

Gestion de configuration en multi-sites

Étude de cas Team Foundation Proxy chez Schneider Electric



Nicolas VAN VOOREN & Olivier DUPONT
Avril 2007

Lancer aujourd'hui un projet informatique sans évoquer les termes d'industrialisation logicielle ou de *software factory* relève d'un exercice périlleux. En effet, ce « nouveau » credo de l'industrie des services informatiques a mis du temps à s'imposer comme une nécessité pour garantir une production logicielle fiable du point de vue fonctionnel et qualitatif, dans des délais toujours plus courts. Il y a encore quelques années la production de logiciels résultait souvent d'une alchimie complexe mêlant compétences individuelles et processus « maison », faute de pouvoir s'offrir des ateliers logiciels souvent complexes et perçus comme coûteux. Simple prise de conscience ou atteinte d'une certaine maturité, on peut espérer que cette tendance s'inscrive dans la durée. Ainsi, Microsoft — reconnu pour la richesse et la qualité de sa plateforme de développement Visual Studio — a étendu sa gamme avec **Team System**, un outil dédié à la mise en œuvre d'une véritable fabrique logicielle dont la tête de pont serait **Visual Studio 2005** et démocratise ainsi des concepts déjà existants chez d'autres outils. L'éditeur américain vient ainsi sur un marché où il entend bien rivaliser avec d'autres acteurs, voire offrir une alternative concrète aux solutions Open Source ayant fleuri ces dernières années.

Team System en quelques mots...

Cette étude n'ayant pas pour objectif de présenter en détail la gamme des outils Team System, nous nous contenterons d'en tracer les grandes lignes. Visual Studio Team Systems (abrégé VSTS) est une déclinaison de Visual Studio 2005. Il s'agit d'une plateforme s'articulant autour d'un serveur, Team Foundation Server, gérant un référentiel de développement et exposant des services pour y accéder. Les modules clients se déclinent en quatre éditions simplifiées — Edition for Software Architects, Edition for Software Developers, Edition for Software Testers et Edition for Database Professionals — et une édition complète Team Suite. Ces éditions s'intègrent directement au sein de Visual Studio 2005 Professional. Il est important de noter que la couche serveur de Team System s'appuie sur différentes technologies

déjà (re)connues telles que SQL Server 2005, Reporting Services et Sharepoint Services.

Les outils fournis dans VSTS permettent de traiter tous les aspects du cycle de vie d'un projet de développement informatique, de la conception à la fourniture des livrables. Toute la communauté des producteurs de logiciels est donc concernée : architecte, chef de projet, concepteur, développeur, gestionnaire de configuration, testeur, administrateur des données, etc.

Nous vous invitons à consulter les pages dédiées à Team System sur le site français de Microsoft (http://www.microsoft.com/france/msdn/vstudio/team_system/default.aspx) pour de plus amples informations ou de consulter le livre blanc de BÉNARD & MÉRAND (2005) introduisant les concepts d'industrialisation logicielle au travers de ce nouvel outil. Pour les aspects plus techniques, vous pouvez consulter l'ouvrage de LEVISON & NELSON (2006) ou encore celui de DAVID *et al.* (2006).

Team System et l'accès distant

Notre étude de cas a pour objectif de présenter une fonction essentielle du dispositif client/serveur de Team System : **l'accès distant**. En effet, la mise en œuvre d'une plateforme logicielle industrialisée doit répondre à l'exigence d'une répartition des équipes sur différents sites géographiquement distants (*offshore*, *nearshore* ou inter-agences). Team System est conçu pour répondre à cette exigence. Nous abordons donc cette problématique en présentant tout d'abord l'architecture technique correspondante puis en présentant les résultats d'une étude menée pour le compte de la division Customer Software de Schneider Electric.

Architecture technique

L'architecture client/serveur

L'architecture client/serveur de Team System s'articule autour du serveur principal, le Team Foundation Server, qui délivre les services applicatifs et gère les différents référentiels. Ainsi, chaque poste doit être connecté à son serveur TFS pour bénéficier des fonctionnalités de la plateforme. La

connexion s'effectue par enregistrement de l'adresse du serveur et par authentification de l'utilisateur selon un mécanisme classique pour Windows. Ce principe s'applique à tout poste présent sur le réseau local du TFS ou sur un poste distant, y compris si celui-ci n'appartient pas au domaine dans lequel est enregistré le TFS ; dans ce dernier cas, il suffira de s'authentifier avec un compte du domaine en question.

Dans le contexte d'un accès distant, la qualité de bande passante du réseau longue distance (WAN) est bien sûr essentielle pour assurer des temps de réponse satisfaisants pour les utilisateurs concernés. Pour améliorer le processus de récupération des fichiers disponibles via le contrôleur de code source, Team System propose la mise en place d'un **serveur proxy** (fig. 1).

