Supplément
Team Foundation Server

PROGRAMMEZ !

Le magazine du développement

www.programmez.com

Team Foundation Server : Tout pour la qualité logicielle

- Cycle de vie des applications : pourquoi y aller ?
- Les différents tests
- Comment automatiser les tests sous Visual Studio ?
- TFS Lab Management, l'art de maîtriser les tests
- Découvrez la prochaine version

Microsoft® Visual Studio®

édito

Qualité logicielle, vous ferez !

Une fois n'est pas coutume, nous parlerons de tests, de cycle de vie de l'application, de bonnes pratiques, bref tout ce qui fait la qualité logicielle. Impensable de déployer en entreprise ou auprès du public, une application, quelle qu'elle soit, sans avoir respecté un minimum de règles de développement, de suivi de projet, ou de bonnes pratiques sur le code, l'architecture et les tests.

La qualité logicielle ne s'improvise pas. Il faut utiliser les bons outils, les bonnes méthodes et règles au bon moment. Mais l'outillage n'est pas une finalité. Car durant des années, ces outils restaient sagement dans les armoires sans être utilisés. Il faut que les développeurs, chefs de projet, testeurs, utilisateurs, architectes, soient sensibilisés, formés à mettre en œuvre des suites logicielles telles que Team Foundation Server (TFS) de Microsoft.

Dites-vous que la qualité logicielle n'est pas négociable dans un monde informatique hétérogène, massivement distribué et connecté, où la moindre erreur de fonctionnement peut figer l'activité d'une entreprise toute entière. Le cloud computing impose une nouvelle rigueur ainsi que les applications smartphone, dans un monde de plus en plus souvent connecté aux serveurs. Mais au-delà de la technique pure, la communication entre chaque membre de l'équipe est vitale. La bonne information au bon moment prend tout son sens.

Dans ce cahier spécial, nous allons nous plonger dans TFS. Nous aborderons les bonnes pratiques (notamment Scrum), comment définir et mettre en place un plan de tests, comment automatiser les tests, utiliser toute la puissance de Lab Manager. Enfin, nous ferons un point sur la prochaine version.

François Tonic

sommaire

Supplément de PROGRAMMEZ N°149 Février 2012 - Ne peut être vendu séparément.
Rédaction : François Tonic, rédacteur en chef. Experts : Vivien Fabing, Sylvain Gross, Laurent Abid, Didier Davar, Salam Elias, Partrick Catro- Brouillet, Cedric Noureau, Guillaume Rouchon, Patrice Lamarche.
Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris. Directeur de la publication : Jean-Claude Vaudecrane. Dépôt légal : à parution
Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, Bruxelles.

A vous de choisir !

Avec Visual Studio (VS) 2010, Microsoft a rationalisé l'ensemble de la gamme afin de la rendre plus compréhensible. D'autre part, pour répandre l'usage de l'ALM, TFS est disponible par défaut avec plusieurs éditions de Visual Studio 2010 :

Editions de Visual Studio 2010	TFS
Professional avec MSDN	Oui
Premium avec MSDN	Oui
Ultimate avec MSDN	Oui
Test Professional avec MSDN	Oui

TFS peut être acheté séparément si vous n'utilisez pas VS 2010 (version antérieure ou outils non .Net). Précision importante : TFS 2010 est disponible uniquement version 64-bit. D'autre part, selon la version de VS 2010, le niveau de fonctionnalités dans les tests, ALM... diffère beaucoup. Voici un petit tableau explicatif :

	VS 2010 Professional avec MSDN	VS 2010 Premium avec MSDN	VS 2010 Ultimate avec MSDN	VS 2010 Test Professional avec MSDN
Debugging et diagnostics	●	●	●●●	
Outils de tests	●	●●	●●●●	●●●
IDE	●●●●	●●●●	●●●●	●
Développement pour bases de données	●	●●●●	●●●●	
Plateforme de développement	●●●●	●●●●	●●●●	
Architecture et modélisation		●	●●●●	
Lab Management			●●●●	●●●●
TFS	●●●●	●●●●	●●●●	●●●●
Contrôle de source	Oui	Oui	Oui	Oui
Gestion des workitems	Oui	Oui	Oui	Oui
Build	Oui	Oui	Oui	Oui
Reporting et BI	Oui	Oui	Oui	Oui
Guide de planning agile	Oui	Oui	Oui	Oui
Team Explorer	Oui	Oui	Oui	Oui
Gestion des cas de test			Oui	Oui

● *faible support,* ●● *support moyen,* ●●● *bon support,* ●●●● *support total*

Un déploiement à multiples couches

Le déploiement d'une plateforme TFS implique le déploiement des éléments suivants :

- application logique, données, et les clients tiers pour Team Foundation
- localisation physique des serveurs
- Team Foundation Build et la fabrication des build
- Team Foundation Server Proxy

TFS implique un déploiement d'un environnement de données complet pour gérer les données projet, les sources, le reporting. Il se décompose en 4 couches :

- TFS Configuration : base stockant les catalogues de ressources et les informations de configuration de TFS.
- TFS Warehouse : base de données stockant les données des rapports
- TFS Analysis : base multidimensionnelle stockant les données agrégées provenant des équipes projets.
- Databases for team project collections : chaque collection d'une équipe projet possède sa propre base dans laquelle sont stockées les données.

La mise en place et la gestion des bases de données TFS (déploiement, maintenance, mise à jour...) est une des tâches sensibles dans l'administration de son TFS. La configuration après l'installation de TFS se fait dans le Configuration Center. L'installation et la configuration ont été grandement améliorées par rapport à la version 2008.

Si vous disposez d'un annuaire d'entreprise, cela facilitera grandement la mise en place des droits d'accès et la déclaration des utilisateurs. Ce centre de configuration permettra de créer la collection TFS dans la base de données. La collection permettra d'y déposer vos projets. Pour toute la partie collaborative, accès web, partages, Sharepoint est nécessaire. Le déploiement d'un TFS prendra plusieurs jours. En cas de doute sur l'architecture à adopter, consulter un expert TFS, un intégrateur certifié.

Définir son infrastructure matérielle et logicielle

Comme nous l'avons vu plus haut, TFS se compose de différents outils et couches. Il faut disposer de TFS, Visual Studio, SQL Server, Windows sur le poste client, Windows Server sur les serveurs (avec IIS pour lancer les services web nécessaires).

Idéalement, dans une infrastructure à montée en charge et pour assurer une fiabilité de l'ensemble, 1 serveur applicatif = 1 serveur physique. Ainsi, la base de données, le build, TFS en lui-même seront à déployer sur des serveurs différents. Il faut au moins disposer de deux serveurs physiques :

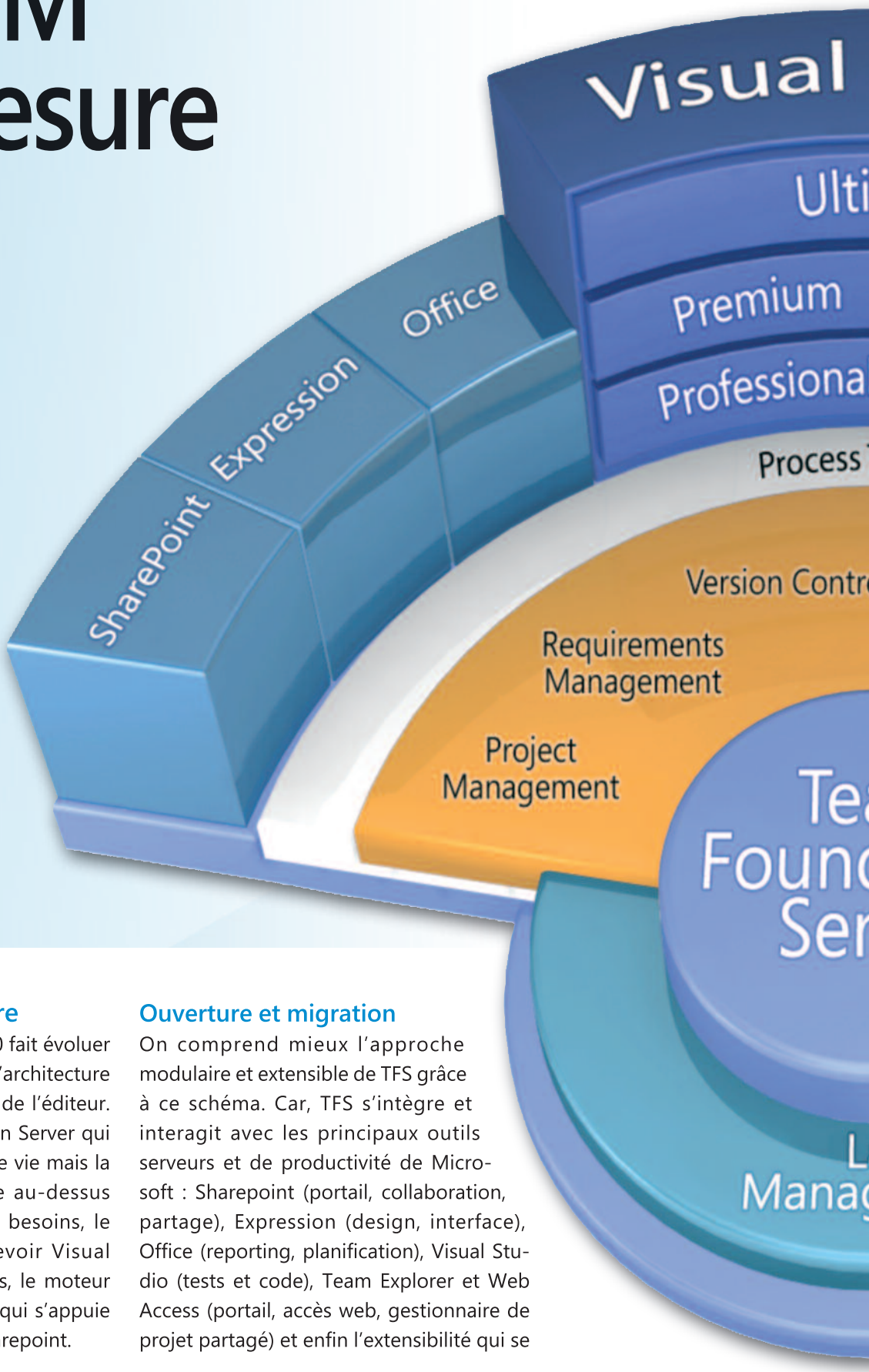
- 1 pour Windows Server, la base de données, TFS...
- 1 pour le build.

Ensuite il faudra tailler votre réseau et le nombre de serveurs selon le nombre d'utilisateurs et la charge attendue. En effet, vous ne dimensionnez pas de la même façon une infrastructure TFS pour 15-20 utilisateurs et une infrastructure pour 200 ou 2000 utilisateurs !

Au-delà du matériel, il ne faut pas négliger les ressources : mémoire vive, stockage, nombre de processeurs. Pareillement, sur les postes de travail des développeurs sur lesquels les outils de développement seront déployés (que ce soit Visual Studio ou Eclipse). Pour les autres utilisateurs (architecte, direction, testeurs), les exigences matérielles seront moindres.

Un ALM sur mesure

Avec sa gamme 2010, Microsoft a fait le ménage à la fois dans Visual Studio et ses outils ALM, de tests et de versionning. La gamme Visual Studio par exemple a été réduite de moitié afin de la rendre plus lisible.

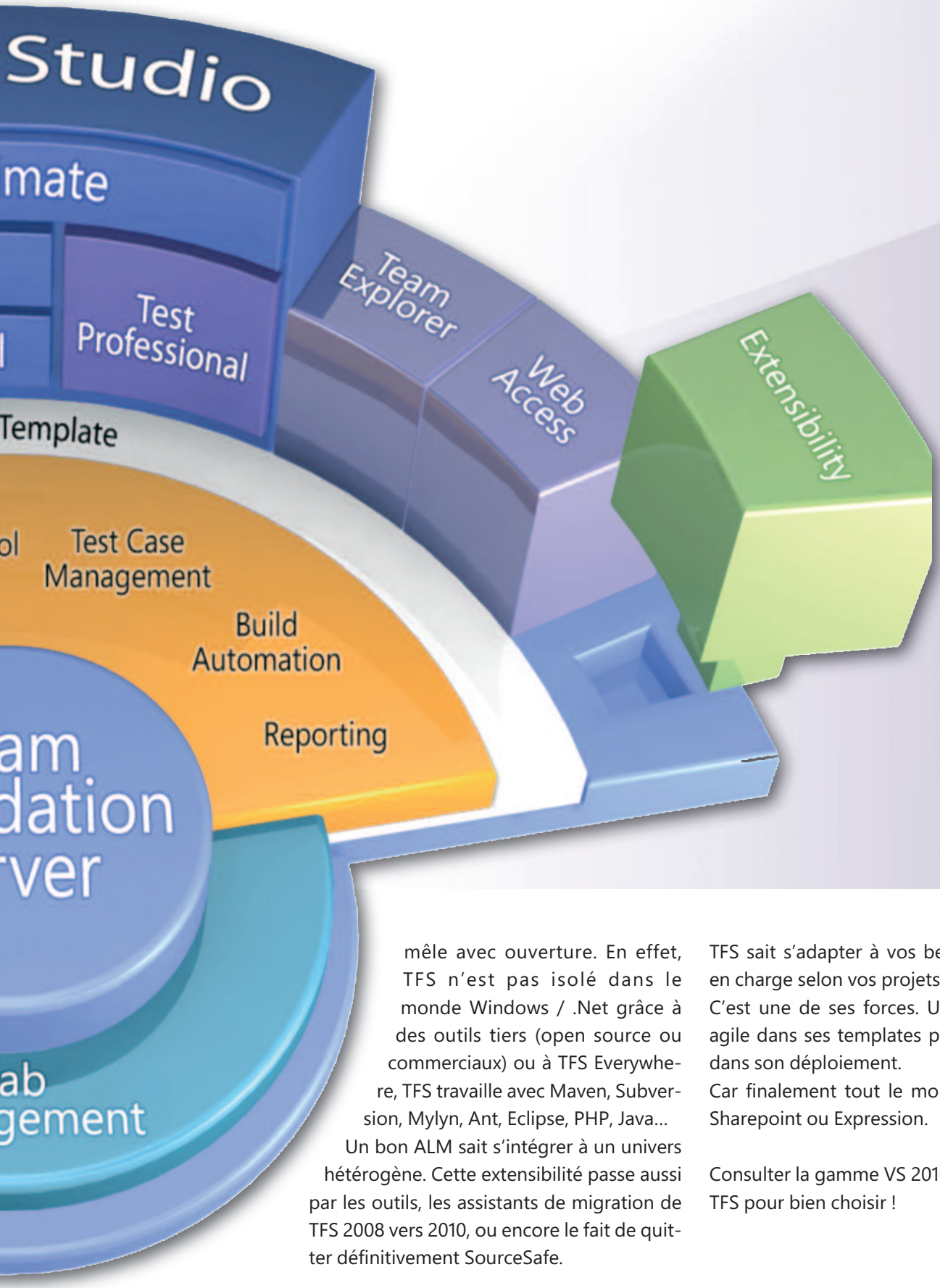


Une approche modulaire

Concernant TFS, la version 2010 fait évoluer l'approche fonctionnement et l'architecture globale de la plateforme ALM de l'éditeur. Le cœur reste Team Foundation Server qui constitue le moteur de cycle de vie mais la réelle valeur ajoutée se situe au-dessus avec la gestion projet et des besoins, le contrôleur de version (au revoir Visual SourceSage), les outils de tests, le moteur de build et la partie reporting qui s'appuie sur les derniers outils Excel, Sharepoint.

Ouverture et migration

On comprend mieux l'approche modulaire et extensible de TFS grâce à ce schéma. Car, TFS s'intègre et interagit avec les principaux outils serveurs et de productivité de Microsoft : Sharepoint (portail, collaboration, partage), Expression (design, interface), Office (reporting, planification), Visual Studio (tests et code), Team Explorer et Web Access (portail, accès web, gestionnaire de projet partagé) et enfin l'extensibilité qui se



mêle avec ouverture. En effet, TFS n'est pas isolé dans le monde Windows / .Net grâce à des outils tiers (open source ou commerciaux) ou à TFS Everywhere, TFS travaille avec Maven, Subversion, Mylyn, Ant, Eclipse, PHP, Java...
Un bon ALM sait s'intégrer à un univers hétérogène. Cette extensibilité passe aussi par les outils, les assistants de migration de TFS 2008 vers 2010, ou encore le fait de quitter définitivement SourceSafe.

TFS sait s'adapter à vos besoins et monter en charge selon vos projets.
C'est une de ses forces. Un ALM doit être agile dans ses templates projets mais aussi dans son déploiement.
Car finalement tout le monde n'utilise pas Sharepoint ou Expression.

Consulter la gamme VS 2010 et les fonctions TFS pour bien choisir !

Cycle de vie des applications : pourquoi y aller ?

Le déroulement incertain des projets informatiques, que ce soit sur les aspects délai, budget, périmètre, ou encore qualité serait-il un fait avéré, une tautologie ? Nous allons tenter de répondre à cette question dans le cadre de cet article en analysant les tenants et les aboutissants d'un projet informatique et de démontrer comment l'ALM apporte une réponse pertinente au travers d'un ensemble riche de méthodes et outils.

D.R.

LES PRINCIPALES CAUSES D'ÉCHECS PROJETS

Les raisons majeures qui sont à l'origine des échecs projets sont de deux ordres : Génériques et opérationnelles. Les causes génériques sont :

Le Facteur « Time to Market » toujours plus agressif entraînant des rythmes projets difficilement soutenables et des dérapages dans les délais qui statistiquement constituent une des premières causes de conflit sur projet

Des besoins métiers toujours plus complexes côté client pour assurer la compétitivité, induisant de la complexité côté implémentation technique

Une évolution incessante et très rapide des technologies et des méthodes de développement apportant une surcouche de complexité supplémentaire dont tout le monde se serait bien passé.

