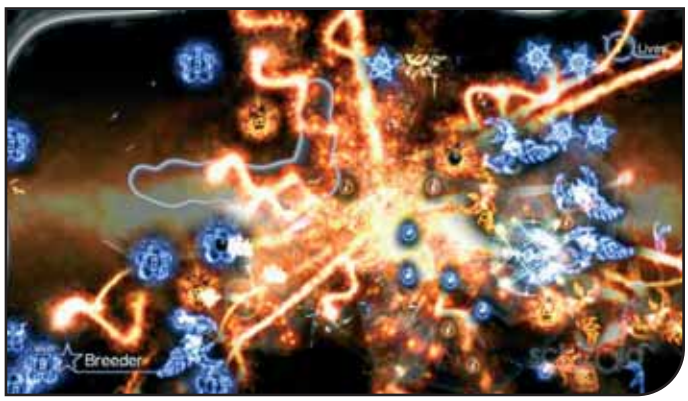
XNA est un framework de développement de jeu vidéo dont le maître mot est la simplicité. Ses avantages sont nombreux : productivité offerte par .NET, code managé, CrossPlatform, le ContentPipeline, etc. Cette technologie désormais éprouvée permet de concentrer essentiellement son attention sur le code de son jeu et non les contraintes techniques autour. Nous allons créer un projet afin d'appréhender tous les concepts clés autour du développement du Windows Phone. Ouvrez Visual Studio et créez un nouveau projet Game Windows Phone avec XNA 4.0. Plusieurs méthodes sont automatiquement créées dans la classe Game pour gérer votre jeu vidéo : Initialize, LoadContent, Update & Draw. Le Constructeur de la méthode Game a quelques propriétés intéressantes :

```
TargetElapsedTime = TimeSpan.FromTicks(333333);
```

XNA tourne nativement à 60 FPS, c'est-à-dire que les méthodes Update et Draw sont appelées 60 fois par seconde. Comme la plupart des autres terminaux mobiles, le Windows Phone a une vitesse de rafraîchissement de 30 FPS. Le TargetElapsedTime est ainsi initialisé pour éviter de gaspiller des calculs inutiles.

```
graphics.PreferredBackBufferWidth = 480;
graphics.PreferredBackBufferHeight = 800;
```

Cela permet d'initialiser les dimensions du backbuffer de l'application avec la résolution native du Windows Phone (800 par 480). Tant que le ratio est respecté, vous pouvez modifier ces valeurs (notamment pour des raisons de performances) et XNA se chargera automatiquement d'adapter le BackBuffer à la résolution de votre Windows Phone. Certains constructeurs proposeront l'année prochaine une seconde résolution sur l'appareil (480 par 320) et vous n'aurez donc absolument rien à gérer pour que tout soit parfaitement compatible.



LET'S START THE GAME !

Commençons par créer une Balle. Dans l'explorateur de solution, faites un clic droit sur le projet, ajoutez un nouvel élément et sélectionnez GameComponent. Créez ensuite 3 variables : un Rectangle Position, un Vector2 Direction et un float pour la Vitesse. En rajoutant l'instance de la classe Ball à la collection de Components de la classe Game, la méthode Update de la balle sera automatiquement appelée.

```
private void InitializeBall()
{
  ball = new Balle(this, new Rectangle(15, 300, 30, 30), new Vector2
(1, -0.5f), 0.2f);
  this.Components.Add(ball);
}
```

Ajoutez 2 méthodes dans la classe Balle pour que cette dernière puisse se mouvoir en fonction de sa direction et vitesse et qu'elle rebondisse contre les murs.

```
private void Move(GameTime gameTime)
{
  Position.X += (int)(Direction.X * Speed * gameTime.Elapsed
GameTime.Milliseconds);
  Position.Y += (int)(Direction.Y * Speed * gameTime.Elapsed
GameTime.Milliseconds);
}

private void ChangeDirection()
{
  if (Position.X < 0 || Position.X + Position.Width > Game.
GraphicsDevice.Viewport.Width)
  {
    Direction.X = -Direction.X;
  }

  if (Position.Y < 0 || Position.Y + Position.Height > Game.
GraphicsDevice.Viewport.Height)
  {
```

```
    Direction.Y = -Direction.Y;
  }
}
```

LA MAGIE DU CONTENT PIPELINE

Le Content Pipeline est un des plus gros atouts de XNA. Il balaye par sa simplicité une des plus grosses contraintes de la plupart des développements de jeu : la gestion de tout son contenu (sons, images, etc.), grâce à notamment 2 outils : l'*importer* et le *processor*. Le premier sert à convertir les fichiers en objet DOM (document object model) que le processor va alors compiler en fichier binaire (.XNB) et pouvoir utiliser dans XNA Game Studio. Véritable framework dans le framework, il est complètement extensible afin de permettre aux artistes de ne pas être bridé sur le choix de leurs outils / format de travail et de répondre également à certains besoins côté programmation. Sur le site du Creator Club, on peut trouver des exemples de *content importer* très utiles : optimisation des SpriteFont pour prendre en charge uniquement tous les caractères contenus dans le jeu, caractères asiatiques inclus ou encore le *SpriteSheet* qui crée automatiquement une texture unique à partir d'autres plus petites pour des questions d'optimisation sans qu'un graphiste perde un temps précieux à la faire et la maintenir. Les possibilités sont infinies. Pour ajouter du contenu, il suffit simplement de les drags'n'droper dans Visual Studio au sein du *project content* créé automatiquement. Nous aurons besoin en tout de 2 images, 1 effet sonore et 1 musique mp3. En cliquant dessus, on peut modifier plusieurs propriétés : l'*AssetName* (nom utilisé par Visual Studio), le *Content Importer* et le *Content Processor*. Notez, pour les musiques, qu'il faut toujours modifier ce dernier et le passer à Song au lieu de SoundEffect. En déroulant le Content Processor, on trouve de nouvelles options d'optimisation intéressantes. Le taux de compression pour un fichier sonore et pour une texture, l'activation du mipmap (image contenant plusieurs résolutions de cette même image) ou le SurfaceColor qui, passé en DxtCompressed, peut permettre de baisser drastiquement le poids de l'image.
XNA étant une technologie multiplateforme (Xbox 360, Windows, Windows Phone), des profils matériels ont été créés et scindés en 2 catégories : Hi-Def (Xbox 360 et Windows avec DirectX 10) et Reach (Windows Phone & Windows avec DirectX 9). Cela entraîne quelques incidences : pas de custom shaders en Reach, la taille maximale des textures est 2048 (contre 4096 en Hi-Def) et 512 pour la taille maximale des cubemap (4096 en Hi-Def). Détail qui a son importance : les mipmaps et la DXT Compression ne peuvent être utilisés que sur des textures dont la taille est une puissance de 2. Autrement dit, puisque la résolution du téléphone est 800 par 480, combiner 2 textures de cette dimension, et même une partie « vierge » pour atteindre une résolution de 1024 par 1024 permet d'obtenir en sortie un XNB 2 fois moins lourd que si les 2 fichiers avaient été séparés ! Repassons de la théorie au code. Créons d'abord quelques variables correspondant à nos ressources :

```
Texture2D ballTexture;
Texture2D brickTexture;
Song music;
SoundEffect sound;
```

Dans les propriétés du *Content Project,* une bonne astuce est de mettre un point dans le RootDirectory pour ne pas avoir à s'en soucier lors de l'appel des ressources. Dans la méthode *LoadContent,*

On initialise d'abord notre *ContentManager* qui nous permettra ensuite de loader nos ressources en passant leur chemin de destination dans le *Content Project.* Il n'est pas obligatoire d'indiquer l'extension des fichiers. Sur Windows Phone, il n'est en effet pas possible d'utiliser le moteur audio XACT.

```
this.Content = new ContentManager(this.Services, «»);
ballTexture = Content.Load<Texture2D>(«balle»);
brickTexture = Content.Load<Texture2D>(«brick»);
music = Content.Load<Song>(«music»);
sound = Content.Load<SoundEffect>(«hit»);
```

Il ne reste plus qu'à gérer le rendu. Le SpriteBatch permet de communiquer avec la carte graphique. La méthode Begin() permet de la « préparer » alors que la méthode End() finalise le rendu pour l'envoyer au backbuffer pour l'afficher ensuite à l'écran. Ces opérations prennent un certain temps à se réaliser alors que la différence de temps de traitement n'est quasiment pas perceptible qu'il y ait un seul Draw ou une centaine. C'est pourquoi il est recommandé d'effectuer tout le rendu du jeu dans un seul bloc Begin / End.

```
spriteBatch.Begin();
spriteBatch.Draw(ballTexture, ball.Position, Color.Blue);
  spriteBatch.End();
```

