POLYNESIE FRANÇAISE

MINISTERE
DU DIALOGUE SOCIAL

SERVICE DU PERSONNEL
ET DE LA FONCTION PUBLIQUE

CONCOURS EXTERNE D'INGENIEURS DE CATEGORIE A

SPECIALITE : INFORMATIQUE – ETUDE, DEVELOPPEMENT D'APPLICATIONS

EPREUVE ECRITE D'ADMISSIBILITE

Rédaction d'une note de synthèse à partir de l'analyse d'un dossier technique portant sur la spécialité retenue

Durée : 4H00 – Coefficient 5

MARDI 26 OCTOBRE 2004
de 13h00 à 17h00

B.P. 124 , 98713 Papeete – TAHITI ,Polynésie française – Avenue Prince Hinoï, Immeuble MOEHAU
Tél. : (689) 47 79 00 - Fax. : (689) 53 31 12 - Email : direction.pel@personnel.gov.pf

CONCOURS D'INGENIEUR EN INFORMATIQUE

OPTION : ETUDES, DEVELOPPEMENT D'APPLICATIONS

A partir des documents joints qui présentent les solutions J2EE et .NET, faire une synthèse des technologies présentées et en donner les avantages et inconvénients.

Dans le cadre de la migration d'une application spécifique développée en Visual Basic, Oracle Developper 2000 avec un système de gestion de base de données ORACLE sur des serveurs SUN (Solaris) et WINDOWS NT vers une application en technologie WEB, quelles préconisations feriez vous pour opter pour l'une ou l'autre des solutions ? Argumentez vos propositions en proposant un modèle d'architecture logicielle et d'infrastructure.
Cette nouvelle application a vocation à être disponible dans un réseau intranet et pour des clients de l'Internet. Il convient d'intégrer dans votre proposition d'architecture la prise en compte de l'intégrité et la sécurité des données et des accès.

SOMMAIRE

> Livre Blanc

Développement d'applications J2EE : Acheter ou développer un framework technique ?

IdeoTechnologies
www.ideotechnologies.com
124, rue de Verdun • 92800Puteaux
Tél. : +33 (0)1 46 25 09 60 • Fax : +33 (0)1 46 25 90 09

Vous avez choisi de développer des applications Java J2EE ?

Après quelques expériences pilotes, vous vous êtes rendu compte du surcoût lié à l'utilisation de Java par rapport à d'autres langages ou plateforme. Vous avez constaté qu'il n'existait pas d'environnement de développement *intégré* pour Java. Vous cherchez des solutions d'amélioration de la productivité. Toutes ces réflexions vous conduisent peut-être à vous intéresser aux frameworks techniques.

Il existe une littérature abondante sur Internet concernant le développement J2EE et les frameworks techniques.

Tous les retours d'expérience de sociétés ayant développées puis déployées des applications J2EE tendent à démontrer le surcoût (de 20 à 30%) des applications développées en J2EE. Les raisons principales de ce surcoût tiennent à la complexité de J2EE, à la grande latitude laissée aux développeurs ainsi qu'aux manques d'outillages.

Il est impératif pour une organisation d'étudier et d'installer un système capable de satisfaire ses besoins actuels et futurs.

Le framework technique dans ce type de contexte apporte une partie des réponses : structuration des développements, accélération des développements par l'usage de composants réutilisables, notamment. Dès lors que l'on veut utiliser un framework technique se pose la question du choix : **développer son propre framework ou acquérir un framework commercial ?**

La décision entre le développement interne en intégrant éventuellement des composants Open Source et l'achat d'un logiciel commercial est à la fois importante et complexe. **Le développement interne est-il la solution la moins coûteuse ? Un logiciel commercial fournira-t-il les fonctionnalités dont vous avez besoin ?**

Ce document vise à vous aider à décider entre le développement d'une solution spécifique, le paramétrage d'une solution open source, le développement par une société extérieure, ou l'achat d'un framework commercial comme ***SweetDEV***.

Le point essentiel est la compétence métier de votre entreprise. Si vous pouvez allouer les ressources • temps, effort et finances • nécessaires au développement d'un logiciel en interne, vous pouvez envisager le développement d'un framework technique. Sinon, un framework commercial, tel que ***SweetDEV***, fournit une solution probablement **meilleure et plus économique**.

> *Développement*

Concentration sur la problématique principale

Développer un framework technique est plus facile à dire qu'à faire.
Quelle option est la meilleure ?
Quelle architecture retenir ?
Vous pouvez développer une solution spécifique s'appuyant sur des développements internes. Vous pouvez choisir d'agréger plusieurs composants Open Source. Vous pouvez également sous-traiter le développement.

Développer une solution peut être extrêmement coûteux. Lorsque vous établissez le budget pour votre framework, **pensez à l'investissement à long terme.** Il faut estimer les coûts initiaux • tels que la mise en œuvre ou la recherche des composants nécessaires. Gardez également à l'esprit les autres coûts : documentation, formation, support, maintenance, développement de nouvelles fonctionnalités, etc.

Mme Hendrickson écrit : « *Attention aux fausses économies. Lorsque vous calculez le prix de l'outil, pensez aux heures qui seront passées à son développement, sa maintenance, sa documentation, et son support.* »

> *Développement* maison *à base de composants Open Source*

Il peut-être tentant de construire son propre framework à partir de composants internes ou Open Source. Bien que le code soit gratuit, il existe des coûts incompressibles. Il est nécessaire de collecter sur Internet les composants adéquats. Puis, il faut que votre équipe s'autoforme pour comprendre le fonctionnement de ces différents composants. La documentation disponible n'est pas nécessairement d'une qualité uniforme. **Qui assurera la maintenance de ce framework une fois réalisé ? Quel est le coût de rédaction de la documentation ? Que se passera-t-il lorsque les experts ayant créé ce framework quitteront votre entreprise ?**

> *Sous-traitance*

Un framework technique peut également être développé et géré par un sous-traitant. **Cette solution est généralement la plus coûteuse**, mais peut vous fournir ce dont vous avez besoin. Elle présente les avantages et les inconvénients d'un développement spécifique. (coût et délais des évolutions, pérennité,...).

Dans un article intitulé *Build It or Buy It* de l'édition de Mai/Juin 2001 de STQE, *Elisabeth Hendrickson* écrit :
« *Sauf si le métier de base de votre entreprise ou de votre service est de développer des logiciels, il est fort probable que nous ne puissiez pas développer de meilleur produit qu'un outil commercial. Souvenez vous que les sociétés qui distribuent des outils commerciaux ont investi un temps important et des efforts considérables à développer leur produit. Votre équipe informatique peut-elle créer un meilleur outil en quelques semaines.* »

> *Achat*

Appuyez-vous sur une solution éprouvée

Les frameworks commerciaux, tels que ***SweetDEV***, doivent satisfaire à vos besoins en terme de développement d'applications J2EE. Ces solutions offrent une excellente **adéquation entre les fonctionnalités, l'architecture, la documentation, le support, et l'évolutivité.** Correctement choisi, la mise en oeuvre d'un framework améliorera la qualité, la maintenabilité de vos applications et la productivité de vos équipes. Elles pourront se concentrer sur les composants *métiers* des applications.

Prenez le temps d'examiner les frameworks commerciaux disponibles. La plupart des sociétés commerciales proposent une évaluation gratuite de leurs produits. Après l'installation et l'utilisation du logiciel, vous pourrez rapidement évaluer si le produit répond à vos besoins.

Acheter un framework technique commercial est la solution **la plus économique.** Cette solution permet à vos développeurs de se concentrer sur les projets générateurs de revenus et de fournir des réponses plus rapides à vos clients.

Si le cœur de métier de votre société n'est pas le développement de logiciels techniques, vous tirerez parti de l'utilisation d'une solution développée par une société qui se concentre sur la fourniture de la meilleure technologie possible.

> *Développement ou achat*

Votre temps, c'est de l'argent

Prenez un peu de temps pour effectuer une analyse comparative, développement ou achat.

Dans un article intitulé *Build vs Buy* posté sur asp.com, *Jit Argawal* écrit : « *Votre analyse des coûts réels doit inclure non seulement la totalité des charges de structures pour les employés qui seront affectés au projet, mais également le manque à gagner représenté par le fait qu'ils ne travaillent pas sur des projets plus rentables. On a souvent de mauvaises surprises lorsque l'équipe a dépassé le point de non retour, et que des opportunités intéressantes se présentent, mais que les ressources ne sont pas disponibles.* »

Critères	Dév. Interne	Achat
Temps de développement réduit		X
Disponibilité du produit		X
Coûts initiaux réduits	X	
Coûts récurrents de développement	X	
Evolutions constantes du produit		X
Support complet du produit		X
Documentation du produit		X
Formations disponibles		X
Compétences de la société		X

Tableau 1 • Analyse développement ou achat

Pour terminer cette analyse, examinez les coûts évidents et les coûts cachés des quatre solutions :

- Développer une solution maison
- Utiliser une solution open source pour développer et personnaliser une solution
- Sous-traiter le projet
- Acheter un logiciel commercial immédiatement disponible

Un résumé des critères de décision est présenté dans le **tableau 1**. L'achat d'un framework commercial procure davantage de bénéfices, tels que l'économie de temps, d'efforts et de coûts à long terme.

A l'heure du choix

Dans le numéro du 20 Février 2002 de *iSource Business, Chris Watts*, abordant la question du choix entre le développement et l'achat, écrit : « *Il est impératif pour une entreprise d'étudier et d'installer un système qui soit capable de satisfaire ses besoins présents et futurs. Ceci conduit à l'inévitable question sur le choix entre la conception et le développement d'un système spécifique, ou l'achat d'un outil commercial. La plupart des analystes de l'industrie nous disent que la meilleure solution consiste à tirer parti de progiciels.* »

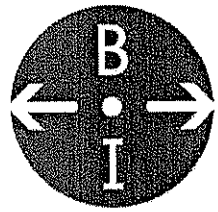
SweetDEV, de *Ideo Technologies*, fournit un ensemble cohérent (Briques Techniques Industrialisées, application Exemple, documentations, formations, support). ***SweetDEV*** représentait fin 2003, un volume de développement de plus 15 années/homme. ***La documentation en ligne*** du produit couvre l'ensemble des fonctionnalités au travers de 600 pages. ***La Javadoc*** fournie comprend les diagrammes UML facilitant l'utilisation du produit. ***La formation initiale de 3 jours*** peut être complétée par un module d'approfondissement de 2 jours. ***Les consultants d'Ideo Technologies*** et/ou de ses partenaires peuvent accompagner vos équipes lors des phases de lancement des projets.

SweetDEV est une solution, immédiatement disponible et déjà utilisée par plusieurs dizaines de développeurs.

SweetDEV permet à vos équipes de se concentrer sur des développements à haute valeur ajoutée pour vos clients.

PASSAGE À .NET :
QUELS GAINS POUR L'ENTREPRISE ?

Version 2.0



LIVRE BLANC

Emmanuel Henrion

François de la Villardière
Fondateurs & Dirigeants

Nous sommes heureux de vous présenter la nouvelle version de notre white paper sur .NET. Dans un contexte économique plus tendu où « consolidation » est devenu le maître mot des systèmes d'information, quelle valeur ajoutée peut apporter à l'entreprise la nouvelle offre .NET de Microsoft ? Comment la position de .NET s'affirme-t-elle au fil du temps ? Business Interactif fait le point sur le sujet…retours d'expérience à l'appui.

Business Interactif en quelques mots

Business Interactif conçoit et réalise de grands projets e-business à forte composante technologique, couvrant les aspects front, middle et back-office. Avec plus de 200 collaborateurs, dont 60% sur les aspects technologiques, Business Interactif accompagne ses clients sur des projets stratégiques en France et à l'étranger (Bureaux à Paris, Lyon, New-York, et Tokyo).

Microsoft a récemment confié à Business Interactif la réalisation de l'ouvrage « Quel Portail pour les entreprises agiles » qui fait le point sur la nouvelle offre portail .NET de Microsoft (téléchargeable sur notre site).

Pour plus de renseignements, nous vous invitons à consulter notre site : www.businessinteractif.fr

Jean-Louis Bénard
Directeur Technique
de Business Interactif

Présentation du white paper sur .NET

Les équipes de Business Interactif interviennent depuis 1996 au cœur des systèmes d'information sur des architectures complexes. CORBA, COM, J2EE…Nous mettons à profit le meilleur des technologies pour créer de la valeur ajoutée. C'est dans cet esprit que depuis 2000 nous avons investi sur l'offre .NET de Microsoft. Avec aujourd'hui des retours d'expérience concrets et plus de 75% des effectifs techniques opérationnels sur le framework et les serveurs .NET, nous avons souhaité publier un white paper dont l'objectif est double : d'une part expliquer simplement ce qu'est .NET, répondre aux grandes questions qui gravitent autour de .NET (positionnement par rapport à J2EE, etc.); d'autre part partager notre retour d'expérience, nos préconisations de mise en oeuvre, et ce au-delà du message marketing qui entoure « .NET ». En six mois l'actualité .NET a été particulièrement riche, c'est pourquoi il était nécessaire de publier une nouvelle version de ce document. N'hésitez pas à consulter notre offre et nos événements (séminaires, conférences) concernant .NET en accédant à notre site www.businessinteractif.fr

Contacts Clients :

Angela Apovo – angela.apovo@businessinteractif.fr

Business Interactif
8, Rue Fournier – 92110 Clichy
Tél. 01 49 68 12 12 - Fax: 01 49 68 12 13
RCS Nanterre B 403 578 198 - APE 722Z

TABLE DES MATIERES

1 EXECUTIVE SUMMARY

1.1 POURQUOI UN WHITE PAPER SUR .NET

.NET a suscité un nombre impressionnant de publications en tout genre. Pourquoi un nouveau white paper sur le sujet ? Depuis 2000, Business Interactif a investi sur les serveurs et sur le framework .NET et consolide aujourd'hui un fort retour d'expérience sur des cas concrets de mise en œuvre. Nous avons souhaité partager ce retour d'expérience par le biais d'un document de synthèse. Ce document n'a pas pour vocation de présenter dans le détail ce qu'est « .NET », mais plutôt de répondre à des questions que se posent régulièrement maîtrises d'ouvrage, directions informatiques et directeurs de projets : que contient réellement .NET ? Comment gérer l'existant ? Comment se positionne .NET par rapport à J2EE ? Quels sont les scénarios de mise en œuvre de .NET dans le système d'information de l'entreprise ? Autant de questions auxquelles nous nous sommes efforcés de répondre simplement, en s'appuyant notamment sur des études de cas concrètes. Cette seconde édition permet de prendre en compte les nouvelles orientations prises par Microsoft sur l'année 2002, et leur impact au cours de 2003, notamment pour tout ce qui concerne les choix d'architecture, les extensions au niveau des web services, l'intégration des serveurs .NET avec Office...

1.2 A QUI S'ADRESSE CE DOCUMENT ?

Ce document s'adresse en priorité aux décideurs informatiques (directeurs de système d'information, responsables d'études, directeurs de projet) et aux interlocuteurs techniques des maîtrises d'ouvrage. Un certain nombre de points sont étayés par des « zooms » assez techniques, plus à destination d'architectes. Ces zooms techniques sont indiqués par le sigle suivant :

Par ailleurs les points de synthèse importants sont mis en valeur par un fond grisé :

Point de synthèse

1.3 VOUS SOUHAITEZ ALLER PLUS LOIN

N'hésitez pas à contacter nos équipes. Nous étudierons avec vous comment tirer au mieux parti de .NET pour créer de la valeur au sein de votre système d'information.

2 QU'EST CE QUE .NET

Définir .NET simplement relève d'un exercice somme toute assez complexe. En effet, la succession des messages marketing diffusés par Microsoft sur le sujet finit par brouiller les cartes. Pour mieux comprendre les enjeux .NET de Microsoft, rappelons que les premières annonces se sont déroulées à un moment où J2EE se révélait comme une architecture très compétitive pour les entreprises. La complexité du message tient probablement au fait que « .NET » désigne plusieurs facettes de l'offre Microsoft :

- Les « serveurs .NET » : la suite de briques applicatives de Microsoft, plus ou moins proches de l'infrastructure, plus ou moins positionnés sur des couches applicatives « hautes ».
- Le « framework .NET » et Visual Studio .NET : la boîte à outils pour construire des applications
- Les services .NET : ensemble de services prêts à l'emploi fournis par Microsoft et utilisables par les développeurs, à l'image de Passport, le système d'authentification de Microsoft.

L'ensemble est rassemblé sur un schéma de synthèse mis en avant par Microsoft (Figure 1).