Ceci étant dit, l'exercice qu'il est intéressant de mener est ce que l'on appelle une « Root Cause Analysis ». L'idée ici étant de déterminer quels sont les facteurs qui vont contribuer le plus fortement au fait que les projets, au plan opérationnel, soient en difficulté.

Tout le monde connaît ces principales causes d'échecs, simplement, la difficulté va consister à les prioriser, nous allons donc proposer cette liste, qui bien évidemment n'est pas exhaustive:

- Gestion de Projet Lacunaire
- Manque de Réactivité Face au Changement
- Manque de communication / collaboration, information disséminée
- Spécification incomplètes ou surréalistes
- Mauvaise gestion des risques

Gestion de projet lacunaire

La gestion de projet reste un élément clé dans la réussite d'un projet. Cette activité va permettre de s'assurer de la satisfaction du client qui est un des objectifs majeurs de tout projet par la capacité à communiquer au travers d'un certain nombre de rapports sur l'état d'avancement, la qualité, le budget et le planning en vue de détecter tout écart et enclencher si nécessaire des actions correctives au plus vite afin de réaligner le projet avec les attentes du client et les objectifs stratégiques de l'entreprise.

Manque de réactivité face au changement

Entre le moment ou un projet démarre et sa livraison, beaucoup de changements vont intervenir, c'est inévitable. Cet état de fait est particulièrement impactant dans des contrats de type forfait dans lesquels les intégrateurs s'engagent à délivrer un périmètre en s'appuyant sur le fameux cycle en V, voire Waterfall (Cascade), rendant encore plus compliqué tout changement, du fait de la lourdeur des processus. Tout changement doit au contraire être accueilli favorablement comme avantage compétitif pour le client.

Manque de communication, information disséminée

Pour faire simple, la communication est la clé de voûte d'un projet informatique, pour comprendre comment ce point est fondamental en projet, on parle souvent du syndrome de la Tour de Babel. La tour de Babel a été le premier grand projet de l'humanité ayant subi un échec, il y a près de 5000 ans.

Ce projet, surréaliste en termes d'exigences au demeurant, a échoué à partir du moment où les bâtisseurs se mirent à parler des langues différentes et donc ne plus communiquer efficacement, la leçon de cette histoire nous a démontré la nécessité qu'ont les parties prenantes de se parler, de se comprendre pour réaliser de grands projets avec succès.

Spécifications incomplètes ou surréalistes

La gestion des spécifications constitue indéniablement une des principales sources de problèmes sur les projets lorsque l'on interroge des parties prenantes. Tout le monde le sait et pourtant la situation continue. Démarrer un projet avec un périmètre insuffisamment défini ou surréaliste est une ineptie qui va précipiter l'équipe en charge du projet dans des difficultés de tout instant.

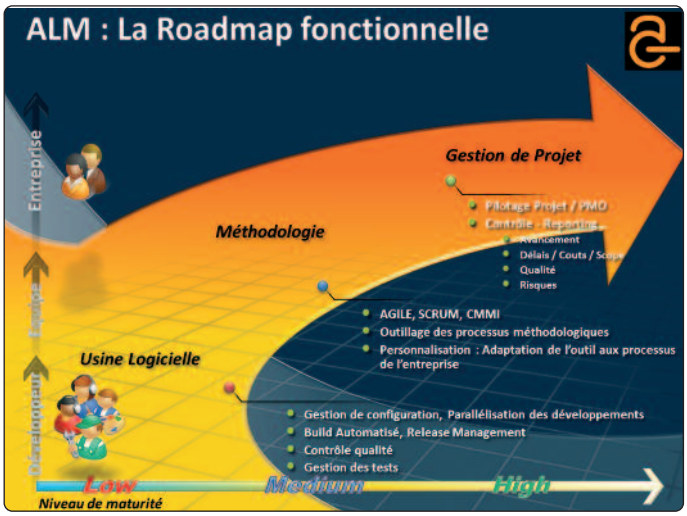
Mauvaise Gestion des risques

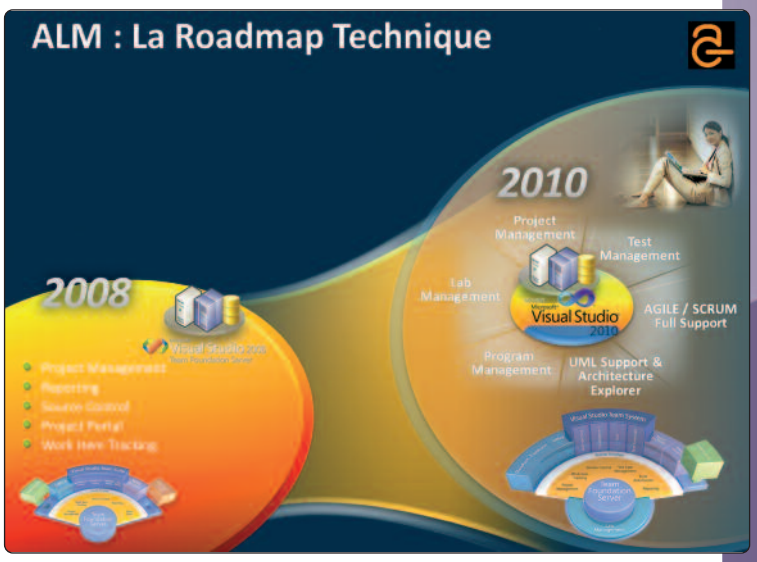
Qui gère ses risques sur un projet ? A cette question suit inéluctablement une réponse évasive. Ceci est pourtant une erreur stratégique. Chaque membre de l'équipe projet est conscient des risques potentiels et pourtant, ils ne sont pas gérés et s'imposent comme une fatalité.

LA SOLUTION ALM DE MICROSOFT : VISUAL STUDIO 2010 / TEAM FOUNDATION SERVER 2010

La plateforme ALM Microsoft TFS 2010 fournit un ensemble d'outils très riche fonctionnellement, permettant d'adresser l'ensemble des activités liées au cycle de vie du développement d'une application de la définition des exigences jusqu'aux phases de Test et Déploiement avec un support riche pour la méthodologie et la gestion de projet.

Ci-contre la Roadmap fonctionnelle de l'ALM TFS 2010 apporte un grand nombre de fonctionnalités à haute valeur pour la réalisation de projets dont :





Maturité ALM et Roadmap Fonctionnelle

Un support hiérarchique pour les Work Items : Cette fonctionnalité permet une véritable activité de Gestion de projet avec le support d'un WBS Projet (Work Breakdown Structure qui représente l'arborescence hiérarchique des livrables d'un projet), ceci permet par exemple de gérer une arborescence de type Projet / Phase / Module / Exigence / Tâche en mode CMMI ou bien User Story / Task pour un Carnet de produit en mode Agile ou SCRUM, et de se synchroniser avec un fichier MS Project ou encore avec Project Server. TFS 2010 intègre de plus un mécanisme de Roll up permettant la consolidation hiérarchique de données horaires ou financières.

Une Plateforme de Test : Avec Test & Lab Center, Microsoft propose une offre complète pour des équipes de test professionnelles, la plateforme est entièrement intégrée à TFS 2010 et propose un client dédié aux testeurs fonctionnels avec des outils de planification de campagnes de test, un outil d'exécution des tests incluant des possibilités d'automatisation et de gestion de paramètres de tests pour améliorer la productivité et la fiabilité lors de la phase d'exécution. Avec cette plateforme, Microsoft adresse le syndrome bien connu de la non-reproductibilité des bugs en fournissant un mécanisme de collecte d'artefacts contextuels au test qui sont téléchargés automatiquement dans les données lors de la création d'un bug dont : les étapes détaillées du test avec les résultats, le Recording Video de la session de test, les données Intellitrace (débogueur intelligent Microsoft) permettant au développeur de connaître le code fautif avec les données source de l'exception, la pile d'appel (Mini dump mémoire) et également des données de type Event Log et Informations Système. De plus avec Lab Center, Microsoft fournit une plateforme de Provisioning d'environnement pour les tests avec des capacités de déploiement des binaires applicatifs et d'exécution de tests de manière automatisée unitaire ou fonctionnelle. Cette dernière fonctionnalité améliore considérablement la productivité et la qualité sur les projets car il est bien connu que le Provisioning et la maintenance d'environnements d'exécution pour les tests est coûteuse en temps et répétitive, tout ce que l'on veut éviter sur un projet.

Le Support de l'UML : Des cas d'utilisation ou tout type de diagramme UML peuvent être créés avec ce module et surtout on notera la possibilité de les relier à n'importe quel autre artefact projet.

Une Plateforme de Build : Une nouvelle version de Team Build basée sur le moteur dernière génération Microsoft Workflow Foundation 4.0 s'impose comme un outil exceptionnel pour l'ensemble des concepts d'intégration continue pour une usine logicielle. Cette nouvelle version permet en plus des options standard d'Analyse de qualité du code, d'exécution de listes de tests unitaires avec le calcul de la couverture de code, de personnaliser simplement un processus de Build avec par exemple le rajout d'activités de déploiement en fin de Build ou encore la génération de release note automatique.

Program and Portfolio Management : Le produit vient en standard avec une intégration avec l'EPM Project Server 2010 ou 2007 offrant des fonctionnalités de consolidation niveau entreprise pour la gestion des ressources, de Programmes et ou Portfolio au-dessus de TFS 2010 qui représente le suivi opérationnel projet par projet. Un des facteurs clé de TFS est un ensemble très complet de rapports fournis en standard qui correspondent à plus de 80% des besoins de Reporting.

Par exemple, le rapport ci-dessous nommé « **Requirement Overview** » qui donne un vue complète du statut d'un projet avec les exigences du projet à gauche, et en regard, l'avancement, le reste à faire, le nombre de tests écrits par exigence, l'état et le résultat de l'exécution des tests et les bugs ouverts et résolus. On notera la notion de roll up des données du niveau exigence jusqu'au niveau projet.

Un autre exemple de rapport est le Build Success Over Time montrant la stabilité du processus de Build avec une légende pour le niveau de qualité de chaque Build en fonction du résultat de la compilation mais également des tests et de la couverture de code jour après jour.

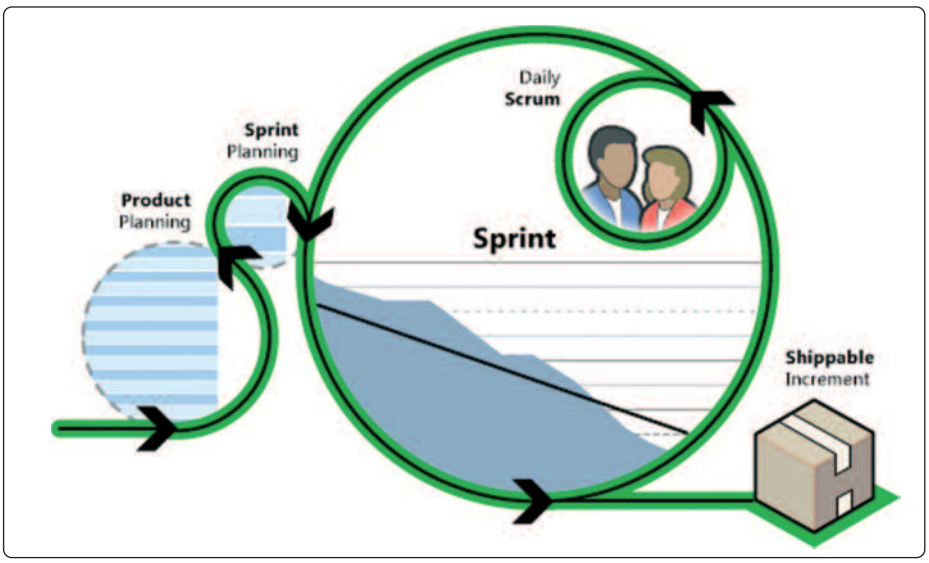
Pour finir

En conclusion, la mise en œuvre d'une plateforme ALM adresse réellement l'ensemble des besoins du cycle de vie projet avec un support fort d'activités en mode Collaboratif, un socle pour la méthodologie de type Agile, Scrum ou CMMI, des fonctions de Contrôle de version, la Gestion des Builds, des rapports et tableaux de bord, un centre de qualité logicielle et un ensemble d'outils connectés adapté aux différents rôles présents sur le projet, leur permettant de mieux collaborer et communiquer en partageant la même information avec leurs outils du quotidien. Des fonctions de gouvernance niveau entreprise peuvent également appuyer une gestion de portefeuilles projets de l'identification d'une opportunité jusqu'au déploiement en production. Enfin, au plan technique, la plateforme ALM de Microsoft est interopérable avec différents types de langages et technologies permettant d'élargir son périmètre d'adoption et surtout de répondre aux exigences d'entreprises exploitant un large panel de technologies Microsoft et non Microsoft.

Philippe **Puschmann**
Architecte senior – capability group application development, Avanade

Visual Studio et l'agilité

Visual Studio est décliné sous différentes versions qui permettent chacune de couvrir différents niveaux de ce que l'on appelle la « Gestion du cycle de vie d'une application » (soit Application Lifecycle Management ou ALM pour les intimes). La version la plus complète, Visual Studio Ultimate avec MSDN Ultimate, couvre la totalité du cycle de vie du développement d'une application, démarrant de la spécification des différentes fonctionnalités à l'aide de User Stories ou de Product Backlog, jusqu'aux tests fonctionnels des utilisateurs, en passant par des rapports sur le taux d'avancement de l'itération en cours, etc. L'offre Visual Studio 2010 est une gamme de produits qui vont permettre à chaque acteur d'un projet de développement d'interagir avec un outil adapté…



Les grands principes d'organisation de Scrum

Mais avant de poursuivre, il est nécessaire de définir et clarifier un terme régulièrement entendu/utilisé concernant la gestion de projets en environnement Microsoft : Le produit Team Foundation Server (abrégé TFS) est un service qui va permettre de centraliser toutes les informations manipulées au travers des différents outils qui composent la suite Visual Studio. Il est très commun de faire l'amalgame avec TFS et l'ALM bien que le premier soit un outil qui permet de mettre en pratique le second.

S'il est possible d'utiliser ces différents outils pour une gestion de projet classique, ceux-ci prennent tout leur sens lors qu'ils sont utilisés pour suivre une méthodologie de type Agile. En effet de nombreuses étapes spécifiques à cette dernière (détaillées tout au long cet article) y sont simplifiées et optimisées.

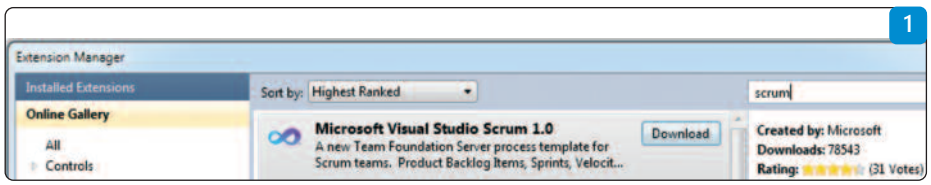
La matière première de cette méthodologie, ce sont les différents éléments de travail (appelés Work Items) qui seront manipulés tout au long du projet. Le découpage du projet se fera très souvent par fonctionnalités définies dans des User Stories (ou Product Backlog en Scrum). Celles-ci seront réparties dans des itérations, chaque itération correspondant à une livraison d'un lot de fonctionnalités. À ces fonctionnalités seront associées des tâches (Task) à réaliser, des cas de tests (Test cases) qui permettront de s'assurer de leur fonctionnement, ou encore différents bugs qui seront trouvés tout au long du cycle de vie de l'application.

Le suivi du cycle de vie de ces différentes tâches possède également un rôle important dans la prise de décision et la priorisation des différentes actions à effectuer. Pour cela un tableau de bord, disponible au travers d'un portail Share Point, présente de nombreux rapports de suivis simples, auxquels s'ajoutent des rapports plus complexes générés à partir d'un cube OLAP. Parmi ces derniers, on pourra trouver des rapports mettant en forme des courbes, présentant l'état d'avancement des Work Items au cours des différentes itérations ou Sprints. L'un des plus notables et des plus connus est le rapport de Burndown qui permet de visualiser le comparatif de la courbe idéale et réelle de l'avancement des tâches d'une Itération.

Autre sujet très important en développement Agile : S'assurer fréquemment de l'intégrité des livrables produits, soit faire ce que l'on appelle de l'intégration continue. Pour cela, la partie Team Build de TFS se chargera de compiler et de tester régulièrement le code source du projet (à des dates précises, voire à chaque modification du code source). Cela permettra également de livrer à intervalles réguliers une version des exécutables qui certes ne possèdera pas toutes les fonctionnalités de l'application finale, mais sur laquelle les clients finaux pourront émettre des critiques et vérifier les fonctionnalités.

Intimement liée à l'intégration continue, la partie Test de TFS joue un rôle important dans la vérification de la qualité logicielle et de la conformité du projet d'équipe avec les attentes des clients. De nombreux outils pour faire de la modélisation, de l'analyse de code ou encore des tests unitaires permettent de vérifier le projet d'équipe très régulièrement (Ex. à chaque compilation locale ou serveur).

L'importance des retours clients a également



été prise en compte en mettant à disposition l'outil Test Manager qui permet la mise en place de tests fonctionnels et facilite leur automatisation.
Élément non spécifique à la méthode Agile, le versioning du code source est également géré dans cette offre.