L'architecture client distant

Le Team Foundation Proxy Server agit comme un serveur de cache pour maintenir, sur un site distant, tout ou partie des fichiers contenus dans le référentiel du contrôleur de code source. Ainsi, la récupération des fichiers effectuée depuis les postes distants s'effectue plus rapidement dès lors que ces fichiers sont présents dans le cache (fig. 2). Les différentes versions d'un fichier (demandées par les utilisateurs) sont même copiées

entièrement sur le proxy et non la différence entre les différentes versions. Chaque version peut ainsi être restituée plus rapidement sans qu'il soit nécessaire de reconstituer le fichier attendu. Mais alors le serveur proxy peut-il contenir l'ensemble du référentiel de code source ? En théorie pourquoi pas. L'espace disque alloué au cache du proxy est fixé à travers un fichier de configuration (`proxy.config`), il peut être défini selon une taille fixe (en Mo) ou selon un pourcentage de l'espace disque disponible. C'est donc la taille du disque et votre paramétrage qui influencent directement la capacité de stockage du cache. Il faut d'ailleurs noter que lorsque le cache s'approche de sa limite, un algorithme prend en charge la suppression des fichiers les plus anciens présents dans celui-ci afin de rendre possible la récupération de nouveaux fichiers ou de nouvelles versions de fichiers déjà présents.

Enfin, il faut savoir que **ce proxy n'affecte que les opérations de lecture des fichiers** du contrôleur de code source. Toutes les autres opérations, notamment l'archivage (*check-in*) ou la mise sur étagère (*shelving*) impactent directement le Team Foundation Server principal.