TOUT EST UNE QUESTION DE NIVEAU

Il peut être intéressant de charger les niveaux dans un fichier externe (pour qu'un level designer puisse les modifier par exemple). On va partir d'un cas un peu sommaire pour illustrer les grands principes du système d'IO : découpons un niveau par « block » de 80 pixels de largeur/hauteur. Dans un simple fichier texte, les niveaux seront donc représentés par 10 lignes de 6 caractères chacun, où chaque X symbolise une brique.

```
X-X-X-
-X-X-X
X-X-X-
-X-X-X
- - -
- - -
- - -
- - -
- - -
- - -
```

Rajoutez le fichier level à votre Content Project. XNA ne compile pas par défaut les fichiers .txt et va donc générer une erreur. Vous avez soit la possibilité d'étendre le ContentPipeline pour créer votre propre type niveau et pouvoir compiler ; soit simplement modifier les

propriétés pour que le fichier ne soit pas compilé mais inclus aux binaires de l'application. Changez la propriété de Build Action à « None » et Copy to Output Directory à « Copy if Newer ».

```
private void LoadLevel(string path)
{
  List<string> lines = new List<string>();
  StreamReader sr = new StreamReader(TitleContainer.OpenStream
(path));
  string line;
  do
  {
    line = sr.ReadLine();
    if(line != null)
      lines.Add(line);
  }   while (line != null);

  int sizeBlock = 80;
  for (int y = 0; y < GraphicsDevice.Viewport.Height/size
Block; ++y)
    for (int x = 0; x < GraphicsDevice.Viewport.Width / size
Block; ++x)
      if (lines[y][x] == 'X')
        this.brickList.Add(new Brick(this, new Rectangle(x * size
Block, y * sizeBlock, sizeBlock, sizeBlock)));

  nbBallActive = this.brickList.Count;
}
```

Attention à ne pas oublier de rajouter les Brick à la liste de Composents de la classe Game et de les afficher. Le code suivant permet de gérer les collisions entre la balle et les briques de manière très simple en les considérant tous deux comme deux rectangles. Pour un résultat plus précis (mais beaucoup plus gourmand), on vous renvoie aux samples du site Creator Club sur le pixel Perfect.

```
private void CheckCollisionBrick()
{
  for(int i = 0; i<brickList.Count; i++)
  {
    if (IsBrickCollision(i))
    {
      this.brickList[i].Visible = true;
      this.nbBallActive—;
      sound.Play();
    }
  }
}

private bool IsBrickCollision(int i)
{
  return (!this.brickList[i].Visible && brickList[i].Position.
Intersects(this.ball.Position));
}
```

On a notamment profité pour jouer l'effet sonore à chaque collision. La gestion des musiques est un peu différente puisqu'il faut passer par la librairie MediaPlayer.

```
  MediaPlayer.Play(this.music);
```

UN WHISKY ? JUSTE UN DOIGT !

Les Windows Phone peuvent gérer jusqu'à 4 points de contact sur l'écran, chacun représenté par une instance de la classe TouchLocation. Dans notre cas, on va gérer le multitouch de manière simple : à chaque fois qu'un doigt posé sur l'écran se retire, la balle part alors dans la direction opposée à ce dernier.

```
private void UpdateInput()
{
  TouchCollection touchCollection = TouchPanel.GetState();
    foreach (TouchLocation tl in touchCollection)
      if (tl.State == TouchLocationState.Released)
      {
        this.ball.Direction = new Vector2(this.ball.Position.X,
this.ball.Position.Y) - tl.Position;
        this.ball.Direction.Normalize();
      }
}
```

À l'initialisation de la Classe Game, mettre le jeu en plein écran permet d'éviter quelques imperfections de précision sur le système d'input.

```
            graphics.IsFullScreen = true;
```

PLUS IL Y A DE PLATFORM, PLUS ON RIT !

Il faut bien garder à l'esprit que nous développons sur un Émulateur et non un Simulateur (du moins pour l'instant). Même si celui-ci est performant, il y a toujours des différences mineures d'exécution, etc. Si la meilleure solution reste de déployer sur un vrai Device quoiqu'il arrive, une bonne astuce est de développer le jeu également sur Windows et ainsi de bénéficier de puissants outils de débogage.
Ce qui tombe particulièrement bien puisque XNA permet un développement cross-platform sur PC, Xbox 360 et aussi sur Windows Phone 7 de manière simplifiée. Dans l'explorateur de solution, faites un clic droit, puis « Create Copy of Project for Windows ». Le projet Windows apparaît alors et les développements d'un projet sont automatiquement reportés sur l'autre. Pour lancer le jeu sur une plateforme particulière, sélectionnez la solution dans l'explorateur. Dans les propriétés, vous pourrez alors modifier le mode de compilation (DebuglWindows Phone, etc.) et le projet de démarrage. Nouveauté de XNA 4.0, le Content Project est désormais partagé entre les projets. Simple, je vous l'avais bien dit !
En sus des deux profils hardware évoqués plus haut, certaines plateformes ont évidemment quelques spécificités propres sur certaines fonctionnalités ou même API : manette sur PC et Xbox 360, gestion de l'overscan sur Xbox 360, etc. Les directives de préprocesseurs nous permettent de ne cibler qu'une ou plusieurs plateformes. Revenons à notre exemple d'Input. Notre jeu est désormais jouable sur les écrans PC et téléviseurs Tactile.
Si on le souhaite, on pourrait rajouter des contrôles clavier ou manette pour gérer différemment la balle uniquement sur PC. A noter d'ailleurs que les appels aux fonctions d'autres systèmes d'input (keyboard, gamepad) provoquent pour l'instant de gros soucis de performance sur l'émulateur.

```
#if WINDOWS
            UpdateKeyBoardCommand(gameTime);
#endif
```

ADMIREZ LE PAYSAGE !

Pour l'instant, XNA 4.0 CTP ne gère pas encore le mode landscape (paysage). Un problème réglé pour la version finale. Certains types de jeux offrant un meilleur confort de jeu en mode paysage (course, baston, plateforme, etc.), voici une petite astuce permettant de pallier ce désagrément temporaire. Commençons par créer un RenderTarget2D, un buffer où l'on va dessiner tout ce qu'il faut afficher, un peu comme une capture d'écran.

```
    renderTarget = new RenderTarget2D(graphics.GraphicsDevice,
graphics.PreferredBackBufferHeight,   graphics.PreferredBack
BufferWidth);
```

On va remplacer les appels des méthodes spritebatch.Begin() et spriteBatch.End() par ces méthodes ci-dessous. Le rendu est désormais en 2 étapes : dessin dans le RenderTarget2D que l'on affiche ensuite à l'écran en l'orientant à 90°. Et voilà le travail. Plus qu'à apporter quelques simili corrections, notamment inverser la position X & Y des input.

```
  public void SpriteBatchBeginLandscape()
  {
    this.GraphicsDevice.SetRenderTarget(renderTarget);
    this.GraphicsDevice.Clear(Color.Black);
    spriteBatch.Begin(SpriteSortMode.Immediate, BlendState.Non
Premultiplied);
  }

  public void SpriteBatchEndLandscape(GameTime gameTime)
  {
    spriteBatch.End();
    base.Draw(gameTime);
    this.GraphicsDevice.SetRenderTarget(null);
    DrawLandscapeMode();
  }

  private void DrawLandscapeMode()
  {
    spriteBatch.Begin();
    spriteBatch.Draw(renderTarget, new Vector2(240, 400),
      null, Color.White, MathHelper.PiOver2,
      new Vector2(400, 240), 1f, SpriteEffects.None, 0);
    spriteBatch.End();
  }
```

DIVISER POUR MIEUX RÉGLER !

Si l'on peut facilement utiliser des fichiers provenant du content pipeline comme on vient de le voir, le système de fichiers du Windows Phone est assez restrictif pour le développeur pour des raisons de sécurité : création de fichiers possible, pas d'accès à la base de registre, etc. Et d'un autre côté, lorsqu'un jeu utilise des fichiers, l'idéal est que ces derniers restent à la fois cloisonnés au jeu seul (pas d'écrasement possible par une autre application), que leurs accès soient le plus transparent possible sans se soucier de problèmes de droits en écriture/lecture. Autant de problématiques sur Windows Phone que l'on peut facilement résoudre grâce aux fonctions de l'Isolated Storage qui tout comme son nom l'indique va créer un espace isolé où l'on peut manipuler des fichiers à sa guise. Voici un petit exemple où l'on enregistre les logs des dernières parties.

```
  private void UpdateStorage()
  {
#if WINDOWS_PHONE
    IsolatedStorageFile  store  =  IsolatedStorageFile.GetUser
StoreForApplication();
    string filePath = «LogLastPlay.txt»;

    if (!store.FileExists(filePath))
    {
      IsolatedStorageFileStream  rootFile  =  store.CreateFile
(filePath);
      rootFile.Close();
    }
    StreamWriter sw = new StreamWriter(store.OpenFile(filePath,
FileMode.Open, FileAccess.Write));
    sw.WriteLine(«Last Game : « + DateTime.Now.ToShortDateString());
    sw.Close();
#endif
  }
```

PARTAGE TON XAP !