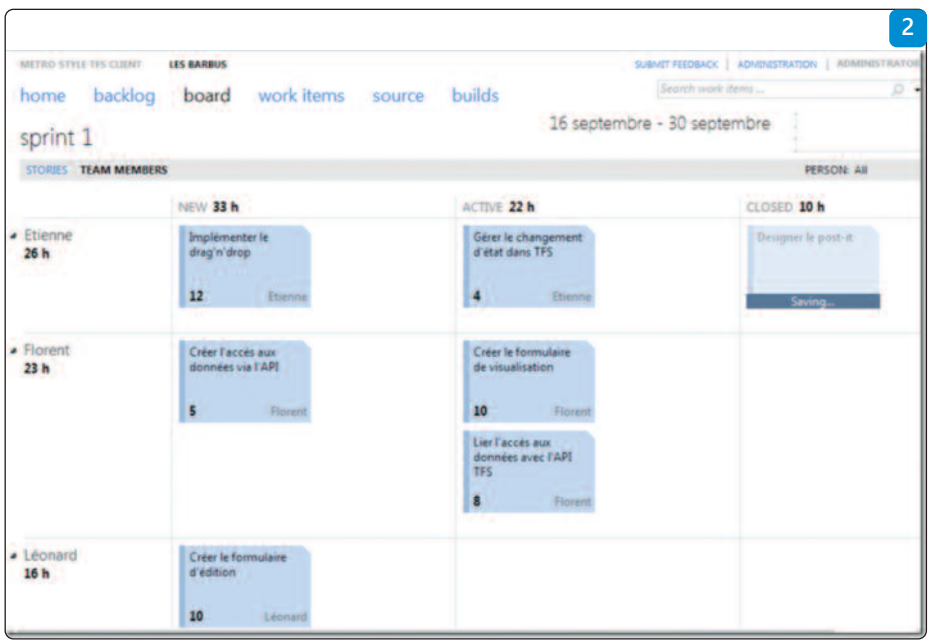
Le Process Template (Modèle de processus)

Si tous ces composants de Team Foundation Server aident à la mise en place de certains processus de la méthodologie Agile, ils peuvent posséder un comportement similaire ou différent selon le projet d'équipe. Tout cela est défini dans un groupement de fichiers XML appelé Process Template. Ce dernier, qui sera différent suivant la méthodologie choisie, définira le comportement des différentes parties de TFS et pourra être personnalisé. On pourra ainsi y trouver les itérations du projet, les workflow de Build qui seront disponibles pour l'intégration continue ou encore quelques rapports de base.
C'est également au travers du Process Template que seront définis les différents Work Items du projet. Ces derniers étant la pierre angulaire de la gestion de projet avec TFS, il est très important d'utiliser des Work Items qui sont pertinents selon la méthodologie choisie. Il sera alors possible d'en ajouter de nouveaux, d'en supprimer ou même de personnaliser ceux déjà présents dans le Process Template de base.
Les caractéristiques d'un Work Item se trouvent dans une « Work Item Type Definition » dans laquelle sont définis :

- Le nom, le type des différentes données stockées, si celles-ci doivent être reportées dans les rapports du cube OLAP (exemple : Title, AssignedTo, RemainingWork, Status)
- L'affichage graphique de la fiche du Work Item (Affichage sous Visual Studio et web)
- Le workflow du Work Item : Les différentes manières de le passer d'un état à un autre (exemple : Fermer une tâche, rouvrir un Bug, mettre à jour un cas de Test)

Un Process Template Agile est fourni par défaut lors de l'installation de Team Foundation Server 2010 (accompagné d'un Template CMMI) et un nouveau Process Template Scrum sera disponible de base dans la nouvelle version de TFS 11 (également disponible dans la galerie des extensions de Visual Studio) **(Fig. 1).**

Ces deux Process Template se différencient principalement par leurs différents types de Work Items.
Le Process Template Agile est inspiré de la méthode Scrum mais reste plus large que ce dernier. Il sera ainsi possible d'utiliser quasi n'importe quelle méthode de gestion de projet avec celui-ci. On parlera ainsi de User Story (Récits utilisateurs) exprimant des besoins qui vont être regroupés temporellement dans des itérations. Elles contiendront des tâches sur lesquelles il faudra saisir le temps passé, l'estimation d'origine, etc.
Le Process Template Scrum quant à lui est plus spécifique à la gestion de projet utilisant la méthodologie Scrum. De nouveaux types de Work Items y sont ainsi définis et de nouveaux rapports y font leur apparition. On parlera alors cette fois-ci de Product Backlog qui seront regroupés temporellement dans des Sprints. Pour implémenter ces premiers, des éléments tâches seront toujours utilisés, mais cette fois-ci, seul le temps restant sera nécessaire.



Un outil pour les gouverner tous, et dans TFS les lier

Tous ces éléments mis à disposition par TFS permettent d'implémenter les bonnes pratiques de la méthodologie Agile. Mais cette méthodologie, aussi efficace soit elle, ne peut être mise en place de manière convenable si elle n'est pas bien outillée, et demande ou perturbe trop les différents intervenants du projet. C'est pourquoi Visual Studio ALM fournit différents outils, adaptés aux rôles de chacun afin de leurs permettre d'accéder à ces informations centralisées depuis leur environnement de prédilection.
La nouvelle version de Visual Studio actuellement disponible en « Developer Preview », a vu son site web Team Web Access enrichi de nombreuses fonctionnalités spécifiques à la gestion en mode Agile. Souvent délaissé dans les versions précédentes au profit du Portail d'équipe SharePoint, celui-ci revient au centre de la gestion de projet d'équipe et s'adresse à tous ses participants, incluant de nouvelles interfaces de saisies:

- Les Sprints (ou Itérations) y sont maintenant planifiés directement à coup de Drag and Drop et une notion de début et de fin a été ajoutée à ceux-ci pour contextualiser les différents rapports générés.
- Il est possible de spécifier le temps de travail disponible pour chacun des développeurs lors du Sprint. On pourra alors y préciser le nombre de jours par semaine que chaque participant du projet y consacrera afin de pouvoir répartir de manière la plus pertinente possible les différentes tâches qui constitueront le Sprint.
- Il existe également une interface de gestion des tâches en mode Post-it, bien connue des habitués de la méthodologie Agile. On pourra tout simplement y Drag and Droper les différents Post-it pour leur faire changer d'état **(Fig. 2)**.
- Le rapport de Burndown est dorénavant disponible directement depuis le tableau de Post-It du Team Web Access. De plus, il est maintenant directement mis à jour lors des modifications effectuées sur les Work Items et ne nécessite plus d'attendre la reconstruction quotidienne du cube OLAP.

Bien entendu, le client Team Explorer pour Visual Studio, principalement destiné aux développeurs, est toujours de la partie. Celui-ci se voit ajouter de nouvelles interfaces d'interaction avec TFS, en plus de bénéficier d'un nouveau design beaucoup plus clair et intuitif.

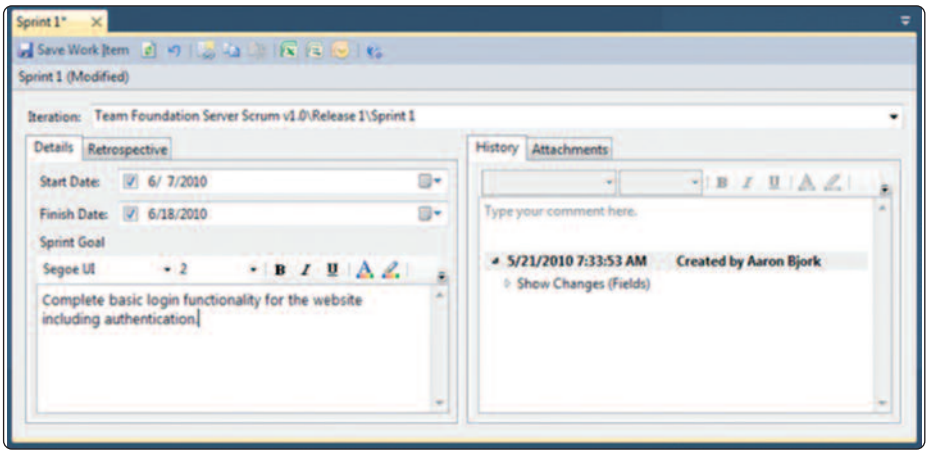
Il reste le moyen le plus complet de manipuler les différents éléments de TFS : Gestion du code source et des Work Items, création et visualisation des Builds serveurs, liens vers le portail d'équipe et le site de Reporting, accès aux documents du site SharePoint, tout y est.

Une nouvelle section nommée My Work permettra de visualiser et modifier en un clic l'état des différents Work Items (ex. Tâches ou bugs) qui nous sont assignés. Il sera alors plus intuitif pour le développeur de sélectionner directement depuis Visual Studio les tâches sur lesquelles il décide de travailler, avant même de commencer à modifier le code source.

Un nouveau Work Item verra également le jour, nommé Code Review. Ce dernier facilitera l'ajout d'un processus de revue de code lors du développement du projet d'équipe. Les développeurs pourront alors, en lieu et place de réaliser un archivage de leurs modifications de code source, décider de créer une nouvelle demande de revue de code et d'y spécifier un autre développeur. Celui-ci sera alors chargé de regarder, commenter puis valider le code source qui lui sera envoyé. Bien évidemment, les demandes de revue de code apparaîtront automatiquement dans la section My Work des développeurs.

Pour les chefs de projet, MOA, etc. le couple Microsoft Excel et Microsoft Project est toujours disponible et permet de manipuler les différents Work Items, Product Backlog, Tâches et autres Bugs. Il est ainsi possible de sauvegarder ses feuilles personnalisées et de les synchroniser à volonté avec le serveur TFS.

Microsoft Project rajoute une couche supplémentaire à Excel : Il permet de visualiser plus facilement la planification des différents Work Items et introduit une notion de dépendance entre ceux-ci.



Un work-item dans VS Scrum 1.0

Un petit nouveau fait également son apparition dans la nouvelle version de TFS 11 : la possibilité de faire du Storyboarding à partir de Powerpoint. En effet, ce dernier étant initialement très pratique pour créer rapidement une interface, il aura suffi de rajouter quelques options à celui-ci pour faire de cet outil connu un outil utile pour montrer l'enchaînement de différents écrans. Il sera alors possible de créer au travers de l'onglet Storyboarding ce que l'on appelle des Shapes (Formes), de les enregistrer puis de les réutiliser à travers divers écrans de l'application.

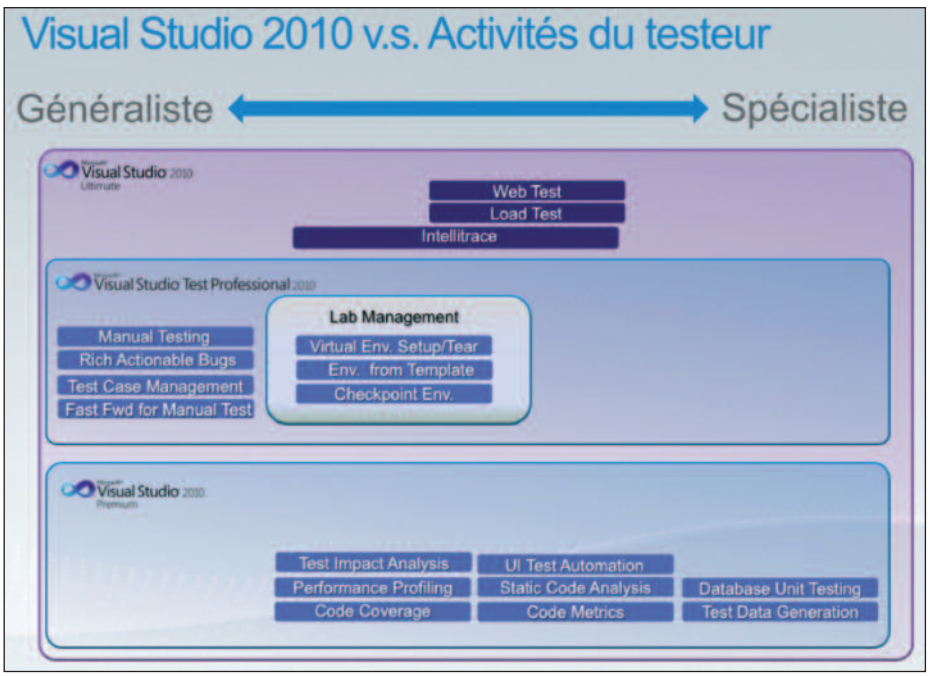
Enfin, Microsoft Test manager s'occupera, comme son nom l'indique, de gérer les différents aspects des tests de votre projet d'équipe. Il sera toujours aussi pratique d'y définir et vérifier des tests concernant les différentes fonctionnalités de votre application, puis de les automatiser au travers d'une Build serveur pour éviter les régressions.

Sa grande force réside dans sa capacité à récolter de précieuses informations de la part de n'importe quel utilisateur (Enregistrements, commentaires, Work Items, etc.)

S'il était assez simple de définir manuellement les scénarios des différents cas de tests fonctionnels de votre projet d'équipe, Team Foundation Server 11 apporte une nouvelle fonctionnalité appelée « Tests exploratoires ». Ceux-ci vont permettre de tester l'application sans scénario prédéfini, puis de créer automatiquement, par exemple lors de la rencontre d'une erreur, un bug ainsi qu'un cas de test associé. Ces derniers possèderont comme scénario, afin de reproduire ce bug ou vérifier ce test, les différentes actions enregistrées durant le test exploratoire.

Une nouvelle fonctionnalité de Feedback a été mise en place. Il sera alors possible de lancer une campagne de Feedback à partir de TFS et d'inviter chacun des participants à tester les différentes fonctionnalités implémentées durant un Sprint défini puis de fournir son Feedback. Celui-ci pourra comporter un enregistrement vidéo voire audio, des notes écrites et des Screenshots ainsi qu'un système de notation de la fonctionnalité testée.

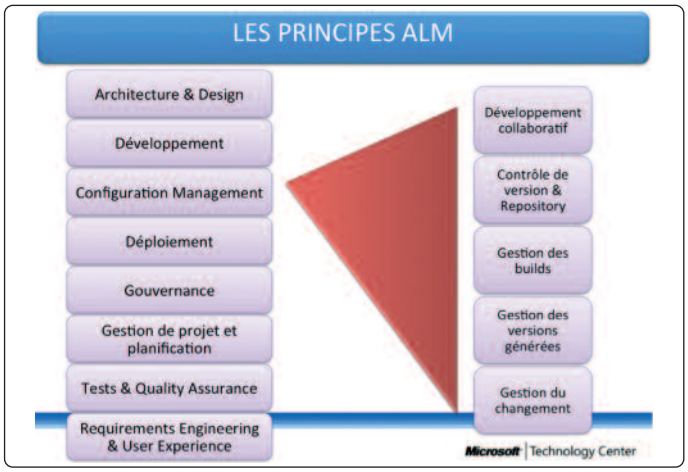
Actuellement en Developper Preview, la nouvelle version de TFS met encore plus l'accent sur l'agilité en mettant en avant le Process Template Scrum et en fournissant des outils de plus en plus perfectionnés pour chacun des membres du projet d'équipe, clients et testeurs compris.



Vivien **Fabing**
Consultant ALM chez Infinite Square
http://blogs.developpeur.org/vivien

Collaboration, partage, communication : les clés d'une équipe agile et de qualité

Un projet « moderne » doit s'appuyer sur une suite logicielle capable de fournir des services à la fois verticaux et transverses. Transverses, dans le sens où cette suite sert de fondation technique, de référentiels à l'ensemble des outils utilisés par la ou les équipes. Verticaux, car chaque catégorie d'intervenant (technique, architecture, métier, utilisateur, direction) a accès aux fonctions et outils nécessaires pour mener à bien sa mission, son travail.



Décloisonner pour mieux travailler ensemble

Une entreprise, qu'elle soit grande ou petite, possède différents départements (finance, direction, commercial / marketing, informatique…) avec différentes équipes. Encore aujourd'hui, ces départements travaillent en silo et en communication limitée avec l'environnement. Une plate-forme ALM va contribuer à décloisonner les équipes, les personnes pour assurer le succès des projets. Sans partage, sans collaboration, le projet ira à l'échec ainsi que l'ALM. Pour cela, il faut faire travailler ensemble trois grandes entités : la demande (utilisateur, client, direction, le métier), l'informatique (équipe technique) et la partie déploiement (administrateurs réseaux, responsable projet…).

Cependant, n'oublions pas que l'outillage ne fait pas tout. Il faut que les équipes et l'ensemble des intervenants soient sensibilisés à la démarche ALM et qualité logicielle. Or, le changement des habitudes et le cloisonnement constituent des remparts à la démarche ALM.

Préparer soigneusement son ALM

Avant tout déploiement d'une suite de cycle de vie / qualité, il faut impérativement préparer la gestion du changement, faire participer les équipes, informer, former, expliquer. Cette phase amont est cruciale car elle conditionne l'utilisation de l'ALM au quotidien.

Un projet ALM ne s'improvise pas, cette approche bouleverse l'organisation des équipes et de l'entreprise.

Il est tout d'abord important d'auditer et de définir vos besoins, les outils déjà utilisés, le patrimoine applicatif à maintenir (et donc son code) induisant une future migration et de définir un organigramme précis des équipes. A partir de cet audit, vous pouvez définir les besoins réels et moduler l'ALM en conséquence. Il n'est pas utile de déployer un ALM complet si les besoins se limitent au contrôle de version, à la génération et l'intégration continue.