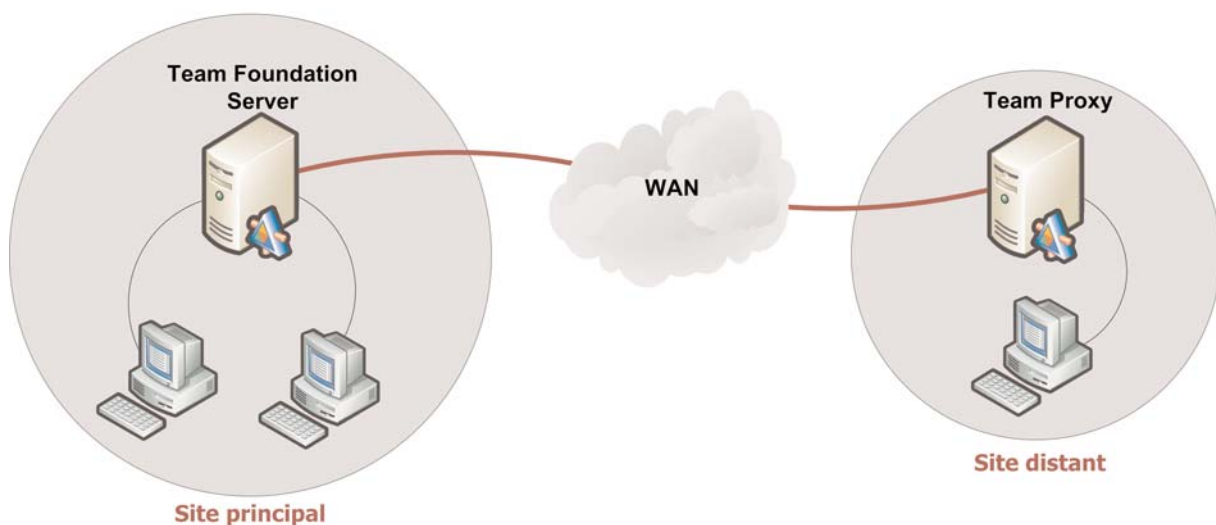


Fig. 1 – Architecture client distant de Team System

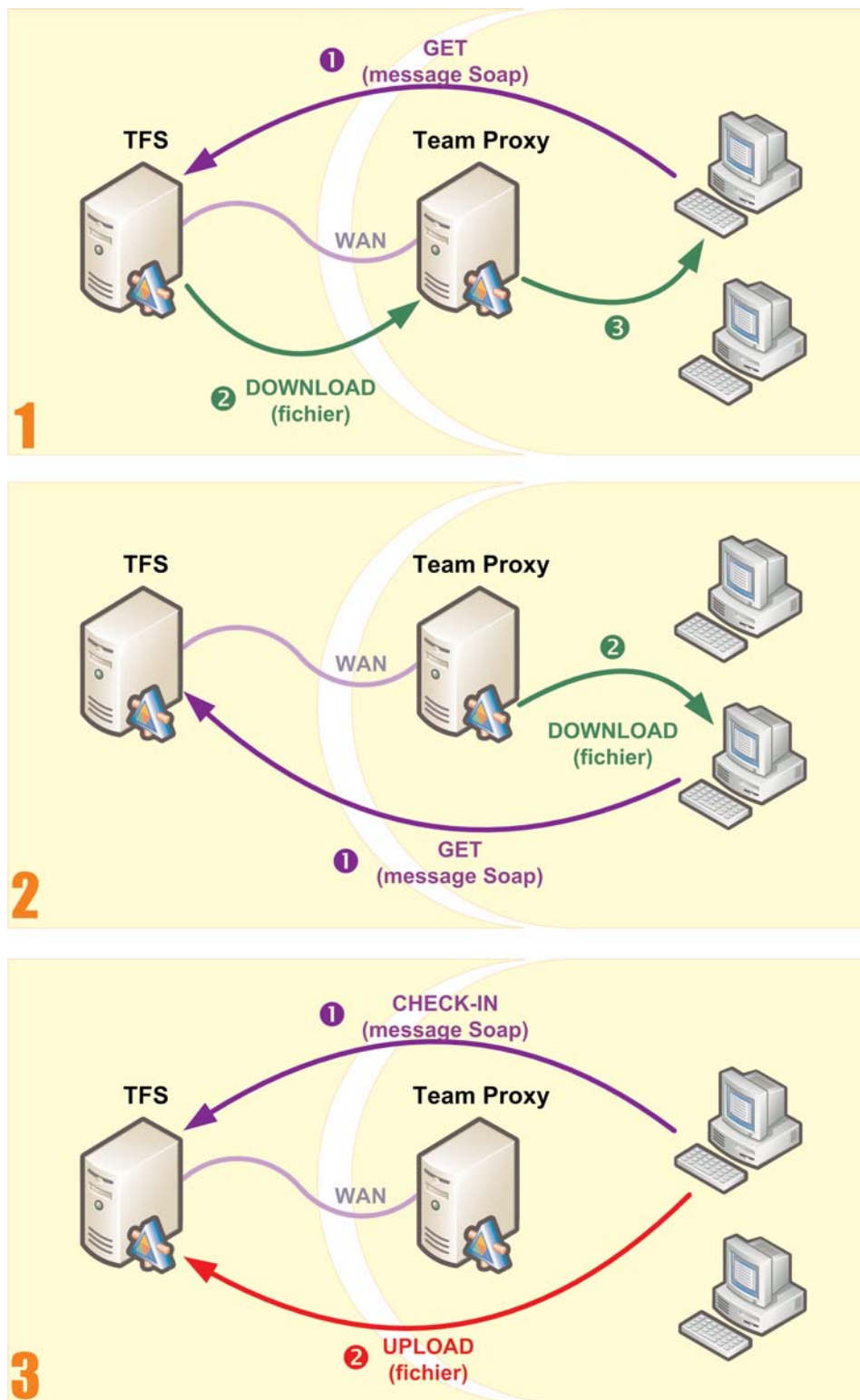


Fig. 2 – Architecture simplifiée des accès via le proxy

Étape 1 : première demande de récupération d'un fichier.

Étape 2 : deuxième demande de récupération du même fichier par un autre poste.

Étape 3 : mise à jour du fichier et réintégration dans le TFS source control.

Utiliser les API Team System

Actuellement, le proxy ne peut pas être paramétré pour automatiser son rafraîchissement, c'est-à-dire la capacité de déclencher à intervalle régulier l'équivalent du GET LATEST VERSION. Une telle fonction aurait l'avantage de masquer à l'utilisateur l'étape de téléchargement vers le système de cache.

Pour pallier à ce point, il est néanmoins possible de développer, par exemple, un service Windows qui prendrait en charge cette tâche. L'exemple d'utilisation des API .Net dédiées à Team System ci-dessous illustre la récupération de fichiers dans le *source control*.

Il est important de noter ici que la récupération des fichiers devant s'effectuer via le proxy, on peut éventuellement vérifier l'existence de ce dernier par une vérification dans la base de registre.

```
...
using Microsoft.Win32;
using Microsoft.TeamFoundation.Client;
using Microsoft.TeamFoundation.VersionControl.Client;
...
//Lire l'adresse du serveur proxy
private string GetProxyAddress()
{
    RegistryKey key = Registry.CurrentUser.OpenSubKey
(@"Software\Microsoft\VisualStudio\8.0\TeamFoundation\SourceControl\Proxy");
    if (key == null)
        throw new ApplicationException("Proxy non détecté");
    return key.GetValue("Url").ToString();
}

//Récupérer les fichiers présents sur le TFS
public void DownloadFilesFromServer(string pAddress)
{
    string projectName = "TEST_VSTS";
    string workspaceName = "wkSTEST_VSTS";

    //vérifier le proxy
    GetProxyAddress();

    //se connecter au TFS
    TeamFoundationServer tfs = TeamFoundationServerFactory.GetServer(pAddress,
new UICredentialsProvider());
    tfs.EnsureAuthenticated();

    //se connecter au Source Control
    VersionControlServer vcs = (VersionControlServer) tfs.GetService
(typeof(VersionControlServer));
    workspace wks = vcs.GetWorkspace(workspaceName, @"\MyDomain\Robot");

    //récupérer les fichiers
    ItemSet its = vcs.GetItems("$/" + projectName, RecursionType.Full);
    foreach (Item it in its.Items) {
        if (it.ItemType == ItemType.File) {
            GetFile(wks, it.ToString());
        }
    }
}

// Récupérer un fichier dans le source control
private void GetFile(Workspace pworkspace, string pNameFile)
{
    GetRequest getRequest = new GetRequest(pNameFile, RecursionType.None,
VersionSpec.Latest);
    GetStatus getStatus = pworkspace.Get(getRequest, GetOptions.Overwrite);
}
}
```