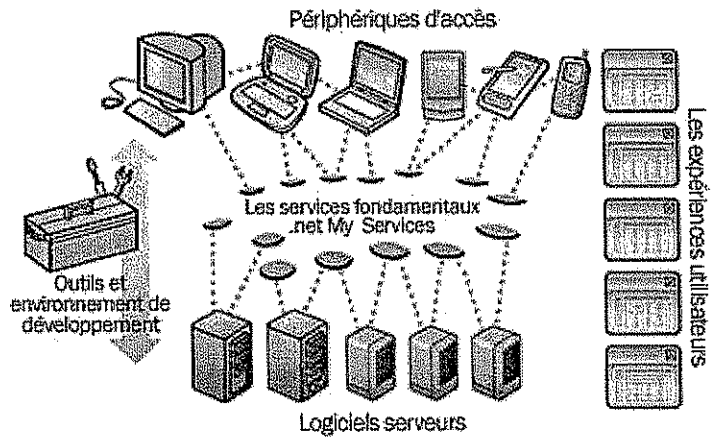


Figure 1 - La vision .NET - source : Microsoft

Nous allons – de manière macroscopique, car, nous l'avons dit, l'objet de ce document n'est pas de détailler ce qu'est .NET – passer en revue les serveurs .NET et le Framework, pour pouvoir répondre à des questions simples comme : y a-t-il une convergence entre les serveurs .NET et le framework ? Comment se positionne .NET par rapport à J2EE ?

2.1 LES SERVEURS .NET

Les serveurs .NET désignent l'ensemble des briques applicatives de Microsoft, que ce soit des briques plutôt orientées infrastructure (Exchange, Biztalk) ou plutôt progiciels (Commerce Server par exemple). Pour rappel :

- **Windows Server 2003** est le système d'exploitation de Microsoft, héritier de Windows 2000. Le serveur d'exploitation est la pierre d'angle des serveurs .NET. Les aspects disponibilité et scalabilité ont été particulièrement renforcés. Le système d'exploitation intègre (à la différence du monde J2EE) les éléments constitutifs du serveur d'application (moniteur transactionnel, etc.). Parmi les nouveautés importantes de .NET Server, on retrouve :
  - L'intégration en standard du framework .NET
  - Le serveur web IIS6, une nouvelle fois optimisé mais surtout installé par défaut avec toutes les options de sécurité activées. Le travail de l'administrateur consiste donc à ouvrir des portes de sécurité plutôt qu'à les fermer
  - L'intégration de services UDDI
  - Des possibilités de clustering plus avancées que celles disponibles via la version 2002
  - La nouvelle version des services Windows Media (9)

- **Application Center** assure le déploiement et la gestion des applications Microsoft. La richesse mais aussi la complexité des architectures à haute disponibilité rend peu à peu incontournable l'utilisation de solutions de ce type pour gérer et automatiser le déploiement des solutions Microsoft évoluées. Ce serveur reste encore peu connu des entreprises, qui n'ont pas forcément atteint le niveau d'industrialisation justifiant l'utilisation d'Application Center.

- **BizTalk Server** est la solution EAI de Microsoft (intégration d'applications). Elle couvre essentiellement des fonctions de mapping de données (transformation de documents en d'autres documents) et des fonctions d'orchestration (création de workflows de processus, orchestration de web services, etc.). Biztalk, face à un marché pourtant très solide (WebMethods, Seebeyond, etc.) a trouvé sa place, notamment du fait d'une politique tarifaire très agressive. L'intégration de Visio pour définir les workflows, l'utilisation native d'XML, mais aussi la complémentarité avec les autres solutions logicielles de Microsoft ont constitué des arguments auxquels les entreprises ont été sensibles.

- **Commerce Server** est probablement le plus mal nommé des serveurs Microsoft. Sorti à la grande époque du commerce électronique, Commerce Server porte avec difficulté un nom qui n'est pas adapté, alors qu'il adresse des problématiques réelles du système d'information, souvent hors d'un contexte « commerce électronique grand public » : gestion de catalogue de produits, fonctions de personnalisation et de profiling, etc. C'est un outil très riche, souvent mal connu ou sous-utilisé.

- **Content Management Server (CMS)** est l'un des plus récents de l'offre Microsoft et en même temps l'un des plus importants. Issu d'un rachat, repackagé, puis redéveloppé en partie, Content Management Server est l'offre de gestion de contenu de Microsoft. Encore une fois, l'éditeur s'attaque à un marché très convoité sur lequel des acteurs de poids (Documentum, Vignette, Divine, Interwoven, Tridion, etc.) se positionnent depuis un moment. Pour se démarquer de ses concurrents, Microsoft joue une fois de plus l'intégration du produit avec le reste de la suite logicielle : CMS est et sera de plus en plus intégré à Commerce Server (alimentation du contenu rédactionnel associé aux produits du catalogue), et à Sharepoint Portal Server (l'offre portail de Microsoft sur laquelle nous allons revenir).

- **Exchange Server** est l'offre de messagerie et de groupware de Microsoft. C'est à la fois un pilier de la suite logicielle Microsoft, concurrent de poids face à IBM, mais c'est aussi un serveur qui a connu et connaît une histoire complexe. En particulier, son positionnement par rapport aux fonctionnalités de travail en groupe (également proches des problématiques de portail) n'a jamais été et n'est pas encore clair. Microsoft, au fil des années, a hésité à intégrer à Exchange des fonctionnalités applicatives avancées. Résultat : le développement d'applications de groupe reste complexe, même si les fonctionnalités supportées par Exchange sont extrêmement riches. La répartition de la logique applicative entre le serveur Exchange et les clients Outlook a d'ailleurs fortement contribué à cette complexité plus qu'apparente. Aujourd'hui toutefois la sortie de la version 2003 d'Exchange Server (nom de code Titanium, disponible uniquement sur Windows Server 2003) laisse entrevoir des avancées significatives :
    - Certaines fonctionnalités ont été sorties du produit car désormais couvertes par d'autres serveurs. Citons par exemple les outils collaboratifs temps-réel.
    - Une meilleure gestion du tandem Outllok – Exchange, notamment grâce à la sortie d'Outlook 11, qui dispose de fonctionnalités « cache » particulièrement efficaces, permettant de minimiser le dialogue entre le poste client et le serveur.
    - Une meilleure gestion des aspects sécurité, que ce soit au niveau de l'authentification Kerberos, ou de la sécurité offerte sur Outlook Web Access
    - Le support de RPC sur HTTP, permettant d'utiliser Exchange sur Internet sans passer par des réseaux privés virtuels.

- Des améliorations sensibles d'Outlook Web Access, avec notamment la mise à disposition d'une version avancée utilisant toutes les fonctionnalités d'Internet Explorer.
- Le support des protocoles Wireless, permettant notamment la synchronisation de clients Pocket PC dans sa déclinaison téléphone.
- Une amélioration sensible des aspects scalabilité, tolérance de panne et administrabilité.

- **Host Integration Server** permet la liaison avec les données et les applications de systèmes IBM SNA et UNIX. Outre l'accès aux données, il permet par exemple à des programmes CICS de participer à des transactions COM.

- **Internet Security and Acceleration Server (ISA)** est la solution firewall et proxy de Microsoft. La sécurité est un marché sur lequel Microsoft a encore clairement du mal à se positionner ; l'éditeur n'a pas toujours eu une très bonne image sur le sujet. Aujourd'hui ce marché reste dominé par des serveurs Unix et notamment Linux, et il est probable que l'évolution soit lente. En effet, l'argument habituel de « l'intégration facile » des différents produits n'est en l'occurrence pas vraiment un argument (voire un contre-argument) en terme de sécurité.

- **Mobile Information Server** fournit une plate-forme pour la gestion des applications mobiles, plutôt bien intégrée à Exchange Server. Il est probable que les évolutions de cette offre seront importantes, notamment avec la sortie des kits de développement d'applications mobiles pour Visual Studio .NET

- **Project Server** est la déclinaison server du produit Microsoft Project. L'outil de gestion de projet de Microsoft renforce sa dimension collaborative avec cette version Server qui permet aux différents collaborateurs du projet de suivre les tableaux de bord, d'effectuer la saisie de leurs temps passés via une interface Web ,etc.

Sharepoint Portal Server est une brique de l'offre portail de Microsoft. Aujourd'hui elle supporte des fonctions de gestion de documents (publication, archivage, recherche) au sein « d'un portail ». Le portail s'appuie sur la technologie Digital Dashboard (autrefois vendue séparément, désormais intégrée à Sharepoint), qui permet notamment la réalisation de « portlets ». L'un des ses principaux points forts est l'intégration avec la suite Office : grâce au protocole WEBDAV, l'utilisateur peut publier ou accéder à des documents sur Sharepoint Server comme s'il s'agissait d'un répertoire physique. Un atout de poids pour minimiser la charge de formation associée

à la mise en place de l'outil. L'intégration avec la suite Office ne devrait cesser de se renforcer, comme nous l'évoquons un peu plus loin dans ce document. Microsoft propose désormais une offre cohérente adressant le marché des portails d'entreprise. Cette offre combine l'ensemble des progiciels de la gamme, et également les outils poste clients. L'ensemble de cette offre est présenté un peu plus loin dans ce document.

- **SQL Server** est le serveur de base de données de Microsoft. Parti au départ avec un handicap d'image assez lourd, SQL Server, au fil des années, a acquis ses lettres de noblesse, jusqu'à rivaliser favorablement avec Oracle et DB2 sur des aspects performance et fiabilité. SQL Server est un peu le pivot de la suite progicielle de Microsoft, puisque pratiquement tous les serveurs Microsoft s'appuient sur SQL Server. Parmi les projets de Microsoft figure l'utilisation de SQL Server comme système de gestion de fichiers des futures versions du système d'exploitation Microsoft.

2.2 LE FRAMEWORK .NET

Le framework .NET constitue avec les serveurs .NET la deuxième pierre d'angle de l'offre de Microsoft. Il s'intègre particulièrement bien avec l'outil de développement Visual Studio .NET.

2.2.1 Le framework

Bon nombre de clients ont été désarçonnés par la terminologie « framework ». De quoi s'agit-il ? Le framework .NET est constitué d'une machine virtuelle chargée de l'exécution de programmes, la *Common Langage Runtime* (CLR), et d'un ensemble de classes techniques (Figure 2):

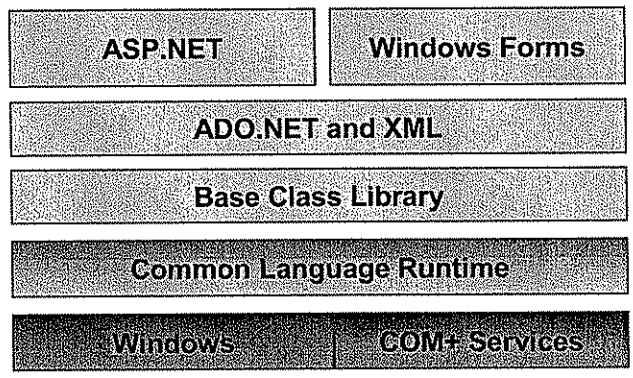


Figure 2 - Cartographie du framework .NET

2.2.1.1 La Common Language Runtime (CLR)

La CLR est une machine virtuelle (un peu comparable à la machine virtuelle Java, voir la section de ce livre blanc consacrée à J2EE versus .NET). Cette machine virtuelle est responsable de l'exécution des programmes .NET dans un environnement sécurisé. Parmi les points essentiels à retenir sur cette machine virtuelle :

- Une gestion de la mémoire optimisée par le biais d'un ramasse-miettes (garbage collector).
- Un packaging d'applications sous forme d'assemblies (remplaçant les DLLs), permettant de gérer les versions des briques applicatives avec une granularité très fine. Ces assemblies réduisent largement la complexité de déploiement (plus d'enregistrement en base de registres) et disposent d'un système de gestion des versions avancé.
- Une gestion de la sécurité plus fine, et une bonne résistance aux effets collatéraux dûs aux plantages d'une application (sécurité du code)

A la différence de la machine Java, la CLR permet l'exécution de programmes écrits dans différents langages (VB.NET, C#, J#, Pascal, Eiffel,...), les programmes étant compilés quelque soit leur langage d'origine dans un langage intermédiaire (*Intermediate Langage*) exécuté par la CLR. La CLR introduit la notion de type indépendant du langage.

A l'occasion de la sortie de la CLR, Microsoft a donné naissance à trois nouveaux langages :

- VB.NET, l'héritier objet de VB (qui n'a plus grand-chose à voir avec son prédécesseur en fait)
- C#, sorte de compromis entre C++ et Java qui séduira notamment les développeurs Delphi (le père de Delphi est le père de C#)
- J#, le langage Java de Microsoft pour la CLR

Enfin notons que la CLR repose – point non négligeable – sur les services COM+. Il n'y a donc pas à proprement parler de « rupture » par rapport à l'approche DNA. Nous reviendrons en détail sur les points communs avec l'architecture DNA Microsoft et sur la manière dont .NET peut s'intégrer à un existant DNA.

On parlera de code « managé » lorsque celui-ci est géré par la CLR. La facilité de déploiement, la meilleure gestion de la mémoire et la sécurisation du code induites par la CLR constituent des avancées importantes.

2.2.1.2 Les bibliothèques de classe

Au-delà de la CLR, Microsoft livre un ensemble de librairies de classes pour accélérer le développement : des classes techniques de base, des classes d'accès aux données (ADO.NET), des classes de manipulation XML très

puissantes (XML et XSD), et des classes gérant l'interface utilisateur, que l'on soit dans un environnement WEB (ASP.NET) ou dans un environnement client Windows lourd (Windows Forms appelés encore WinForms).

Ces librairies de classes sont extrêmement riches, à tel point que Microsoft a commencé à sortir de nouvelles librairies packageant des fonctionnalités sous forme de composants plus simples. C'est notamment le cas du DAAB (*Data Access Application Block*), permettant une mise en œuvre simplifiée d'ADO.NET., que nous présentons un peu plus loin dans ce document.

Contrairement à ce qui avait pu se produire dans le passé, Microsoft a particulièrement soigné les bibliothèques de classes pour les interfaces utilisateurs, que ce soit en environnement deux tiers ou trois tiers. Résultat : un ensemble cohérent qui permettra aux développeurs de passer rapidement d'un environnement à un autre. L'occasion de faire table rase d'un ensemble de librairies « historiques » qui cohabitaient sans réellement de vision unifiée.

Nous reviendrons plus en détail sur les librairies liées au développement ASP.NET ; en effet celles-ci permettent d'avoir une approche événementielle et orientée composant très proche de celle habituellement utilisée en client-serveur. Le développement ASP.NET s'appuie sur les pages ASPX, remplaçantes des pages ASP. Parmi les fonctionnalités importantes amenées avec APS.NET, la possibilité réelle de faire du « databinding », c'est-à-dire d'utiliser des contrôles orientés données.

2.2.2 Visual Studio .NET

Visual Studio .NET est l'environnement de développement de Microsoft, héritier de Visual Studio (Figure 3).

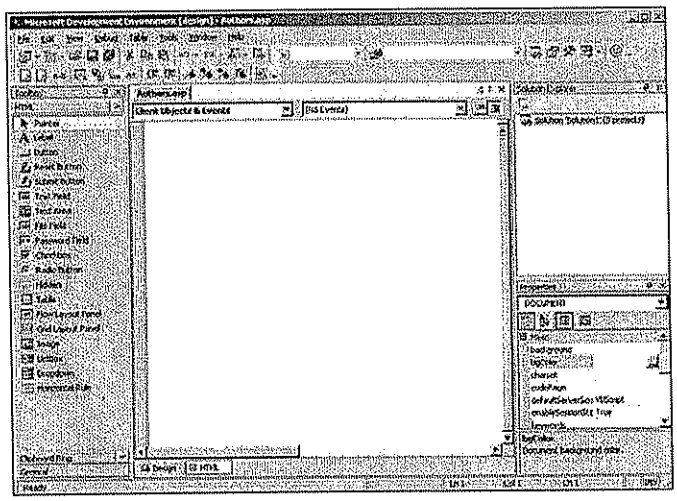


Figure 3 - L'environnement de développement Visual Studio.NET

Microsoft a fait converger ses outils de développement en un seul environnement capable de supporter plusieurs langages (VB.NET, C#...). Dans la mouvance « une seule machine virtuelle, plusieurs langages » Microsoft se devait de faciliter le développement multi-langages par le biais d'un environnement unique.