Il ne faut jamais oublier qu'un ALM doit être lui-même agile, c'est-à-dire modulaire et capable de s'adapter à vos besoins, aux exigences de chaque projet. D'autre part, il ne se déploie pas d'un coup de baguette magique, mais progressivement. D'abord sur des équipes réduites pour des projets non critiques. La montée en charge doit s'étaler sur plusieurs mois afin que les administrateurs ALM et les équipes adoptent les outils et acquièrent les nouveaux réflexes. Avec un modèle trop rigide de sa qualité logicielle on prendra un risque d'échec de sa plate-forme ALM. Bref, une solution ALM n'est pas la solution à tous vos problèmes, ni un objectif ultime. Un ALM ne servira à rien si l'organisation, les méthodes de travail ne changent pas. Et il ne faut pas que votre ALM soit isolé du reste du système d'information, sa réussite dépend de son intégration et de la bonne volonté de tout le monde !

Développeurs, testeurs, utilisateurs, chef de projet, direction… même combat !

Nous avons l'habitude de dire : avoir la bonne information au bon format et au bon moment. Un ALM est comme un gros référentiel traitant et stockant diverses données. La plate-forme traite et cible les informations selon le profil du collaborateur. Ainsi, un développeur aura des besoins techniques (code, description des bugs, planning des itérations, information sur les build, etc.), alors qu'un testeur demandera surtout le bugtrack, la traçabilité des corrections, l'accès aux modules de tests, une vue de planning plus haute que le développeur, etc.).

La collaboration entre tous les membres des équipes est donc primordiale, même si chaque profil a son proche planning, ses propres démarches. L'ALM gérera en interne tout cela. La plateforme met à la disposition des intervenants une communication élargie : messagerie, messagerie instantanée, VoIP, gestion de la présence, partage documentaire, suivi des documents, gestion des itérations, etc.

Vous devez tendre vers le cercle vertueux.

François **Tonic**

A RETENIR

Une solution de gestion de cycle de vie de l'application doit :

- fluidifier les processus, les flux de données, d'information entre les équipes et chaque personne
- aligner le métier / la logique business et la technique
- fournir la bonne information à la bonne personne, au bon moment
- établir un référentiel unique et commun à tout le monde
- créer un cercle vertueux dans la vie du projet de l'expression des demandes à la maintenance

Tests unitaires, d'interface utilisateurs, de charge : une palette d'outils pour maîtriser votre qualité logicielle

La qualité d'un logiciel passe obligatoirement par sa conformité par rapport aux spécifications. Qu'il s'agisse de spécifications techniques ou fonctivonnelles, les créateurs de logiciels doivent garantir cette conformité, afin de démontrer la qualité de leur production.

Heureusement, les outils dont disposent les équipes projets sont de plus en plus performants, et accessibles. Nous allons passer en revue dans cet article les très nombreuses possibilités qu'offre la plateforme d'industrialisation logicielle de Microsoft : Visual Studio 2010 et Team Foundation Server 2010.

Tester d'accord, mais tester quoi ?

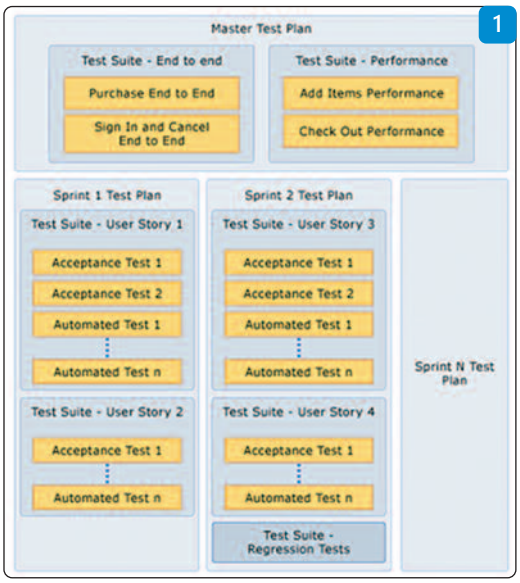
Le logiciel doit être conforme à différents niveaux :

- Conformité des traitements : les algorithmes et les traitements sont-ils correctement implémentés, et les fonctions renvoient-elles le bon résultat ?
- Conformité des scénarios d'utilisation : l'enchaînement des écrans, le libellé des messages, des boutons et des liens, la présence et les propriétés des contrôles sont-elles conformes aux spécifications ?
- Conformité de l'apparence de l'interface utilisateur : les couleurs et les images sont-elles correctement définies, les contrôles sont-ils placés au bon endroit ?
- Conformité des temps de réponses : les pages apparaissent-elles après un délai acceptable ?
- Aptitudes à la montée en charge : les performances sont-elles toujours conformes lorsque de nombreux utilisateurs sont connectés et utilisent l'application ?

L'enjeu d'une plateforme d'industrialisation logicielle est de fournir aux équipes projets une solution adaptée pour tester chacun de ces niveaux, soit de manière automatique, soit manuellement.

Mais la réponse à la qualité ne se situe pas uniquement dans l'outil ! Encore faut-il aborder le test avec méthodologie, en sachant placer l'effort de test sur les zones réellement à risque, avec une approche pragmatique permettant de soutenir l'effort de test dans la durée. Une application conforme à la première version aurait rapidement une image détestable si les versions suivantes fourmillaient de bug…

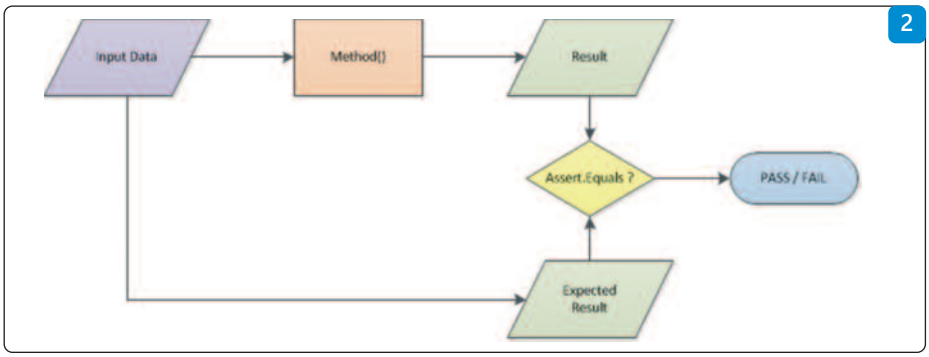
La stratégie de test d'une application va donc permettre de définir la manière dont l'application va être testée, en combinant les différents types de tests **(Fig. 1)**.



Exemple de plan de test

Au début était le test unitaire…

Le premier élément à garantir dans une application est le respect des règles de traitements. Ceux-ci sont généralement implémentés dans la couche Business de l'application, sous forme de méthodes de service. Chaque méthode de service peut ainsi être validée par un test, qui consistera à injecter un jeu de données d'entrée à cette méthode, et de comparer la réponse de la méthode au résultat attendu. Il s'agit du principe du test unitaire, car il s'intéresse à la seule fonction, sans s'occuper de son environnement **(Fig. 2)**.



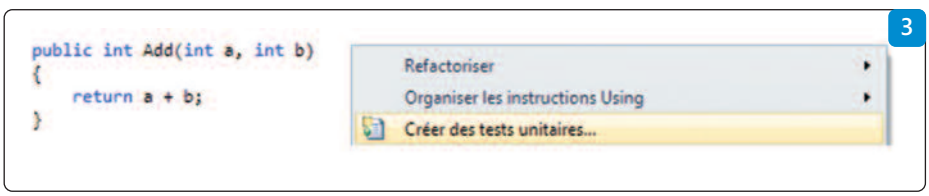
Les tests unitaires

Avec Visual Studio 2010, la mise en place de test unitaire est extrêmement simple. Il suffit de se placer sur la fonction que l'on souhaite tester, d'effectuer un clic droit, et de sélectionner **Créer des tests unitaires (Fig. 3)**.

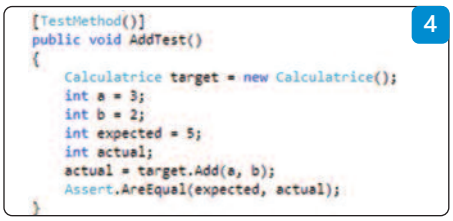
Quelques validations plus tard, un nouveau projet est ajouté à notre solution : il s'agit d'un projet de test, qui permettra de gérer le code source de test de notre application.

Une méthode de test a été créée, et nous n'avons plus qu'à renseigner les paramètres d'entrée, et le résultat attendu **(Fig. 4)**.

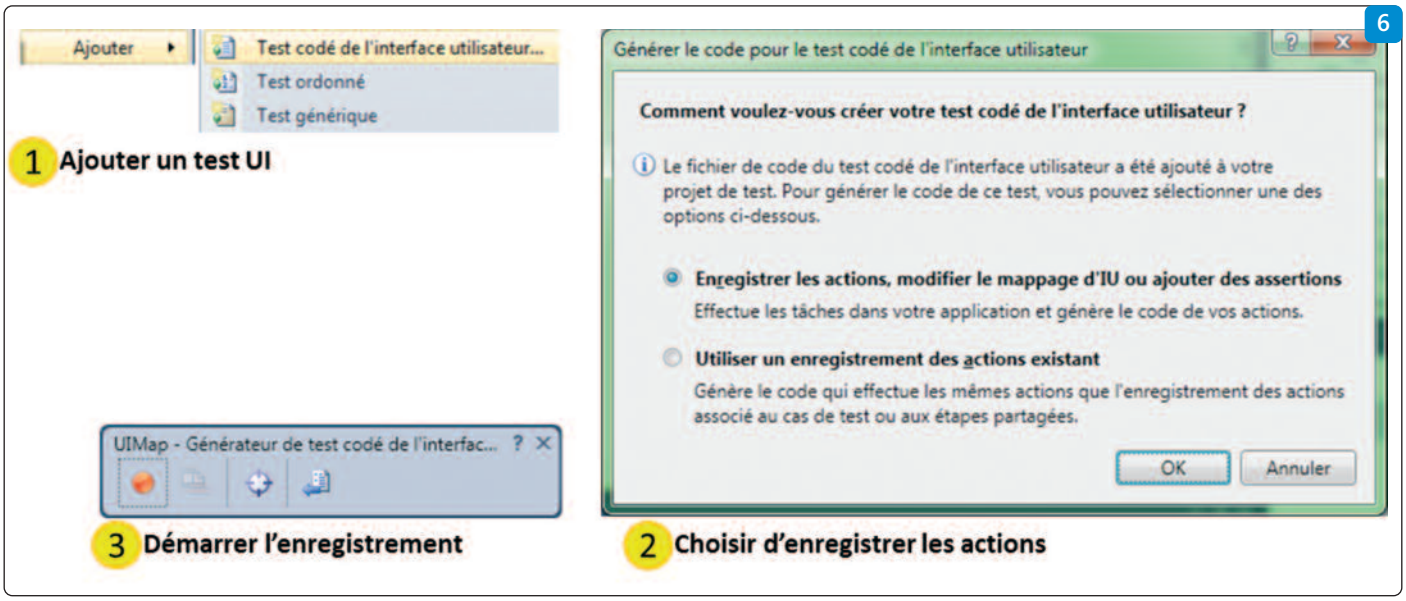
Le test apparaît alors dans la fenêtre d'Affichage des tests, d'où il est possible de l'exécuter : le résultat est conforme **(Fig. 5)**.



Créer des tests unitaires



Code de test unitaire



Créer un test codé d'Interface Utilisateur

...puis vint le test codé d'interface utilisateur

Le test des méthodes de services représente forcément une part importante de la qualité d'une application. Cependant, il ne permet qu'une validation technique, très proche du code.

Mise en place des tests codés d'interface utilisateur

Une nouveauté très intéressante, apparue avec Visual Studio 2010, consiste à tester notre application à partir de son interface utilisateur. Ce test se rapproche des tests fonctionnels, en simulant un utilisateur réel. Ces tests d'UI (User Interface) vont nous permettre d'exécuter automatiquement des actions préalablement apprises, d'effectuer des mesures sur l'interface graphique, afin de valider ou d'invalider la conformité du logiciel. Un assistant permet d'effectuer très simplement ces différentes opérations **(Fig. 6)**.

Au cours de l'enregistrement, et du paramétrage des points de mesure, du code source est généré dans le projet de test.

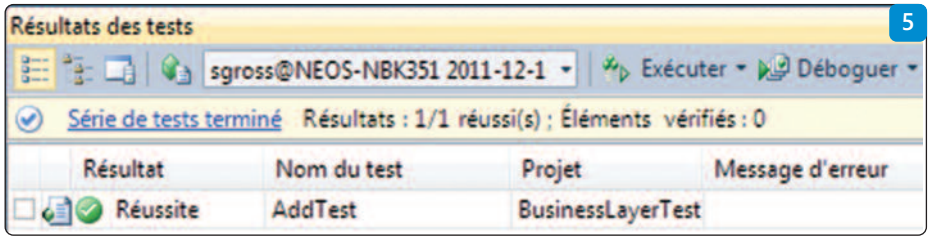
```
[CodedUITest]
public class CodedUIFromHomePage
{
    public CodedUIFromHomePage()
    {
    }

    [TestMethod]
    public void CodedUIValidateAboutPath()
    {
      this.UIMap.StartApplication();
      this.UIMap.ControlHomePage();
      this.UIMap.NavigateAbout();
      this.UIMap.ValidateAboutContent();
      this.UIMap.CloseApplication();
    }
// ...
  }
```

L'exécution de ce test va automatiquement démarrer l'application, et effectuer chacune des étapes définies lors de l'enregistrement, qu'il s'agisse d'actions, ou de contrôles.

Paramétrage

Mieux, à l'aide de ces tests, il devient possible d'entrer automatiquement des valeurs dans des champs de formulaires. Ces différents jeux de données d'entrées seront stockés dans une base de données, ou dans des fichiers, ce qui permet d'exécuter plusieurs fois un même scénario, avec des valeurs différentes.



Résultat de test unitaire

La mise en place de ce mécanisme de Binding avec des données externes ne demande que quelques modifications dans le code généré, et le paramétrage de la connexion à la source de données externe.

Performance et charge : la cerise sur le gâteau !

Les tests unitaires et les tests d'interfaces utilisateurs permettent de valider la qualité fonctionnelle de l'application. La conformité de l'application ne se limite néanmoins pas à sa qualité fonctionnelle : il faut s'assurer que l'utilisation restera opérationnelle, dans des conditions réelles d'utilisation, avec un nombre important d'utilisateurs, sur une longue durée d'utilisation. Pour ce faire, il existe d'autres types de tests :

- Les tests de performance Web
- Les tests de charge

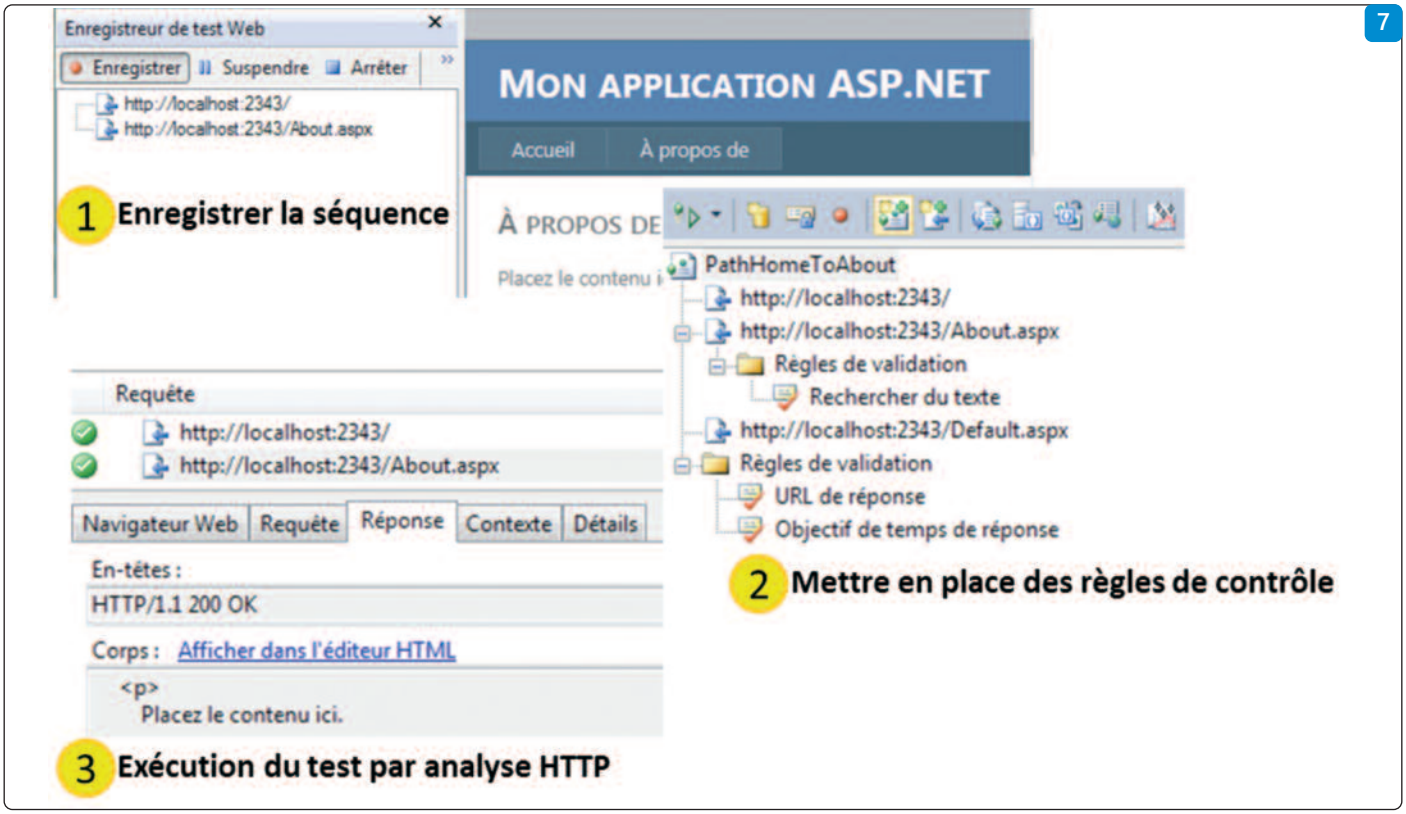
Les tests de performance Web

Les tests de performances Web ont ceci en commun avec les tests d'interface utilisateurs qu'ils permettent d'exécuter des scénarios d'utilisation, en effectuant des contrôles durant l'exécution. La différence principale est que dans le cas des tests de performance Web, seuls les échanges HTTP sont analysés, et non leur rendu à l'écran. C'est l'analyse des trames HTTP qui va permettre de contrôler la conformité des retours serveur de l'application **(Fig. 7)**. L'un des avantages de cette méthode est sa capacité à être intégrée dans des tests de charge.

