

La plate-forme Microsoft .NET

En Juin 2000, lors de sa conférence annuelle la société Microsoft annonçait sa nouvelle stratégie d'entreprise. Une plate-forme technologique intégrant un environnement de développement unifié, basés sur des standards tels que XML et les Web Services. La plate-forme .NET était née. Elle devra succéder à DNA (Distributed Internet Applications), l'environnement de composants applicatifs distribués que Microsoft avait mis en avant et n'ayant pas rencontré le succès attendu. Après une année 2002 - 2003 consacrée aux tests et aux projets pilotes, le bilan de .Net vu par Microsoft est plutôt satisfaisant. Mais qu'en est-il vraiment...

Qu'est ce que la plate-forme .NET ?

Contrairement à beaucoup d'idées reçues, la plate-forme .NET n'est pas uniquement un environnement de développement **RAD** (Rapid Application Development) et multi-langage; .NET est la nouvelle stratégie de Microsoft, une vision de sa prochaine génération des systèmes d'exploitation, des logiciels spécialisés ainsi que des outils de développement pour les systèmes d'information d'entreprise.

La plate-forme Microsoft .NET repose principalement sur quatre piliers :

- ❑ Le **Framework .NET**
- ❑ L'environnement de développement **Visual Studio .NET**
- ❑ **.NET Server** : La future version « Server » de Microsoft (qui changera peut-être de nom).
- ❑ **.NET Enterprise Servers**. Les composants serveurs (Application Center, SQL Server, Biztalk Server, Content Management Server, Commerce Server, etc.) qui se grefferont à .NET Server.

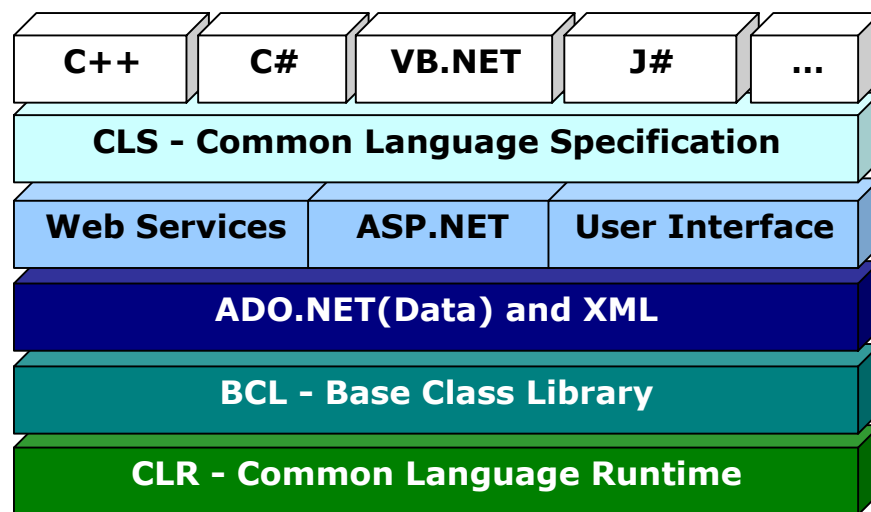
L'objectif que s'est fixé Microsoft est pour le moins ambitieux, tant du point de vue technique que stratégique. Il est important de noter que la nouvelle plate-forme .NET n'est pas une évolution de sa technologie actuelle DNA mais bien une nouvelle orientation stratégique, qui risque de remettre en cause bien des acquis et qui va apporter son lot de nouveautés au prix d'une compatibilité avec l'existant, pas toujours garantie. Pour s'en rendre compte, analysons les quelques éléments suivants :

- ❑ Le serveur Web IIS de Microsoft, abandonne son ancien modèle multi-thread certes performant, mais fragile au profit d'un modèle « multiprocesseur », ce qui n'est pas sans rappeler le nouveau modèle du serveur Web d'Apache.
- ❑ La technologie **ASP** (Active Server Pages) basée sur des scripts interprétés cède la place aux pages ASP.NET dont le code est compilé dès la première invocation, à la façon des pages **JSP** (Java Server Pages).
- ❑ Les API Win32 telles que **ATL** (Active Template Library) et les **MFC** (Microsoft Foundation Classes) sont remplacées par un ensemble plus cohérent de classes de base du Framework .NET .
- ❑ Le langage VB.NET n'assure plus la compatibilité descendante depuis Visual Basic 6. VB.NET est dorénavant 100% orienté objet de façon à remplir le contrat de services de la **CLS** (Common Language Specification) .NET .
- ❑ La venue d'un nouveau langage nommé **C#** (prononcez « c sharp ») voit le jour. Il s'agit d'un langage objet moderne, synthèse entre C++ et Java. Le concepteur de C# n'est autre que **Anders Hejlsberg**, qui fût l'architecte de plusieurs langages et outils chez Borland, dont le célèbre Delphi.