La principale caractéristique de Visual Studio .NET est de fournir un environnement capable d'accélérer considérablement la productivité du développeur, sur l'ensemble du cycle de vie du projet. Nous reviendrons un peu plus loin sur ce sujet.

Toutefois – au moins officiellement – Microsoft ne souhaite pas avoir une approche hégémonique concernant l'environnement et le langage de développement. L'un des slogans accompagnant le lancement de .NET était d'ailleurs « le meilleur langage c'est le votre ». L'essentiel étant de disposer d'un compilateur générant de l'*Intermediate Language (MSIL)*.

Microsoft ne pousse pas forcément à un passage vers ses langages de développement. Il sera possible de conserver son langage habituel dès lors qu'un compilateur vers l'Intermediate Language sera disponible.

2.2.3 Le Framework et Visual Studio dans leurs versions 2003

Microsoft a récemment sorti la version 2003 du Framework .NET. Outre un certain nombre d'améliorations, les principales nouveautés de ce framework sont les suivantes :

- le « Mobile Toolkit » permettant de développer des applications mobiles a été intégré au framework .NET. Le framework s'est donc étendu de composants qui permettent d'adapter le rendu visuel à des écrans de téléphones mobiles (smartphones, I-Mode...) ou de PDAs. La déclinaison « Compact » du Framework .NET permet quant à elle d'adresser tous les besoins de type PocketPC (y compris la version mobile de SQL Server).
- Le framework supporte désormais la coexistence de frameworks de versions différentes. Ainsi les applications développées pour le framework dans sa version 1.0 et le nouveau framework 1.1 peuvent coexister sur un même serveur.
- Les bases Oracle sont désormais nativement supportées par le Framework, sans passer par l'accès OLE DB.
- Côté Visual Studio, on retrouve un certain nombre de nouveautés, dont la possibilité de crypter les assemblies grâce à Obfuscator.

2.3 LES SERVICES .NET

.NET c'est aussi un ensemble de services mis à disposition par Microsoft pour être directement exploités par les développeurs, que ce soit sous forme de web services ou non.

Le premier de ces services est Passport, le système d'authentification de Microsoft. Un service qui a fait couler beaucoup d'encre, et qui reste un vrai sujet d'interrogation pour les entreprises. Car l'externalisation de bases de données utilisateurs (même si l'hébergement des serveurs Passport se fait chez des tiers et non chez Microsoft) n'est pas simple à envisager pour de grandes entreprises, pour des raisons évidentes de sécurité. La suite permettra de voir si des services techniques moins « sensibles » trouveront un meilleur écho. Aujourd'hui, de nouveaux services tels que Mappoint (service de géolocalisation) rencontrent un écho beaucoup plus favorable auprès des clients.

2.4 CONVERGENCE DES SERVEURS ET DU FRAMEWORK

L'offre Microsoft .NET pose naturellement la question de la convergence entre les « serveurs .NET » et le framework. Au-delà du discours marketing, qu'en est il réellement ?

Il faut reconnaître que pour l'instant un certain nombre de serveurs n'ont de .NET que le nom ; le cœur est écrit sous forme d'objets COM+, qui, bien qu'intégrables à des applications .NET, n'apportent pas réellement de ce point de vue une valeur ajoutée.

Pour mieux comprendre la tendance – et la stratégie de Microsoft – il est intéressant d'examiner en détail les toutes dernières versions des produits.

2.4.1 Content Management Server 2002 : un exemple significatif

Le produit, assez monolithique dans sa version 2001, a évolué dans sa version 2002 pour être facilement intégré à des applications Visual Studio .NET :

CMS 2002 dispose d'une interface d'administration dédiée, mais se présente surtout sous forme d'un ensemble de composants directement utilisables depuis Visual Studio .NET. Ainsi, la création de « modèles » de pages se fait de manière conviviale, par drag-and-drop de composants orientés gestion de contenu, et ce en restant dans l'environnement Visual Studio.

Il s'agit d'un avantage clé par rapport à bon nombre d'outils de gestion de contenu, qui, pour la création de modèles de pages, n'offrent qu'un éditeur de texte assez basique. Ici l'éditeur est Visual Studio.NET, un environnement qui va permettre à l'intérieur des modèles de pages de placer du code applicatif, et pourquoi pas, l'appel à d'autres composants de l'offre progicielle de Microsoft.

Les modèles ainsi créés pourront d'une part être utilisés par des contributeurs non techniques pour alimenter en contenu un site, mais pourront également intégrer une logique applicative évoluée. Illustrons ceci par quelques copies d'écrans : la création d'un projet « Content

Management » se fait depuis Visual Studio .NET, en utilisant un assistant (Figure 4) :

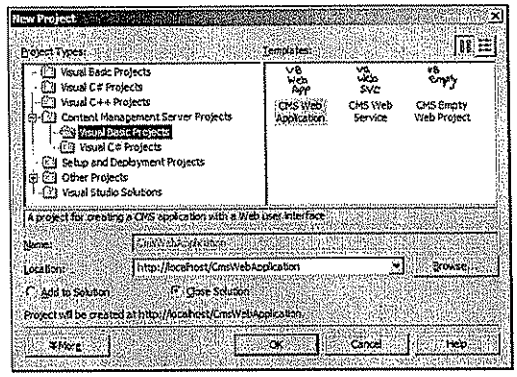


Figure 4 - Assistant Visual Studio.NET pour la création de projets CMS2002

Le projet ainsi créé, le développeur peut utiliser et modifier ses modèles (templates) de gestion de contenu, en positionnant des contrôles (PlaceHolders) destinés à recevoir la saisie des contributeurs (Figure 5) : les placeholders disponibles dans la palette de gauche sont disposés sur le template central. A droite, un navigateur de templates permet de passer rapidement d'un template à l'autre.

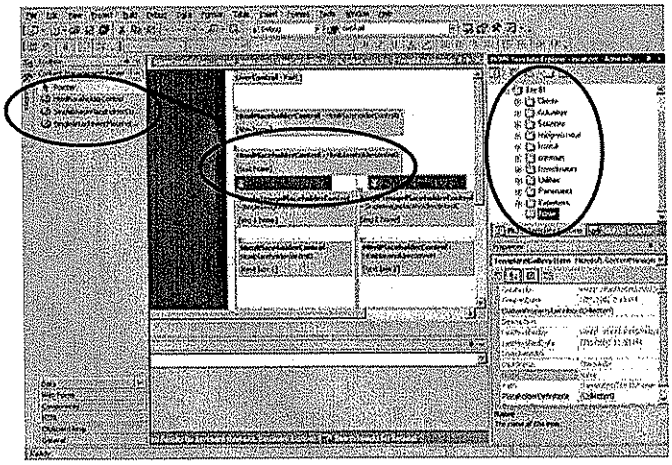


Figure 5 - Utilisation de CMS2002 dans Visual Studio

Les modèles ainsi créés sont des pages ASPX (les héritières des pages ASP) et à tout instant le développeur peut basculer vers le code source pour intégrer du code applicatif et de l'appel à des objets métier.

CMS 2002 va ainsi permettre aux développeurs d'adjoindre à leurs applications des fonctionnalités de gestion de contenu, en profitant des basiques CMS : gestion de workflow de publication, versionning, sécurité des contenus, etc.

2.4.2 Généralisation

L'exemple CMS 2002 illustre bien la tendance : les serveurs Microsoft vont être à la fois disponibles sous forme d'applications utilisables telles quelles, mais seront également accessibles sous forme de briques, de composants applicatifs utilisables directement dans Visual Studio .NET.

De ce fait, l'outil de développement de Microsoft joue de plus en plus un rôle d'agrégateur, « d'assembleur de pièces détachées » pour créer rapidement des applications à partir de briques existantes, qu'elles soient Microsoft ou non. Visual Studio .NET est donc un pilier (avec Biztalk) de l'intégration des serveurs .NET nouvelle génération.

2.5 CONVERGENCE DES SERVEURS .NET ET DU POSTE CLIENT : LA NOUVELLE OFFRE PORTAIL DE MICROSOFT

Microsoft se démarque de ses concurrents par un positionnement original sur le marché du portail. Alors que la plupart des grands éditeurs prônent un client léger (navigateur Web) comme interface unique du portail, Microsoft adopte une démarche complètement différente :

Tout en reconnaissant l'importance du client léger, réceptacle favori des applications 3 tiers, Microsoft met également en lumière le fait que les outils de productivité de l'utilisateur ne peuvent se résumer à un simple navigateur Web. Au quotidien, ce sont les outils bureautiques qui sont les plus utilisés, et ce sont eux qui doivent devenir la porte d'entrée privilégiée de l'utilisateur vers le portail.

2.5.1 Office 11, le client « intelligent »

L'offre portail de Microsoft est donc une offre qui s'appuie sur un « client intelligent », riche, regroupant à la fois le navigateur Web mais aussi l'ensemble de la suite Bureautique Microsoft. Ces différents outils dialoguent directement avec les briques du portail, notamment par le biais de Web Services et de « smart tags », liens intelligents qui, depuis un document Word ou Excel par exemple, permettent d'accéder à des fonctionnalités applicatives avancées du portail.

L'ensemble de la suite Office offre une prise en charge d'XML assez avancée, que ce soit au niveau de Word (Figure 6) ou de « XDocs » (Figure 7), la nouvelle application qui permet la création et l'alimentation de formulaires reposant sur une structure XML.

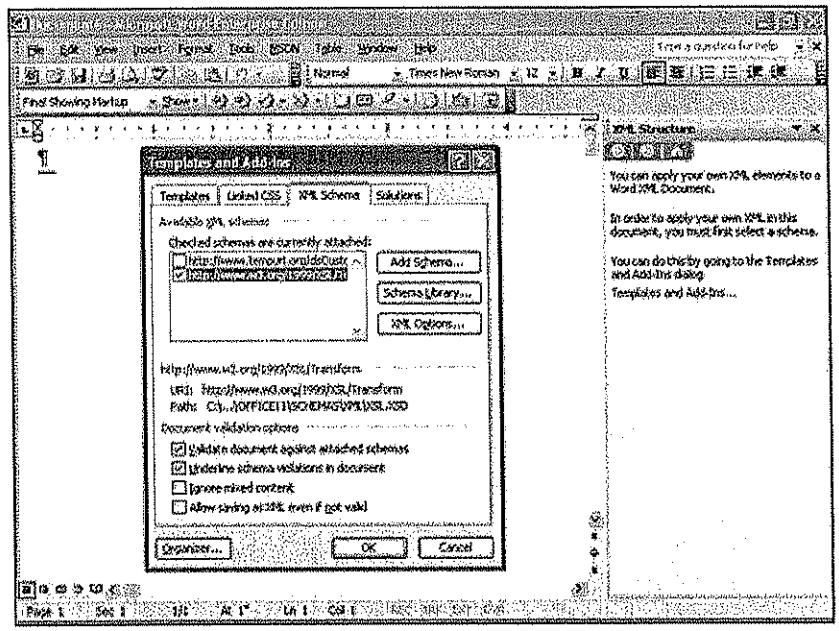


Figure 6 - Word 11 permet la création de documents s'appuyant sur un schéma XML - Source : Microsoft

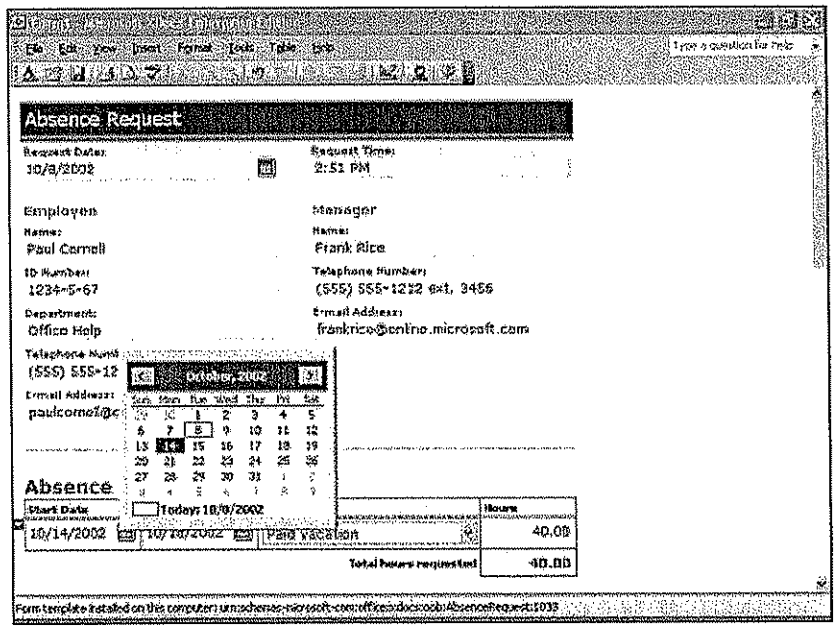


Figure 7 - "XDocs" permet la création et la saisie de formulaires reponsant sur une structure XML - Source : Microsoft

La sortie de Office 11 constituera l'aboutissement final de cette approche. En particulier, la possibilité de profiter des fonctionnalités de Visual Studio 2003 et de .NET pour étendre les fonctionnalités d'Office fait partie des points très attendus.

Côté Portail, l'offre a également été largement revue pour intégrer différentes briques applicatives. En particulier, Sharepoint Portal Server et Sharepoint Team Services ont été revus pour constituer un framework désormais cohérent.

Nous invitons le lecteur à se reporter au document « *Quel Portail pour les entreprises agiles* », réalisé par Business Interactif pour Microsoft, et qui permet d'appréhender en détail l'ensemble de l'offre Portail.

Nous présentons en Figure 8 le schéma de synthèse de l'offre :

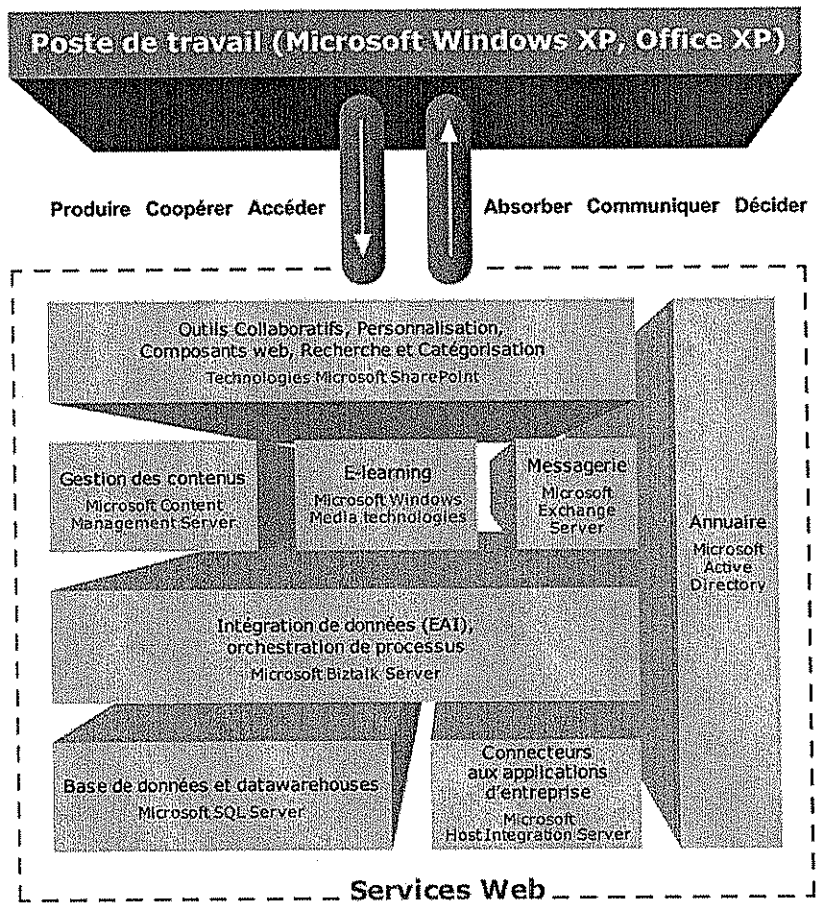


Figure 8 - L'offre Portail de Microsoft - source Microsoft "Quel Portail pour les entreprises agiles"

2.6 CONVERGENCE DNA / .NET

DNA (Distributed Network Architecture) désigne l'architecture qui précédait .NET, et plus spécialement les briques constituant le serveur d'application Microsoft intégré au système d'exploitation : les objets COM+, reposant

notamment sur le moniteur transactionnel de Microsoft MTS et le serveur de messages applicatifs MSMQ.

