Evolution des composants sous .NET

NB : ce fascicule fait partie d'un travail de diplôme sur le sujet « Technologie ActiveX & Visual Basic 6 ».
Les autres fascicules peuvent être demandés à fcomte@caramail.com.

Critiques, remarques, questions ? fcomte@caramail.com

1 - Microsoft.NET et Visual Basic.NET

Qui n'a pas entendu parler de la suite .NET ? Microsoft multiplie les salons sur le sujet. Mais, à moins d'avoir participé à l'un d'entre eux, il n'est pas possible d'avoir une vision claire de ce que propose .NET. Le chapitre suivant tente néanmoins de présenter brièvement la suite .NET, ainsi que les perspectives qui se présentent pour l'évolution de Visual Basic, et de COM.

1.1 La suite .NET en général…

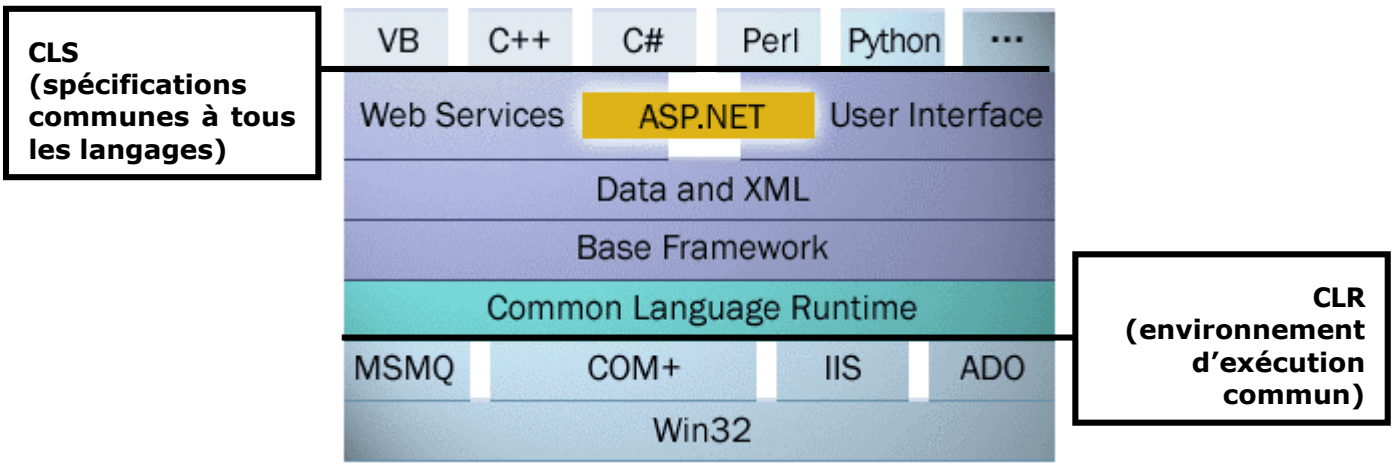
Pour comprendre le positionnement de Microsoft.NET, comparons avec un environnement « classique » :
Sur la plate-forme Windows, les applications qui respectent la norme d'interface COM sont compatibles et peuvent (merveille !) communiquer. Pour développer ce type d'application, Microsoft fournit :

- une boîte d'outils : l'api Win32,
- une environnement de développement : Visual Studio,
- un outil de développement rapide : Visual Basic.

Transposé au web : il s'agit maintenant de faire communiquer des entreprises sur la plate-forme Internet, par le biais de Web Services basés sur une interface XML.
Dans cette optique, Microsoft.NET propose :

- une boîte à outils : le Framework.net
- un environnement de développement : Visual Studio.NET,
- un outil de développement rapide : Visual Basic.NET.

Le framework (cadre de travail qui définit les standards de développement) s'appuie sur un système d'exploitation, mais il n'est pas pour autant dédié à cet OS. Son intérêt est de permettre aux développeurs de ne plus développer pour le système d'exploitation (lié à un matériel, à un constructeur, à une interface, etc.) mais pour le framework à proprement parler.



Ce dernier comprend une couche fondamentale dénommée CLR (Common Language Runtime) sur laquelle se greffe tout un ensemble de bibliothèques riches et évoluées, et sur lesquelles s'appuient à leur tour les différents langages de programmation (compilateurs C++, VB, C# et Jscript intégrés en standard) à travers un mécanisme unificateur dénommé CLS (Common Language Specification). Le CLR assure la compilation en code natif des programmes (les logiciels .NET sont compilés), la gestion mémoire, la gestion des erreurs, et les contrôles de sécurité. Il fournit l'accès à l'OS et aux ressources système.

Contrairement aux « purs » burger de McDo, « .NET » est plus qu'une mode de langage qui s'appliquerait à des produits connus. Les produits .NET sont en effet de conception radicalement différente de leurs homologues classiques.
Ainsi, bien que la compatibilité avec COM soit conservée, il est clair que cette technologie sera – à terme – abandonnée. Microsoft mise sur la migration des applications COM vers .NET.