L'étude Schneider Electric

Contexte

Leader mondial en appareillage électrique et en automatismes, Schneider Electric a confié à sa division *Customer Software* (CS) le développement de ses logiciels d'avant-vente. En aidant ses clients dans la sélection de produits, à l'assemblage et au chiffage de solutions, les logiciels Schneider Electric leur permettent de concevoir des solutions adaptées à leurs besoins et d'améliorer les performances dans leur propre domaine d'activité. Pour le renouvellement de sa gamme de 16 logiciels destinés à une clientèle internationale, CS a sélectionné une architecture de type SOA (*Service Oriented Architecture*) sur la plate-forme de développement Microsoft .NET.

Début 2006, dans la lignée de l'adoption de Visual Studio 2005 en tant que plateforme de développement, CS a décidé de réaliser une évaluation objective de **Visual Studio 2005 Team System** de Microsoft. L'enjeu principal de cette évaluation consistait à établir — dans un premier temps — si les fonctions dédiées à la **gestion de configuration** de VSTS répondaient aux besoins et aux contraintes de l'environnement de développement de la division CS, et plus généralement si les outils étaient à même de répondre aux différentes problématiques de la conduite de projet chez Schneider Electric :

- modèle d'architecture logicielle ;
- équipes réparties sur des sites distants ;
- applications multilingues ;
- prise en compte des normes applicables ;
- suivi des versions.

Cette étude a été menée en quatre phases :

- phase initiale : tests sur une machine virtuelle (VPC), effectués au printemps 2006 ;
- phase 1 : simulations LAN (été 2006).
- phase 2 : simulations WAN (été 2006).
- phase 3 : capacités en mode client/serveur, accès distants (oct./nov. 2006).

La phase 3 concernait tout particulièrement les aspects liés à l'installation de Visual Studio Team System (poste client et serveur), ainsi que les fonctions d'accessibilité à distance (avec ou sans proxy) de Team

Foundation Server dans un contexte fonctionnel et technique similaire à l'architecture cible. En cela, l'étude est représentative des besoins d'un industriel dont les équipes informatiques sont réparties sur plusieurs sites y compris à l'étranger.

Tekigo a été sollicité en tant qu'expert technologique .Net et centre de compétence Team System pour accompagner cette démarche d'évaluation. Tekigo est intervenue dans les phases « initiale » et 3. L'étude a été pilotée par Luc FILLET, *Configuration Manager* au sein de la division CS ; il a notamment établi les projets sur lesquels scénariser les tests, notamment pour tenir compte des différentes volumétries de fichiers habituellement traitées.

Méthodologie

Concernant l'infrastructure mise en œuvre, sur le site France ont été installés :

- 1 Team Foundation Server – serveur Dell Pentium IV 2,8 Ghz – RAM 2 Go
- 1 poste client – Dell Optiplex Gx270SF – Pentium IV 2,8 Ghz – RAM 1 Go.

Sur le site Inde :

- 1 poste client – configuration identique au poste client France.
- 1 serveur proxy – serveur de type Pentium IV – RAM 1 Go

Les postes clients ont été équipés de Windows XP Pro SP2 et de Visual Studio 2005 Team Suite. Les serveurs ont été équipés de Windows 2003 Standard.

Le réseau sur le site France est à 10 Mbit/s, le réseau Inde est à 100 Mbit/s. Entre les 2 sites, le réseau WAN s'appuie sur une liaison British Telecom de 2 Mbit/s.

Concernant l'**installation de Team System**, celui-ci a été installé en single-server, mode workgroup.

Concernant l'**environnement fonctionnel**, plusieurs projets TS ont été créés et alimentés sur la base d'éléments significatifs sur le plan de la taille et du nombre de fichiers (et de dossiers) des projets habituellement traités par la division CS.