Aujourd'hui, l'architecture DNA est peu à peu éliminée du discours marketing au profit de .NET. Pas vraiment de rupture selon Microsoft, puisque le framework .NET s'appuie (par le biais de la CLR) sur les services COM+. Néanmoins, le développement .NET est toutefois bien différent du développement DNA. Il suffit de regarder le déploiement des applications pour s'en convaincre. Les objets COM+ ne s'appellent eux-mêmes plus COM+ mais « Enterprise Services ». Comment retrouver ses petits ? Quelles sont les stratégies de migration ?

Nous verrons dans la seconde partie de ce document quelles sont les stratégies de migration ou de cohabitation DNA/.NET. Toutefois, afin de faciliter la lecture du document, nous dressons ici un petit tableau de correspondance (si correspondance il peut y avoir) qui permettra aux lecteurs familiers des anciennes architectures Microsoft de s'y retrouver.

Architecture DNA	Architecture .NET
Pages ASP	**Pages ASPX, ASCX**
Services COM+ (MTS/MSMQ)	**Enterprise Services**
Objets COM, COM+	**Serviced Components**
Visual Basic, Visual C++...	**Visual Studio .NET**
DLL	**Assembly**
Scripting ASP	**Jscript, C#, VB.NET, Delphi...**

2.7 ARCHITECTURE .NET VERSUS J2EE

L'une des questions récurrentes sur .NET concerne son positionnement par rapport à l'architecture J2EE. Quelles sont les différences entre les deux approches ? Quelle est la meilleure ? Coupons d'emblée court à toute polémique : il n'y a pas de « gagnant » dans ce duel.

Les deux architectures ont un certain nombre de similitudes (patterns de conception assez proches), et les points sur lesquels les spécialistes concentrent leurs joutes (notamment la persistance relationnelle-objet) sont des aspects des deux frameworks encore relativement peu mis en œuvre.

Regardons de manière synthétique les deux schémas d'architecture .NET et J2EE, : la partie cliente (client léger ou client lourd), la partie objets métier et accès aux données (Figure 9) :

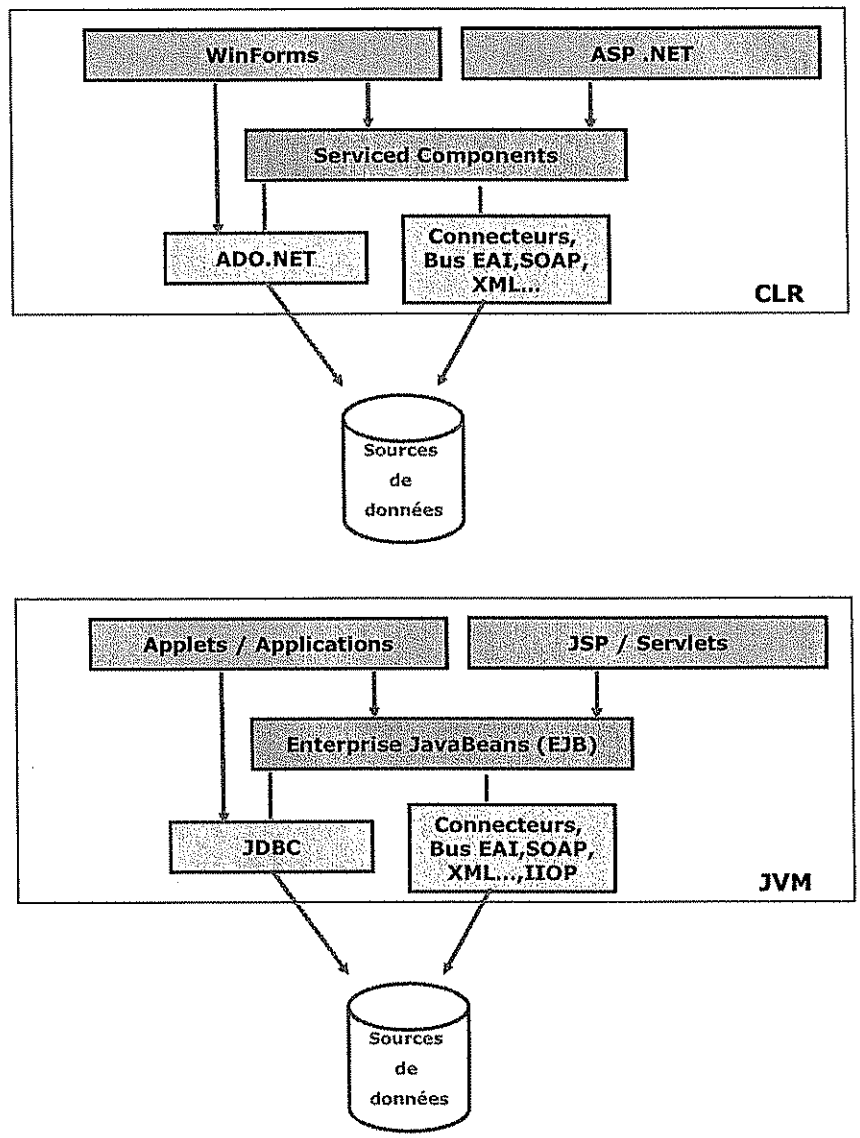


Figure 9 - Schémas de synthèse (simplifiés) des architectures .NET et J2EE

2.7.1 Machine virtuelle

Les deux architectures disposent d'une machine virtuelle qui exécute les programmes et qui prend en charge la gestion de la mémoire, notamment par le biais d'un garbage collector (ramasse-miettes). Dans les deux cas, on se trouve à mi-chemin entre le « compilé pur » et « l'interprété », avec une performance réduite de 10% par rapport à du compilé pur (variable selon benchmarks). Dans le cas de Java, cette performance est obtenue par une compilation (just-in-time ou non) qui fait généralement perdre au byte-code son caractère multi-plateforme.

2.7.2 Interface utilisateur

Côté interface utilisateur, il faut distinguer deux types de client : client léger et client lourd.

Le client léger est pris en charge dans .NET par les pages ASPX, par les pages JSP et les servlets côté J2EE.

Dans les recommandations liées à l'architecture J2EE et dans l'ancienne architecture Microsoft DNA, les pages JSP ou ASP ne prennent en charge pratiquement pas de mise en forme : les pages ASP/JSP appellent des objets (sur lesquels nous allons revenir), qui renvoient des flux XML. Les pages mappent alors ces flux avec des feuilles de style XSL pour renvoyer au final du HTML.

Le framework .NET, grâce aux fonctionnalités de CodeBehind et au principe des contrôles serveurs (présentées dans la partie suivante de ce document), permet d'avoir une approche de l'interface utilisateur légèrement différente, en particulier lorsque l'on cible des applications back-office usuellement réalisées en client-serveur.

L'approche XML/XSL n'est plus nécessaire pour tous les types de développement .NET, en particulier lorsque l'on souhaite bénéficier de la productivité « client-serveur » tout en s'appuyant sur une architecture trois-tiers. C'est une différence importante par rapport à l'architecture actuelle J2EE, même si ce type d'approche peut être couverte en partie par des bibliothèques Open Source complémentaires.

Pour ce qui est du client lourd, on s'appuyera sur des applets ou des applications Java standalone en J2EE, avec des frameworks type Swing, JFC, et sur les Winforms en .NET. L'architecture J2EE offre une bonne portabilité de l'application (approche mutli-plateforme) mais se révèle par contre moins performante que les Winforms sur plate-forme Microsoft.

2.7.3 Couche objets

Côté J2EE, ce sont les Enterprise Java Beans (EJB). On distingue les EJB sessions et les EJB Entities, ces derniers assurant la persistance sous forme d'objets métiers permettant d'isoler l'application du type de stockage utilisé au final.

Côté .NET, ce sont les Serviced Components, équivalents des EJB sessions. La partie persistance n'a pas aujourd'hui d'équivalent aux EJB Entities, le framework correspondant (ObjectSpaces) n'étant pas encore disponible en version finale.

Pour les deux architectures, les objets intégrant la logique applicative disposent d'un ensemble de services (gestion de la montée en charge, de la tolérance de panne, moniteur transactionnel, message queuing...), les

services EJB d'un côté, les services COM+ de l'autre (qui deviennent les « Enterprise Services »).

Pour ce qui est des appels entre objets distribués, on s'appuie en environnement J2EE sur les web services et sur les implémentations RMI/CORBA, tandis que côté Microsoft on utilisera les web services et .NET Remoting.

La différence essentielle sur la couche objet est liée à l'implémentation de la persistance. Cette différence se comble avec l'arrivée d'Objectspaces. Dans tous les cas cette différence n'est pas actuellement un critère significatif car l'implémentation de la persistance, notamment pour des applications de gestion, est peu mise en œuvre, même en environnement J2EE.

2.7.4 Accès aux données

L'accès aux bases de données se fait (frameworks de persistance relationnel-objet mis de côté) par JDBC en J2EE, et par ADO.NET en .NET (ADO.NET permettant l'accès à des drivers non managés pour des problématiques de compatibilité). Nous reviendrons plus en détail sur ADO.NET.

Pour ce qui est des autres types d'accès aux données, on s'appuyera soit sur des web services, des bus EAI ou des connecteurs spécifiques dans les deux cas.

2.7.5 Tableau de synthèse

Nous avons rassemblé dans un tableau comparatif les deux frameworks, même si cette comparaison doit être prise avec du recul, les périmètres de chacun n'étant pas exactement les mêmes.

Architecture J2EE	Architecture .NET
JVM	**CLR**
Pages JSP, Servlet	**Pages ASPX**
Swing, JFC…	**Winforms**
EJB Sessions	**Serviced Components**
EJB Services (JTA/JTS…)	**Enterprise Services (MTS…)**
EJB Entities / JDO	**ObjectSpaces**
JDBC	**ADO.NET**
RMI/CORBA	**.NET Remoting**

Les deux architectures ont beaucoup de similitudes, sans qu'il soit possible de déterminer si l'une surpasse l'autre. L'un des facteurs clé de choix pourra être la productivité des outils de développement associés, la stabilité des implémentations des deux architectures et l'historique de l'entreprise.

3 QUELS BENEFICES DANS LA MISE EN ŒUVRE DE .NET

3.1 INTRODUCTION

La section précédente de ce livre blanc a permis de présenter de manière macroscopique ce qu'était .NET, ainsi que son positionnement par rapport à l'architecture J2EE.

L'architecture semble intéressante, mais finalement justifie-t-elle de migrer un existant, d'acquérir de nouvelles compétences, etc. ? Nous avons rassemblé ici quelques éléments – non exhaustifs – mais qui illustrent les gains de productivité, de performance et de maintenabilité amenés par .NET. Les points étudiés sont les suivants :

- L'accès aux données
- La gestion de la performance
- La productivité d'un développement client-serveur nouvelle génération
- La réutilisabilité
- La couverture complète du cycle logiciel
- L'opportunité de mixer progiciel et développement sur mesure pour créer de la valeur

3.2 ACCES AUX DONNEES

L'accès aux données joue un rôle majeur dans les applications de gestion. Les difficultés rencontrées par Microsoft dans des vies antérieures sur le sujet (notamment avec les limites d'ODBC) illustrent bien le caractère critique de cet accès. De ce point de vue, ADO.NET apporte des gains sensibles, tant en termes de performance, que de productivité.

3.2.1 Disponibilité de drivers natifs

Microsoft a finalement répondu à une demande très forte de la part de la communauté développeurs : au-delà d'un accès générique aux bases de données, la disponibilité d'un driver natif pour SQL Server et pour Oracle, optimisé pour chaque base de données. Une avancée significative, en particulier pour le monde Oracle (Figure 10) :

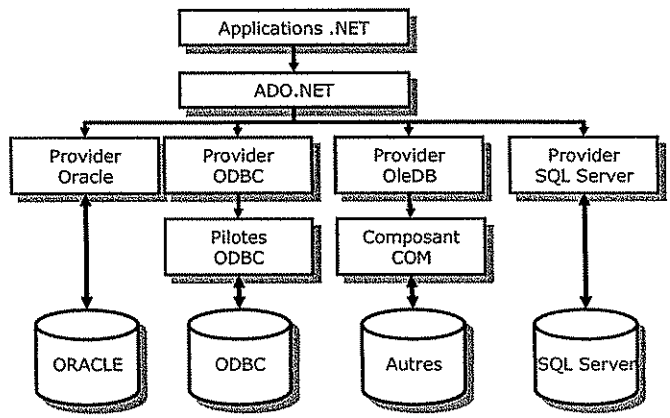


Figure 10 - Types de données accessibles via ADO.NET

La disponibilité d'un driver natif et optimisé pour Oracle est une avancée essentielle.

3.2.2 Des fonctionnalités comparables aux outils client-serveur les plus évolués

Concernant la librairie de classes accessibles par le développeur, Microsoft a fait là aussi des progrès importants. En effet, des fonctionnalités importantes, utilisables dans des ateliers client-serveur très évolués comme Delphi, n'étaient pas disponibles, ou restaient du moins complexes à implémenter. Aujourd'hui elles sont pratiquement toutes disponibles dans ADO.NET ;

Un exemple : la mise à jour différée, qui consiste à embarquer hors de la base de données un jeu d'enregistrements, à effectuer des manipulations sur ce jeu (insert, delete, update) puis à renvoyer le jeu de données à la base sous forme de transaction. Il s'agit d'une opération usuelle en client-serveur, très utile car elle permet de soulager la base de données en diminuant sa sollicitation. Il s'agit aussi d'une opération assez complexe, dans la mesure où il faut pouvoir gérer finement les problèmes liés à cette mise à jour différée : conflits de transactions, changements de valeur en base. Cette mise à jour différée et transactionnée est désormais disponible, et son implémentation (notamment grâce au modèle de programmation événementielle) reste rapide à implémenter, même si elle demande le plus grand soin.

L'évolution d'ADO.NET pour prendre en compte des problématiques client-serveur complexes en mode semi-déconnecté permet d'accompagner l'évolution des applications multi-tiers (« web ») vers des fonctionnalités complexes tirant réellement partie du système d'information.

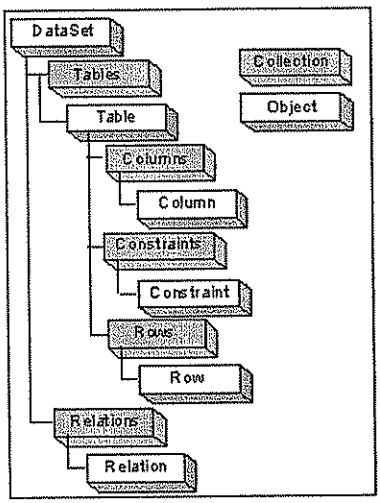


Figure 11 - Le Dataset

Ces fonctionnalités sont notamment permises par l'apparition d'un nouvel objet, le Dataset (remplaçant très évolué du recordset) qui permet d'embarquer en mémoire des ensembles de données et leurs dépendances, en gérant en cache les mises à jour.

3.2.3 ADO.NET et XML

Mais l'une des avancées les plus significatives de .NET en termes d'accès aux données se situe sur la gestion des données XML. C'est un point extrêmement sensible, puisque les échanges inter-applicatifs, qu'ils reposent ou non sur le protocole SOAP des web services s'appuient fortement sur XML.

Aujourd'hui une part importante du temps des développeurs, autrefois consacrée aux requêtes SQL, se partage maintenant entre ces mêmes requêtes SQL et du formatage de données XML.

Pour accélérer le travail du développeur, le framework .NET offre plusieurs outils :

- D'une part un ensemble de classes qui facilitent la manipulation de données XML, le parcours d'arbres, la vérification de la conformité d'un document par rapport à un schéma, etc.
- L'objet Dataset lui-même. En effet, le développeur peut, grâce à l'objet Dataset, avoir pour un même ensemble de données une vue relationnelle (« SQL ») et une vue XML, ces deux vues étant totalement synchronisées (voir schéma). Ainsi, selon le contexte et le type d'opération, le développeur utilisera la vue relationnelle ou la vue XML, pouvant passer instantanément de l'une à l'autre (Figure 12) :

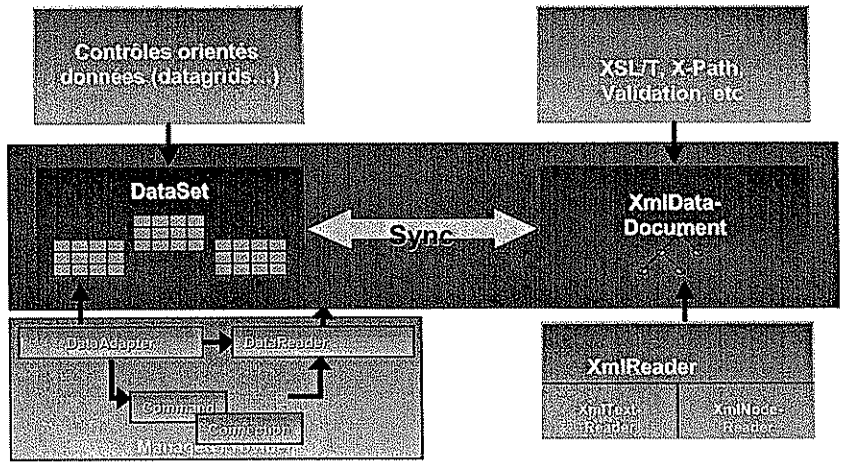


Figure 12 - Vue XML et relationnelle du dataset

En offrant une vue relationnelle et XML synchronisées des ensembles de données, le framework .NET accélère considérablement les développements dès que des manipulations de données XML issues de bases de données sont nécessaires (web services, EAI, etc...).