Lorsque votre jeu est compilé pour Windows Phone, un fichier xap est généré. C'est celui-ci qui sera déployé sur un Windows Phone Device. En attendant, vous aurez sans doute envie de partager sans nécessairement distribuer les sources du jeu. La procédure ci-dessous permet de lancer un xap sur l'émulateur.

1. Ouvrez maintenant l'invite de commande (« cmd.exe » dans la recherche Windows)
2. Glissez-Déposez le fichier wp.exe dans l'invite de commande. Ce dernier se trouve dans le dossier C:\Program Files (x86)\Microsoft XNA\XNA Game Studio\v4.0\Tools (chemin par défaut).
3. Saisissez le mot-clé « install »
4. Glissez-Déposez le fichier xap (disponible dans \bin de votre jeu)
5. Copiez collez le GUID de votre jeu (disponible dans AssemblyInfo.cs)
6. Glissez-Déposez le GameThumbnail (disponible dans les binaires)
7. Glissez-Déposez le fichier XapCacheFile.xml (disponible dans \obj)
8. Terminez la commande avec l'instruction /clean et Appuyez sur entrée. L'émulateur devrait se lancer et déployer votre application.
9. Saisissez le mot-clé « launch » suivi du GUID de votre jeu. Le jeu devrait logiquement se lancer sur l'émulateur.

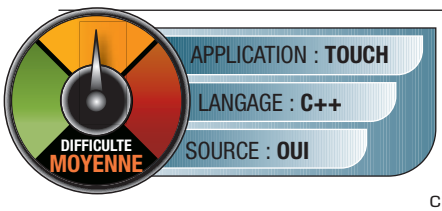
Enfin, il est tout à fait possible d'utiliser quelques briques de Silverlight avec XNA et réciproquement. Quelques exemples foisonnent sur le net comme l'utilisation d'un client Twitter ou simuler l'accéléromètre … avec une télécommande Wii !

■ Frédéric **Pedro**
Gérant de POHLM Studio - Developpeur Windows Phone 7
Frederic.pedro@supinfo.com

La programmation MultiTouch sous Windows 7

2e partie

Le mois dernier, nous avons fait connaissance avec le maniement de l'API Windows Touch en code C++ et Win32 pur et dur. Nous regardons aujourd'hui comment les choses se passent du côté de .Net et de C#



Résumons la situation. Nous savons que Windows 7 propose une API pour construire des applications capables de réagir aux sollicitations de l'utilisateur sur un périphérique tactile. Le code C++ que nous avons écrit était verbeux et difficile à organiser. C'est une caractéristique constante des applications Win32 :-{ Cependant le travail que nous avons fait nous a permis de nous familiariser avec l'API et ses particularités. Ainsi nous savons qu'elle propose deux niveaux d'abstraction. Un niveau 'Gesture' pour lequel le système reconnaît les gestes de l'utilisateur parmi un ensemble de gestes prédéfinis, et un niveau 'Touch', de plus bas niveau, qui fournit des données brutes et laisse le programmeur se débrouiller pour interpréter les actions de l'utilisateur. Le programmeur peut pour cela s'aider d'un jeu d'interfaces COM. Nous n'avons pas travaillé avec ces classes le mois dernier, car cela aurait conduit à du code volumineux. Nous en concluons, maintenant que nous connaissons l'API Windows Touch, qu' il est sage de s'abstenir de la programmer directement. D'autant plus que, lorsqu'il s'agit de répondre à des actions de l'utilisateur, les performances de C++ perdent une grande partie de leur intérêt. Nous nous tournons donc maintenant vers .Net et vers C# en particulier. Là, malgré quelques petites difficultés, les choses sont beaucoup plus aisées. Il y a trois approches pour faire de la programmation multi-touch en .Net. La première approche est celle donnée par le Windows 7 SDK. Elle consiste à définir une méthode WndProc dans une application Windows Forms et à travailler avec les API natives et les structures de données natives à grands coups d'attributs et autres marshalling. On se retrouve avec une procédure de fenêtre aussi laide que la procédure native et globalement le code C# est plus lourd et pire que le code natif. Windows 7 SDK ou pas, c'est une approche qui est maladroite. On jettera un bref coup d'œil aux exemples, par curiosité, puis on oubliera tout cela rapidement. Les deux autres possibilités, fort heureusement, s'appuient sur des librairies qui proposent un niveau d'abstraction beaucoup plus élevé. La première possibilité de travailler en .Net est offerte par une librairie que l'on téléchargera à http://code.msdn.microsoft.com/WindowsTouch. Cette librairie, malheureusement non documentée, est destinée à la plate-forme .Net 3.5 SP1. Elle permet de coder des applications Multi-Touch en WPF, mais aussi, et en cela elle est unique, en Windows Form. Son emploi s'imposera donc si l'on veut absolument travailler avec les Windows Form, ou si l'on craint l'absence de .Net 4 lors du déploiement de l'application. La dernière possibilité consiste à utiliser l'encapsulation de Windows Touch avec .Net 4, qui était disponible en Release Candidate au moment de la rédaction de cet article. Nous avons écrit le code avec Visual Studio 2010, également en Release Candidate. Cette dernière possibilité est de très loin la plus agréable, mais elle ne permet que des applications WPF, Microsoft semblant bien décidé à faire petit à petit oublier les Windows Forms.

1 GESTURES EN .NET 3.5

On téléchargera le code de la librairie à l'adresse donnée plus haut. On compilera le code, ce qui produira des assemblies qu'il suffira de référencer dans vos projets pour être opérationnel. Avec cette librairie, nous nous intéressons aux applications Windows Forms seulement. Voici un premier exemple, disponible sur notre site, qui traite les gestures :

```
using Windows7.Multitouch;
using Windows7.Multitouch.WinForms;

namespace BasicGestureCSharp
{
 public partial class Form1 : Form
 {
   Brush brush = new SolidBrush(SystemColors.WindowText)
   private readonly
     Windows7.Multitouch.GestureHandler gestureHandler;
   private string msg;
   double Angle = 0.0;

 public Form1()
 {
   InitializeComponent();

   if(!Windows7.Multitouch.TouchHandler.
     DigitizerCapabilities.IsMultiTouchReady)
   {
     MessageBox.Show(«Multitouch indisponible»);
     Environment.Exit(1);
   }

   gestureHandler = Factory.CreateHandler<Windows7.
       Multitouch.GestureHandler>(this);

   gestureHandler.Pan += OnPan;
   gestureHandler.Rotate += OnRotate;
   gestureHandler.TwoFingerTap += OnTwoFingerTap;
   gestureHandler.Zoom += OnZoom;
   gestureHandler.PressAndTap += OnPressAndTap;
 }

 private void OnPan(object sender, GestureEventArgs args)
```

```
{
  msg = String.Format(
    «Pan : Location X: {0} Y: {1} Translation Width: {2}, Height{3}»,
    args.Location.X,
    args.Location.Y,
    args.PanTranslation.Width,
    args.PanTranslation.Height);
  Invalidate();
}

private void OnRotate(object sender, GestureEventArgs args)
{
  Angle += args.RotateAngle;
  msg = String.Format(
    «Rotate : Centre X: {0} Y: {1} Angle: {2}»,
    args.Center.X,
    args.Center.Y,
    Angle);
  Invalidate();
}

private void OnTwoFingerTap(object sender, GestureEventArgs args)
{
  msg = String.Format(«Two Finger Tap : Location X: {0} Y: {1}»,
  args.Location.X,
  args.Location.Y);
  Invalidate();
}

private void OnZoom(object sender, GestureEventArgs args)
{
  msg = String.Format(«Zoom : Facteur {0}»,
    args.ZoomFactor);
  Invalidate();
}
```

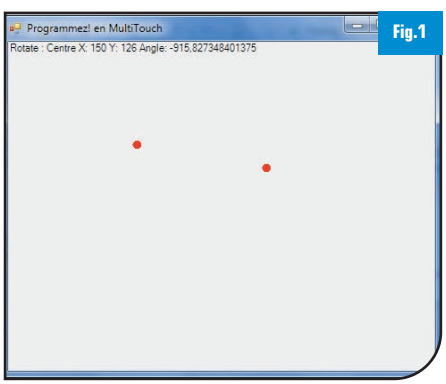


Fig.1

Notre application de reconnaissance des gestures en action

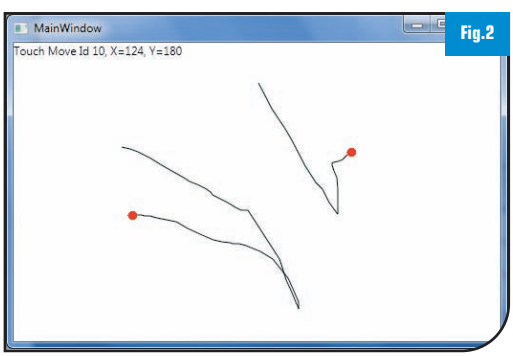


Fig.2

Démonstration rudimentaire de multi-touch en .Net 4

```
 private void OnPressAndTap(object sender, GestureEventArgs args)
 {
   msg = String.Format(«Press and Tap»);
   Invalidate();
 }

 private void OnPaint(object sender, PaintEventArgs e)
 {
   Graphics g = e.Graphics;
   g.DrawString(msg, Font, brush, 0, 0);
 }
 }
}
```

Pour tester ce code [Fig.1], nous utilisons le même émulateur que le mois dernier. Ce code nous apprend, dans le constructeur de la Form, comment vérifier que le MultiTouch est disponible. Puis nous voyons comment mettre en place le mécanisme de réception des événements multi-touch. Nous constatons à cette occasion qu'il n'est nul besoin d'une initialisation particulière pour certaines gestures, contrairement à ce qui se passe en Win32.