Comme nous venons de le voir plus haut, les changements sont profonds, surtout dans le cas d'un existant basé sur les MFC, **VB** (Visual Basic) 6 et versions antérieures, sans oublier les applicatifs COM/DCOM/COM+ qui ont fait pendant longtemps la clé de voûte de l'architecture traditionnelle Microsoft et qui représentent la majorité des développements Windows existants. Il est toujours possible néanmoins d'utiliser des objets COM/DCOM/COM+1.0 ou les MFC avec .NET, mais il est important de savoir que ces derniers ne font plus partie intégrante du Framework Microsoft .NET .

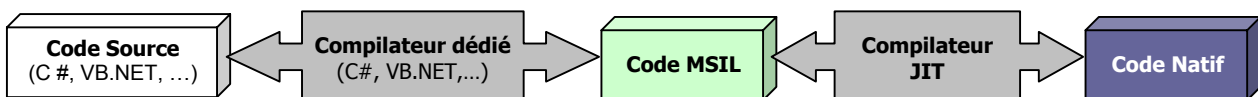
Le Framework .NET

L'architecture du Framework .NET est composée de plusieurs couches, comme l'illustre le diagramme suivant :



La machine virtuelle .NET

La **CLR** (Common Language Runtime), qui se place juste au dessus du système d'exploitation, est l'un des piliers centraux du Framework. A la manière de la plate-forme JAVA de Sun Microsystems, Microsoft a choisi d'intégrer à la plate-forme .NET une machine virtuelle nommée CLR ainsi qu'un code intermédiaire, le pendant du Byte Code JAVA, le **MSIL** (MicroSoft Intermediate Language) qui une fois généré est compilé à la volée en code natif, grâce à un compilateur JIT (Just In Time).

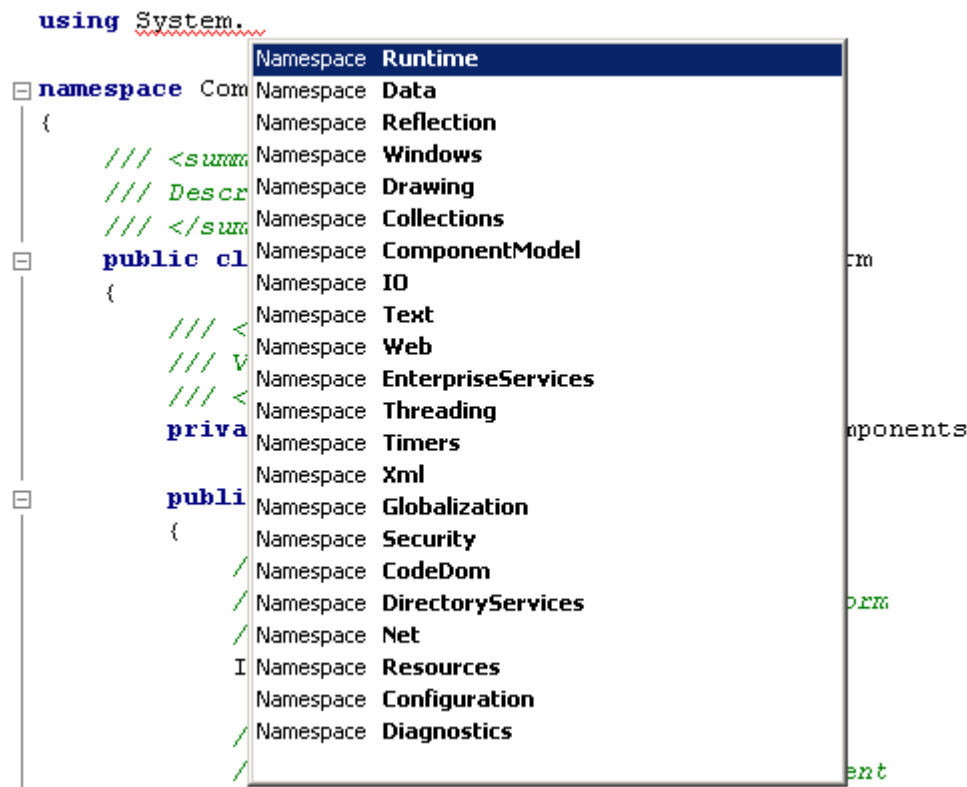


La CLR intègre aussi un mécanisme de ramasse-miettes (garbage collector) qui gère et contrôle l'espace mémoire ainsi que le cycle de vie des objets référencés par une application .NET. Tout au long de l'exécution d'une application .NET, la sécurité applicative est aussi prise en charge par la CLR. Un développeur a donc le choix de donner des droits restrictifs à une application .NET, en lui interdisant par exemple, l'accès aux services réseau.