3.3 PERFORMANCE

La gestion de la performance des applications reste une priorité pour les entreprises. En particulier sur les architectures trois-tiers (« web ») où l'on est passé d'applications peu sensibles à des applications transactionnelles critiques.

Aujourd'hui, même si les solutions multi-tiers permettent la mise en place de fermes de serveurs pouvant supporter la charge, la nécessité de réduire le coût hardware / licences favorise la recherche de la performance par processeur.

3.3.1 Amélioration « brute » de la performance

De ce point de vue, comme l'atteste l'étude PC Magazine Nile Application Test (Figure 13), la performance délivrée par ASP.NET n'a plus grand-chose à voir avec son prédécesseur. En particulier, ASP.NET exploite au mieux la présence d'un grand nombre de processeurs.

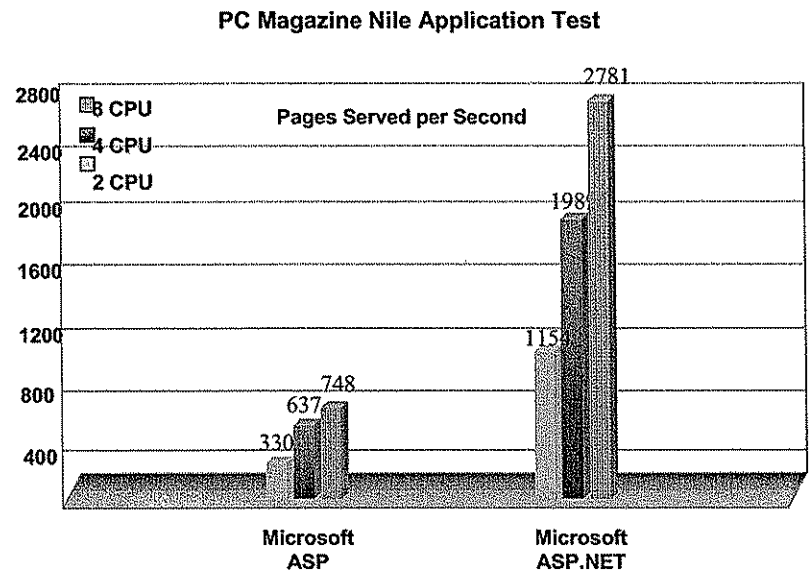


Figure 13 - Comparatif des performances ASP et ASP.NET

3.3.2 Pré-compilation et gestion du cache

L'amélioration de la performance des applications ASP.NET est liée à un certain nombre de facteurs, qu'il est important de comprendre pour bien discerner la valeur ajoutée du Framework. Les pages ASP.NET, à la différence de leurs ancêtres les pages ASP, sont compilées lors de leur première version sous forme d'assemblies (les remplaçantes des DLL, dont nous avons déjà parlé).

Autrement dit, dès le second appel d'une page ASP.NET, la performance est identique à celle d'un objet compilé. C'est une amélioration majeure.

Mais l'amélioration de la performance tient pour beaucoup à la mise à disposition d'un ensemble de solutions de « cache ». Le principe du cache est de ne pas recalculer un élément si celui-ci a déjà été demandé antérieurement, pour le délivrer directement.

Le cache qui était partiellement mis en œuvre en ASP a été totalement revu. Il permet d'agir avec une granularité très fine et reste compatible avec le déploiement de fermes de serveurs, sans développement complémentaire.

Le gain de temps pour le développeur est important. A la clé, des gains de performance très sensibles.

Le cache peut-être décomposé en plusieurs sous-ensembles, rassemblés sur le schéma Figure 14.

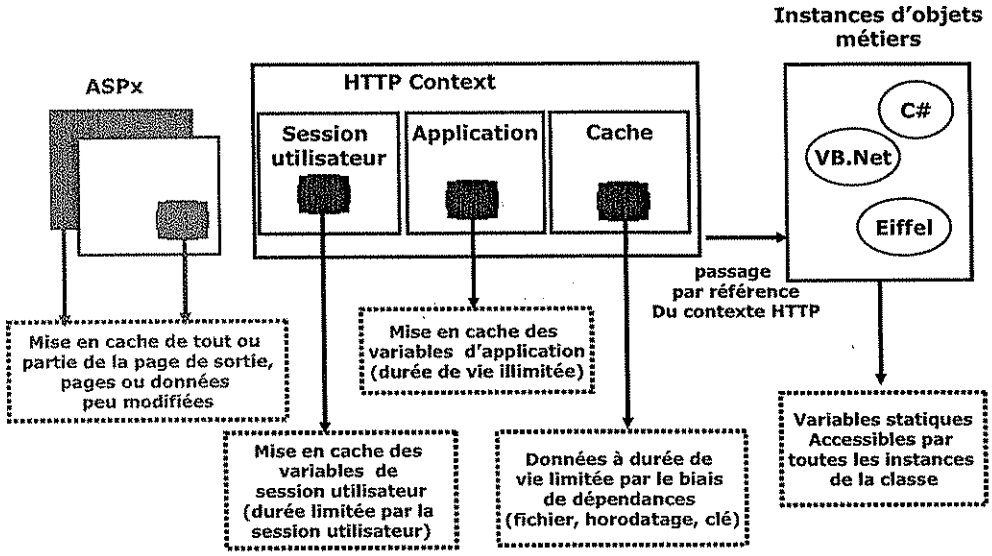


Figure 14 - Les mécanismes de cache associés à ASP.NET

Le cache de page : il permet, pour tout ou partie de la page, de « cacher » le résultat envoyé au client comme s'il s'agissait d'une page statique. Un certain nombre de paramètres permettent de définir dans quelles conditions il faut rafraîchir le résultat renvoyé.

Le cache applicatif : ce sont les objets Session et Application (qui existaient déjà mais qui ont été largement revus, notamment pour supporter le load-balancing et les fermes de serveur), et surtout l'objet Cache. Ce dernier permet de stocker des données ou des objets à durée de vie limitée, l'expiration du cache associé pouvant s'appuyer sur des priorités, des dépendances entre objets, des règles définies par le développeur, etc. L'objet cache supporte les fermes de serveurs et les accès multi-threadés, de manière transparente pour le développeur.

Les variables statiques de classe : il s'agit d'un mécanisme fort intéressant, mais que nous ne détaillerons pas ici ; il reste à manipuler avec prudence, notamment pour gérer les accès multi-threadés.

Le framework met à disposition des développeurs des outils de cache très puissants, qui demandent toutefois une certaine pratique pour être mis en œuvre correctement et dégager toute leur valeur ajoutée.

3.4 LE DEVELOPPEMENT « CLIENT-SERVEUR DEUX TIERS ET TROIS TIERS »

3.4.1 Un maître mot : productivité

Un effort important a été fourni par Microsoft, tant au niveau du framework que de Visual Studio, pour améliorer la productivité du développeur. Le résultat est assez probant, en particulier pour le développement d'applications trois-tiers (web).

3.4.2 Le fossé entre le développement client-serveur et le développement web se réduit fortement

Le couple Framework.NET / Visual Studio .NET permet en effet d'avoir une approche du développement quasiment similaire, que l'on réalise une application « client lourd » ou « client web ».

En particulier, les développeurs habitués au client-serveur pourront aborder beaucoup plus facilement le développement Web, en retrouvant une démarche et une productivité pratiquement identiques.

Cette similitude tient pour beaucoup à l'adoption au niveau du développement Web de concepts qui ont fait leur preuve dans le client-serveur :

Le développement visuel (WebForms) : le développeur positionne des contrôles (comme il le faisait autrefois avec des contrôles VB par exemple) sur l'interface utilisateur, par drag and drop. Ces contrôles disposent de propriétés directement modifiables via un inspecteur de propriétés. Un certain nombre de ces contrôles sont orientés données (datagrid, etc.), leur alimentation étant gérée automatiquement. Ces composants vont au final générer un code qui sera compatible avec l'interface utilisateur (HTML, WML, etc.) : ils s'auto-adaptent à l'environnement utilisateur dans lequel ils évoluent. Si Microsoft fournit un certain nombre de « contrôles serveurs » en standard, de nombreux éditeurs ont également migré leurs composants client-serveur vers le développement WebForms. Les développeurs ont naturellement la possibilité d'implémenter leurs propres contrôles.

Ainsi, une partie importante du temps consacré dans le développement web à l'interface utilisateur (IHM) et à l'implémentation de contrôles orientés données se fait désormais par drag and drop et par paramétrage visuel de composants. L'essentiel du temps développeur est désormais concentré sur le développement « métier » où il a la plus forte valeur ajoutée.

La Figure 15 permet d'illustrer ce propos avec le datagrid :

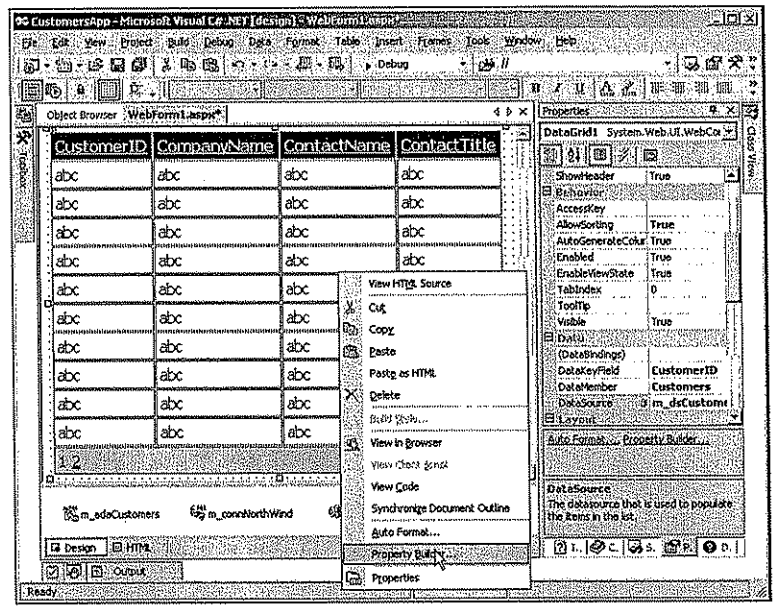


Figure 15 - Proactivité des contrôles serveurs : l'exemple de la datagrid

Le modèle événementiel : Microsoft a implémenté un modèle événementiel proche de celui du client-serveur, mais adapté au développement trois tiers.

Ce modèle permet d'attacher aux contrôles vus ci-dessus des événements déclencheurs d'actions (clic, sélection d'un item dans une liste déroulante, etc.), et de créer ses propres événements. Le développeur peut sélectionner rapidement un événement et implémenter le code devant être appelé lors de cet événement. Lors de l'apparition d'un événement, un aller-retour serveur permettra de traiter cet événement « server-side », cet aller-retour étant totalement transparent pour le développeur.

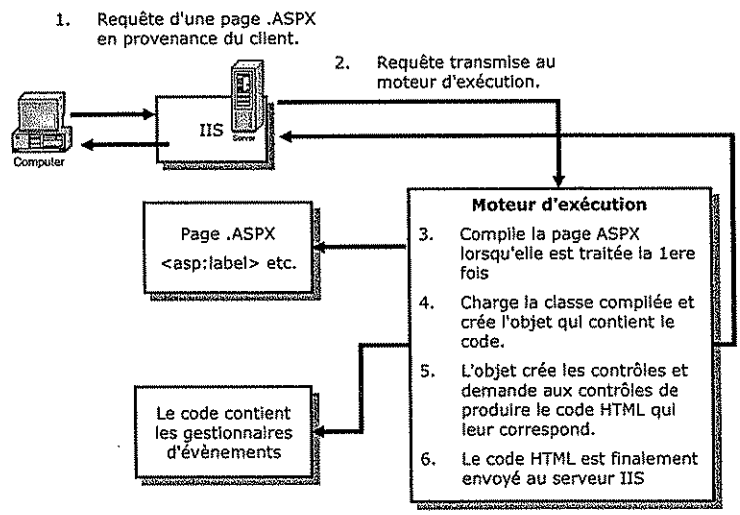


Figure 16 - Génération HTML par les contrôles serveurs et prise en compte des événements

La Figure 16 illustre à la fois le modèle événementiel et la manière dont le code HTML correspondant aux contrôles serveur est généré à la volée (remarquons qu'il est là aussi possible de bénéficier des fonctionnalités de cache).

Attention : l'utilisation répétée et incontrôlée de ce modèle événementiel (le « postback ») peut nuire aux performances. L'utilisation systématique de ce modèle événementiel est donc à déconseiller.

3.4.3 Maintenabilité des développements Web

L'un des principaux reproches faits au développement Web (que l'on adresse le monde Microsoft ou le monde Java) tient aux difficultés de maintenance – et a fortiori de réutilisation – du code écrit. La montée en puissance des langages de script a favorisé le développement facile, « vite fait mal fait ». En particulier, le code lié à l'interface utilisateur (IHM) est souvent mélangé avec le code de gestion, ce qui est contraire aux principes fondamentaux du développement. Aujourd'hui les modèles J2EE et .NET, par l'utilisation du design pattern « MVC » (modèle-vue-contrôleur) favorisent une séparation claire du code de gestion et du code de présentation.

Microsoft a notamment travaillé dans le sens de cette séparation avec le principe du « *code behind* ». Une page ASPX pourra ne contenir que le code lié à l'interface utilisateur (du HTML évolué, incluant les déclarations des contrôles serveurs), la partie événementielle du code (correspondant en fait aux appels au code de gestion) étant séparée dans une autre page (la première « héritant » de la seconde, voir Figure 17).

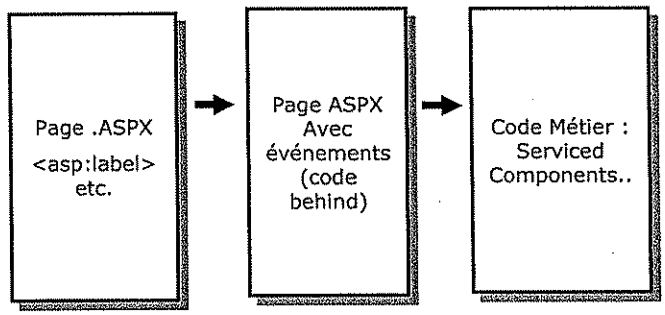


Figure 17 - Principe du code behind

Le « *code behind* » permet de séparer clairement le code de l'interface utilisateur, le code événementiel, et le code de gestion encapsulé dans des objets. Ce modèle de développement est idéal pour la maintenance.

3.5 REUTILISABILITE : DE REELLES OPPORTUNITES

La réutilisation fait partie des grands mythes de l'informatique de gestion, servis régulièrement par les éditeurs d'outils de développement.

Le développement objet, clé de la réutilisation, reste souvent un obstacle pour un certain nombre de développeurs, insuffisamment formés pour atteindre le niveau de maturation nécessaire à la réutilisation.

Pour aborder ce volet du développement, Microsoft propose plusieurs types d'approche :

La réutilisation par l'objet : les développeurs sont invités à créer des objets réutilisables, qu'ils soient techniques ou métier, notamment sur la couche serviced components. Au-delà de l'héritage, le polymorphisme est un outil puissant, souvent sous-utilisé.