1.2 … Visual Basic.NET en particulier

Attachons-nous plus précisément à Visual Basic.NET, la version RAD (Rapid Application Development) de la suite.
Que permet Visual Basic.NET ? Visual Basic permettait le développement rapide de fenêtres Windows et de composants COM. De même, quelques clics dans Visual Basic.NET permettent de créer un formulaire html complexe, le code associé, et de publier un Web-Service basé sur une interface XML.

Le langage quant à lui possède 2 évolutions très attendues : il devient purement objet, et il gère les exceptions. En outre, basé sur le C#, VB.NET est à priori portable sous d'autres environnements comme Linux.

1.3 Qu'en est-il de la migration de VB à VB.NET ?

Il est trop tôt pour s'avancer sur la pertinence de cette migration, essentiellement liée aux évolutions de la plate-forme Windows 2000.
Mais pour ce qui est de la migration elle-même, il est impératif de garder en vue qu'il s'agit ici de 2 produits différents.

Si Microsoft fournit un outil de migration qui permettra de migrer aisément les applications purement Visual Basic, en revanche il est conseillé d'observer la plus grande prudence sur la migration d'applications plus complexes. Il faut en effet s'attendre à rencontrer des problèmes sur tout ce qui touche à la gestion de la mémoire, ou bien au cycle de vie des objets (pour ne citer qu'un exemple, les types vb.net ne sont plus contigus en mémoire… ce qui élimine de la migration de nombreux appels à des fonctions C ou Win32).

2 - Positionnement de COM

Quand on a compris le rôle du framework .NET dans tout ce qui relève de la « plomberie » nécessaire à l'interopérabilité des langages, pour assurer la compatibilité binaire, la gestion de la mémoire et des cycles de vie des objets, on peut se demander où se situe COM. C'est en fait assez simple : COM est tout simplement obsolète ! La construction de composants ayant les mêmes propriétés qu'un composant COM est simplifiée à l'extrême quel que soit le langage (plus de IUnknown ni de AddRef…). La base installée des composants COM est par contre préservée, grâce à une interopérabilité dans les deux sens. Une application .NET peut appeler les services d'un composant COM comme s'il s'agissait d'un composant .NET vis-à-vis du client. Si COM est obsolète, en revanche le cas de COM+ et de ses services middleware (transactions, pooling, messages asynchrones,…) est différent : en effet, on ne dispose pas d'équivalent dans .NET, du moins dans sa première version.

2.1 L'enfer des DLL serait terminé ?

Hélas pour eux, le travail des développeurs ne s'arrête pas à l'écriture du code mais intègre le déploiement sur les postes clients. Or le déploiement sous Windows est devenu un problème de plus en plus complexe dont Microsoft n'a pris conscience que récemment. Lors de l'installation d'un programme, il y a de fortes chances pour que d'autres programmes déjà installés ne fonctionnent plus, les DLL nécessaires ayant été écrasées par d'autres, parfois plus récentes, mais non compatibles. La gestion des versions des composants COM est encore plus problématique, et a été rendue presque opaque par la base de registres, qui n'a pas le mérite de la simplicité.

Avec .NET, Microsoft se lance le défi de remédier une bonne fois pour toutes à « l'enfer des DLL ». Au cœur de chaque programme – et donc du bytecode MSIL – sont incorporées des méta-données appelées manifestes, décrivant les dépendances de chaque module, ainsi que les numéros de version. Les modules sont ensuite réunis logiquement au sein d'une assemblée. Ces assemblées peuvent être partagées ou privées, décision prise de manière explicite par le développeur. Ces informations permettent l'installation d'une application par un simple copier/coller de fichiers. Plusieurs versions d'assemblées (portant le même nom de fichier) peuvent ainsi coexister au sein d'une machine. Au-delà du simple XCOPY des fichiers EXE et DLL, il reste possible d'utiliser des packages .MSI, introduits avec Windows Installer depuis Windows 2000. L'installation d'une application n'est alors plus faite par un programme tiers mais par un service de l'OS. Ce dernier dispose d'un système de téléchargement incrémental intelligent, où les modules seront chargés uniquement selon les besoins, en respectant l'état de la machine et des programmes installés.

2.2 Positionnement d'ActiveX

Faites l'expérience : allez sur la msdn (http://msdn.microsoft.com) et effectuez une recherche sur .NET et ActiveX. Les trop rares pages que vous trouverez ne vous mèneront pas loin… Est-ce parce que la technologie ActiveX n'a pas eu le succès escompté et que Microsoft tend à faire disparaître cette norme pour une nouvelle (et donc sous un nouveau nom…) ? Ou parce que la documentation à ce sujet n'est pas encore disponible ? Mystère…
Après avoir posé la question de la place d'ActiveX dans l'architecture .NET dans un forum de discussion Microsoft, voici ce qu'on m'a répondu :