La « Base Class Library »

Le Framework .NET ainsi que sa **BCL** (Base Class Library) disposent d'une architecture complètement objet contrairement aux API Windows, qui ne sont en général qu'une liste de fonctions système enrichies. Les classes du Framework sont ordonnées hiérarchiquement à la manière des packages JAVA. Ces packages sont nommés **namespace** en .NET.

Les namespaces sont utilisables et restent les mêmes pour tous les langages du Framework .NET que cela soit C#, VB.NET ou encore C++. L'ensemble des classes de base du Framework est hiérarchisé et regroupé par domaines à la manière du **JDK** (Java Development Kit). Le namespace racine de la BCL est « System ». Il regroupe tous les types de base du Framework ; string, int, bool, etc. Plusieurs classes s'y trouvent aussi comme la classe « Console » bien utile dans la gestion de l'affichage et à la lecture d'informations en mode console, par exemple. Ensuite viennent les autres namespaces spécialisés, comme le montre la capture d'écran suivante, qui contiennent eux mêmes des classes qui leur sont propres :



On peut citer le namespace « System.IO » regroupant les classes utiles à la gestion des entrées/sorties, le namespace « System.Net » utile aux services réseau, le namespace « System.Xml » spécifique aux traitements des **XML** (eXtended Markup Language) ou encore la persistance qui est gérée par ADO.NET et qui est regroupée dans les classes du namespace « System.Data ».

Microsoft a mis au point un standard de développement basé sur le Framework .NET. Ce dernier est consultable à l'adresse suivante :

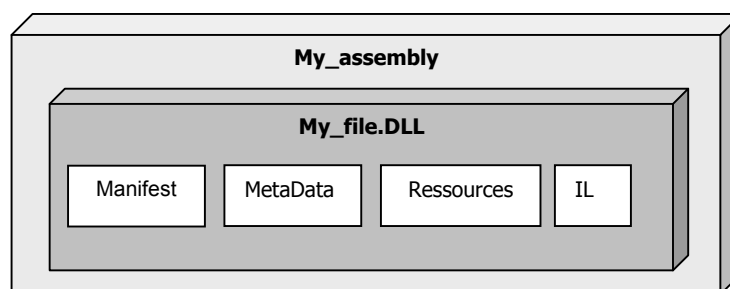
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconnetframeworkdesignguidelines.asp>

Les assemblies .NET

Microsoft a su profiter du passage au Framework .NET pour régler un ensemble de problèmes qui lui sont caractéristiques, comme par exemple celui des **DLL** (Dynamic Link Library).

En effet, que cela soit au niveau de l'installation ou du déploiement, il était impossible d'avoir plusieurs versions d'une DLL chargées en même temps, ce qui posait des problèmes de compatibilité entre applications. Il arrivait souvent qu'au sein d'une même entreprise l'on change de version de DLL et que certaines applications ne marchent tout simplement plus, dû au fait qu'elles n'étaient plus compatibles avec la nouvelle version de la DLL installée.

Avec le Framework .NET lorsque vous concevez une application Windows Form ou Web, un **assembly** est automatiquement généré. Un assembly peut prendre la forme d'un fichier exécutable ou d'une DLL, mais correspond à un conteneur de classes et de fichiers ressources au sein de votre projet qui peut être utilisé par plusieurs autres applications, à la manière d'une archive JAVA (**JAR**). Grâce à ce conteneur physique de classes et de ressources, il devient possible d'avoir plusieurs versions du même assembly chargés en mémoire. Ce dernier peut contenir un fichier manifest, des Metadata, des fichiers de ressources (images, fichiers d'aide, etc.), encapsuler une DLL, ainsi que le code intermédiaire **IL** (Intermediate Language) de l'application.



Un système de cache a aussi été mis au point; son nom le **GAC** pour Global Assembly Cache. Il contient toutes les assemblies du Framework. Il est aussi possible de déployer ses propres assemblies dans le GAC pour des raisons de performance ou de commodité, sachant que la CLR commence par rechercher ces derniers tout d'abord dans le GAC.