La réutilisation par les composants et les contrôles : c'est celle qui a connu et connaît encore le plus grand succès, car elle reste assez simple et intuitive à mettre en œuvre : un ensemble de fonctionnalités sont packagées sous forme d'un composant, accessible via des propriétés, des méthodes et/ou des événements. Dans le cas du développement Webforms, il sera possible de créer deux types de composant :

- Les *« custom controls »* : ce sont des contrôles que les développeurs utilisent ensuite par drag and drop, depuis la palette de composants de Visual Studio. Ainsi il est possible de compléter la librairie de composants mis à disposition par Microsoft avec ses propres composants. On pourra ainsi créer des composants visuels (exemple : datagrid) mais aussi des composants dont le résultat ne sera pas visuel (exemple : des composants pour effectuer le contrôle de saisie des champs). Dans les deux cas, ces composants sont simples à réutiliser car le développeur les manipule par le biais des propriétés et des événements. Ces composants sont « templatés », c'est-à-dire que leur représentation visuelle peut facilement être paramétrée sans rentrer dans le code.
- Les *« user controls »* : ils permettent aux développeurs, très simplement, de réutiliser du code ASPX sous forme de composant, et offrent la possibilité de faire disparaître les traditionnels « include ».

Microsoft propose donc à travers le framework .NET deux types d'approches pour la réutilisation : l'approche objet et l'approche composant.

3.6 LE CYCLE LOGICIEL COUVERT DE BOUT EN BOUT

L'un des principaux reproches faits à Microsoft était la faible couverture de Visual Studio sur le cycle projet. Il s'agissait d'un outil de développement, ni plus ni moins. Face à des ateliers de génie logiciel souvent très complets, notamment dans l'univers J2EE, Microsoft se devait de proposer quelque chose qui puisse améliorer la productivité sur l'ensemble du cycle projet.

3.6.1 Conception

Sur les phases de conception, et notamment de modélisation objet, deux possibilités :

- **Visio** permet de concevoir des modèles UML et de générer les squelettes de classe associés.
- **Rational XDE**, est une alternative plus puissante, car elle permet de gérer du bi-directionnel (synchronisation permanente des modèles avec le code, possibilité de modifier le code ou le modèle et de voir l'autre partie se synchroniser). Les modèles sont visualisables et modifiables directement dans Visual Studio, sans changer d'environnement (Figure 18) :

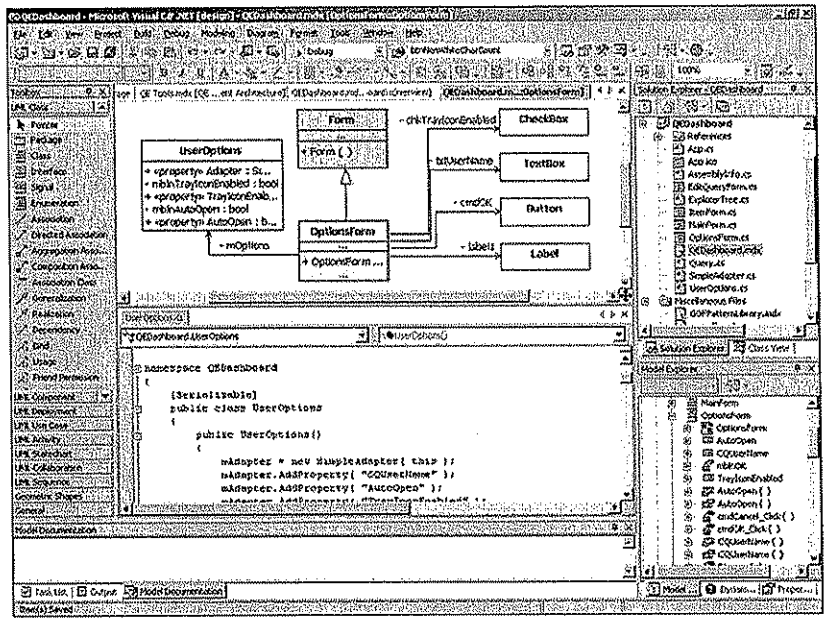


Figure 18 - Rational XDE - Source : Rational

3.6.2 Développement – EDI

Côté environnement de développement, il s'agit plus d'un aboutissement d'un savoir-faire déjà mis en œuvre de plusieurs années. L'environnement est globalement convivial, intuitif, multi-langages. Un certain nombre de fonctions sont disponibles au niveau de l'éditeur de code, notamment la possibilité d'avoir une vue arborescente du code (masquage ou affichage de portions de code en cliquant sur un nœud).

3.6.3 Debugging

C'est côté débugging d'applications Web que le travail le plus important a été réalisé. A la clé, une réelle amélioration de la productivité.

En effet, le débugging d'applications Web était souvent préhistorique (alimentation d'un fichier de log pour examiner des valeurs de variables, etc.).

Il est désormais possible de faire du débugging pas à pas (Figure 19), à travers le code source de l'application trois-tiers, en profitant de toutes les fonctions habituellement utilisables sur du debugging d'applications client-serveur (pas-à-pas, examen/modification de valeurs de variables, etc.). Il est même possible de suivre le pas-à-pas à l'intérieur des procédures stockées appelées par les objets.

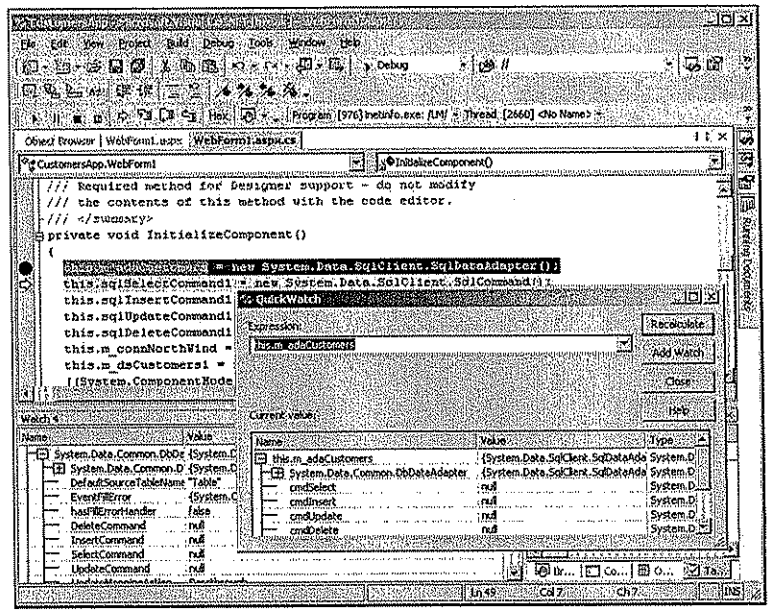


Figure 19 - Mise en oeuvre du debugging dan Visual Studio et de la gestion des traces

3.6.4 Tests

Côté tests, et sans prétendre à la richesse de solutions type Mercury ou Rational, Microsoft intègre dans son offre Visual Studio .NET l'outil Application Center Test (ACT), héritier de WAS, qui permet à la fois de faire du test fonctionnel et du test de charge, y compris sur des applications Web. Ce type d'outil se révèle particulièrement pratique dans une approche Extreme Programming.

3.6.5 Déploiement

Côté déploiement, la grande nouveauté concerne les applications trois-tiers Web (pour les applications client-serveur, un utilitaire permet de créer un setup de déploiement, très classique). En effet, les composants utilisés dans les applications ASP.NET ne nécessitent plus d'enregistrement en base de registres. C'est l'une des grandes forces du framework : les assemblies, remplaçantes des DLLs, s'auto-décrivent. Le déploiement consiste donc à placer les assemblies dans des répertoires. Ce déploiement – autre nouveauté très importante – peut se faire à chaud : lors de l'écrasement d'une assembly par une nouvelle version, les demandes en cours continueront d'être traitées avec l'ancienne, et les nouvelles seront prises en charge par la nouvelle assembly.

Ces deux points signifient des gains de productivité très importants.

3.6.6 Documentation – Maintenance

Côté documentation, Microsoft propose sensiblement l'équivalent de Javadoc (uniquement disponible en C#), avec la possibilité de personnaliser les commentaires générés. L'approche est une approche XML qui permet d'adapter à ses propres besoins le système de génération de documentation.

Toutefois, on s'orientera souvent, notamment dans le cadre d'une maintenance applicative, vers des outils comme ceux de Cast Software.

Microsoft a donc travaillé sur la productivité de l'ensemble du cycle projet. Les progrès sont très importants sur les phases de debugging et de déploiement. Côté conception, l'utilisation de Visio ou Rational XDE peut accélérer la transition de la modélisation vers le développement.

3.7 LES « PROGICIELS ADAPTABLES »

Au-delà des gains de productivité amenés par le framework .NET, l'une des clés du retour sur investissement d'un passage à .NET est l'intégration entre le « développement Framework .NET » et les progiciels Microsoft (Commerce Server, Biztalk, la future offre ERP de Microsoft, etc.). Nous renvoyons à ce titre le lecteur à la première section de ce document.

Ce que nous laissent entrevoir des produits comme CMS 2002, c'est la possibilité de profiter des avantages du progiciel ET du développement sur mesure : les progiciels Microsoft deviennent utilisables sous forme de briques, intégrables dans un développement spécifique léger, mais beaucoup plus puissant et plus personnalisable qu'un simple paramétrage. Reste à voir si ce pari sera réussi pour l'ensemble de la gamme progicielle Microsoft.

4 METTRE EN OEUVRE .NET DANS LE SI

4.1 INTRODUCTION

Après avoir présenté les avantages d'un passage à .NET, nous allons maintenant nous intéresser au « comment » : comment faire pour gérer l'existant, comment migrer, comment tirer parti des avantages apportés par .NET pour créer de la valeur au sein du système d'information.

Nous verrons dans la dernière section de ce document quelques retours d'expérience significatifs qui permettent de mieux matérialiser les pratiques évoquées ici.

4.2 GERER L'EXISTANT

La gestion de l'existant est bien sûr le point qui doit être abordé avant toute autre considération. Il serait vain de vouloir tirer partie des avantages de .NET si l'on doit pour cela se couper de l'existant. Cet existant peut être à la fois Microsoft (VB6 par exemple, architectures DNA) ou J2EE.

Deux options sont possibles : jouer la carte de l'interopérabilité ou celle de la migration. Dans le cas d'un existant Microsoft, la migration sera à terme incontournable, même si elle peut être décalée à court terme en jouant l'interopérabilité.

4.2.1 Interopérabilité DNA / .NET

Pour ce qui est de l'interopérabilité entre un existant Microsoft « DNA » (comprenons COM, ASP...) et des applications .NET, il est possible de s'appuyer sur des ponts proposés entre les deux mondes par Microsoft.

Ces ponts entre les deux mondes adressent essentiellement l'interopérabilité entre COM+ et le code managé .NET

En effet, un composant .NET pourra appeler un objet COM+ par le biais d'un « wrapper » (enveloppe) , le runtime callable wrapper (RCW) :

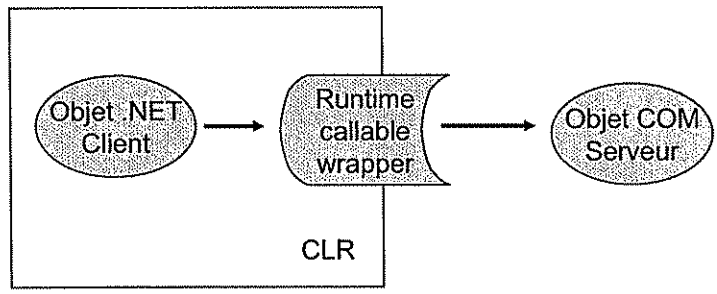


Figure 20 - appel d'un objet COM depuis un objet .NET

Le client .NET peut ainsi accéder à d'autres objets .NET ou à un serveur COM (non managé) par le biais de ce wrapper.

Réciproquement, un objet COM pourra accéder à un objet .NET par le biais d'un wrapper, le COM Callable Wrapper.

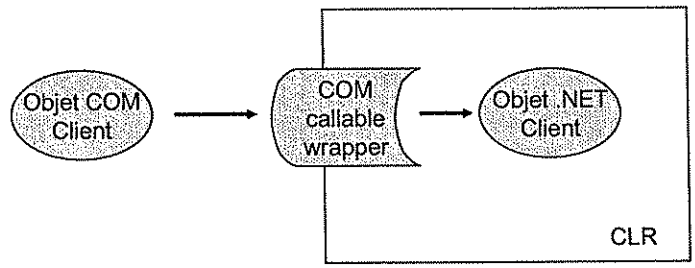


Figure 21 - Appel d'un objet .NET depuis un objet COM

L'impact du passage par un wrapper en termes de performance reste assez faible, pourvu que les appels ne soient pas permanents (exemple : lecture très fréquente d'une valeur de propriété par le biais d'un wrapper).

Il est également possible d'appeler des fonctions de DLLs (écrites en C par exemple) depuis .NET, grâce au PInvoke.

Si les composants peuvent ainsi facilement discuter entre eux, le dialogue entre des pages ASP et ASPX est nettement moins simple. Les deux peuvent coexister, mais ne se voient pas (les sessions et les variables associées ne sont pas visibles d'un monde à l'autre). L'interopérabilité entre des applications ASP et ASP.NET devra donc passer par des appels à des objets, des cookies, des sessions stockées en formulaire, ou des données échangées par l'intermédiaire du serveur de base de données.

Ainsi, l'interopérabilité entre COM+ et .NET reste très bonne. En revanche, le dialogue entre applications ASP et ASP.NET nécessite du travail, du fait du cloisonnement des sessions. Les préconisations Microsoft concernant le développement Web en s'appuyant sur des objets COM+ et des sessions gérées en base s'avèrent donc payantes dans le cas d'un passage à .NET.

Une autre approche pour gérer l'interopérabilité consiste à encapsuler des fonctionnalités existantes sous forme de web services. Cette approche reste identique à celle adoptée dans le cas d'un existant J2EE, nous la détaillons dans le paragraphe qui suit.

4.2.2 Interopérabilité J2EE - .NET

La coexistence des deux architectures est désormais incontournable. Elle ne doit pas nécessairement être vue comme un « fardeau », elle présente l'avantage de ne pas mettre tous ses œufs dans le même panier, tout en restant sur des standards.

Si un certain nombre de middlewares passerelles ont vu le jour au cours des dernières années (passerelles CORBA – COM par exemple), il nous semble

important de jouer la carte de la prudence. Ces middlewares sont généralement proposés par des éditeurs à la surface financière assez faible et à la pérennité souvent incertaine.

On préférera aux middlewares bas niveaux et propriétaires une solution appuyée sur des échanges XML, si possible par le biais de web services.

La démarche reste assez simple : elle consiste à poser une enveloppe web services (« wrapper ») au dessus de fonctionnalités métier. Ce sont donc des méthodes de « serviced components » côté .NET et d'EJB côté J2EE qui seront exposées à l'autre monde par le biais de web services. Ces web services sont généralement faciles à implémenter, les éditeurs fournissant de part et d'autre des assistants ou des attributs adaptés.

Mais si l'implémentation technique de web services reste assez simple, les choix d'architecture associés (granularité des web services, web services techniques ou fonctionnels) demandent de l'expérience et sont fortement structurants, notamment en matière de performance.

4.2.3 Migration applicative : les étapes

Alternativement à l'interopérabilité, il est possible de s'orienter vers une migration applicative, en particulier si l'on dispose d'un existant Microsoft vieillissant.

La migration d'application est un exercice qui demande une bonne maîtrise de .NET et de la technologie préexistante.

Sans présenter l'ensemble des alternatives – qui dépassent le cadre de ce document – il est possible d'adopter deux grandes familles de stratégies :

On pourra s'orienter vers des migrations « horizontales » ou « verticales ». Dans les deux cas, il est essentiel d'adopter une démarche de maîtrise des risques pro-active (validation de pilote / exploration des points critiques très en amont).

4.2.3.1 Migration horizontale

La migration horizontale consiste à migrer un « tiers » de l'application dans son ensemble. Ainsi, pour une application web écrite à base de pages ASP et d'objets COM, la couche métier étant encapuslée dans les objets COM et la présentation dans les pages ASP :

On pourra choisir de réécrire d'abord la couche présentation en ASPX (webforms), en appelant les objets COM+ à travers des « wrappers » .NET.

Réciproquement, on pourra réécrire les objets COM+ sous forme d'objets .NET et les faire apparaître comme des objets COM+ pour les pages ASP.