2 MULTITOUCH EN .NET 3.5

Voici maintenant un extrait de notre second exemple, BasicTouchC-Sharp disponible sur notre site, qui illustre comment travailler en Multi Touch

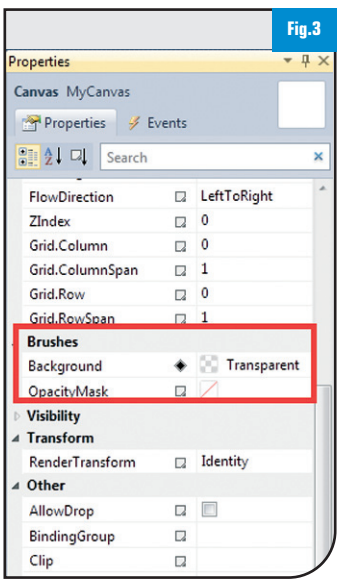
```
public Form1()
{
   //etc
            touchHandler = Factory.CreateHandler<Windows7.
     Multitouch.TouchHandler>(this);

   touchHandler.TouchDown += OnTouchDown;
   touchHandler.TouchMove += OnTouchMove;
   touchHandler.TouchUp += OnTouchUp;
}

private void OnTouchDown(object sender, TouchEventArgs args)
{
 msg = String.Format(
   «TouchDown : Location X: {0} Y: {1} Id: {2}»,
   args.Location.X,
   args.Location.Y,
   args.Id);
 Invalidate();
}
```

Ce code ne présente aucune difficulté. Une seule remarque : alors que l'API Win32 transmet des tableaux de points de contact dans les messages, ici les délégués sont invoqués pour chaque point, ceux-ci étant différenciés par un identificateur. Dans cet exemple, si nous voulions identifier la gesture effectuée par l'utilisateur, nous devrions utiliser les encapsulations des interfaces COM dont nous avons parlé plus haut. Par exemple Windows7.Multitouch.Manipulation.ManipulationProcessor. Nous laissons ce point à la sagacité du lecteur pour aborder le travail le plus intéressant, celui en .Net 4

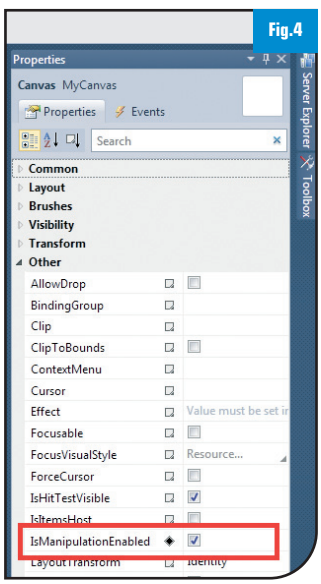
3 LE MULTITOUCH EN .NET 4

Le travail avec .Net 4 est celui qui est destiné à s'imposer dans un très proche avenir. Il est limité à WPF, mais cela se comprend, car la nature vectorielle de WPF convient fort bien aux périphériques tactiles. En outre, les classes .Net procurent le plus haut niveau d'abstraction et présentent l'intérêt majeur de gommer les niveaux Gesture et Touch de l'API Win32, ainsi que nous le verrons un peu plus loin. Commençons par écrire une application très simple, qui trace une ligne suivant le parcours du doigt de l'utilisateur sur le périphérique tactile. [Fig.2]. Vous trouverez cet exemple, DemoTouch, sur notre site. Si toutefois vous souhaitez construire une application avec Visual Studio 2010, voici comment procéder. Créez une application WPF. Nous savons qu'une application Windows Form ne fonctionnerait pas. Placez, à partir du concepteur visuel, un Canvas (par exemple) et configurez-le pour qu'il remplisse la fenêtre. Nous voulons que les événements tactiles arrivent à ce Canvas. Pour cela nous devons faire quelques petites manipulations. La toute première est de définir la brosse (Brush) pour la couleur de fond de la fenêtre [Fig.3]. WPF a ceci de particulier que si aucune brosse n'est définie pour un composant, les événements ne sont pas routés pour celui-ci ! La première chose à faire est donc de définir une brosse dans l'éditeur de propriétés. Une brosse transparente (donner Transparent comme valeur) conviendra très bien. Maintenant nous pouvons recevoir des événements dans le Canvas, mais pas encore les événements tactiles. Pour cela nous devons positionner à true la propriété IsManipulationEnabled dans l'éditeur de propriétés, comme illustré [Fig.4]. Nous en avons terminé avec les configurations. Il suffit maintenant d'écrire des gestionnaires d'événements. Nous le faisons dans le code ci-dessous, pour les événements TouchDown et TouchMove.

```
namespace DemoTouch
{
 public partial class MainWindow : Window
 {
   private TextBlock textblock = new TextBlock();
   private string msg = «»;
   private Brush brush = new SolidColorBrush(Colors.Black);
   private Point point = new Point();
   private int currentId;

public MainWindow()
{
  InitializeComponent();
}

private void OnLoad(object sender, RoutedEventArgs e)
{
  textblock.Text = msg;
  Canvas.SetTop(textblock, 0);
  Canvas.SetLeft(textblock, 0);
  MyCanvas.Children.Add(textblock);
}

private void OnTouchDown(object sender, TouchEventArgs e)
{
  Trace.WriteLine(«Touch Down»);
  TouchPoint tp = e.TouchDevice.GetTouchPoint(MyCanvas);
  msg = String.Format(«Touch Down Id {0}, X={1}, Y={2}»,
    tp.TouchDevice.Id,
    tp.Bounds.X,
    tp.Bounds.Y);
  textblock.Text = msg;
  point.X = tp.Bounds.X;
  point.Y = tp.Bounds.Y;
  currentId = tp.TouchDevice.Id;
  e.Handled = true;
}

private void OnTouchMove(object sender, TouchEventArgs e)
{
  Trace.WriteLine(«Touch Move»);
  if (e.TouchDevice.Id != currentId)
  {
    e.Handled = true;
    return;
  }

  Line line = new Line();
  line.X1 = point.X;
  line.Y1 = point.Y;

  TouchPoint tp = e.TouchDevice.GetTouchPoint(MyCanvas);
  line.X2 = tp.Bounds.X;
  line.Y2 = tp.Bounds.Y;

  line.Stroke = System.Windows.Media.Brushes.Black;
  MyCanvas.Children.Add(line);
  point.X = tp.Bounds.X;
  point.Y = tp.Bounds.Y;
  currentId = tp.TouchDevice.Id;


  msg = String.Format(«Touch Move Id {0}, X={1}, Y={2}»,
    tp.TouchDevice.Id,
    tp.Bounds.X,
```



En WPF, un composant doit avoir une brosse définie pour recevoir des composants.



Nous devons demander explicitement à recevoir les événements tactiles.

```
      tp.Bounds.Y);
    textblock.Text = msg;

    e.Handled = true;
    }
  }
}
```

Ce code est très simple, mais il mérite quelques remarques. La plus importante est qu'il n'est pas multi-point. Chaque contact tactile prend la main sur le précédent. La raison en est simple : de même qu'avec la librairie pour .Net 3.5, les classes .Net ne fournissent pas un tableau de points de contacts dans les événements, mais génèrent des événements pour chaque point de contact, événements qui tous aboutissent au même délégué. Il est donc du travail du programmeur de repérer à chaque fois de quel contact il s'agit en examinant la propriété Id de la propriété TouchDevice de l'événement passé en argument au délégué. Pour obtenir un vrai comportement multi-touch, on peut par exemple gérer les points de contact dans un dictionnaire. Ceci ne pose aucune difficulté et est laissé au lecteur.