Les tests de charge

Le test de charge permet de valider la capacité de montée en charge de l'application, en exécutant les différents tests unitaires ou de perfor-

7



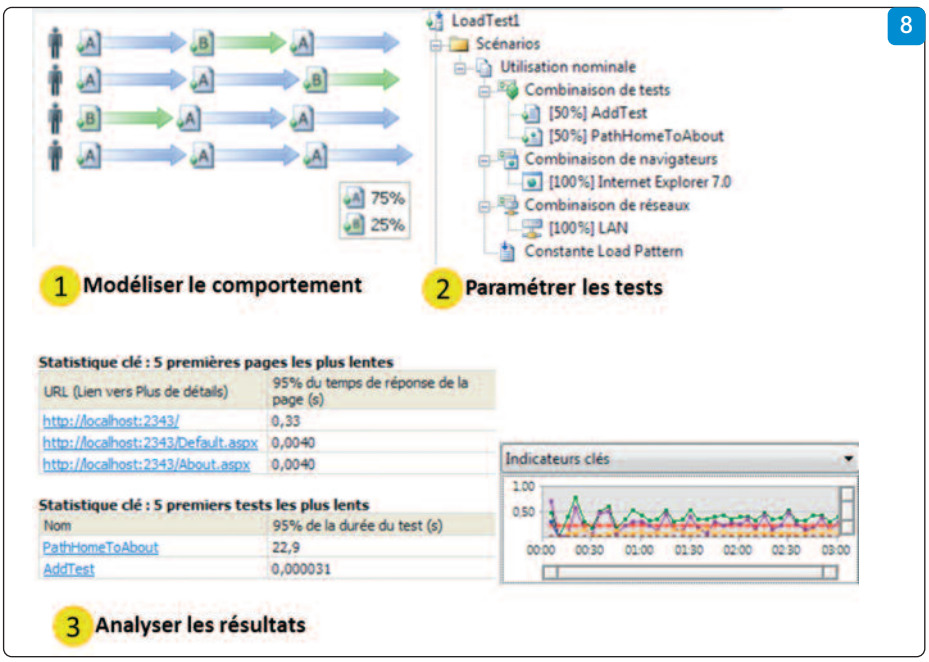
Mettre en place d'un test de performance Web

mance, à l'aide d'un grand nombre d'utilisateurs virtuels simultanés. Il est possible de paramétrer le comportement des utilisateurs : temps de réflexion, alternance de scénarios, variation du nombre… La durée du test de charge est paramétrée, avec la possibilité de prévoir des temps de chauffe, et un temps de montée en charge.

Durant ces tests de charges, différents compteurs tels que mémoire disponible, taux d'utilisation de CPU, nombre d'erreurs, etc… sont échantillonnés, et stockés dans une base de données de résultats.

L'analyse de ces résultats se fait depuis l'interface Visual Studio, qui présente différents rapports graphiques ou des tableaux permettant d'apprécier les résultats des tests. L'analyse peut se faire à postériori, les données étant stockées dans une base de données **(Fig. 8)**.

8



Mettre en place un test de charges

Comment y aller ?

Méthodologie

La mise en œuvre de ces différents types de tests est un moyen de maîtriser la qualité de ses applications. Cependant, ces outils seuls ne suffiront pas : la mise en place d'une méthodologie adaptée est nécessaire.

Cette méthodologie doit permettre :

- De concentrer l'effort de test sur les parties les plus critiques
- De systématiser le test pour chaque livraison, voire pour chaque archivage de code source afin de détecter au plus tôt les anomalies, et y remédier.

La mise en place de l'automatisation des tests devient rapidement nécessaire afin d'offrir les garanties proposées par les outils. Par l'utilisation conjointe de Visual Studio, et de TFS, cette automatisation des tests est facilement accessible. Vous rêviez d'un moyen de tester automatiquement vos applications toutes les nuits, que ce soit sur l'IHM, les traitements ou les performances et la charge, de façon à disposer chaque matin d'une liste d'anomalies à corriger ? La plateforme Visual Studio 2010 et Team Foundation Server 2010 rend ce rêve accessible, avec un coût de mise en place raisonnable.

Sylvain **Gross**
BU Manager Western Europe chez Neos-SDI
Responsable Technique Offre ALM/PLM
http://www.neos-sdi.com
sylvain.gross@neos-sdi.com
@SylvainGrossNeo

L'automatisation de tests sous Visual Studio

Si l'on compare les cycles de vie des projets informatiques de nos jours à ceux d'il y a une ou deux décennies, on constate des délais de livraison de plus en plus rapprochés, des utilisateurs finaux de plus en plus exigeants en termes de qualité, une démultiplication des configurations soft qui doivent être maintenant compatibles.

Les systèmes développés sont de plus en plus complexes et la concurrence ainsi que le « time to market » représentent un danger de plus en plus pressant.

Dans le même temps les outils de test permettant d'automatiser des processus métier ont été contraints d'évoluer pour correspondre aux besoins du marché

Ils doivent être orientés objets, proposer une interface de programmation puissante et assistée. Dans ces conditions les problèmes de reconnaissance d'objet à l'enregistrement comme à l'exécution devront être réduits au minimum. La maintenance devra, elle, être réduite à son plus strict minimum.

En suivant les mouvances méthodologiques telles que Scrum ou toute autre méthode, les outils devront donc évoluer pour s'intégrer dans la démarche générale du cycle de développement des applications. Nous assistons à une recrudescence de nouveaux outils qui devront nous assister tant au niveau des tests unitaires qu'au niveau des tests fonctionnels utilisateurs, ou bien de performance.

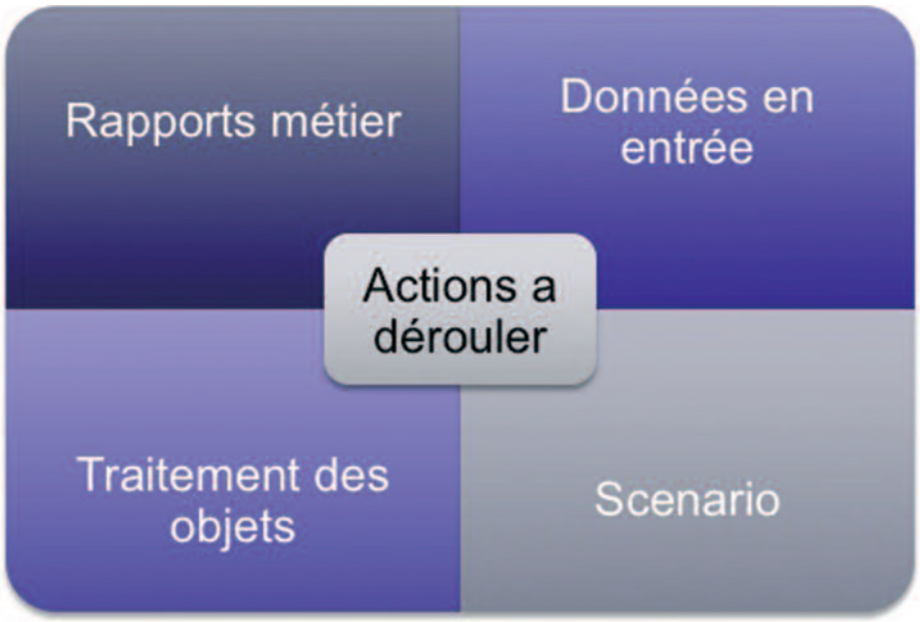
L'automatisation, un challenge à relever en équipe

Nous avons pu constater, en intervenant sur de nombreux projets d'automatisation, que certaines erreurs stratégiques impactent irrémédiablement le succès de la méthode : l'isolement des ressources dédiées a l'automatisation.

Sans une participation active de tous les acteurs dans un projet de développement logiciel, le processus d'automatisation risque d'être long et de ne cibler qu'une partie des points importants a vérifier.

Le partage idéal des tâches permettrait d'utiliser au mieux les compétences des acteurs de la conception logicielle :

- Le développeur pourrait valider les techniques de reconnaissance des objets, les environnements cibles des exécutions, et pourrait donner le support nécessaire pour développer des scripts correspondant aux contenus des versions de builds et environnements cibles.
- L'expert métier pourrait intervenir dans le choix des périmètres fonctionnels à tester, il serait le référent idéal pour concevoir l'ensemble des scénarios ainsi que la matrice des jeux de données à utiliser
- L'ingénieur test et recette pourrait lancer les campagnes de tests automatiques, debugger le niveau le plus simple des échecs d'exécution et analyser les résultats obtenus
- L'expert automatisation aurait pour rôle de mettre en place un framework d'automatisation qui permettrait d'agencer les cas de test, de maîtriser les exécutions, de scripter les processus métiers de manière modulaire et de les mettre à la disposition des experts métiers qui organiseront les scénarios.



De cette manière la montée en compétence de chacun restera concentrée sur les domaines maîtrisés et ainsi la couverture des tests automatiques ne sera pas remise en question. Un réel gain de temps pourra être alors constaté lorsque chaque acteur aura participé a son élaboration.

L'Automatisation par couche

Pour permettre à chacun de tirer un profit maximal de l'automatisation il est indispensable d'en dissocier ses composantes. (voir diagramme ci-dessus).

De cette manière chacun pourra apporter sa pierre à l'édifice automatique sans interférer sur les autres composantes.

Dans le cas d'un changement d'environnement (donc de jeux de données à modifier), les données étant stockées dans un objet indépendant du script, et le mode opératoire ayant changé, alors la couche de gestion des processus métiers sera maintenue.

Une force de frappe adaptée au besoin

Coded UI est une réponse appropriée à la mouvance du monde automatique.

Profondément ancré dans le socle du développement logiciel, Coded UI s'intègre nativement aux cycles de livraison des versions logicielles produites par Visual Studio. Ainsi grâce à l'interface de configuration de la build nous pourrons facilement lancer les package de tests automatiques correspondant aux tests d'environne-

ment, au bon fonctionnement minimum ou la non Régression au moment ou la version est « construite »

Les campagnes de test ciblées sont aussi beaucoup plus simples à définir grâce à TFS. En effet TFS permet d'accéder à toutes les données cruciales du projet, il est trivial de définir les cas de test impactés par un changement dans le code source. Ce niveau de précision est bien plus utile qu'un simple lien à une exigence métier. Cette même fonctionnalité demande dans des outils comparables un développement conséquent dont la stabilité n'est pas toujours à toute épreuve

Fluidité dans l'automatisation

Imaginez de plus que le même moteur permettant aux développeurs de créer des objets (IHM ou autre ADO etc.) soit à votre disposition pour reconnaître puis interagir avec vos applications, et ce en mode Web ou Windows front end.

Une automatisation par couche Native

Durant des années nous avons recherché et développé des solutions qui nous permettaient une dissociation entre les différentes composantes business, technique, Data ou scénarios, avec la solution d'automatisation de Visual studio et de par son architecture anticipée vous pourrez héberger chacune des couches dans des fichiers différents

Un moteur de rapport modulable et puissant

Grâce aux Report services de TFS nous disposons de toutes les informations nécessaires à l'élaboration de rapports explicites, orientés métier ou technique, et portables. Nous pourrons donc consolider les indices de qualité pour faire naître un tableau de bord facilement optimisé, et si besoin, lié à un accès SharePoint.

D'un point de vue technique

La participation de tous les acteurs d'une cellule de qualification logicielle outillée peut être une pierre à l'édifice d'une campagne de tests automatisée. En effet, un testeur manuel a des connaissances métier précieuses pour l'automaticien et il convient de les mettre à profit dans l'automatisation des tests.

Ce testeur manuel a donc à sa disposition un client lourd lui servant de référentiel de tests et d'assistant à leur bon déroulement : Visual Studio Test Manager. Dès lors qu'il est prêt à exécuter manuellement ses tests, VSTS lui propose d'enregistrer toutes ses actions à la manière des macros Word ou Excel **(Fig. 1)**.

Dès lors, son cas de test est enregistré et il peut déjà se vanter : « Je fais de l'automatisation de tests ». Il pourra s'en servir pour rejouer immédiatement son cas de test avec la ligne suivante du jeu de données du test et même lors des prochaines exécutions de celui-ci dans les jours, semaines ou mois à venir en cliquant l'icône Play lors de la prochaine campagne.

Ce n'est pas là leur seul point commun avec ces macros Excel qui ont tant fait pour nous faciliter le travail redondant. En effet, ces actions enregistrées peuvent être récupérées sur l'environnement de développement de Visual Studio. Le code ainsi généré est parfaitement compréhensible, retouchable, et entièrement personnalisable.

Cette génération de code fait apparaître plusieurs nouveaux fichiers dans notre projet .NET :

- CodedUITest1.cs : contient la classe représentant notre Coded UI. Assure la couche métier du test dans laquelle on va combiner les différents test steps pour former un test case. Elle assure également la récupération des lignes des jeux de données de tests depuis Test Manager. A noter que la source de données (Test Manager par défaut) est modifiable vers une base SQL Server, un fichier XML, Excel, Access, txt, etc.
- UIMap.uitest : contient le modèle XML du mapping des objets manipulés par le test.
- UIMap.Designer.cs : contient la représentation de code du fichier XML contenu dans le fichier UIMap.uitest, soit les propriétés utilisées pour identifier les objets de test et les méthodes capturées précédemment. Elles portent le nom des pas de tests choisis dans Test Manager. En effet, chaque test step génère sa propre méthode unitaire afin d'optimiser leur réutilisation par la suite. Ces méthodes sont également commentées.
- UIMap.cs : contient les méthodes de manipulation des objets de test personnalisées. Plus elles sont unitaires, plus elles sont réutilisables.

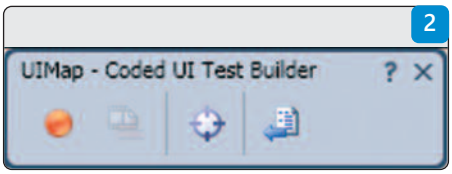
Jusqu'à ce point, seules des manipulations automatiques d'IHM composent notre test automatique. Faut-il encore vérifier le bon comportement de notre application et donc ajouter des points de vérification : l'**assertion**.

En faisant apparaître le codedUITestBuilder, on va pouvoir enregistrer de nouvelles manipulations à automatiser, ajouter de nouveaux objets de l'application à notre projet afin de pouvoir les manipuler ou vérifier les valeurs de leurs propriétés : ce sont les assertions **(Fig. 2)**.

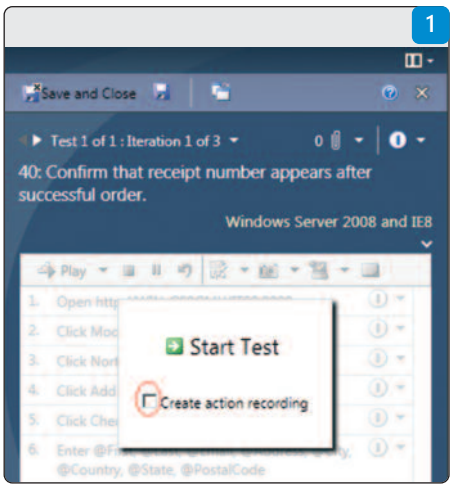
Par un glisser-déplacer depuis l'icône cible du codedUITestBuilder sur un objet composant de l'application à tester, on va le sélectionner. Alors, lui et ses propriétés apparaissent. On peut ainsi sélectionner un couple [propriété – valeur] afin d'ajouter la méthode qui en vérifie le contenu **(Fig. 3)**.

En cliquant sur Add Assertion, on va choisir le modèle de comparaison de la valeur : AreEqual, not equal, starts with, contains etc **(Fig. 4)**.

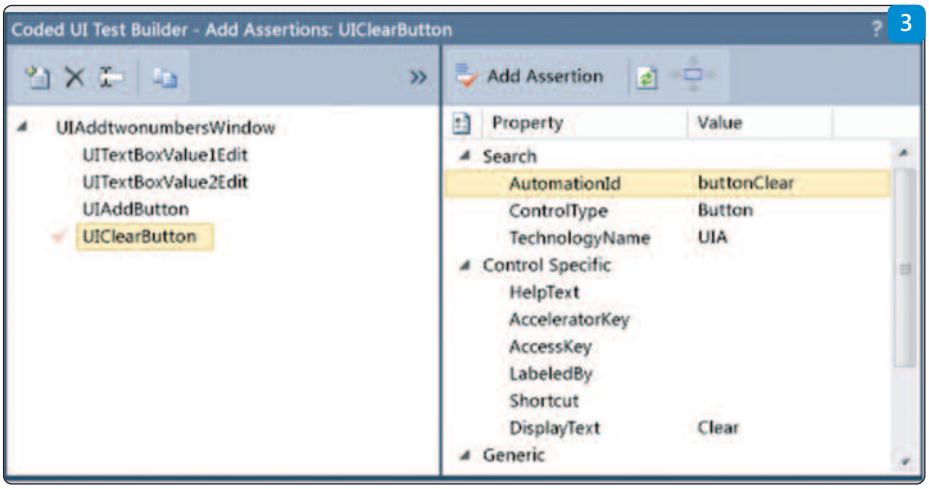
2



1



3



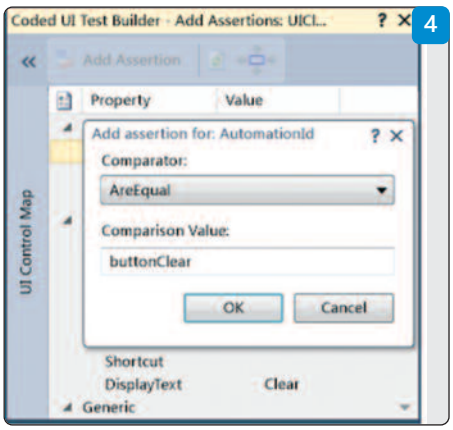
Pour ajouter l'assertion à notre test automatisé, on va générer le code de cette assertion (Dernier bouton du CodedUITestBuilder). Après avoir choisi le nom de la méthode de l'assertion, elle sera ajoutée à la classe UIMap et l'appel de cette méthode sera lui ajouté à la méthode globale dans le fichier CodedUITest1.cs. Néanmoins, vous pourrez déplacer l'appel de cette méthode à votre guise **(Fig. 5)**.