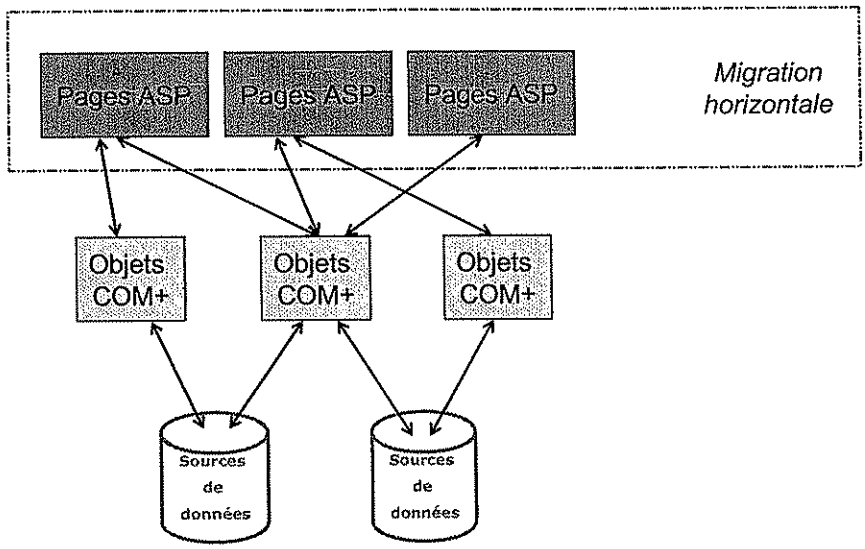


Figure 22 - Principe d'une migration horizontale

Ces migrations, simples sur le papier demandent de l'expertise technique et des choix assez forts (exemple : si les pages ASP utilisaient des feuilles de style XSL, faut-il conserver cette approche XML/XSL ou basculer dans une philosophie complètement webforms à base de contrôles serveurs).

4.2.3.2 Migration verticale

La migration verticale consiste à migrer l'ensemble des couches (présentation et métier) mais simplement pour une partie des fonctionnalités de l'application.

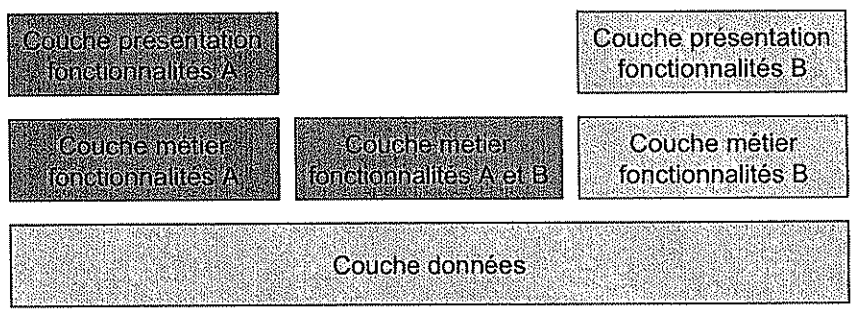


Figure 23 - Principe d'une migration verticale

La complexité consiste dans ce cas à gérer les relations entre les deux parties de l'application qui ne reposent plus sur les mêmes technologies. En particulier pour des applications Web, la session ASP ne pouvant être partagée avec la session ASP.NET.

4.2.3.3 *Maîtrise des risques*

Dans tous les cas la migration doit débuter avec la mise en place d'un plan de maîtrise des risques technologiques. On surveillera notamment tous les points de « frottement » entre les parties à migrer et les parties qui subsistent, et on vérifiera la qualité des interfaces par le biais de prototypes.

4.2.3.4 *Passer par une phase intermédiaire*

Dans le cas où l'application existante DNA ne respecte pas des standards de développement en matière de séparation de couche (présentation, métier, données), il peut être intéressant de remanier l'application existante (refactoring) pour la préparer à une migration, qu'elle soit horizontale ou verticale.

4.3 CONSTRUIRE SUR DU SOLIDE

Les systèmes d'information des années 90 ont été bâtis avec des outils et des architectures souvent contestables. Les faiblesses de Visual Basic, (notamment le fait qu'il ne soit pas orienté objet), mais aussi un développement web construit au départ sur un langage de script (ASP) ont contribué à créer des « zones d'ombre » (pour ne pas dire des marécages) dans le système d'information. Code spaghetti, absence d'architecture, code d'interface mélangé au code de gestion, etc.

Aujourd'hui les outils et l'architecture mis à disposition par Microsoft permettent de faire de l'applicatif « propre ». La balle est désormais dans le camp de la maîtrise d'ouvrage et de la maîtrise d'œuvre. Et les choses ne seront pas simples. Car définir une architecture applicative solide reste un exercice complexe : objets techniques, objets fonctionnels, granularité des services offerts...autant de choix difficiles à faire, même si la technologie facilite le travail.
La mise en place de normes et des guidelines de conception et de développement joue un rôle capital pour construire sur du solide.

4.4 PROFITER DES « PROGICIELS ADAPTABLES »

Ce n'est pas parce que les outils de développement fournis par Microsoft sont puissants qu'il faut partir sur du 100% sur-mesure.

Les progiciels proposés par Microsoft vont permettre de remplacer des pans entiers de développement spécifique par de l'intégration progicielle. L'approche « framework » de ces progiciels permet d'avoir une solution au finale très intégrée, mixant le meilleur du progiciel et du développement spécifique. Le coût des licences associé obligera cependant à un certain nombre d'arbitrages.

4.5 WEB SERVICES ET EAI

Nous consacrons désormais dans la nouvelle version de ce livre blanc une section entière à l'EAI et aux Web Services, (Partie 7).

4.6 QUELS OUTILS DE DEVELOPPEMENT – QUELS LANGAGES

Côté conception, développement et maintenance, un passage à .NET peut être l'occasion de se poser des questions quant aux outils à mettre en œuvre. Voici quelques pistes.

4.6.1 Doit on obligatoirement passer à C# ou VB.NET ?

Le passage à un environnement de développement Visual Studio.NET avec comme langage C# ou VB.NET constitue bien entendu une possibilité intéressante, en particulier lorsque l'on doit évoluer à partir d'un historique VB ou C++. En revanche, si l'existant logiciel est dans un autre langage (Delphi par exemple), on pourra surveiller la sortie des nouvelles versions des outils de développement associés. Il est probable qu'un certain nombre d'entre eux (c'est le cas de Borland) ont planifié la sortie d'un compilateur permettant de générer de l'Intermediate Langage compris par la CLR.

```
<configuration>
  <system.web>
    <compilation debug="true">
       <assemblies>
          <add assembly="DelphiProvider" />
       </assemblies>
       <compilers>
          <compiler language="Delphi" extension=".pas"

type="Borland.Delphi.DelphiCodeProvider,DelphiProvider" />
       </compilers>
    </compilation>
  </system.web>
</configuration>
```

```
<html>
    <script language="Delphi" runat="server">
    procedure ButtonClick(Sender: System.Object; E:
EventArgs);
    begin
        Message.Text := Edit1.Text;
    end;
    </script>
    <body>
    <form runat="server">
        <asp:textbox id="Edit1" runat="server"/>
        <asp:button text="Click Me!" OnClick="ButtonClick"
runat="server"/>
    </form>
    <p><b><asp:label id="Message" runat="server"/></b></p>
    </body>
</html>
```

Figure 24 - Exemple de code ASP.NET utilisant le langage Delphi. Le fichier de configuration du serveur Web permet de préciser quel sera le langage utilisé (preview de Delphi.NET)

La conservation d'un langage historique (en particulier s'il est orienté objet et s'il a fait ses preuves) peut faciliter un passage à .NET

4.6.2 Entre VB.NET et C#, que choisir ?

Il est tentant de migrer vers du VB.NET, en particulier lorsque les équipes de développement sont familières de VB. Néanmoins il ne faut pas se leurrer ; le fossé est pratiquement aussi important entre VB et VB.NET qu'entre VB et C#. Implémentation des concepts objets, gestion propre des erreurs, framework .NET : autant de nouveautés qu'il faut intégrer par de la formation. Ce serait un leurre de croire que la connaissance d'une syntaxe suffit à appréhender autant de nouveaux concepts.

Un passage à C# offre plusieurs avantages :

- C# est le plus abouti des langages orientés objets dédiés à la plate-forme .NET. C'est aussi le langage sur lequel Microsoft misera le plus dans les années à venir
- Un passage à C# est l'occasion de couper net avec des mauvaises habitudes et de démarrer sur de bonnes bases
- VB.NET présente encore des inconvénients assez importants (notamment sur des aspects de transtypage). Il ne supporte pas la gestion paramétrable de la documentation automatique du code.

Même si un passage de VB à VB.NET peut être tentant, un passage à C# est l'occasion de rompre avec les mauvaises habitudes. Attention toutefois à apporter la formation et le soutien technologique nécessaire.

4.6.3 Quels outils pour la conception ?

Pour la conception, plusieurs outils sont possibles :

- Pour ce qui est de la modélisation de données, Visio propose une solution, qui reste toutefois éloignée du formalisme Merisien. La plupart des entreprises françaises préféreront nous semble-t-il, conserver des outils type PowerAMC.
- Pour ce qui est de la modélisation objet, Visio sera intéressant si l'on s'oriente vers une modélisation macroscopique initiale, donnant lieu à la génération de squelettes, enrichis ensuite sans synchronisation des modèles. Si en revanche on s'oriente vers une synchronisation permanente des modèles, l'utilisation de Rational XDE devient intéressante.

4.6.4 Quels outils pour le développement ?

Pour le développement, Visual Studio est un environnement convivial et productif pour les langages poussés par Microsoft. Néanmoins il est possible de s'orienter vers une alternative Open Source, constituée des produits SharpDevelop et WebMatrix.

- SharpDevelop est un environnement de développement intégré, comprenant coloration syntaxique, intégration avec des outils de test open source type NUnit, intégration avec CVS pour la gestion de la configuration.
- WebMatrix est un éditeur visuel de pages ASPX, plutôt bien réussi, même s'il ne bénéficie pas de fonctionnalités type debugging, etc.

Si le rôle de chaque développeur est bien spécifié, il peut être intéressant de déployer un parc mix Visual Studio.NET / SharpDevelop-WebMatrix, un atelier étant utilisé plutôt que l'autre en fonction de la complexité des tâches à accomplir. Un moyen intéressant de baisser le coût des licences, même si l'on risque de perdre un peu en productivité.

4.6.5 Quels outils pour les tests ?

Pour les tests, on pourra s'orienter vers plusieurs types d'outils, en fonction des besoins – et des budgets.

- Nunit est l'outil de test unitaire issu de la vague extreme programming. C'est un bon outil, même s'il demande de fait au développeur beaucoup de travail.
- Application Center Test (la solution de test de Microsoft intégrée à Visual Studio.NET) permettra par contre d'effectuer des tests fonctionnels poussés sans écrire de code, ainsi que des tests de montée en charge.
- Pour des tests plus évolués, notamment concernant la montée en charge et l'analyse des goulets d'étranglement associés, des outils tels que ceux proposés par Mercury ou Rational sont plus adaptés.

4.6.6 Quels outils pour la maintenance ?

Des outils de gestion de configuration (qu'ils soient open source comme CVS ou payants comme Source Safe ou les outils de Rational) s'avèrent sur ces phases de maintenance encore plus utiles que sur les phases de développement.

Enfin le recours à un outil comme celui de Cast Software, permet de cartographier le code, d'analyser les dépendances, de mesurer les impacts de modification sur le code.

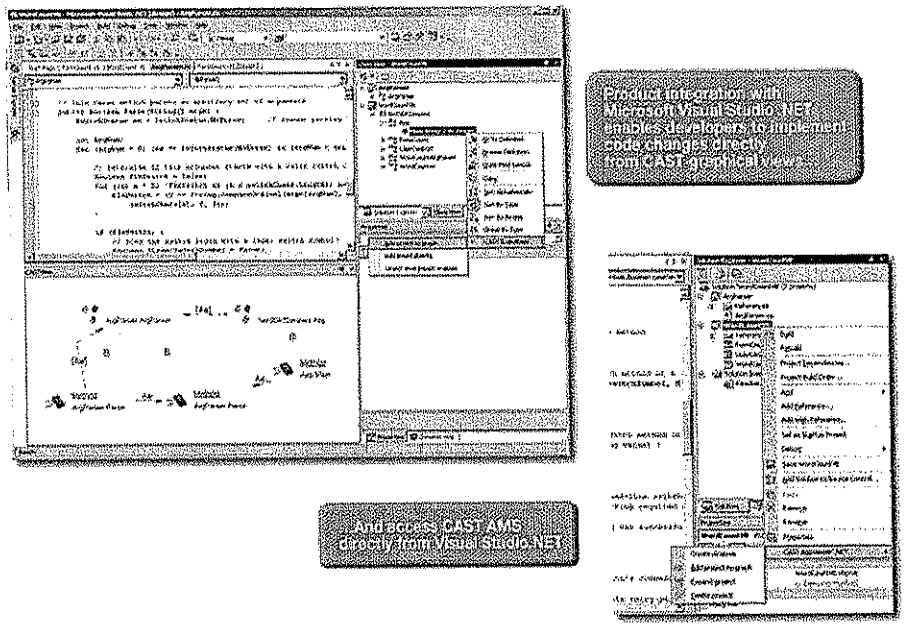


Figure 25 - Cast Application Mining Suite pour Visual Studio .NET – Source CAST

4.7 SIDEWINDER, LA CONTRE-ATTAQUE DE BORLAND

Borland prépare aujourd'hui la sortie de son produit destiné à .NET, dont le nom de code est actuellement SideWinder.

Nul doute que les ambitions de Borland sur ce marché sont importantes.

Borland sera probablement le seul éditeur à proposer une alternative sérieuse et crédible à Visual Studio.NET.

On peut naturellement s'interroger sur l'opportunité de travailler en C# avec l'outil de Borland plus qu'avec l'outil de Microsoft. La réponse est probablement à chercher du côté de la capacité de l'outil à appréhender l'ensemble du cycle de vie de l'application. Aujourd'hui Microsoft dépend fortement de Rational pour la partie conception UML (Rational XDE), et le rachat de Rational par IBM ne va pas forcément dans le bon sens pour Microsoft – en dépit du message rassurant envoyé par les dirigeants de Rational.

Borland, de son côté, et notamment avec le rachat de Together, maîtrise l'ensemble du cycle projet : gestion des exigences Projet, Conception, Modélisation, Tests, Optimisation du Code, Déploiement...en offrant par ailleurs un niveau d'interopérabilité élevé avec le monde J2EE.

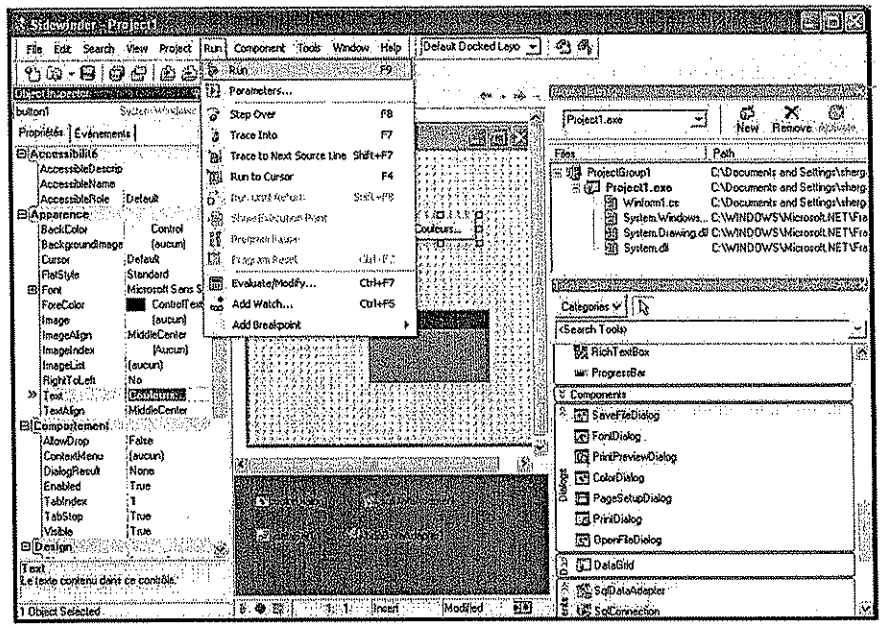


Figure 26 - Environnement de développement Sidewinder - Source : Borland

Pour les clients historiques de Borland, Sidewinder représente une forte opportunité de passer « en douceur » à .NET, que ce soit par le support du langage Delphi ou par les outils de migration disponibles.

5 RECOMMANDATIONS D'ARCHITECTURE

A la différence de l'architecture DNA, Microsoft a accompagné la sortie de .NET d'un certain nombre de documents d'architecture et de bonnes pratiques concernant la réalisation d'applications .NET. De nombreux documents sont disponibles en ligne sur les sites Microsoft. Nous avons souhaité dans ce document donner une vue synthétique des points qui nous paraissent essentiels. Par ailleurs un certain nombre de choix d'architecture sont laissés à l'appréciation des équipes de développement ; nous présentons ici les choix qui nous paraissent de manière générique les plus pertinents, il est essentiel toutefois d'adapter les solutions présentées ici au contexte de l'application.