4 DES MANIPULATIONS AVEC .NET 4

Nous abordons maintenant les manipulations [Fig.5]. Notre exemple est construit comme le précédent. Toutefois les événements ne seront plus reçus par le Canvas, mais directement par l'objet Image que nous avons placé dans le Canvas. Pour cela, la propriété IsManipulationEnabled du Canvas ne sera plus positionnée à true, mais c'est celle de l'Image qui le sera. Le code complet de cet exemple, DemoManipulation, est disponible sur notre site :
Ce n'est que du calcul matriciel effectué à chaque événement reçu. Le runtime envoie ces événements tout au long du mouvement. Comme on le voit dans l'exemple, on ne se préoccupe pas de savoir quel type de mouvement est en cours. On applique simplement le calcul sans questions existentielles et c'est bien pratique.

5 LES MOUVEMENTS À INERTIE

Dans les exemples précédents, les mouvements s'arrêtent net dès que l'utilisateur n'agit plus. Pour des applications tactiles «qui en jettent», on préfère que les mouvements s'amortissent : c'est l'inertie. Pour obtenir cet effet, il suffit de le définir au départ d'un mouvement, dans un gestionnaire d'événement particulier :

Fig.5

Démonstration de manipulation sous .Net 4

```
private void OnManipulationInertiaStarting(object sender,
ManipulationInertiaStartingEventArgs e)
{
 Trace.WriteLine(«Inertia Manipulation Starting»);
 e.TranslationBehavior.DesiredDeceleration = 1.0 * 96.0 /
(1000.0 * 1000.0);
 e.RotationBehavior.DesiredDeceleration = 120 / (1000.0 *
 1000.0);
 e.ExpansionBehavior.DesiredDeceleration = 1.0 * 96 / (1000.0
* 1000.0);
 //e.TranslationBehavior.DesiredDisplacement = 100.0;
 //e.RotationBehavior.DesiredDeceleration = 120.0;
 //e.ExpansionBehavior.DesiredDeceleration = 10.0;

 e.Handled = true;
}
```

Nous renvoyons le lecteur à l'exemple DemoInertie sur notre site pour le code complet. Nous avons deux choix possibles. L'inertie proprement dite, ce que fait notre code, ou simplement définir un mouvement résiduel, ce que fait le code en commentaire. L'effet produit par la deuxième possibilité est beaucoup moins réaliste à mon humble avis. Les formules données pour les inerties viennent de la documentation MSDN. Avec les valeurs proposées, le logo de programmez! freine aussi bien qu'une savonnette mouillée dans une baignoire et il va quitter votre écran au moindre courant d'air.

6 LIMITER LES MOUVEMENTS

Il est donc souhaitable de limiter les mouvements dans certaines situations. Par exemple nous pouvons imiter le comportement de l'interface du Palm Pre (cf. Programmez! 131) en empêchant notre logo de sortie latéralement, mais en le laissant sortir verticalement. Pour obtenir cela, il suffit d'ajouter un tout petit peu de code dans le gestionnaire d'événement OnManipulationDelta de l'exemple précédent (code complet DemoInertieLimites sur notre site) :

```
if (e.IsInertial)
{
 double centrex = centre_image.X + fe.ActualWidth / 2;

 if (centrex <= 0 || centrex >= MyCanvas.ActualWidth)
 {
   e.ReportBoundaryFeedback(e.DeltaManipulation);
   e.Complete();
   e.Handled = true;
   return;
 }
}
```

On signale que le bord de la fenêtre est atteint avec la méthode e.ReportBoundaryFeedback. Mais cela ne suffit pas pour arrêter la procédure d'inertie. Si l'on ne fait rien de plus, notre logo continuera de se cogner à l'infini sur le bord de notre fenêtre qui en serait toute tremblante. Nous devons donc invoquer e.complete() pour signaler au runtime que la procédure d'inertie est terminée.

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

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

Economisez jusqu'à 50%

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 !
www.programmez.com

1 -25%

Abonnement 1 an

49€ au lieu de 65,45 € tarif au numéro - *Tarif France métropolitaine*

2 +0,8€ par mois

Abonnement Intégral : + archives

1 an au magazine + archives sur Internet et PDF

59€ *Tarif France métropolitaine*

3 jusqu'à -50%

Abonnement 2 ans + 1 livre numérique ENI

- **79€** au lieu de 130,90 (valeur de 22 numéros) *Tarif France métropolitaine* + un livre d'une valeur de 23,9 € à 31,9 €, soit un total de 154,8 € à 162,8 €
- **89€** **2 ans au magazine + archives sur Internet et PDF + 1 livre numérique ENI**

OUI, *je m'abonne* *Vous pouvez vous abonner en ligne et trouver tous les tarifs* www.programmez.com

- ☐ **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- ☐ **Abonnement Intégral : 1 an au magazine + archives : 59 €** *Tarif France métropolitaine*
- ☐ **Abonnement 2 ans au magazine + livre numérique ENI : 79 €** *Offre France métropolitaine*
- ☐ **Abonnement 2 ans au magazine + livre numérique ENI + archives : 89 €** *Offre France métropolitaine*

Livres à Choisir : ☐ Visual Studio 2010 ☐ PHP5.3 ☐ Bing Maps ☐ MySQL 5, Administration et optimisation
☐ Java et Spring, Concevoir, construire et développer une application Java/J2EE avec Spring. *Détails sur www.programmez.com/abonnement.php*

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Tél : **(Attention, e-mail indispensable)**

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

Offre limitée, valable jusqu'au 30 septembre 2010

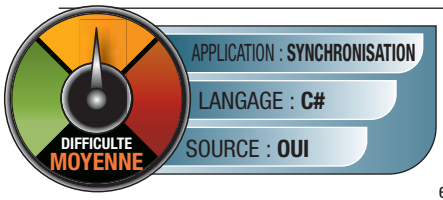
Le renvoi du présent bulletin implique pour le souscripteur l'acceptation pleine et entière de toutes les conditions de vente de cette offre. Conformément à la loi Informatique et Libertés du 05/01/78, vous disposez d'un droit d'accès et de rectification aux données vous concernant. Par notre intermédiaire, 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 133

Le magazine du développement PROgrammez!

Synchroniser vos fichiers sous Windows avec le framework Sync

Avec la multiplication des périphériques de stockage, des ordinateurs portables et/ou en réseau, garder tous ses documents à jour est devenu difficile. Le framework Sync propose une solution élégante, et à la portée de tous.



De nos jours il est très courant de travailler sur plusieurs postes, de passer des fichiers d'un poste à l'autre par des clés USB, ou encore d'être itinérant avec un ordinateur portable qui ne sera connecté que périodiquement au réseau de l'entreprise. Dans toutes ces situations, et il en existe beaucoup d'autres similaires, se pose le problème d'avoir des documents dans une seule et même version sur tous les postes ou tous les périphériques de stockage, autrement dit, d'avoir des données synchronisées.

1 PRÉSENTATION SUCCINCTE DE SYNC

Sync est une plate-forme de synchronisation complète qui prend en charge n'importe quel type de périphérique de stockage et n'importe quel protocole réseau. Si aujourd'hui nous nous intéressons à la synchronisation de fichiers, Sync ne se limite pas à cela, mais au contraire permet de manipuler des types de données personnalisés et également de synchroniser des bases de données SQL Server ou encore des données issues du Web, comme des flux RSS. Pour cela Sync s'articule autour du concept de Provider. Un Provider est un composant logiciel qui sait travailler avec les données d'une source et qui, en même temps, gère des meta-données pour cette source. C'est par l'analyse de ces méta-données qu'un Provider déduira les opérations à effectuer. Le framework Sync vient avec un certain nombre de Providers tout prêts. Ceux-ci peuvent servir de base pour l'écriture d'un Provider dédié à un type de donnée particulier.

Sync supporte tout type de périphérique et tous les protocoles réseau.

2 LES OUTILS

Concrètement, Sync est basé sur la technologie COM de Microsoft. Technologie avec laquelle nous avons souvent travaillé, et dont nous savons qu'elle peut vite conduire à l'écriture d'un code imbuvable en C ou C++. Ceci est encore plus vrai en ce qui concerne Sync, car dans ce cas il est nécessaire d'implémenter certaines interfaces COM pour déclarer des fonctions de rappel. Fort heureusement Sync vient avec des assemblies .Net qui encapsulent tout cela. Considérant que pour des opérations de synchronisation la performance du code ne compte pas devant les opérations de lecture/écriture sur disque ou à travers le réseau, nous laisserons sagement C++ au profit de C# qui est idéal. N'importe quel autre langage .Net conviendra également, bien entendu. Nos exemples ont été écrits avec Visual Studio et sont disponibles sur notre site. Pour pouvoir les utiliser vous devez télécharger Sync à partir du site de Microsoft (http://www.microsoft.com/downloads/details.aspx?displaylang=fr&FamilyID=89adbb1e-53ff-41b5-ba17-8e43a2e66254) Il existe une version avec documentation en français. Une fois Sync installé vous trouverez, dans le sous-répertoire runtime les assemblies Microsoft.Synchronizationet Microsoft.Synchronization.Files qui devront être ajoutées en référence à tous nos exemples. Il n'est pas nécessaire que Sync soit présent sur toutes les machines et périphériques à synchroniser. Seule sa présence sur la machine qui héberge l'application de synchronisation est requise.