Tests effectués

Les tests effectués ont consisté à mesurer les temps de réponse perçus par l'utilisateur final lors des différentes sollicitations réalisées dans Visual Studio. Ces tests ont été traités en trois étapes :

- tests locaux (échanges entre le TFS et le poste client France) ;
- tests distants sans proxy (échange entre le TFS et le poste client Inde) ;
- tests distants avec proxy (idem précédent en insérant le proxy). Les mesures ont été effectuées une première fois (chargement du cache) puis une deuxième (chargement à partir du cache).

Afin de compléter le dispositif de mesures, les compteurs de performance système et spécifiques de Team System ont été activés et enregistrés au cours des différents tests. Enfin, les tests ont été effectués en dehors des plages habituelles de travail afin de disposer de mesures d'impact sur le réseau WAN.

Résultats et analyse

① Analyse des compteurs de performance

Cette analyse a mis en lumière une faible sollicitation des ressources du serveur (CPU, mémoire) pour les activités les plus courantes telles que la récupération individuelle de fichiers. En revanche, la création d'un projet TS ou le chargement massif de fichiers dans le Source Control entraînent des consommations de temps CPU allant de 20 à 50 %. Il en va de même pour la consommation mémoire¹ qui ne fluctue que ponctuellement au gré de l'invocation de certains services. Les compteurs Team System — dans le cadre de nos tests — se sont révélés finalement peu pertinents, ne s'agissant pas de tests de montée en charges.

② Analyse de l'impact sur le réseau WAN

L'enseignement principal des mesures effectuées par l'entité supervisant le réseau global de Schneider Electric a été de noter, le jour des tests, un temps de latence moyen

¹ Team Foundation Server utilise environ 1 Go de mémoire vive pour faire fonctionner l'ensemble de ses composantes.

de 200 ms, avec un pic à 310 ms, donc assez éloigné des 350 ms maximum requis par Team System (voir configuration requise à l'adresse [http://msdn2.microsoft.com/fr-fr/library/ms253078\(VS.80\).aspx](http://msdn2.microsoft.com/fr-fr/library/ms253078(VS.80).aspx)).

③ Mesures relevées au cours des tests

Remarques pour la lecture des tableaux : les temps de réponse marqués à « 0 s » correspondent à une réponse perçue comme « instantanée ». Les cases grisées indiquent qu'aucune mesure n'a été effectuée dans ce contexte.

Les tests « locaux » — client France — ont servi de valeurs étalons par rapport aux autres mesures.

On peut également préciser que le mode « Workgroup » de Team System présentent intrinsèquement des performances dégradées par rapport au mode « Domaine ».

Au vu des résultats du **tableau 1**, on peut d'ores et déjà noter que la présence du proxy permet de retrouver des performances équivalentes à celles d'un accès local à partir du moment où les fichiers sont présents dans le cache. On mesure ici tout l'intérêt du dispositif en comparant les résultats entre la colonne « Client Inde / Sans proxy » et celle « Client Inde cache rempli / Avec proxy ». Certes le premier accès est pénalisé du fait d'un pré chargement dans le cache (voir schéma d'architecture en p. 3) mais les suivants y trouvent leur compte.

Autre enseignement, la récupération d'un gros fichier prend plus de temps que, à taille équivalente, *n* petits fichiers. L'idée d'utiliser le Source Control pour stocker des fichiers volumineux n'apparaît donc pas comme une solution pertinente notamment si ces fichiers doivent être lus à partir de sites distants.

Les résultats du **tableau 2** sont très satisfaisants dans le contexte architectural de Team System, sachant également que des opérations de type *check-in* sur un grand nombre de fichiers simultanément ne se produisent pas fréquemment. Là encore on constate que la taille individuelle des fichiers influence fortement l'*upload* durant la phase d'archivage.

Récupération de fichiers via le Source Control « Get latest version »			Temps mesurés (en s)			
			Sans proxy		Avec proxy	
Éléments chargés	Taille (en Mo)	Nb éléments	Client France	Client Inde	Client Inde cache vide	Client Inde cache rempli
File1.cs, 5 ko	–	1	0	3	3	3
File2.cs, 25 ko	–	1	0	3	3	3
File3.dll	0,1	1	0	3	4	3
File4.dll	0,5	1	1	6	6	3
File5.dll	100	1	235	780	950	16
Projet R1a	14	210	12	37	54	9
Projet R1b	18	260	15	62	77	10
Projet R2a	19	300	11	50	114	10
Projet R2b	19	240	18	58	68	8
Projet R2c	50	170	28	129	118	13
Projet R3	86	860	76	300	330	28
Projet D	275	5000	150	867		
Projet M1	200	1300	94	615	489	48
Projet M2	77	275	44	276	199	15