Bien que chaque test case codé dans le codedUITest soit directement exécutable, on peut les combiner dans un Ordered Test pour créer et exécuter une campagne complète à la demande, cet ordered test devenant lui-même un test case **(Fig. 6)**.

Sans sortir de notre sujet, on peut ajouter que l'ordered test peut contenir non seulement des tests case automatisés mais également des tests de performance.

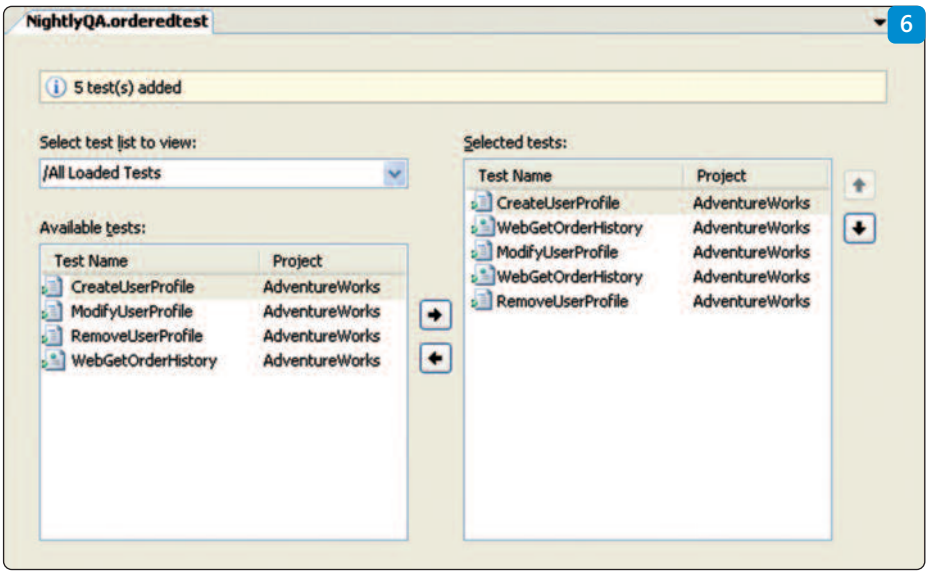
TFS inclut un processus complet de build de vos applications. Sans en expliciter les détails ici, on peut déjà dire que construire une build de son application ne se résume plus à une simple compilation d'un projet Visual Studio. On a aujourd'hui besoin, en plus de compiler ses pages de code, de créer un numéro de release, de la différencier en détail de la précédente, d'effectuer toutes sortes d'actions au niveau de la base de données, de créer une nouvelle instance de campagne de tests et beaucoup de choses encore. TFS va nous aider en fournissant un outil de configuration de build qui nous permet, une fois mis en place, de builder l'application en un seul click mais au-delà de cela, de la planifier.

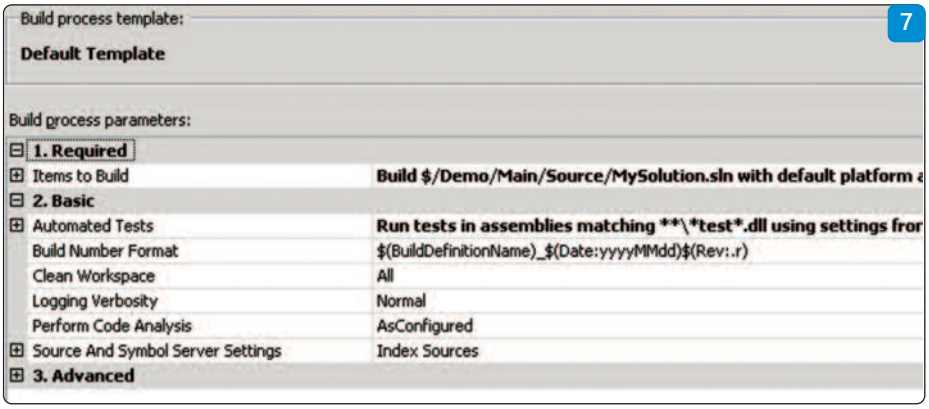
Mais pour revenir à notre sujet, ce processus de build automatique peut imbriquer nos campagnes de tests automatiques sur la build faite et déployée. Le LAB CENTER fourni en complément nous laisse gérer nos environnements pour choisir les emplacements de déploiement et même créer à la volée des machines virtuelles, y déployer nos applications buildées et jouer nos campagnes de tests automatiques et tests de performance **(Fig. 7)**.




```
5
public void checkSearchButtonText()
{
    Variable Declarations

    // Verify that 'Search' button's property 'DisplayText' equals 'Search'
    Assert.AreEqual(this.checkSearchButtonTextExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);
}
public void CodedUITestMethod1()
{
    // To generate code for this test, select "Generate Co
    // For more information on generated code, see http://
    this.UIMap.SeconnecteràURL();
    this.UIMap.Entrervilledanslechampville();
    this.UIMap.checkSearchButtonText();
    this.UIMap.CliquersurSearch();
    this.UIMap.Fermerlenavigateur();
```





Bref, Microsoft fournit ici un outil puissant, simple à prendre en main, économe de son temps, et complet. Imaginez : Partir du bureau le soir en ayant terminé ce qui était à faire dans votre journée, et vous le savez grâce aux indicateurs fournis.

Votre infrastructure se met en marche pendant que vous dormez : Votre application se compile, toutes les opérations de build se font, le produit se déploie sur des VM créés au bon moment, les tests automatisés se déroulent, jusqu'aux tests de performance.

Vous arrivez le lendemain matin et trouvez les résultats complets de qualimétrie de vos projets, que de temps gagné !!!

En conclusion

Coded UI de VSTS est donc un outil d'avant-garde qui peut s'intégrer nativement au cycle de développement de vos applications. N'hésitez pas à nous contacter.

Laurent **Abid**
CEO, T.T.C Testing Training & Consulting Ltd
Laurent@TTC-testing.com

Didier **Davar**
Consultant Senior Démarche outillée, T.T.C Testing Training & Consulting Ltd
didier.d@ttc-testing.com

Planifier et organiser les tests avec Microsoft

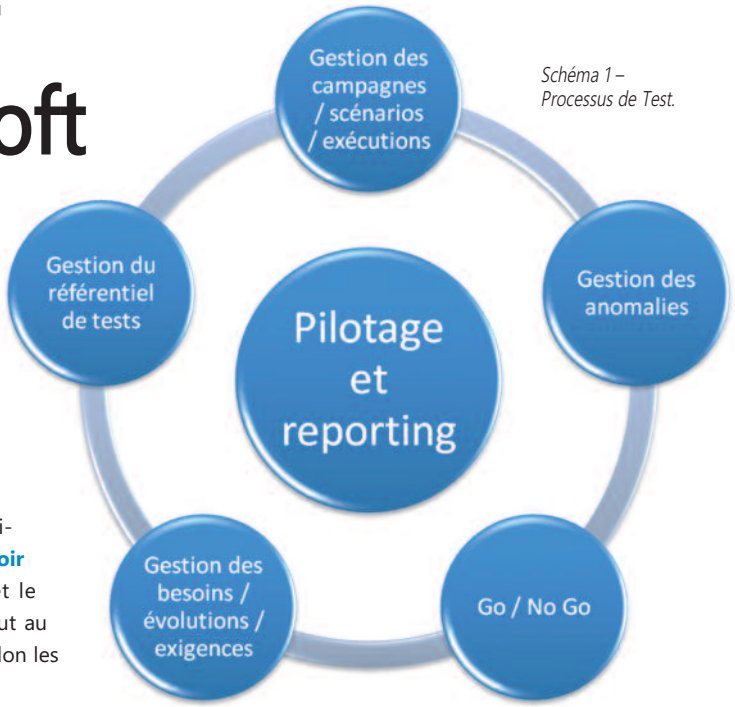


Schéma 1 – Processus de Test.

Microsoft intègre une série d'outils de tests dans sa solution Visual Studio 2010 (Application Life cycle Management). Un processus complet de suivi des plans de test y est proposé avec des solutions de collaboration entre développeurs et testeurs.

Les objectifs sont de :

- simplifier la compréhension du besoin et du test associé,
- améliorer le dialogue et la visibilité entre le demandeur et le développeur tout au long du cycle de développement,
- faciliter la remontée d'informations sur les anomalies aux développeurs, par exemple, avec l'enregistrement vidéo des exécutions des tests.

Microsoft vise également la démocratisation des tests plus accessibles et économiques. C'est une réponse aux besoins croissants des projets qui recherchent plus de qualité, de performance et une mise en production plus rapide.

La capacité de supporter des processus agiles est un facteur décisif d'adoption des outils de tests de Visual Studio. Près de 80% des entreprises affirment qu'avec son approche Microsoft Agile, Visual Studio est une étape stratégique vers une organisation Agile. Toutefois, cela ne signifie pas que l'on peut se passer de quantifier les bénéfices de l'assurance qualité et des outils de test. Il est vivement recommandé de mettre rapidement en place des indicateurs de qualité et d'amélioration du cycle de développement.

Par où commencer ? Les processus de tests et de transformation de la qualité s'articulent toujours autour de mêmes fondamentaux : la gestion des exigences, du référentiel des tests, des campagnes d'exécutions, des anomalies et la décision du « Go/no Go » - **voir schéma 1**. Le pilotage et le reporting sont assurés tout au long du cycle des tests selon les indicateurs établis.

Sur la base des exigences de tests, les responsables des tests élaborent un référentiel afin de gérer l'effort pour chaque itération programmée. L'objectif est ainsi d'avoir une vue complète sur tous les éléments (les cas de tests, les configurations). Plusieurs référentiels de tests peuvent être gérés pour un projet donné, toutefois, afin de faciliter la capitalisation, il est préférable de n'en prévoir qu'un par projet.

Chaque plan de test correspond à une mise en situation différente. Il permet de définir les conditions spécifiques des tests tels que le nom, le contenu du plan de test, son responsable ou son état d'avancement. Notons qu'avec la fonctionnalité Build de TFS 2010, l'équipe informatique peut optimiser ou automatiser certaines étapes de l'ALM en vue d'une méthodologie agile en accélérant la cadence de montée en production de l'application tout en maintenant un niveau de qualité élevé et en garantissant le contrôle des coûts **(schéma 2)**.

A partir d'un plan de test, il est possible d'exécuter des tests manuels et d'automatiser les tests en utilisant Microsoft Test Manager avec Visual Studio Test Professionnel 2010 ou Visual Studio 2010 Ultimate. Quel que soit le test lancé, l'utilisateur peut sauvegarder les résultats des tests directement dans le projet d'équipe sur Team Foundation Server. Cela permet de voir simultanément où en sont les tests automatisés et les tests manuels réalisés.

Avec Visual Studio 2010 Ultimate, le pilotage de la correction des bugs se réalise soit avec une vision des bugs priorisés soit avec la vision de couverture des exigences de tests, puisque les anomalies et les tests sont liés aux exigences. Cela permet de concentrer son effort de correction sur ce qui est important. Le traitement des bugs est aussi aisément confié à un autre utilisateur, car Visual Studio 2010 donne accès à plus d'informations partagées. Une fois le bug traité, il peut être assigné à un testeur pour vérifier que le problème a été réglé sans incidences secondaires sur l'application.

Les utilisateurs de Visual Studio 2010 Ultimate ou Premium peuvent également automatiser les tests de l'interface utilisateur (coded UI tests). L'automatisation des tests, qui est un projet dans le projet, permet de rejouer, à volonté et plus rapidement que les tests manuels, un ensemble de tests prioritaires et importants à moindre coût. Dans la mesure où la maintenance de ces tests peut être lourde, il faut définir le ROI et le risque du référentiel de tests pour choisir les éléments à automatiser. C'est la prochaine étape pour transformer la qualité, l'efficacité et la performance du développement d'applications de l'entreprise tout en réduisant les coûts.

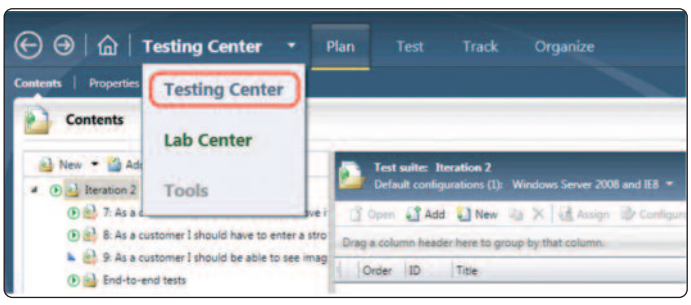


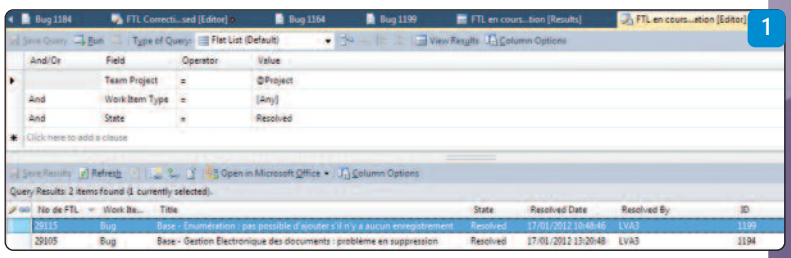
Schéma 2 – Définition des Plans de test à partir de Test Center de Visual Studio 2010 / Microsoft.

Salam **Elias**,
Responsable de l'Alliance Microsoft et Directeur de projets chez Cognizant (http://www.cognizant.com). Consulter son blog : http://salam.hd.free.fr/BlogEngine/

Témoignage

Améliorer la qualité logicielle avec Team Foundation Server 2010

La société Log'in Space est un éditeur de logiciels dans le domaine de l'immobilier (gérance, syndic, location saisonnière, etc.). L'éditeur fait partie des leaders dans le domaine en France.


1

Avec une gamme existante de logiciels développés dans des technologies comme Cobol et 4D, la société a fait le choix stratégique de redévelopper la gamme de logiciels avec les technologies .net, en utilisant entre autres, des technologies comme WPF pour proposer une ergonomie originale et efficace, et WCF pour l'exposition de la logique métier et d'accès aux données sous forme de services. Afin d'encadrer le développement de l'applicatif, le choix de Team Foundation Server a été fait afin de profiter de ses différentes fonctionnalités et de son intégration au sein de Visual Studio. Nous vous proposons au travers de cet article de vous présenter deux éléments mis en oeuvre pour améliorer la qualité de nos produits grâce à TFS 2010.

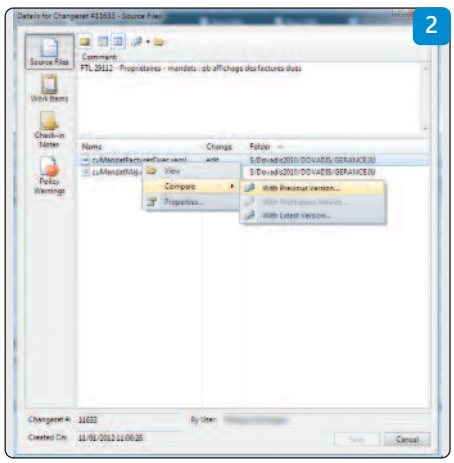
Les revues de code technique

En plus des sessions de pair programming qui permettent de partager des points de vue sur la manière d'écrire du code, mettre en place des revues de code permet de s'assurer que le code « checkiné » est correct, qu'il ne comporte pas de potentiels bugs, qu'il respecte les normes de développement, etc. Visual Studio 2010 ne propose pas de fonctionnalité permettant de mettre en place de manière naturelle ce type de revue de code (la prochaine version corrigera ce défaut en proposant un outil de revue de code intégré). Il est cependant possible de mettre en place efficacement ces revues en profitant de la traçabilité offerte entre les Work Items (tâches et bugs) et les changesets. Mettre en place ce type de revue de code est simple. Assurez-vous que l'ensemble des développeurs associent leurs changesets avec un work item lors de leur checkin, en mettant en place une Checkin-Policy ou en en indiquant cette obligation dans les normes de développement.

Afin de savoir quels sont les éléments à réviser, créer ensuite une requête de Work Item qui permet de lister l'ensemble des WorkItems qui sont en état Resolved. Pour chaque tâche/bug terminé/résolu par un développeur **(Fig. 1)**.