3 UN EXEMPLE BASIQUE

Voici un code qui synchronise le contenu de deux répertoires maître (master) et esclave (slave), sur le disque D: (le lecteur adaptera le code en fonction de ses besoins propres) :

```
using Microsoft.Synchronization;
using Microsoft.Synchronization.Files;

namespace MasterSlave
{
 class Program
 {
   private string masterpath = «D:\\master»;
   private string slavepath = «D:\\slave»;

   private void Run()
   {
     FileSyncProvider masterProvider =
       new FileSyncProvider(masterpath);
     FileSyncProvider slaveProvider =
       new FileSyncProvider(slavepath);

     SyncOrchestrator orchestrator = new SyncOrchestrator();
     orchestrator.LocalProvider = masterProvider;
     orchestrator.RemoteProvider = slaveProvider;
     orchestrator.Direction = SyncDirectionOrder.Upload;
     orchestrator.Synchronize();
   }

   static void Main(string[] args)
   {
```

```
        Program p = new Program();
        p.Run();
    }
  }
}
```

Quelques lignes de C# en tout et pour tout et voici le contenu de nos deux répertoires synchronisés, y compris en cas de présence de sous-répertoires. On ne peut imaginer plus simple. Ici nous travaillons sur un même disque par commodité pour les manipulations, mais cela ne présente bien sûr pas d'intérêt pratique. Voulez-vous synchroniser vos données sur une clé USB identifiée par F: ? Il vous suffit de modifier une seule ligne :

```
private string slavepath = «F:\\slave»;
```

Si vous souhaitez synchroniser vos données sur un ordinateur nommé par exemple SOLEIL, c'est tout aussi simple :

```
private string slavepath = «\\\\SOLEIL\\slave»;
```

Dans de dernier cas, cela tombe sous le sens, le répertoire distant doit être un répertoire partagé et vous devez avoir la permission d'y accéder. Le principe de notre code est très simple. On instancie deux Provider, un par répertoire. On les met ensuite en relation par le biais de l'objet SyncOrchestrator. Vient ensuite un point très important, on définit le sens de la synchronisation, puis on lance celle-ci. Dans notre exemple, nous avons choisi *Upload* comme sens de synchronisation. Cela signifie que si des fichiers sont présents sur *master* et qu'ils ne sont pas présents sur *slave*, alors les dits fichiers seront dupliqués sur *slave*. Cette opération est unidirectionnelle. Ainsi si vous supprimez manuellement un fichier sur *slave* et que vous relancez la synchronisation, le fichier ne sera pas uploadé une seconde fois sur *slave* depuis *master*, le runtime considérant que l'opération a déjà été effectuée. Comment le runtime déduit-il cela ? Lors de la première opération, aucune metadonnée n'était présente dans aucun répertoire. Le runtime a donc uploadé le contenu entier de *master*. Puis il a écrit ces fameuses métadonnées, afin de réduire les opérations lors de la prochaine synchronisation. Ces métadonnées résident dans les fichiers filesync.metadata qui sont apparues après la première synchronisation. Pour avoir une synchronisation bidirectionnelle, il suffit d'écrire :

```
orchestrator.Direction = SyncDirectionOrder.UploadAndDownLoad;
```

ou

```
orchestrator.Direction = SyncDirectionOrder.DownloadAndUpload;
```

Question piège: si en l'état nous donnons UploadAndDownLoad et que nous relançons la synchronisation, le fichier manuellement supprimé sur *slave* sera-t-il uploadé de nouveau ? La réponse est : non seulement il ne sera pas uploadé de nouveau, mais le fichier équivalent sur *master* sera effacé ! Par quelle magie ? Par la magie des métadonnées. En elles est consigné le fait que la synchronisation de tous les fichiers a été effectuée une première fois. Si depuis un fichier a été supprimé, demander une synchronisation bidirectionnelle revient à demander la répercussion de la dernière opération en date sur tous les répertoires, donc à demander la suppression du fichier. À la réflexion, tout cela est logique, mais on en conclura que ce n'est pas une bonne idée d'écrire une application de synchronisation qui permette à l'utilisateur d'en changer la direction. Il est à peu près certain que tôt ou tard l'utilisateur fera une erreur de raisonnement et perdra des données.

4 APPLIQUER UN FILTRE

Il est possible d'exclure des fichiers d'une opération de synchronisation. Pour cela on définit un filtre, comme dans l'exemple ci-dessous, extrait de MasterSlaveFilter sur notre site. Seules quelques lignes sont nouvelles :

```
FileSyncScopeFilter filtre = new FileSyncScopeFilter();
filtre.FileNameExcludes.Add(«nocopy*.*»);
filtre.FileNameExcludes.Add(«*.doc*»);

FileSyncProvider masterProvider =
  new  FileSyncProvider(masterpath,  filtre,  FileSyncOptions
.None);
FileSyncProvider slaveProvider = new FileSyncProvider(slavepath);
// etc, etc.
```

Dans cet exemple des fichiers Word, .doc et .docx seront exclus de la synchronisation. Un fichier nommé nocopythisfile.txt le sera également.

5 DÉFINIR DES CORBEILLES

Les exemples précédents présentent l'avantage d'une extrême simplicité, mais on voit, et c'est dommage, qu'ils ne permettent pas de conserver quelque part une copie des données modifiées en cas de conflit. On a par exemple conflit quand un fichier est modifié manuellement sur le master et sur l'esclave. Cette possibilité de conserver une copie est toutefois offerte par le framework. Ainsi il est possible de définir des répertoires 'Corbeille', ou d'utiliser la corbeille du système (Recycle Bin) afin que des anciennes versions de fichiers y soient déposées. Le code est à peine plus compliqué, nous renvoyons le lecteur au source MasterSlaveCorbeille sur notre site.

6 ETUDIER LE COMPORTEMENT DU RUNTIME EN CAS DE CONFLIT

Pour ne pas avoir de mauvaises surprises, il est bon d'expérimenter avec le runtime. Ainsi par exemple vous constaterez qu'il est sensible à la casse des noms de fichiers, alors que Windows ne l'est pas. Vous constaterez que ses réactions en cas de conflit ne sont pas toujours ce à quoi on s'attend. Prenons deux exemples, en mode bidirectionnel. Soit un conflit de création pour notre premier exemple. On crée manuellement un fichier sur l'esclave, puis on crée manuellement un fichier de même nom (attention à la casse !) sur le maître. Pour résoudre le conflit, le runtime se basera sur les timbres à date des fichiers. Le fichier sur le maître, parce que plus récent est le gagnant du conflit. Ce fichier est donc uploadé sur l'esclave et le fichier perdant de conflit est placé dans la corbeille du perdant de conflit (donc l'esclave). Ce comportement est conforme à la documentation qui nous dit que les fichiers sont toujours placés dans les corbeilles des perdants de conflit. Tout va bien. Mais j'ai eu l'occasion de constater des comportements différents en cas de conflit de modification, comme dans ce deuxième exemple : soit deux fichiers texte identiques (synchronisés), un sur le maître et un sur l'esclave. On modifie manuellement le contenu du fichier texte sur le maître, puis on modifie manuellement le contenu du fichier texte sur l'esclave. La version sur l'esclave sera la dernière en date, donc la gagnante du conflit. Elle sera donc downloadée sur le maître. Jusqu'ici tout est normal. Le maître étant perdant de conflit, son fichier perdant devrait être placé dans la corbeille du maître.