« Les composants sont *exposés* à la mode COM mais ne sont pas des composants COM. J'ai plutôt l'impression que l'interopérabilité avec COM n'est là que comme étape transitoire vers un environnement où tous les composants seront NETifiés... »

et

« Il est beaucoup plus simple d'utiliser et même écrire des composants COM avec .NET. En fait il existe deux utilitaires en ligne de commande, un pour importer un ActiveX dans le monde .NET (tlbimp.exe) et un autre pour exporter un composant .NET en ActiveX (tlbexp.exe). L'exportation est même très bien faite et expose des interfaces que peux de développeurs proposent habituellement. Le pont .NET vers ActiveX a été exploité aux maximum des possibilités. Dans Visual Studio.NET l'incorporation est ultra simple, il suffit de choisir le composant à inclure dans le projet depuis une liste. »

De plus, lors d'une visite sur la MSDN, je suis tombé sur un article traitant des différences entre les contrôles ActiveX de VB6 et les contrôles .NET, dans lequel l'auteur faisait entendre qu'à terme la technologie ActiveX serait abandonnée au profit de .NET.

Mais Microsoft distille tout de même quelques informations au sujet d'ActiveX par rapport à .NET. Morceaux choisis[1].

« Avec C#, tout objet est automatiquement un objet COM. Les développeurs n'ont plus à implémenter explicitement IUnknown et autres interfaces COM. A la place, ces fonctionnalités sont construites automatiquement. De plus, les programmes C# peuvent utiliser les objets COM existants, sans se soucier du langage dans lequel ils ont été développés[2] ».

« Un des rôles majeurs du framework .NET est de simplifier le développement d'objets COM. Une des choses les plus compliquée sous COM étant de saisir complètement son infrastructure, .NET automatise toutes les manipulations sophistiquées comme le compteur de références, la description des interfaces et l'inscription (registration). »

« Il est important de noter que cela ne signifie pas que les composants du framework .NET ne sont pas des composants COM. En fait, un développeur COM travaillant sous Visual Studio 6 pourrait appeler un composant .NET, et il apparaîtrait à ses yeux comme un composant COM. Inversement, un composant COM peut être appelé dans Visual Studio.NET et être vu comme un composant .NET. »

« Notons une mise en garde à cette relation : les développeurs COM doivent faire plusieurs choses manuellement que le framework .NET effectue automatiquement. Par exemple, la sécurité d'un composant COM doit être écrite à la main[3], sa mémoire ne peut pas être gérée automatiquement, et, pour installer un composant COM, des entrées doivent être écrites dans la base de registres. Tout ceci n'a plus court avec les composants .NET, car les composants s'auto décrivent et peuvent être installés sans passer par une inscription dans la base de registres. »

[1] Tirés de http://msdn.microsoft.com/vstudio/nextgen/technology/csharpintro.asp et http://www.microsoft.com/net/developer/framework_com.asp.
[2] C'était déjà le cas avant .NET… (Note de l'auteur)
[3] Avec l'interface `IObjectSafety`, cf. chapitre "ActiveX et la sécurité"

2.3 En guise de conclusion…

.NET (et donc VB.NET) paraît être un produit complètement révolutionnaire, car définitivement totalement orienté réseau (Internet). Avec ces grands changements, c'est une multitude de nouveaux termes et nouveaux concepts que les développeurs devront apprendre : un nouveau langage, le C#, qui d'après Microsoft va peu à peu remplacer C++, le XML, qui semble incontournable, les services Web, qui (toujours d'après la même firme) permettront tout et même plus sur le net, etc.

Cependant, on peut tout de même se demander si la philosophie résolument tournée vers une interopérabilité totale et une ouverture à toutes sortes de supports et périphériques fonctionnera. Microsoft a dans le passé réussi à prouver que ces idées restaient généralement couchées sur le papier, normalisées à l'extrême, et – en pratique – difficilement réalisables. La vision d'un univers qui voit migrer tous les produits, services et données vers Internet, où ces dernières vivraient indépendamment du support qui les exploite, et où le Web ne serait plus seulement le lien entre un client et un serveur d'informations mais un espace d'échange de données inter-applications, inter-périphériques, inter-métiers, etc., est certes alléchante, mais… devrons-nous acheter des PC Microsoft, des téléphones Microsoft, des PDA Microsoft, bref DU Microsoft, pour faire fonctionner le tout ? La réponse dans quelques années…

Bibliographie critique et liens Internet

- Innovatica News, N°12 et N°13, journal de l'entreprise Innovatica, articles de Mr. Grégoire Nogier, qui a écrit deux articles sur la compatibilité COM/DCOM-Java et sur Microsoft.NET et VB.NET.

- PC Expert, n°112, septembre 2001. Magazine dont un des articles « le piège .NET » m'a aidé à faire le tour du problème, tout comme le magazine suivant.

- Programmez !, n°35, septembre 2001 (article « .NET, c'est parti ! Comment Microsoft va tout changer »).

- http://msdn.microsoft.com, le lien INCONTOURNABLE pour développer sous plate-forme Windows.