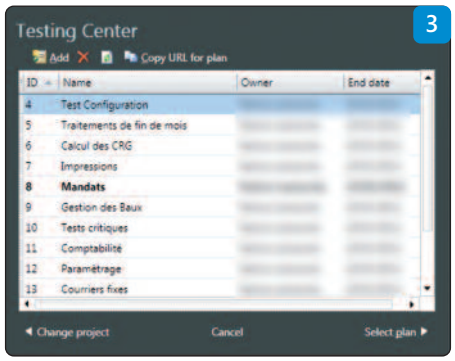
Il vous est ensuite possible d'effectuer des revues de code quotidiennes en analysant le différentiel de code implémenté pour résoudre les bugs concernés **(Fig. 2)**.

La mise en place de ce type de revue de code a cependant un inconvénient majeur : les revues de code ne sont effectuées que sur du code qui est déjà présent en source control.


2

La mise en place de tests UI

La mise en place de tests est fondamentale pour parvenir à améliorer la qualité d'un projet, réduire le risque de régressions et travailler sereinement. La mise en place à posteriori de tests unitaires et de tests d'intégration peut être assez complexe et coûteuse en temps. Une première étape assez simple à mettre en œuvre est d'automatiser les tests manuels. Pour cela Microsoft propose un outil nommé Microsoft Test Manager. Cet outil permet de gérer des plans de tests qui contiennent des cas de tests destinés à vérifier le bon fonctionnement d'une fonctionnalité **(Fig. 3)**.

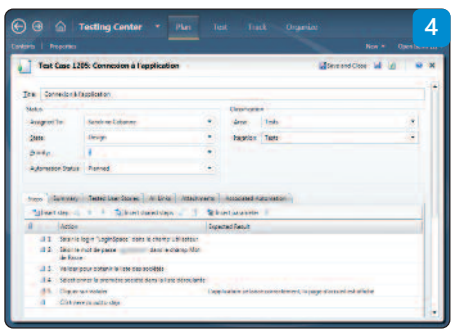


3

L'outil permet d'enregistrer des séquences d'actions effectuées sur l'interface utilisateur afin d'être capable de les reproduire très rapidement sans avoir à effectuer de manipulations manuelles. Ceci implique que le résultat final doit toujours être vérifié manuellement par le testeur en fonction des indications fournies par l'outil **(Fig. 4)**.

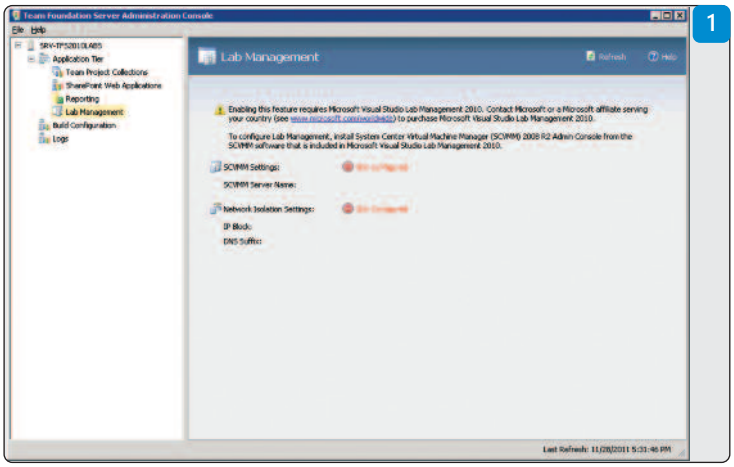
Afin d'arriver à effectuer automatiquement les vérifications (les asserts) il est nécessaire d'utiliser Visual Studio afin de générer un Coded UI Test à partir du cas de test auparavant enregistré. Celui-ci va générer le code nécessaire à l'exécution du test, et va permettre d'écrire (ou de générer) le code nécessaire à la définition de l'Assert. A noter que la définition de tests UI ne se fait pas aussi idéalement qu'on le souhaiterait. Il est fort probable que les développeurs aient à revenir sur le code XAML de l'application pour ajouter différentes informations telles que la propriété Name si elle est manquante par exemple. De plus, si la création d'assertions peut être très simple dans certains cas, il n'est pas rare qu'il soit nécessaire d'écrire différents outils pour simplifier l'écriture de celles-ci. Je peux vous recommander la suite d'outils de Jonathan Antoine dans ce domaine : http://bit.ly/xRBNL9

Patrice **Lamarche**
Leader Technique (Log'in Space)


4

Intégration continue et tests avec TFS Lab Management

« Sur ma machine, ça fonctionne pourtant ». « Impossible de reproduire ce bug ». N'avez-vous pas déjà entendu ces phrases dans vos projets de développement ? Avec leur complexité croissante, les outils d'ALM apportent de nouvelles solutions pour assurer une forte qualité logicielle traçable et automatisée. Team Foundation Server propose la fonctionnalité Lab Management afin de faciliter la reproduction des bugs, l'automatisation des tests et des déploiements ainsi que le suivi de la qualité. Dans cet article, nous verrons comment le mettre en place puis son utilisation et ce qu'il peut nous apporter.



1

Infrastructure et déploiement

La réticence à déployer Lab Management découle essentiellement de la complexité de son infrastructure. Pour gérer les environnements virtuels des projets d'équipe, TFS requiert un serveur System Center Virtual Machine Manager 2008 R2.

On peut résumer très synthétiquement SCVMM comme un centre de contrôle pour hyperviseurs. Avec la mouture actuelle, vous pouvez gérer de manière agrégée le déploiement et la maintenance de machines virtuelles réparties sur plusieurs hôtes aussi bien Hyper-V que VMware, par une interface graphique ou avec du PowerShell. Cet article n'a pas pour but de détailler le déploiement d'un SCVMM mais il faut savoir que c'est en lui-même un projet à part entière.

Quand on ouvre les propriétés de Lab Management dans la console d'administration de TFS, on constate qu'on ne peut rien faire tant que la Console d'administration de SCVMM n'a pas été installée. **(Fig.1)**

Il faut donc l'installer depuis le disque/iso d'installation de SCVMM et s'assurer que le compte de service de TFS dispose du rôle administrateur dans SCVMM.

Une fois ceci fait, on peut commencer à configurer les options de base de Lab Management, à savoir référencer le serveur SCVMM par son nom complet puis les paramètres d'isolation réseau. L'isolation réseau permet à TFS d'instancier des VM sans polluer le réseau existant mais aussi d'avoir plusieurs environnements d'un même projet en même temps (pour plus de détails : http://msdn.microsoft.com/en-us/library/ee518924.aspx#NetworkIsolationChecklist).

Il y a plusieurs choix de topologies possibles mais qui ont des complexités différentes en fonction de l'infrastructure existante.

Après avoir configuré les bases au niveau global TFS, il faut maintenant configurer plus spécifiquement au niveau de chaque collection de projets **(Fig.2)**.

Ces options sont le cœur du Lab Management puisque c'est ici que l'on va configurer les bibliothèques de VM, les groupes d'hôtes (hyperviseurs) sur lesquels elles seront déployées et le compte de service de workflow qui sera utilisé pour les builds et les tests. L'intérêt des groupes d'hôtes est que l'on n'a pas à se soucier de savoir où sont placées les VM : SCVMM le fait pour nous en calculant, d'après les préférences établies par l'administrateur, l'hyperviseur le mieux noté pour la tâche en question.

On profite là de la première itération des outils MS pour le Cloud privé appliqué au développement. L'intérêt que ces paramètres soient au niveau collection de sites réside bien entendu dans le fait de pouvoir avoir différentes configurations. On peut ainsi avoir une collection de projets prioritaires qui bénéficiera d'hyperviseurs plus véloces et de VM avec plus de ressources par rapports aux projets moins importants.

N.B. : *actuellement, TFS 2010 ne supporte pas les hôtes en cluster pour le provisionnement automatique des VM MAIS il fonctionne parfaitement pour des VM provisionnées et configurées manuellement ou avec VM prep tool .*

Afin de pouvoir dispatcher les tests automatisés, on doit installer un contrôleur de tests, idéalement sur un serveur dédié, provenant du même disque que les agents.

L'installation et la configuration est très simple et la capture **(Fig.3)** parle d'elle-même.

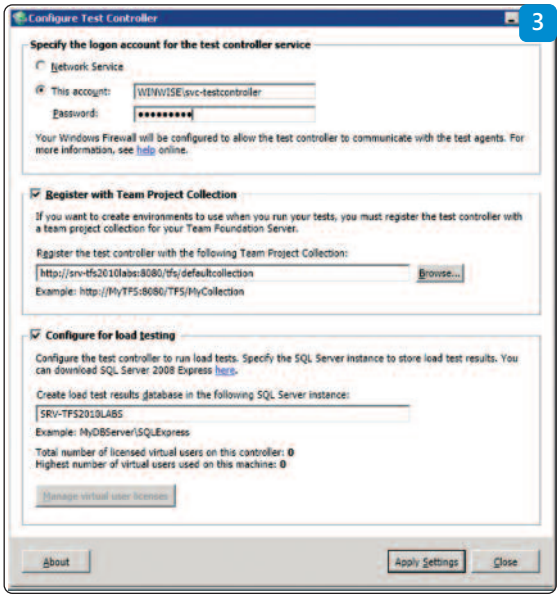
Enfin, pour avoir une configuration supportée et bénéficier des dernières fonctions de TFS, il faut installer un hotfix disponible à l'adresse suivante : http://support.microsoft.com/kb/983578.

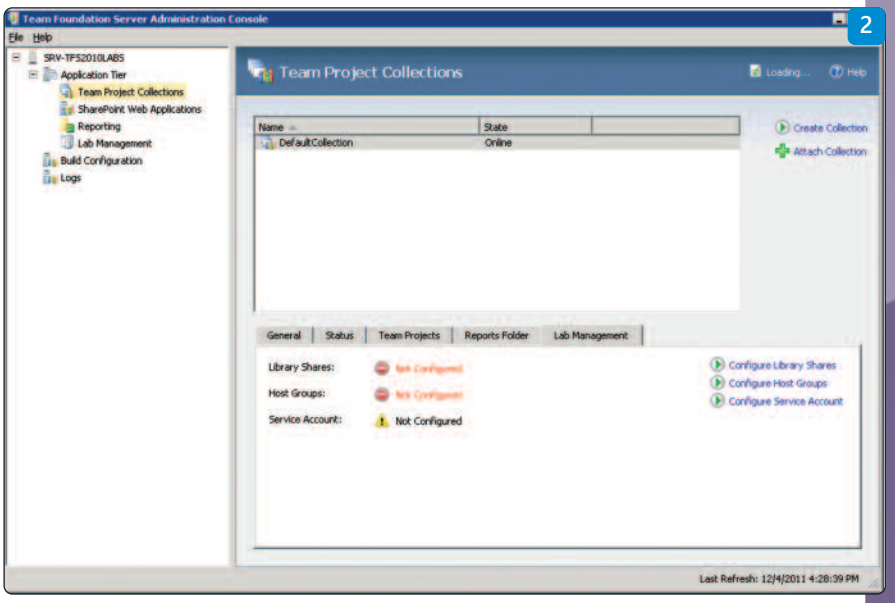
Utilisation de Test Manager

Maintenant que la plateforme est mise en place, on ne va pas se priver d'utiliser l'ensemble des fonctionnalités offertes par le Lab Management !

L'outil au cœur de l'utilisation du Lab Management est le Test Manager. Si vous avez déjà l'habitude de définir des plans de test, vous avez déjà eu affaire à lui.

Difficile de choisir par quoi commencer, l'outil est vraiment complet et répond aux problématiques rencontrées le plus fréquemment au cours de la vie d'un logiciel. Nous allons aborder les plus courantes et voir comment le Lab





Management, couplé au reste de la plateforme TFS bien évidemment, y répond.

« Je dois tester le déploiement de mon application sur différents environnements cibles »

Sans le Lab Management, on est obligé de gérer soi-même les machines sur lesquelles on déploie l'application et cela devient vite fastidieux si on doit couvrir un nombre important de systèmes d'exploitation.

Avec, tout devient plus facile :

Commencez par créer à travers SCVMM autant de machines virtuelles que d'environnements différents sur lesquels vous souhaitez tester le déploiement de votre application. Installez également les agents de Build TFS 2010, tests et lab sur chaque machine. Faites un snapshot sur chaque environnement une fois que vous avez installé tous les prérequis nécessaires à votre logiciel.

N.B. : *pour simplifier ces tâches, un outil existe : VM prep tool disponible ici :*
http://archive.msdn.microsoft.com/vslabmgmt

Stockez ensuite ces VM dans la librairie de SCVMM. Vous aurez le choix entre les stocker telles quelles ou les enregistrer en tant que template. Vous choisirez la deuxième option si vous avez besoin d'instancier plusieurs fois une même VM pour vos tests.

Importez dans votre projet TFS les VM dont vous avez besoin à partir de SCVMM. Cela se passe dans le Test Manager. Notez qu'étant donné que les VM sont stockées dans SCVMM, une seule VM peut être réutilisée dans plusieurs Team Projects.

Créez un environnement (toujours via Test Manager) contenant toutes les VM sur lesquelles vous souhaitez déployer votre application.

Créez une build utilisant le template « LabDefaultTemplate.xaml ». Dans l'onglet Process vous aurez accès à un wizard de configuration vous permettant de sélectionner l'environnement précédemment créé.

Il ne vous manque plus qu'à indiquer quelles lignes de commande vous souhaitez exécuter sur chaque VM de votre environnement afin de déployer les différents composants de votre application.

Vous pouvez maintenant lancer votre build et observer les résultats pour contrôler le bon déploiement de votre application sur les différents environnements.

On vient de voir rapidement l'automatisation du déploiement d'une application sur différents environnements cibles. Cela peut paraître un peu long à mettre en place, c'est vrai à l'initialisation, mais vous gagnez :

- L'automatisation du déploiement de votre application, peu importe les modifications de code
- L'évolution simple de vos environnements cibles (il suffit de changer les VM de votre environnement)
- La possibilité de vous connecter sur les VM une fois l'application déployée pour effectuer des tests

Les possibilités en termes de paramétrage des environnements et des build de déploiement sont immenses. Le template de build fourni par défaut peut être modifié (comme tout template de build). Vous pouvez gérer finement les paramètres réseau des VM : fonctionnement dans un réseau isolé, interaction avec le réseau de l'entreprise). Tout a été prévu pour permettre la définition des environnements correspondant au contexte réel de votre application, aussi complexe soit-il. Cela est déjà intéressant mais on peut aller beaucoup plus loin.

« Je veux tester mon application sur différents environnements cibles »

Déployer son application automatiquement, c'est un bon début mais cela ne garantit pas qu'elle fonctionne. Allons un peu plus loin dans l'automatisation et rajoutons des tests à tout cela. La plus grosse partie du travail se passe dans l'outil Test Manager.

Si vous êtes déjà familiarisé avec la création de plans de test vous n'aurez aucune difficulté à définir l'ensemble des tests que vous souhaitez exécuter de manière automatisée.

Il vous faut en effet définir un plan de test composé de suites de tests composées elles-mêmes de cas de tests (automatisés ou non).

Une fois que vos plans de test sont définis, il ne vous reste plus qu'à les associer à la build qui effectue déjà le déploiement de l'application. En relançant le wizard de configuration évoqué précédemment et en suivant l'étape « Test » vous pourrez sélectionner les plans/suites de tests/cas de tests que vous souhaitez exécuter.

Voilà, votre build qui déployait votre application exécute désormais les tests automatisés dans la foulée.

Les résultats sont consultables via le rapport de build ou dans le Test Manager en consultant les résultats des plans de test.

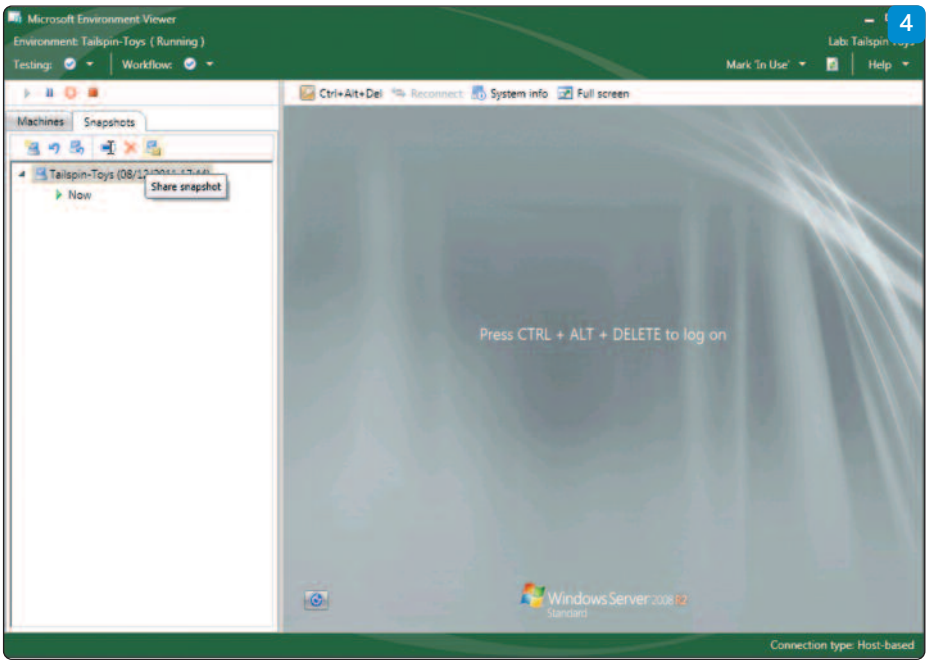
Les tests exécutables de cette manière sont :

- Les tests unitaires
- Les tests web
- Les tests codés d'UI

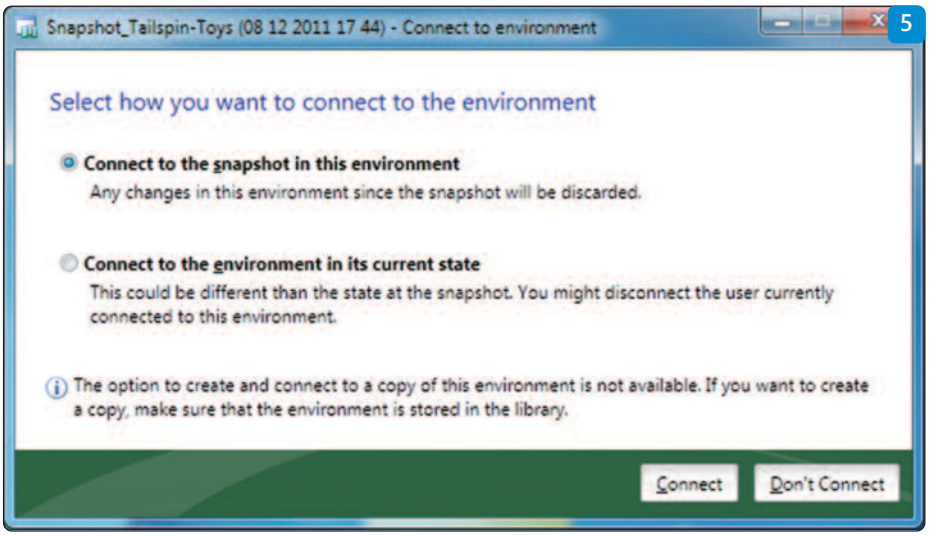
C'est bien me direz-vous mais comment fait-on pour les tests manuels ?