Tableau 1. – Mesures du temps de récupération de fichiers

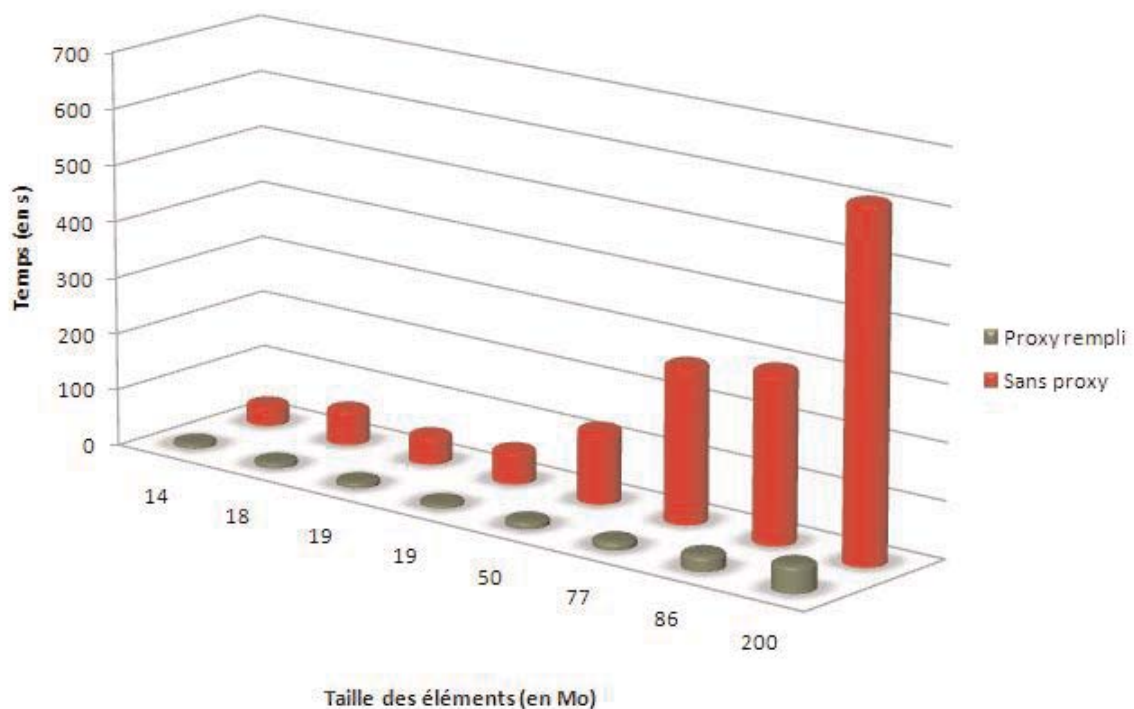


Fig. 3. – Représentation de mesures effectuées sur le site Inde

Check-out Check-in			Temps mesurés (en s)	
	Taille (en Mo)	Nb éléments	Client France	Client Inde cache rempli
Check-out (tous)			0	0
Check-in				
File1.cs, 5 ko	–	1	0	1
File2.cs, 25 ko	–	1	0	1
File3.dll	0,1	1	0	2
File4.dll	0,5	1	1	6
File5.dll	100	1		636
Projet R1	18	260		58
Projet M1	97	800		237

Tableau 2. – Mesures du temps de réponse pour les check-out et check-in

Divers	Client France	Client Inde cache rempli
Création d'un Team Project	1 mn 45 s	14 mn 42 s
Mise en configuration d'un projet Inde		
Projet R1		2 s
Projet M1		11 s
Pose d'un label	0 s	0 s
Accès au portail du projet (Sharepoint)		10 s
Liste des éléments extraits (<i>check-out</i>)		0 s
Comparaison de 2 versions du fichier file2.cs		

Tableau 3. – Mesures du temps de réponse pour différentes opérations

Toutes les actions mesurées dans le **tableau 3** n'étaient bien sûr influencées en rien par la présence du proxy, le seul enseignement important à en tirer est donc que la création d'un projet TS doit être effectuée sur le réseau local du Team Foundation Server si l'on attache de l'importance au temps nécessaire pour cette opération.

Conclusion

Team System est une plateforme client/serveur reposant sur des services web permet-

tant d'effectuer, tout au long du cycle de vie d'un projet, différentes actions et notamment la gestion de configuration. Par cette architecture, tout client distant est capable de gérer cette activité de *source control*. Team System propose cependant une couche logicielle de type proxy destiné à optimiser la récupération des fichiers sources.

Au vu des résultats obtenus dans cette étude, les vertus du Team Foundation Proxy apparaissent comme une évidence. Le gain de performance est indéniable dès lors que le cache de fichiers est opérationnel.

Témoignage client

Par Luc FILLET, *Configuration Manager* à la division CS de Schneider Electric.



Pour le renouvellement complet de sa gamme de logiciels débuté il y a 2 ans, la division *Customer Software* (CS) a choisi une architecture innovante, connue aujourd'hui sous le sigle SOA, avec un socle de composants et les applications proprement dites. Pour leur développement, CS fait appel à des ressources réparties géographiquement sur plusieurs sites.