Pourtant votre serviteur a eu la surprise de voir le fichier uploadé dans la corbeille de l'esclave, c'est-à-dire dans la corbeille du gagnant de conflit, et ceci contrairement à la documentation (est-ce un bug du runtime ?). En revanche, si dans ce conflit c'est le maître qui est gagnant (car modifié en second) alors le comportement est conforme à la logique et à la documentation. Mentionnons en passant qu'il est possible de forcer par programmation les gagnants de conflit. Nous renvoyons le lecteur à la documentation;

7 PRÉVISUALISER UNE SYNCHRONISATION ET AGIR SUR LE DÉROULEMENT DE CELLE-CI

Une synchronisation étant susceptible de nous réserver des surprises, il sera intéressant de pouvoir prévisualiser les résultats de l'opération avant de lancer véritablement celle-ci, et éventuellement d'agir sur le déroulement de l'opération réelle. Le runtime émet divers événements lors d'une opération de synchronisation. L'événement AppliedChange signale qu'une opération élémentaire (le transfert d'un fichier par exemple) a été effectuée. L'événement ApplyingChange est déclenché quand une opération élémentaire débute. Si à ce moment la propriété SkipChange de l'événement reçue est positionnée à true, alors l'opération élémentaire ne sera pas effectuée et un événement SkippedChange sera déclenché à la place. Nous parlons ici d'une opération de synchronisation réelle. Ainsi l'exemple ci-dessous (MasterSlavePreview, au complet sur notre site) ne traitera pas un fichier de nom programmez.txt, s'il en rencontre un.

```
using System;
using Microsoft.Synchronization;
using Microsoft.Synchronization.Files;

namespace MasterSlavePreview
{
 class Program
 {
   private bool previewmode = false;
   private string masterpath = «D:\\master»;
   private string slavepath = D:\\slave»;
   private string mdfile = «filesync.metadata»;
   private string tmp = «D:\\temp»;
   private string masterCorbeille = «D:\\mastercorbeille»;
   private string slaveCorbeille = «D:\\slavecorbeille»;

   private void Run() {
     // pas de filtre
     FileSyncProvider masterProvider =
       new FileSyncProvider(masterpath, null,
         FileSyncOptions.None,
         masterpath, mdfile, tmp,
         masterCorbeille);
     masterProvider.PreviewMode = previewmode;

   masterProvider.AppliedChange +=
     new EventHandler<AppliedChangeEventArgs>(
       OnMasterAppliedChange);
   masterProvider.ApplyingChange +=
     new EventHandler<ApplyingChangeEventArgs>(
       OnMasterApplyingChange);
   masterProvider.SkippedChange +=
     new EventHandler<SkippedChangeEventArgs>(
       OnMasterSkippedChange);

   FileSyncProvider slaveProvider =
     new FileSyncProvider(slavepath, null,
       FileSyncOptions.None,
       slavepath, mdfile, tmp,
       slaveCorbeille);
   slaveProvider.PreviewMode = previewmode;

   slaveProvider.AppliedChange +=
     new EventHandler<AppliedChangeEventArgs>(
       OnSlaveAppliedChange);
   slaveProvider.ApplyingChange +=
     new EventHandler<ApplyingChangeEventArgs>(
       OnSlaveApplyingChange);
   slaveProvider.SkippedChange +=
     new EventHandler<SkippedChangeEventArgs>(
       OnSlaveSkippedChange);

   SyncOrchestrator orchestrator = new SyncOrchestrator();
   orchestrator.LocalProvider = masterProvider;
   orchestrator.RemoteProvider = slaveProvider;
   orchestrator.Direction = SyncDirectionOrder.UploadAndDownload;
   orchestrator.Synchronize();
   }

 private void OnMasterAppliedChange(Object sender,
   AppliedChangeEventArgs evt) {
   Console.WriteLine(«Evenement OnMasterAppliedChange»);

   switch (evt.ChangeType) {
   case ChangeType.Create:
     Console.WriteLine(«Applied CREATE « + evt.NewFilePath);
     break;
   case ChangeType.Delete:
     Console.WriteLine(«Applied DELETE « + evt.OldFilePath);
     break;
   case ChangeType.Update:
     Console.WriteLine(«Applied OVERWRITE « +
       evt.OldFilePath);
     break;
   case ChangeType.Rename:
     Console.WriteLine(«Applied RENAME « + evt.OldFilePath
       + « -> « + evt.NewFilePath);
     break;
   default:
     break;
   }
   Console.WriteLine();
 }

 private void OnMasterApplyingChange(Object sender,
   ApplyingChangeEventArgs evt) {
   Console.WriteLine(«Evenement OnMasterApplyingChange»);
   if (evt.NewFileData.Name.Equals(«programmez.txt»)) {
```

```
    evt.SkipChange = true;
    return;
  }

  switch (evt.ChangeType) {
  case ChangeType.Create:
    Console.WriteLine(«Applying CREATE « +
      evt.NewFileData.Name);
    break;
  case ChangeType.Delete:
    Console.WriteLine(«Applying DELETE « +
      evt.CurrentFileData.Name);
    break;
  case ChangeType.Update:
    Console.WriteLine(«Applying OVERWRITE « +
      evt.CurrentFileData.Name);
    break;
  case ChangeType.Rename:
    Console.WriteLine(«Applied RENAME « +
      evt.CurrentFileData.Name
      + « -> « + evt.NewFileData.Name);
    break;
  default:
    break;
  }
  Console.WriteLine();
}

private void OnMasterSkippedChange(Object sender,
  SkippedChangeEventArgs evt) {
  Console.WriteLine(«Skipped « + evt.NewFilePath);
  Console.WriteLine(«Reason: « + evt.SkipReason.ToString());
}

private void OnSlaveAppliedChange(Object sender,
  AppliedChangeEventArgs evt) {
  Console.WriteLine(«Evenement OnSlaveAppliedChange»);
  switch (evt.ChangeType) {
  case ChangeType.Create:
    Console.WriteLine(«Applied CREATE « + evt.NewFilePath);
    break;
  case ChangeType.Delete:
    Console.WriteLine(«Applied DELETE « + evt.OldFilePath);
    break;
  case ChangeType.Update:
    Console.WriteLine(«Applied OVERWRITE « +
      evt.OldFilePath);
    break;
  case ChangeType.Rename:
    Console.WriteLine(«Applied RENAME « + evt.OldFilePath
      + « -> « + evt.NewFilePath);
    break;
  default:
    break;
  }
  Console.WriteLine();
  }
```

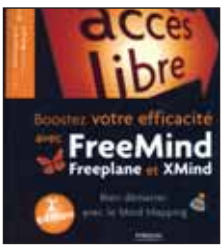
```
 private void OnSlaveApplyingChange(Object sender,
   ApplyingChangeEventArgs evt) {
   Console.WriteLine(«Evenement OnSlaveApplyingChange»);

   if (evt.NewFileData.Name.Equals(«programmez.txt»)) {
     evt.SkipChange = true;
     return;
   }

   switch (evt.ChangeType) {
   case ChangeType.Create:
     Console.WriteLine(«Applying CREATE « +
       evt.NewFileData.Name);
     break;
   case ChangeType.Delete:
     Console.WriteLine(«Applying DELETE « +
       evt.CurrentFileData.Name);
     break;
   case ChangeType.Update:
     Console.WriteLine(«Applying OVERWRITE « +
       evt.CurrentFileData.Name);
     break;
   case ChangeType.Rename:
     Console.WriteLine(«Applied RENAME « +
       evt.CurrentFileData.Name
                        + « -> « + evt.NewFileData.Name);
     break;
   default:
     break;
   }
   Console.WriteLine();
   }

 private void OnSlaveSkippedChange(Object sender,
   SkippedChangeEventArgs evt) {
   Console.WriteLine(«Skipped « + evt.NewFilePath);
   Console.WriteLine(«Reason: « + evt.SkipReason.ToString());
 }

 static void Main(string[] args) {
   Program p = new Program();
   p.Run();
 }
 }
}
```

Si maintenant, avec ce même code, on passe les Provider en mode preview en positionnant à true leur propriété PreviewMove, alors seuls les événements AppliyngChange seront déclenchés, sans aucun effet sur les données, ce qui permet à l'application de constituer un journal et de le présenter à l'utilisateur qui pourra alors décider en connaissance de cause s'il lance ou non l'opération réelle. Notons qu'en mode preview, les événements AppliedChange ne sont jamais déclenchés.

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

PENSÉE

Boostez votre efficacité avec FreeMind



*Difficulté : ***
Editeur : Eyrolles
Auteur : collectif
Prix : 25 €

Vous avez des idées, des suggestions ? Mais comment les conserver, les organiser, les utiliser ? Il existe des outils spécialisés pour le faire comme FreeMind. Il permet de créer des cartes mentales et d'organiser idées et connaissances de façon visuelle. Il s'utilise sous Windows, Linux ou MacOS X et est open source. Cet ouvrage vous fait découvrir un domaine peu connu, voire inconnu. On débute par la prise en main et les fonctionnalités de base puis, les auteurs abordent la cartographie des idées, les notions de brainstorming et la manière dont se constitue une base de données. Vous saurez aussi comment utiliser au quotidien un tel outil : Scripting, alertes, comment créer des documents de projets, un CV, etc. A découvrir !