Rien de plus simple en utilisant les options fournies. On peut en effet indiquer à notre build qui déploie puis teste l'application de faire un snapshot de l'environnement après avoir déployé

l'application et avant d'avoir exécuté les tests. En passant par le Lab Center nous pouvons accéder à notre environnement virtualisé et décider de restaurer le snapshot créé par l'exécution de la build puis nous connecter à l'environnement. Nous nous trouvons donc sur notre environnement cible sur lequel la version de l'application que nous voulons tester est installée. Il ne reste plus qu'à lancer le Test Center depuis cette même machine, de parcourir le plan de test concerné, de sélectionner le cas de test manuel à dérouler et de cliquer sur « Run ». Nous sommes donc en train d'exécuter un test manuel dans les mêmes conditions que les tests automatisés qui ont été déclenchés par la build. Petite cerise sur le gâteau, si jamais vous détectez une anomalie lors du test manuel, vous pouvez prendre un snapshot de l'environnement. Un lien vers ce snapshot sera alors automatiquement ajouté lorsque vous créerez le bug associé au cas de test manuel.
Le partage de snapshot peut aussi se faire en exportant un fichier .lvr qui peut être envoyé par email à un autre collaborateur **(Fig.4)**.



Ce fichier ne contient qu'une simple ligne de connexion (ex « CommandUrl=levr://ConnectToFwdLink?CollectionURI=http://srv-tfs2010labs:8080/tfs/DefaultCollection&ForwardLink=1 »). Cela permet à la personne en charge de la résolution du bug de constater ce qui s'est produit et de pouvoir analyser les logs système et applicatifs générés pas votre test **(Fig.5)**.



Comme cela, fini le souci du « chez moi ça marche » qui ne fait pas avancer la résolution du bug. Vous pouvez tester directement dans les mêmes conditions que la personne qui a détecté l'anomalie.

Conclusion

Nous avons vu dans ces quelques pages une présentation rapide de Lab Management dans TFS mais il faudrait un livre entier pour réellement être exhaustif sur le périmètre fonctionnel de l'outil.
Pour des projets conséquents et, avec de l'investissement en assurance qualité, le TCO arrive à rester modéré.

Qu'en est-il de l'avenir ?

Les gammes System Center 2012 et TFS 11 sont d'ores et déjà en préversions publiques et apporteront leurs lots d'améliorations.
Concernant Lab Management vNext, Brian HARRY, le père de TFS, a déjà annoncé qu'il serait plus simple à mettre en place en ne nécessitant plus d'être intégré à un SCVMM pour fonctionner (http://blogs.msdn.com/b/bharry/archive/2011/10/31/lab-management-improvements-in-tfs-11.aspx).
Les projets pourront ainsi être déployés et testés sur des machines physiques existantes qui auront été configurées en conséquence.
Bien entendu, pour des infrastructures plus conséquentes, l'outil pourra exploiter SCVMM 2012. Il n'y a, à l'heure où ces lignes sont écrites, aucune annonce sur les évolutions prévues sur ce point. On espère que les clusters d'hyperviseurs et les clouds privés pourront être intégrés avec TFS.
Comme pour tous les déploiements, vous devez vous demander s'il est préférable d'attendre avant de mettre en place les vNext. La réponse dépend de la volumétrie à mettre en place. Une forte volumétrie (plusieurs dizaines voire centaines de VM) impliquant un projet d'une durée supérieure ou égale à 10/12 mois pourra justifier d'attendre.

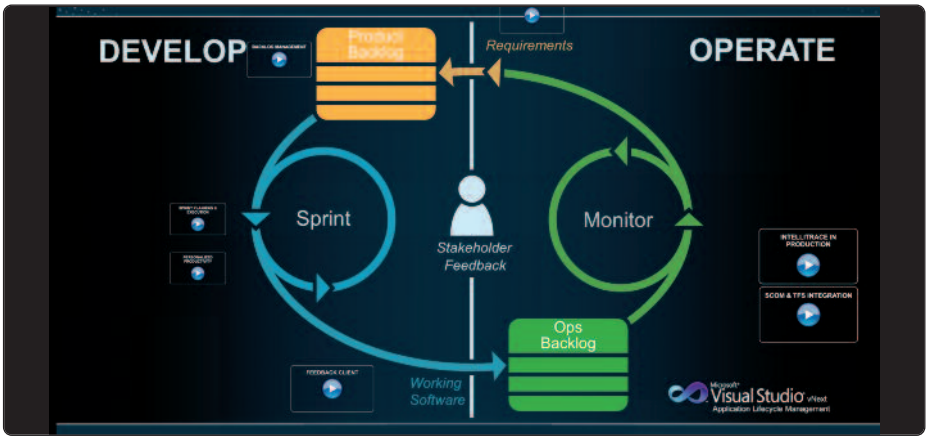
Liens utiles

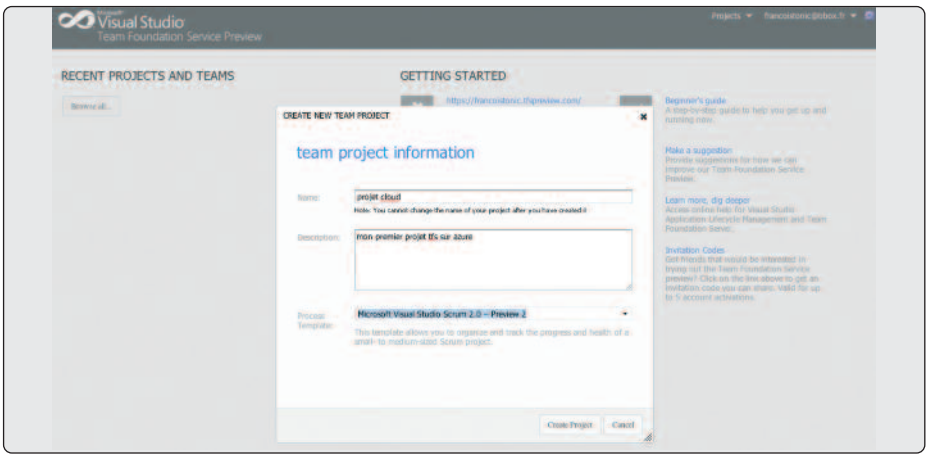
Utilisation d'un lab virtuel pour le cycle de vie de votre application : http://msdn.microsoft.com/fr-fr/library/dd997438.aspx
Le blog d'équipe de Lab Management (en anglais) : http://blogs.msdn.com/b/lab_management/
Les vidéos d'apprentissage MSDN (en anglais) : http://msdn.microsoft.com/en-us/vstudio/gg537361#lab

Pierrick **Catro-Brouillet**
Expert SharePoint et Infrastructure, Winwise
http://blogs.developpeur.org/spbrouillet
Cédric **Noureau**
Consultant/Chef de projets, Winwise

Aperçu de la prochaine version

Collaboration, agilité, gestion du cycle de vie, tests, quatre piliers essentiels qui sont aujourd'hui les fondations de TFS et de Visual Studio. Microsoft travaille activement à la prochaine version de Visual Studio.





La philosophie demeure identique : couvrir toujours mieux le cycle de vie, améliorer le travail d'équipe, mettre des interactions toujours plus efficaces. Voyons ensemble quelques éléments de cette prochaine version.
Pour cette future version, les équipes Microsoft ont concentré leur travail sur quatre éléments :

- collaboration : concentrer les flux entre les membres des équipes en valorisant ce que ces flux peuvent apporter aux collaborateurs
- « feedback actionnable » : il arrive qu'un commentaire, un retour utilisateur soit nécessaire à l'équipe pour corriger, modifier des éléments d'un projet. Mais comment faire concrètement ? L'idée est de pouvoir utiliser directement le commentaire. Par exemple, un testeur pointe un défaut de développement, il devrait pouvoir agrémenter son commentaire de vidéos, images, informations précises sur la configuration, un log Intellitrace. Bref, tout ce qui serait utile aux développeurs pour résoudre, corriger un problème.
- Les styles de travail : adapter les meilleurs outils pour les différents membres des équipes. Car les attentes, les besoins d'un développeur, architecte, testeur… ne sont pas identiques et leurs manières de travailler sont parfois très différentes.
- Processus agiles transparents : que toute l'équipe travaille sur une source unique vérifiée et qu'on puisse avoir à tout moment l'exacte situation du projet, des tâches, des sources.

Une collaboration toujours plus fluide

vNext doit supprimer toutes les petites choses qui détournent testeurs, développeurs et l'ensemble des équipes de leurs tâches, du projet. Il s'agit aussi d'améliorer les flux d'informations entre les utilisateurs et l'équipe et entre le Product Owner (Scrum) et les utilisateurs. Cela passe par une transparence dans les flux, la planification. Il s'agit aussi de pouvoir récolter et utiliser les commentaires, les retours utilisateurs le plus rapidement possible et de les utiliser immédiatement. Pour le développement, il s'agit d'augmenter l'assurance qualité par une amélioration du code, de la revue de code, des frameworks de tests unitaires, dans l'exploration des tests et des tâches au quotidien.
Parmi les nombreuses nouveautés (voir page suivante), le lab management sera plus simple à utiliser et à mettre en œuvre. Pas la peine de disposer d'un serveur Hyper-V et de System Center Virtual Machine Manager. Un simple serveur (virtualisé ou physique, avec tous les éléments nécessaires) sera requis ! Et ça, c'est une super nouvelle ! Les préversions VS 11 et TFS 11 sont disponibles en téléchargement.
Enjoy !

Ressources

Site officiel : http://www.microsoft.com/visual studio/en-us/roadmap
Blog en Français pointu sur le sujet : http://blogs.developpeur.org/etienne/
Blog sur Lab Management : http://blogs.msdn.com/b/bharry/archive/2011/10/31/lab-management-improvements-in-tfs-11.aspx

François **Tonic**

TFS prend le virage cloud !

Dévoilé officiellement en septembre 2011, Team Foundation Server sur Windows Azure (préversion) est un portage TFS sur la plateforme Windows Azure de Microsoft. Il s'agit d'un service hébergé TFS. En quelques minutes, il est possible de mettre en place un projet et surtout de disposer d'un TFS sans avoir besoin de le déployer localement. Si à l'heure actuelle, ce TFS n'est pas complet, il propose tout de même des fonctions très complètes :

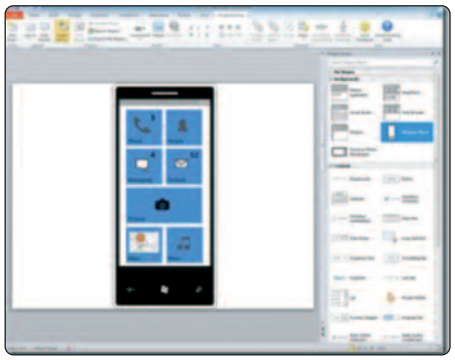
- contrôle de source
- suivi des work item
- gestion des projets agiles (nouveauté de TFS 11)
- outils de tests
- automatisation des builds

TFS pour Windows Azure se veut ouvert et peut s'interconnecter avec Visual Studio 11, Team Explorer Everywhere, VS 2010 et Test Professional 2010. Le démarrage se déroule en trois grandes étapes :
1 - URL du compte pour accéder à son TFS Azure
2 - création de l'équipe projet
3 - installation des clients nécessaires pour connecter l'équipe à TFS…

Les principales nouveautés VS 11 sur le cycle de vie

1 GESTION DES EXIGENCES "LÉGÈRE"

Afin de pouvoir gérer de manière simple et rapide les exigences on aura un plugin pour Powerpoint permettant de rapidement créer une maquette de l'application. L'avantage, outre le fait que Powerpoint est un outil familier, est, qu'une fois la maquette finie (ou suffisamment avancée), on dispose automatiquement d'une présentation simple et efficace permettant de récupérer des commentaires des stakeholders (intervenants)

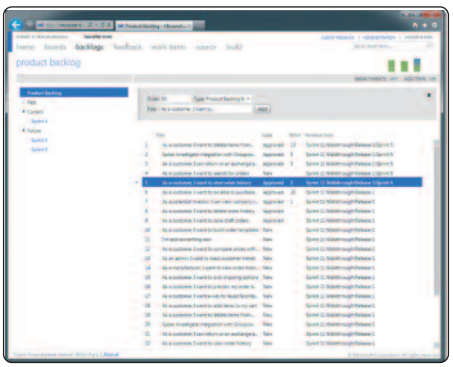


2 COMMENTAIRES/RETOURS DES STAKEHOLDERS

Le maquettage dans Powerpoint permet un premier feedback avant le développement de l'application. Afin de permettre d'obtenir du feedback lors du développement, Microsoft va proposer un outil se basant sur le TestRunner de VisualStudio Test Professionnal 2010. Une fois lancé, l'outil collectera des données (vidéo, son, actions) pendant que l'utilisateur se servira de l'application. Il pourra à tout moment faire un commentaire qui se traduira par la création d'un work item dans TFS contenant l'ensemble des données collectées.

Au niveau du web access on retrouvera aussi un onglet dédié au feedback pour permettre une meilleure communication entre équipe de développement et les intervenants.

3 PLANIFICATION AGILE



Les méthodologies Agile sont de plus en plus utilisées. Microsoft a donc revu son interface web access pour simplifier la partie planification. On va donc retrouver :

- Une gestion du product backlog avec possibilité de réordonner facilement les items par glisser/déposer.
- La gestion de la capacité directement dans l'interface web (il fallait passer par les workbooks Excel en 2010)
- Un taskboard afin de suivre et mettre à jour par glisser/déposer le statut des work items.

Comme vous pouvez le voir aussi sur ces captures d'écran, le style complet du site web access a été repris pour suivre les guidelines METRO. L'utilisation de METRO, du HTML et d'AJAX font que le site est très réactif et devient vraiment utilisable au jour le jour.

4 TESTS UNITAIRES CONTINUS

L'utilisation des tests unitaires n'est plus à démontrer. Par contre dans la pratique cela peut être contraignant car il faut penser à les exécuter, en fonction du nombre, cela peut prendre du temps etc. La prochaine version de Visual Studio apportera plusieurs améliorations:

- L'interface de tests a été revue afin de faire ressortir ce sur quoi le développeur doit se concentrer.
- Les tests sont aussi maintenant exécutés en permanence en tâche de fond.
- L'explorateur de tests unitaires supportera de nouveaux framework de tests !! On retrouvera en plus de MSTest, NUnit et XUnit. Si votre framework préféré n'est pas un de ceux-là, pas de problème, Microsoft a prévu un point d'extensibilité permettant à qui le veut de rajouter le support d'un autre framework ☺

5 TESTS EXPLORATOIRES

Il arrive que certains bogues soient trouvés en dehors de l'exécution des tests de recettes. On parle alors de tests exploratoires où l'utilisateur est libre de ses actions et peut ainsi découvrir des bogues lors d'un scénario non prévu. Il est souvent difficile dans ce cas de reproduire le bogue. Pour améliorer cela, la prochaine version de Visual Studio va supporter ce type de test en proposant un Test Runner "libre" (sans étape). Comme pour le test runner classique, les données et actions de l'utilisateur seront enregistrées.

6 INTÉGRATION AVEC SYSTEM CENTER

Afin de supporter les interactions avec les opérations, Microsoft va fournir un nouveau connecteur pour System Center Operation Manager afin de pouvoir créer des work items dans TFS directement depuis SCOM lorsqu'un problème intervient en production. Pour tous les fans de l'IntelliTrace, sachez que Microsoft prévoit de fournir une version pour la production !

7 MEILLEURE EXPÉRIENCE UTILISATEUR

D'une manière générale, l'onglet Team Explorer de Visual Studio va être repris et devenir un Team Navigator. Il sera plus simple de voir les informations et cela apportera une grande nouveauté : la gestion du contexte de travail. Qu'est-ce que c'est ? Lorsqu'un développeur voudra travailler sur une tâche, il indiquera via l'interface qu'il commence à travailler dessus. Visual Studio pourra alors suivre les modifications qui sont faites et les liera à la tâche en cours. Si le développeur doit changer de tâche (pour corriger un bogue par exemple), Visual Studio prendra l'ensemble du contexte en cours et le sauvegardera en le liant à la tâche. L'utilisateur pourra ensuite travailler sur autre chose et lorsqu'il indiquera qu'il reprend son travail sur la première tâche, son contexte sera automatiquement restauré !

Guillaume **Rouchon**
Wygwam
http://blog.qetza.net/fr/tag/team-foundation-server-vnext/