La colonne vertébrale de cette fabrique logicielle est le système de gestion de configuration logicielle (GCL). Historiquement, en fonction des besoins et des contraintes des projets, CS utilisait l'un ou l'autre de ces deux outils : Visual SourceSafe de Microsoft et Clearcase Multisite de Rational/IBM.

La mise en place de la nouvelle architecture développée sur la plate-forme Microsoft .NET nous a obligé à redéfinir notre GCL et à rechercher la solution la plus performante, avec pour enjeux, l'amélioration de notre productivité et la réduction du temps de mise sur le marché des applications.

Aussi, nos principaux critères de sélection de cet outil étaient :

- la prise en charge de tout le contenu de la fabrique logicielle : de la conception des composants aux versions installables des applications, et d'éléments volumineux soit en taille, soit en nombre,
- son intégration dans l'environnement des développeurs, des architectes...
- la possibilité de faire du développement collaboratif sur plusieurs sites de production,
- sa capacité de permettre un suivi précis du travail en cours.

L'étude réalisée avait donc deux objectifs : comparer les outils Clearcase et Visual Studio 2005 Team System, et mesurer les qualités de GCL de VSTS sur notre chaîne de production logicielle.

À l'issue de cette évaluation réalisée avec le concours de Tekigo, CS a retenu la solution Microsoft : tous les aspects de GCL identifiés dans notre fabrique logicielle étaient traités. Certes, la première version de VSTS ne fournit pas le meilleur outil pour chaque domaine couvert, comme celui de la GCL. C'est l'ensemble des possibilités de VSTS garantissant une cohérence et une fluidité de tous les éléments des projets, ainsi que sa capacité d'ouverture et l'engagement de son éditeur qui nous ont convaincu.

Les entreprises disposant d'équipes de développement géographiquement réparties et souhaitant déployer Team System ont donc tout intérêt à installer un serveur proxy sur leurs différents sites de production. Certes cette stratégie implique l'acquisition d'une licence TFS pour chaque serveur proxy mais le retour sur investissement est pratiquement immédiat compte tenu du gain de performance, un argument qui prend toute sa dimension dans une activité quotidienne pratiquée par les équipes de développement.

Remerciements

Nous tenons à remercier tout d'abord l'équipe Schneider Electric pour sa disponibilité, en particulier Luc Fillet et Alain Fuss, ainsi que l'équipe Microsoft France en charge de la promotion de Team System.



Bibliographie

- Bénard, J.-L., & Mérand, F. 2005. — Approches pragmatiques pour industrialiser le développement d'applications. BrainSonic editions, Paris, 88 p.
- David, J.-L., Gousset, M., & Gunvaldson, E. 2006. — Professional Team Foundation Server. Wrox, Indianapolis, 486 p.
- Levinson, J., & Nelson, D. 2006. — Pro Visual Studio 2005 Team System. Apress, Berkeley, 541 p.
- Microsoft. 2006. — Team Foundation Installation. Disponible sur MSDN.
- Microsoft. 2006. — Team Foundation Administrator's Guide. Disponible sur MSDN.

Qui sommes nous

TEKIGO est une société de conseil et d'expertise, spécialisée dans l'accompagnement technologique des entreprises. La société s'appuie sur une équipe de consultants possédant une forte expertise méthodologique et technique mais aussi une expérience de la conduite de projets, assurant ainsi la prise en compte de l'ensemble des phases d'un projet informatique : assistance à maîtrise d'ouvrage et maîtrise d'œuvre, conception, architecture, monitorat, etc. TEKIGO intervient également dans le cadre d'une externalisation de la R&D. Pour plus d'informations, consulter notre site : www.tekigo.com