Le Framework .NET et le monde JAVA

Après avoir fait un petit tour d'horizon du Framework .NET, on ne peut s'empêcher de faire le parallèle avec JAVA. En effet, beaucoup d'éléments dans l'architecture du Framework .NET sont extrêmement similaires à la technologie JAVA.

Microsoft ne se cache pas d'avoir analysé et puisé de façon pragmatique ses inspirations technologiques dans les domaines qui ont fait le succès du langage JAVA de Sun Microsystems.

On peut constater une ressemblance marquée sur les concepts et aspects suivants:

JAVA	Framework .NET
JDK (Java Development Kit)	SDK .NET (Software Development Kit)
JRE (Java Runtime Environment)	.NET Framework Redistributable Package
JVM (Java Virtual Machine)	CLR .NET (Common Language Runtime)
Le langage intermédiaire Byte Code interprété par la JVM	Le langage intermédiaire MSIL interprété par la CLR .NET
Le langage objet Java	Le C# est très proche de java dans sa syntaxe ainsi que dans plusieurs éléments de son langage. On peut appliquer les mêmes remarques à J#, voir plus.
Compilation des pages web (JSP => servlets)	Compilation des pages web ASP.NET (Contrairement à ASP dont les pages étaient interprétées)
Java RMI (Remote Method Invocation)	Microsoft .NET Remoting

On peut aussi noter un certain nombre d'éléments qui diffèrent de JAVA:

- ❑ La CLR .NET supporte plusieurs langages (C++, C#.NET, J#, VB.NET, Cobol...) grâce au standard CLS (Common Language Specification).
- ❑ Le concept de serveur d'applications **J2EE** (Java 2 Enterprise Edition) devient sous .NET le système d'exploitation Microsoft Windows lui même.
- ❑ Un environnement de développement unifié et multi-langage pour le développement rapide d'applications .NET : Visual Studio .NET.
- ❑ Actuellement, la couche persistante ADO.NET et ObjectSpace sont encore bien loin du monde Java et de ses EJB Entity Bean.
- ❑ Le Framework .NET n'est pas multi plate-forme contrairement à JAVA qui a été porté sur les systèmes d'exploitation suivants: Windows, Linux, MacOS X, Solaris, BSD et plusieurs autres Unix propriétaires.

Les développeurs JAVA et le Framework.NET

Les sept ans d'âge de JAVA lui confèrent une maturité et le support d'une communauté importante et active. JAVA est un langage objet par naissance et non orienté objet comme l'est devenu par exemple VB.NET. Les développeurs Visual Basic et ASP actuels, qui n'ont pas fait l'effort de se confronter à un langage objet ou du moins orienté objet, devront passer un certain temps avant de bien assimiler et bien pratiquer les concepts de la programmation objet et autres « design patterns ». Ces derniers font par contre légion dans la communauté JAVA sans oublier les développeurs C++.

La société Microsoft, bien consciente de la manne et du potentiel des développeurs JAVA, essaie par l'intermédiaire de J#, **JLCA** (Java Language Conversion Assistant) et bien sûr C#, de séduire un maximum de ces développeurs. Devant le nombre de similitudes entre le Framework .NET et JAVA il est évident que les développeurs JAVA avec leurs expériences feraient merveilles sur la plate-forme .NET. Qu'on se le dise, pour pouvoir exploiter au mieux la puissance du Framework .NET Microsoft, la maîtrise de la programmation objet est un avantage indéniable.

Les développeurs JAVA se laisseront-ils tenter ? L'avenir nous le dira ; mais ce qui est sûr, c'est qu'il faut compter dès à présent autant sur le monde .NET que sur le monde JAVA.

Linux, l'Open source et le Framework .NET

Microsoft c'est efforcé de rendre le Framework .NET le plus portable que possible. Une spécification a été déposée dans ce sens par la société de Redmond au près de l'ECMA qui est un organisme de standardisation technologique, dont le siège est localisé à Genève. Des parties de cette spécification sont disponibles à l'adresse suivante: <http://msdn.microsoft.com/net/ecma/>

Mono, .NET et Linux

Le projet Mono (<http://www.go-mono.com>), initiative de la société Ximian qui a été rachetée dernièrement par la société Novell, a pour but de rendre le développement d'applications sous Linux compatible avec le Framework .NET. A fin d'éviter toute confusion, notons bien la différence entre le *Framework .NET*, qui est un sous-ensemble de la plate-forme .NET et la plate-forme elle-même. Le Framework, au vu de ses spécifications, peut être porté en partie sur un autre système d'exploitation. Quand à la *plate-forme .NET*, elle représente la nouvelle stratégie de Microsoft concernant ses systèmes d'exploitation, ses logiciels spécialisés et ses outils de développement. Le projet Mono se limite à porter que les parties les plus importantes du Framework .NET, à fin de rendre son développement compatible sous Linux, en s'appuyant principalement sur les spécifications de Microsoft soumises à l'ECMA.