WEB

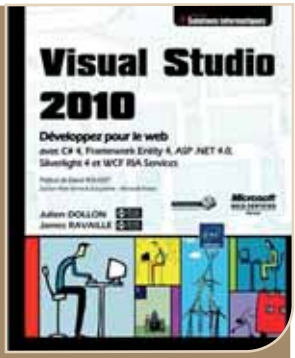
Concevez votre site web avec PHP et MySQL

*Difficulté : ***
Editeur : leLivreduZéro
Auteur : Mathieu Nebra
Prix : 25 €

Vous connaissez le HTML et vous avez toujours rêvé de créer un site web dynamique, avec votre propre blog, vos forums et votre espace membres ? Ne cherchez plus ! Découvrez dans ce livre dédié aux débutants comment utiliser les outils les plus célèbres du web dynamique : PHP et MySQL ! L'auteur fait tout pour simplifier le plus possible les notions de code et l'apprentissage technique qui est fatalement incontournable. Nous retrouvons les chapitres classiques : les variables, la gestion des données, les fonctions, les boucles, etc. Nous avons beaucoup aimé les annexes, notamment la nécessité de coder proprement, comment faire du debug de script ou encore bien utiliser le fichier htaccess. Pour débuter, il s'agit d'un excellent livre !

LIVRE DU MOIS

Visual Studio 2010

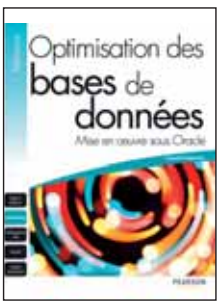


*Difficulté : *** - Editeur : éditions Eni*
Auteur : Julien Dollon & James Ravaille - Prix : 37 €

Voici un indispensable pour celles et ceux qui développent en .Net 4 et avec Visual Studio 2010. Il cible les développeurs et concepteurs déjà habitués à C# et .Net 4. Le fil rouge est la mise en œuvre d'un ensemble de technologies (C# 4, Framework Entity 4, ASP .NET 4.0, Silverlight 4 et WCF RIA Services) permettant de développer une application web à travers la réalisation de projets de gestion de données. Dans un premier temps, après avoir introduit les nouveautés de la version 4 du langage C# et présenté l'architecture des applications, les auteurs abordent la création de composants d'accès et de gestion des données avec le Framework Entity 4 puis la maîtrise de ASP.Net, Ajax et jQuery, sans oublier Silverlight 4. Le livre aborde aussi les services WCF, WCF Data Services et WCF RIA Services. Bref une bonne découverte du nouvel IDE de Microsoft et des différentes piles de .Net et du développement web.

SGBD

Optimisation des bases de données

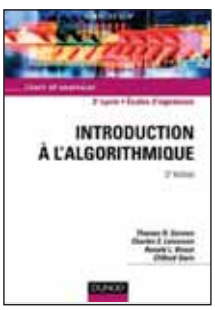


*Difficulté : ****
Editeur : Pearson Education
Auteur : Laurent Navarro
Prix : 26 €

Cet ouvrage a pour objectif de mettre à la portée des développeurs les connaissances utiles à l'optimisation des bases de données. Cette activité est souvent confiée aux administrateurs de bases de données (DBA) une fois les projets terminés, c'est alors au niveau du développement qu'il faut se pencher sur la problématique des performances. Optimiser les données demeure un problème récurrent et peu aisé pour le développeur. Si l'auteur cible le système Oracle, les pratiques peuvent être adaptées à d'autres plates-formes.

TECHNIQUE

Introduction à l'Algorithmique



*Difficulté : *****
Editeur : Dunod
Auteur : collectif
Prix : 59,90 €

Enorme ! Voilà le mot que nous pourrions utiliser pour qualifier ce pavé de 1 200 pages et presque 2 kilos ! Mais au-delà du poids, c'est la maîtrise du sujet par les auteurs et les apports de l'ouvrage à l'approche mathématique du développement, car n'oublions pas que la programmation utilise et abuse de l'algorithmie. Les algorithmes sont rédigés en français et dans un pseudo-code proche des langages Pascal, C et Java. Ils sont analysés en profondeur et complétés par des preuves mathématiques. De nombreux exemples, figures, études de cas et exercices de difficulté graduée complètent les explications. Un must !

MÉTHODOLOGIE

CMMi par l'exemple

*Difficulté : ****
Editeur : Eyrolles
Auteur : François Dufay
Prix : 29,90 €

Le CMMi n'est pas une nouveauté. Pour rappel, il s'agit d'un ensemble de bonnes pratiques pour évaluer et améliorer les systèmes, la fourniture de services. C'est un complément à l'approche ITIL ou encore à CobIT, car chacun agit dans son domaine : gouvernance, production. L'auteur se propose ici de vous plonger dans CMMi et de savoir comment, et où, déployer et utiliser cette approche, qui n'est pas neutre sur l'entreprise, les équipes, l'organisation interne. Deux modèles sont particulièrement mis en avant : CMMi-Dev et CMMi-ACQ. A lire et à comprendre.

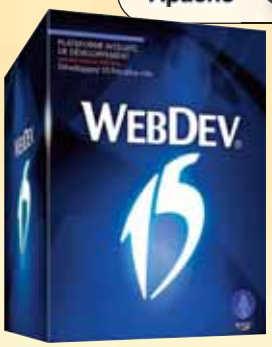
DÉVELOPPEZ
10 FOIS
PLUS VITE

CRÉEZ VOS SITES DYNAMIQUES 10X+ VITE AVEC WEBDEV® 15

EXEMPLES DE SITES EN WEBDEV®

Parmi les **dizaines de milliers** de sites réalisés en WEBDEV, en voici quelques-uns. Retrouvez-en plus de 1000 sur www.pcsoft.fr.

Apache php



Compatible WINDEV et WINDEV Mobile

WEBDEV® 15

ENVIRONNEMENT PROFESSIONNEL DE DÉVELOPPEMENT (AGL)

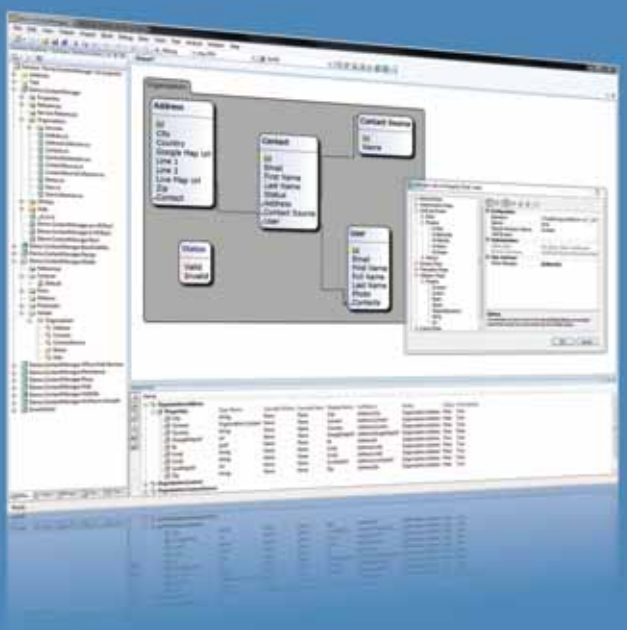
Créez des sites dynamiques (sites temps réel, reliés à des bases de données) Internet, Intranet et SaaS.

PHP, J2EE, Webservices, XML, Ajax, Linux, Windows...

www.pcsoft.fr

Demandez votre dossier gratuit 200 pages Tél: **04.67.032.032** info@pcsoft.fr

Besoin D'AIDE ?
CODEFLUENT ENTITIES



ELÉMENTS APPLICATIFS

- *Localisation*
- *Liaison aux données*
- *Règles & Validation*
- *Concurrence d'accès*
- *Sécurité*
- *Cache*
- *Gestion des BLOBS*

ARCHITECTURES

- *SOA, Client intelligent*
- *Client riche, RIA*
- *Web, Webparts*
- *Client/Serveur, N-Tier*
- *Office*
- *SharePoint*
- *SaaS, Cloud*

TECHNOLOGIES

- *.NET, C#, Linq, Visual Studio*
- *ASP .NET WebForms, MVC*
- *Silverlight, WCF, ASMX*
- *WPF, WinForms*
- *Microsoft SQL Server*
- *Oracle Database*
- *Excel, Access, SharePoint*

LA PREMIERE FABRIQUE LOGICIELLE PLEBISCITEE PAR LES CLIENTS

Face aux enjeux associés à l'**évolution constante des technologies** et à leur intégration, CodeFluent Entities, la fabrique logicielle pilotée par les modèles apporte une solution concrète immédiate.

CodeFluent Entities offre aux architectes et aux développeurs **une méthode structurée** et tous les **outils** pour faciliter le développement des applications .NET, sur tout type d'architecture, à partir d'une **modélisation du métier.**
CodeFluent Entities permet de réaliser des applications **évolutives** de **qualité** avec un niveau de **productivité** inégalé.

*Téléchargez votre licence **GRATUITE** ici :*
http://www.CodeFluentEntities.com/programmez

CodeFluent Entities
the Model Driven Software factOry

Contact: info@softfluent.com
www.CodeFluentEntities.com
***SoftFluent** 3 rue de la Renaissance 92160 ANTONY*