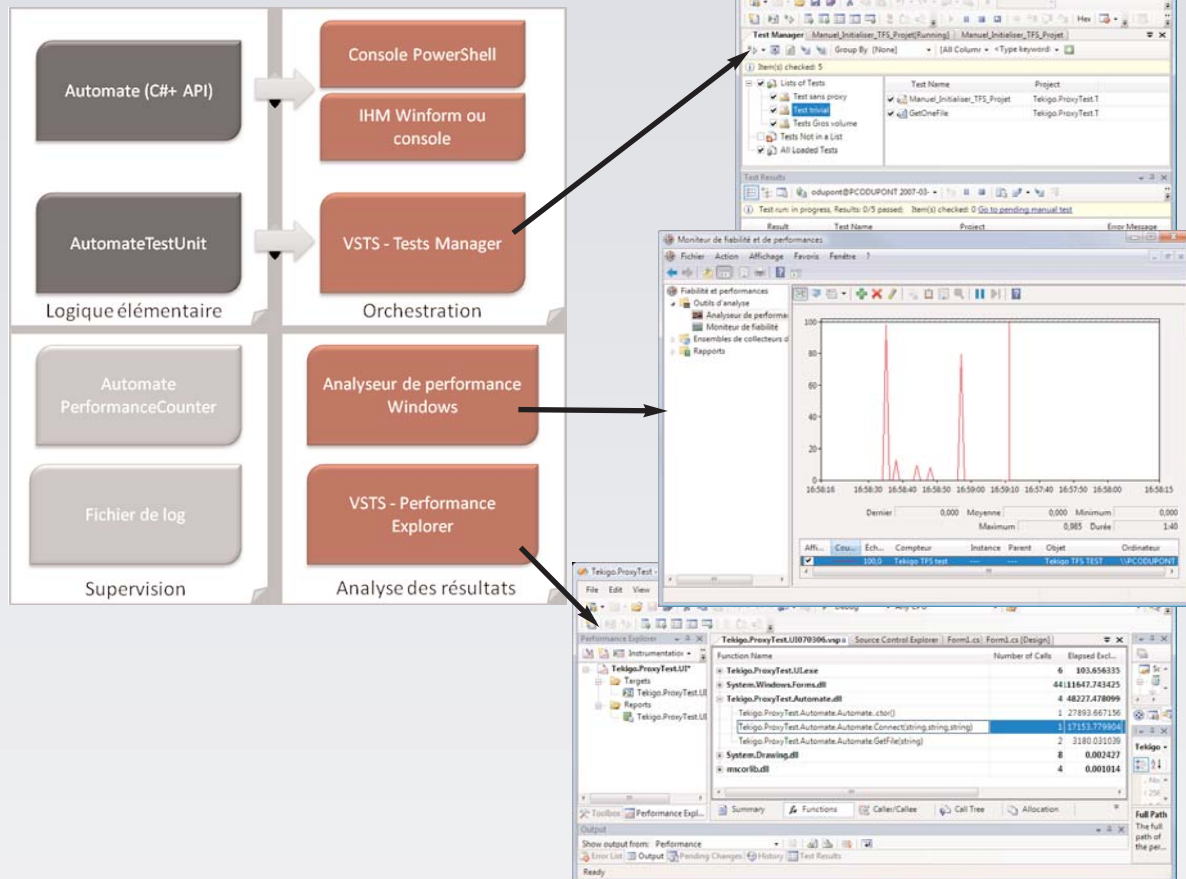


Centre de compétences Team System

Les centres de compétences Team System forment un réseau de partenaires de Microsoft capables d'accompagner les entreprises pour évaluer ou déployer la solution Visual Studio Team System et sa plateforme Team Foundation Server. Fortement impliqué dans la réussite des projets, dans le respect des contraintes spécifiques à chacun de ses clients (ressources, technicité, coûts, délais, qualité...), TEKIGO s'inscrit naturellement dans une démarche à la fois qualitative et adaptative autour des problématiques de *software factory* (industrialisation et cycle de développement), de méthodologie « projet » et de qualité logicielle.

Automatisation de tests

Profitions de ce livre blanc pour faire un petit focus sur les possibilités de VSTS en termes de test. En effet, on peut se demander comment effectuer des mesures les plus proches possible de la réalité et cela sans y passer trop de temps. Afin de collecter les différentes métriques (délais, comptage...) nécessaires à l'analyse de l'utilisation du proxy, il est possible de monter rapidement un automate grâce à VS2005 et les API de Team Foundation Server. Selon la nature et la volumétrie des scénarios de tests à effectuer, ainsi que le détail de reporting souhaité, on peut choisir différentes solutions techniques aisément implémentables tels que :



Dans la démarche on commence par écrire un automate (helper) implémentant les opérations élémentaires de manipulation du module *Source Control* de TFS (connexion, get, check-in, création de fichiers...) à l'aide des API de **Microsoft.TeamFoundation.VersionControl.Client**. On peut ainsi orchestrer un certain nombre de scénarios via un script PowerShell ou via une IHM Winform. Afin d'être plus flexible et induire une stratégie de plan de test plus ambitieuse et à faible coût, on peut s'appuyer sur le Test Manager de VSTS pour organiser des scénarios de tests unitaire complets. Une fois l'orchestration établie, il faut générer les métriques à l'exécution de l'automate. Pour cela, on s'appuie sur les API des capteurs de performances de Windows qui nous permettent de suivre en « temps réel » le comportement de l'automate, et ce de manière graphique. Ces mesures concernent l'usage effectif du TFSSC et se limitent à ce que l'on veut prouver. Pour avoir des informations complémentaires sur des opérations ne concernant pas directement les scénarios de tests, on pourra utiliser le Performance Explorer de VSTS pour isoler techniquement des délais d'exécution telle que la connexion, sans développement supplémentaire.

Plus d'informations : www.tekigo.com/TeamSystem.aspx