Actuellement, Mono atteint une couverture fonctionnelle proche de la version 1.1 du Framework .NET avec une CLR, un compilateur C#, la BCL, une bonne partie d'ASP.NET et des Web Services. L'effort mérite d'être souligné ! Toutefois, on est encore un peu loin d'un environnement complet de développement .NET et véritablement productif sous Linux.

Un autre élément important à noter est que le projet Mono ne s'inscrit pas dans un pur modèle **Open Source/GPL** (General Public License). La licence choisie pour le projet Mono est à cheval entre la **LGPL** (Lesser General Public License) et la MIT X11, ce qui risque de faire grincer les dents de l'importante communauté Linux et du logiciel libre.

L'Open Source et .NET

De plus en plus de vrais projets Open Source pour .NET voient le jour, depuis l'entrée en scène de ce dernier. En voici une petite liste :

- ❑ NAnt (<http://nant.sourceforge.net>) : La version du fameux Ant d'Apache (<http://ant.apache.org>) pour .NET
- ❑ NVelocity (<http://nvelocity.sourceforge.net>) : La version .NET du moteur de templates d'Apache Velocity (<http://jakarta.apache.org/velocity/>)
- ❑ Nunit (<http://www.nunit.org>) : Une version xUnit (<http://www.xprogramming.com/software.htm>) de tests unitaires pour .NET
- ❑ Nmaverick (<http://mavnet.sourceforge.net>) : La version .NET du framework Web MVC Maverick (<http://mav.sourceforge.net>)
- ❑ Cassini (<http://www.asp.net/Default.aspx?tabindex=7&tabid=41>) : Un serveur HTTP/ASP.NET conçu en C# pour .NET

Dernièrement Microsoft a annoncé le développement de sa propre version de NAnt, MsBuild une version propriétaire et malheureusement non Open Source. La firme de Redmond préférant systématiquement re-développer plutôt qu'intégrer des outils Open Source existants.

A terme, cette politique risque d'affaiblir les défenseurs de l'Open Source .NET . Microsoft ne serait-il pas en train de se tirer une balle dans le pied ? Il est en effet dommage que la firme de Redmond préfère favoriser le développement d'outils propriétaires d'un côté et prôner d'un autre, la portabilité d'un Framework .NET pour le système d'exploitation Open Source Linux, élaboré par la société Ximian, elle même issue du monde de l'Open Source.

L'interopérabilité et .NET

De nos jours, il existe différents environnements, systèmes et langages conçus pour des domaines bien spécifiques. Il peut arriver d'implémenter différentes parties d'une application dans un certain langage, car plus adéquat, puis de faire interagir ces différentes parties entre elles au moyen d'un autre langage ou d'un environnement applicatif.

Le principe de l'interopérabilité est justement de permettre à plusieurs plates-formes, systèmes ou langages hétérogènes de communiquer entre eux de façon homogène.

Les Web Services, utilisation et limitation

Quand on parle d'interopérabilité, on mentionne souvent l'utilité des « web services ». Ils sont un excellent moyen pour interopérer des systèmes hétérogènes certes, malheureusement ils ne permettent pas aujourd'hui de gérer des opérations telles que les traitements distribués. Il est ainsi extrêmement difficile de réaliser une application de « trading » en temps réel traitant et affichant une courbe à l'aide des « web services ».

La meilleure utilisation de ces derniers, réside dans une approche orientée document ou résultat plutôt qu'une approche orientée traitements distribués tel que les **RPC** (Remote Procedure Call).

CORBA / IIOP et l'interopérabilité

CORBA (Common Object Request Broker Architecture) qui a été élaboré par le consortium de l'**OMG** (Object Management Group) constitue sans doute l'un des middlewares d'objets distribués les plus utilisés au monde. On ne compte plus aujourd'hui le nombre d'**ORB** (Object Request Broker) gratuits ou payants basés sur ce modèle de développement ou intégrés dans une multitude de serveurs d'applications. La plate-forme JAVA à travers la norme J2EE a aussi contribué à son succès en adoptant le protocole **IIOP** (Inter-ORB Protocol) au sein même de l'architecture de composants distribués Enterprise Java Beans (EJB).

Le couple « CORBA / IIOP » a prouvé à plusieurs reprises sa maturité et sa capacité à traiter de gros volumes de données dans des architectures JAVA / C++, que cela soit au niveau de moteurs financiers ou d'importantes applications industrielles.

Et qu'en est-il de la plate-forme .NET ? Malheureusement, le Framework .NET 1.1 n'intègre pas de couche native « CORBA / IIOP », espérons que Microsoft comblera très vite cette grosse lacune dans les versions suivantes. Une alternative possible est l'utilisation de ponts « JAVA/COM » ou « RMI/.NET Remoting » tels que JaNet, Halcyon ou JnBridge. Ces derniers conviennent parfaitement à certaines applications dites légères mais ne peuvent en aucun cas constituer une réponse unique au problème de l'interopérabilité.

La société Borland qui a conçu plusieurs outils de développement pour la plate-forme .NET (C# Builder, Delphi Studio .NET) et JAVA (JBuilder, Borland Enterprise Server), a sorti un framework nommé **Janeva** (<http://www.borland.com/janeva/index.html>), permettant à des applications .NET d'accéder à des composants « JAVA / J2EE », par l'intermédiaire de « CORBA / IIOP ». Ce framework semble donc très intéressant en terme d'interopérabilité .NET / JAVA.

Une autre initiative tout aussi intéressante et qui mérite d'être soulignée, vient d'un projet suisse Open Source nommé **IIOP.NET** (<http://iiop-net.sourceforge.net/>). Il utilise lui aussi le couple « CORBA / IIOP » pour interopérer entre .NET et JAVA. En voici un petit exemple d'utilisation: http://www.codeproject.com/csharp/dist_object_system.asp

Il est surprenant de constater que Microsoft n'ait pas intégré de couche native « CORBA / IIOP » dans sa version actuelle du Framework .NET. Cette couche aurait permis une interopérabilité plus rapide entre le monde .NET et JAVA et donc une plus grande ouverture. Heureusement, des initiatives et des projets indépendants ont fait leur apparition et nous prouvent que les deux principales plates-formes que sont JAVA et .NET peuvent très bien interagir entre-elles et être complémentaires.

En conclusion

Microsoft a su tirer parti des concepts technologiques de ses concurrents pour élaborer une plate-forme .NET stable et performante. Plusieurs problèmes récurrents de l'architecture DNA ont été résolus avec l'arrivée du Framework .NET 1.1. Pour l'instant, quelques grands comptes ont commencé à utiliser la technologie .NET, mais il existe encore très peu d'applications critiques tournant sur cette plate-forme. Laissons encore du temps à .NET pour savoir s'imposer comme remplaçant de l'ancienne architecture DNA de Microsoft et arriver à être nativement interopérable avec le monde JAVA.

A propos de Digitalis Consulting

La société **Digitalis Consulting** s'est spécialisée dans le conseil, la délégation de compétences et dans la recherche et le développement technologique. Elle développe son expertise et formalise son savoir-faire dans la réalisation d'architectures applicatives centralisées et distribuées en intégrant le meilleur des technologies issues des principaux acteurs du marché.

L'évaluation et l'analyse des plates-formes et technologies émergentes font partie intégrante de Digitalis Consulting ; le but étant de promouvoir une approche pragmatique et architecturale, à fin de mesurer au mieux le risque technologique dans le cadre de l'évolution des systèmes d'information d'entreprise.

Pour toute information complémentaire, n'hésitez pas à nous contacter: info@digitalisconsulting.com

Rédaction Novembre 2003 :

Mohammed-Lotfi Mattou, CTO
Digitalis Consulting Sàrl
<http://www.digitalisconsulting.com/>

Sources

Microsoft .NET : <http://www.microsoft.com/net/>

Microsoft .NET Framework : <http://msdn.microsoft.com/netframework/>

DotNetGuru.org: <http://www.dotnetguru.org>

Dotnet-fr.org : <http://www.dotnet-fr.org>

Adea.pm.gouv.fr: http://www.adae.pm.gouv.fr/upload/documents/Microsoft_SyntheseDotNet.pdf

01 Informatique: http://www.01net.com/article/215702_a.html

Microsoft .NET FrameWork en bref :
<http://www.microsoft.com/France/msdn/technologies/technos/framework/bref.asp>