Si la plupart des applications ont convergé vers un modèle séparant les couches présentation / métier / accès aux données, force est de constater que les interprétations de ce modèle sont nombreuses.

Microsoft propose désormais un modèle cohérent et détaillé pour la mise en œuvre d'applications multi-niveaux .NET.

Pour les besoins du document, nous présentons ici une version simplifiée de l'architecture. N'hésitez pas à nous contacter pour plus d'informations.

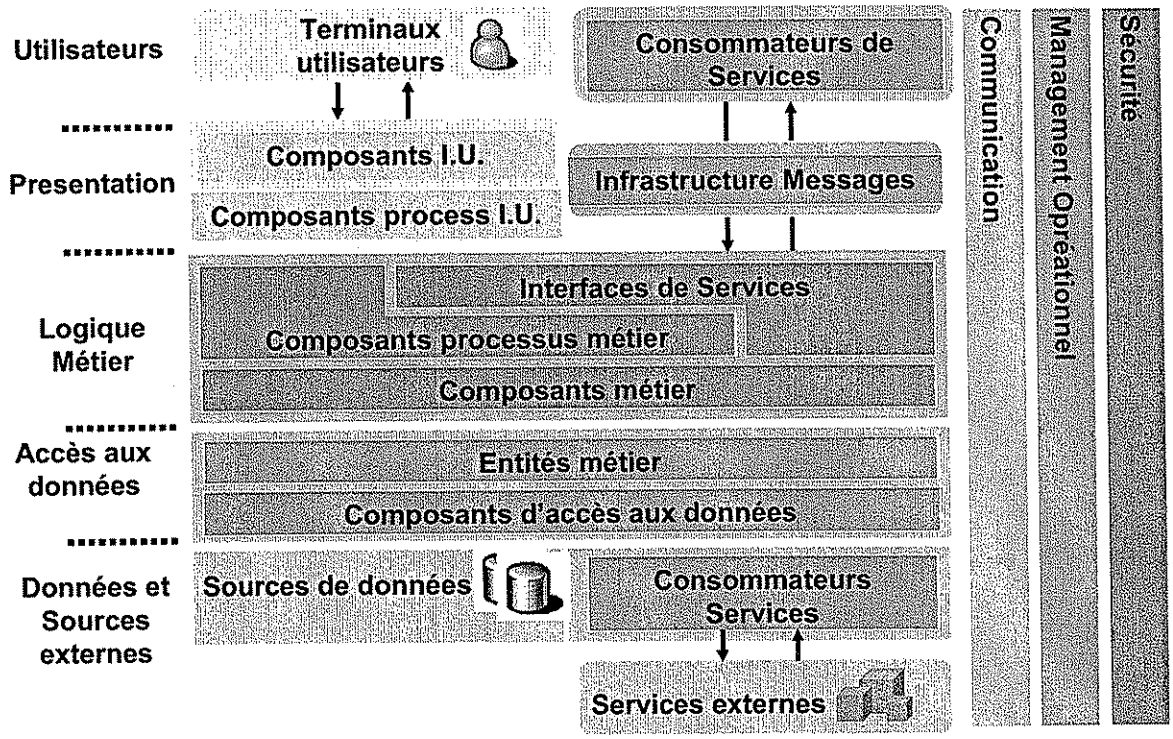
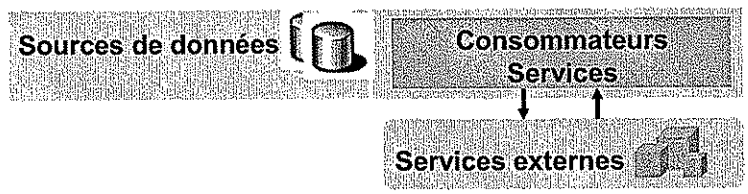


Figure 27 - Architecture applicative - Source : Microsoft (traduction Business Interactif)

Ce modèle est mis en œuvre sur les applications développées par les équipes Business Interactif. Nous présentons ici les différentes couches, ainsi que les modes de mise en œuvre qui nous semblent les plus pertinents.

5.1 COUCHE DONNEES



La couche de données est composée :

- de bases de données « traditionnelles », qu'elles soient relationnelles, sous formes de fichiers, etc.
- de services Web qui vont fournir à distance des ensembles de données, et qui vont être « consommés » par l'application.

La notion de source de données devient donc une notion complètement étendue et distribuée.

5.2 COUCHE D'ACCES AUX DONNEES

La couche d'accès aux données est une couche essentielle, souvent polémique par rapport aux modèles EJB Entities. L'approche Microsoft de la persistance relationnelle-objet est souvent considérée comme légère par rapport à celle de J2EE – à tort nous semble-t-il.

Les entités métier proposées par Microsoft peuvent être de plusieurs nature. Pour simplifier, deux grandes catégories se dégagent :

- Des objets qui mappent la base de données relationnelle (que ce soit à l'aide de frameworks de type Objectspaces ou non).
- Des ***datasets***

Cette deuxième approche nous paraît essentielle et fondamentale, car elle résout la plupart des problèmes liés à la persistance.

Il est en effet possible de créer des entités métier qui seront des datasets, typés, avec une signification métier indépendante de la structure réelle de la base de données sous-jacente.

Ainsi pour une entité Client on pourra définir un type de Dataset composé d'une table ayant par exemple des « colonnes » ID, Nom, Prénom, Adresse alors que la base de données elle-même ne dispose pas nécessairement de ces champs. La définition du type du Dataset est généralement effectué sous la forme d'un schéma XML (nous renvoyons le lecteur au paragraphe 3.2 pour une présentation des capacités du Dataset).

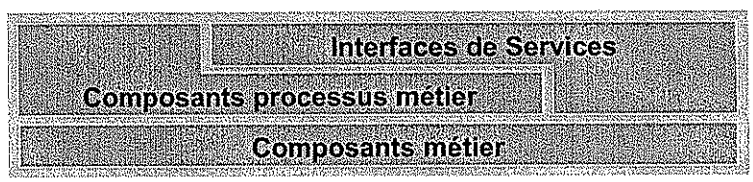
Les composants d'accès aux données sont quand à eux chargés d'alimenter ces datasets typés avec les données réellement en base, par le biais de procédures stockées.

Quel est l'intérêt d'utiliser des datasets métier plutôt que des objets ? En fait les avantages sont multiples :

- Lorsqu'il faut manipuler une liste d'objets entités, plutôt que de créer autant d'instances en mémoire (souvent pénalisant en termes de performances), un seul objet est créé : une instance de dataset...avec autant d'enregistrements qu'il y a d'objets. Lorsqu'on manipule une seule entité, c'est un dataset à une seule ligne qui est manipulé.
- Lorsque le dataset est typé, il est possible d'accéder à chacun des champs des enregistrements par le biais d'accesseurs qui lui donnent une interface objet.
- Lorsqu'il faut gérer des relations 1..n (exemple : on souhaite rapatrier dans l'entité Facture l'ensemble des lignes de la facture), le dataset est particulièrement adapté puisqu'il est capable de supporter plusieurs tables et des relations entre ces tables.
- Les datasets métier sont directement pris en charge par les contrôles visuels orientés données, ce qui évite toute transformation supplémentaire et coûteuse (voir 5.4).

Nous insistons ici sur le fait que les datasets en question sont des datasets métier, ils n'ont pas vocation à nécessairement mapper des tables de la base de données

5.3 COUCHE LOGIQUE METIER



La couche logique métier est la couche qui a en charge la manipulation des entités métiers sur lesquelles des opérations « métier » sont effectuées.

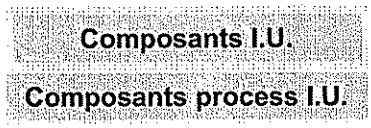
L'exemple classique est le transfert d'argent d'un compte à un autre : les objets « Compte » sont des entités, l'objet « Transfert » est un composant métier.

Les « Composants métier » dialoguent avec la couche supérieure (l'interface utilisateur que nous allons voir dans le prochain paragraphe) par le biais de datasets métier. Ils peuvent aussi bien jouer le rôle de pont entre l'interface utilisateur et les entités, que récréer eux-mêmes des datasets métier (par exemple par agrégation de plusieurs entités). Ce sont ces composants qui contiennent la logique applicative à proprement parler.

Les « Composants processus métier » prennent en charge des « workflows » métier. Ils sont chargés de l'orchestration des différents composants métier. Dans des cas complexes, il sera utile de remplacer ces composants processus par une orchestration Biztalk.

Les « Interfaces de services » ont pour rôle d'exposer les fonctionnalités de la couche métier pour des accès variés, par exemple exposer les fonctionnalités sous forme de web services. IL s'agit d'une surcouche supplémentaire qui pourra effectuer des transformations adaptées à un dialogue de telle ou telle nature.

5.4 COUCHE PRESENTATION



La couche présentation est celle qui gère le dialogue avec les utilisateurs. Elle peut dialoguer avec la couche métier de manière « native » ou en passant par la couche « Interfaces de services » selon les besoins.

Les « Composants I.U. » (pour interface utilisateur) sont les composants visuels, généralement des pages ASPX avec le code behind activé (l'appel à la couche métier se faisant depuis le code behind). Ils intègrent des composants visuels orientés données, fournis par Microsoft, capables d'interpréter directement des datasets métier (exemple : la datagrid permet d'afficher directement sous forme de tableau une table de dataset).

On comprend ici tout l'intérêt de gérer les Entités sous forme de datasets métier : l'affichage ou l'alimentation des datasets depuis l'interface utilisateur ne demande aucune retransformation et exige très peu de code.

Les « Composants Process I.U. » vont être mis en œuvre lorsque l'interface utilisateur contient des workflow utilisateurs assez complexes. Ces composants gèrent en quelque sorte l'orchestration côté interface utilisateur.

Ces composants sont adaptés à une architecture de type « Modèle Vue Contrôleur 2 » (MVC2) qui n'est pas sans rappeler celle de Struts dans le monde Java.

5.5 COUCHES TRANSVERSES

Transversalement par rapport à ces couches, ont identifie des couches de Communication, de Management Opérationnel et de Sécurité.

Ces aspects transverses sont naturellement fondamentaux. En particulier on prêtera une attention toute particulière à la gestion de la sécurité applicative entre les différentes couches, la manière dont est géré le traitement des erreurs (Exceptions) et leur propagation entre les couches.

Ces trois points peuvent faire à eux seuls l'objet d'un document complet tellement ils sont essentiels. Pour laisser à ce document son caractère synthétique, nous avons choisi de ne pas les traiter ici. N'hésiter pas à prendre contact avec nos équipes si vous souhaitez approfondir ces aspects.

6 LES EXTENSIONS DU FRAMEWORK

Microsoft a étendu les possibilités du Framework par un certain nombre de SDKs (Software Development Kits), la plupart disponibles sur MSDN.

Mais, au delà de ces kits de développement, Microsoft propose également des boîtes à outils techniques permettant d'accélérer le développement, tout en fournissant un guideline structurant pour le développement. Ce sont les « Applications Blocks ».

Aujourd'hui deux principaux Blocks sont disponibles, mais nul doute que d'autres viendront compléter la gamme.

6.1 LE DAAB : DATA ACCESS APPLICATION BLOCK

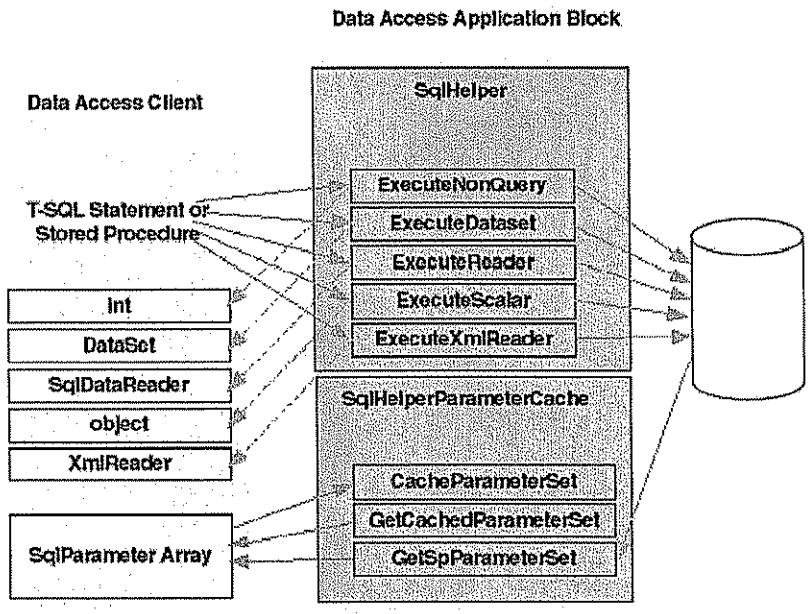


Figure 28 - Architecture du DAAB - Source : Microsoft

Le Data Access Application Block est une surcouche d'ADO.NET. Elle permet un accès simplifié aux bases de données en fournissant un « SQL Helper » et des fonctions de haut niveau, que ce soit pour travailler avec les Datasets et DataReader (ExecuteDataset, ExecuteDataReader) ou pour effectuer des requêtes ne renvoyant pas de résultat (ExecuteNonQuery par exemple).

Aujourd'hui le DAAB est disponible pour SQL Server. Business Interactif a développé une extension et intègre dans ses développements un DAAB compatible DB/2 et Oracle.

6.2 L'EMAB : EXCEPTION MANAGEMENT APPLICATION BLOCK

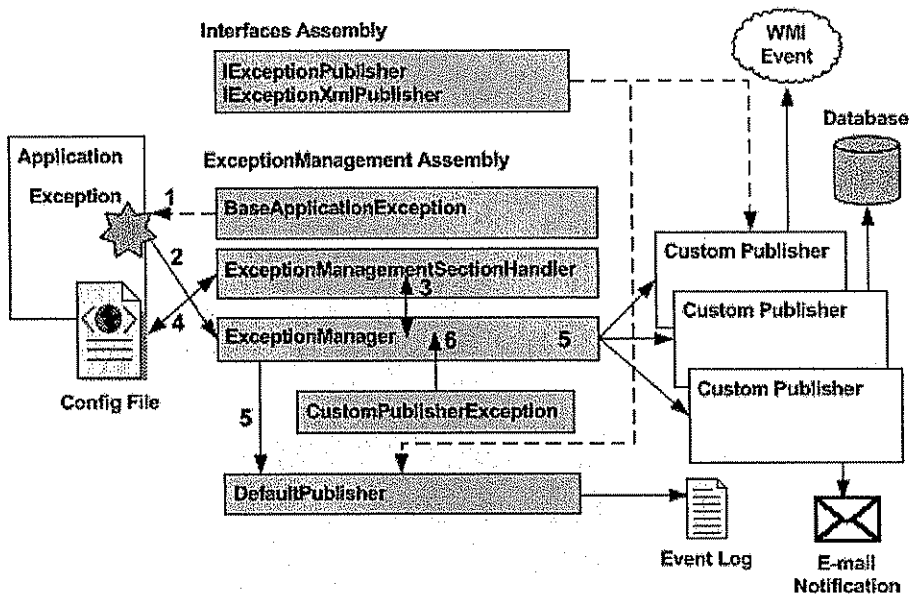


Figure 29 - Architecture de l'EMAB - Source : Microsoft

Le framework .NET fournit en standard un certain nombre de fonctionnalités puissantes pour la gestion des logs (fonction de « trace »), avec la possibilité de rediriger les logs vers des cibles différentes, par le biais de listeners.

L'EMAB fournit un framework pour la gestion et l'implémentation des exceptions. Avec peu de lignes de code, le développeur a la possibilité de tracer et de remonter les exceptions, par exemple vers l'Event Log. L'EMAB fournit également une architecture qui permet de bien séparer le code associé au traitement des erreurs du code applicatif lui-même.

6.3 POSITIONNEMENT PAR RAPPORT AUX GUIDELINES D'ARCHITECTURE

Ces « Applications Blocks » sont naturellement compatibles avec les schémas d'architecture présentés dans la section précédente de ce document.

Ils jouent en fait le rôle « d'accélérateur structurant ». Ainsi le DAAB sera mis en œuvre dans la couche « Composants d'accès aux données ».

7 EAI ET WEBSERVICES

7.1 LES WEBSERVICES : POUR QUOI FAIRE

La grande mode du moment semble bien être celle des web services. Il est vrai qu'il s'agit d'une véritable avancée pour l'interopérabilité et la collaboration des applications, point sur lequel nous reviendrons dans le chapitre suivant consacré à l'EAI et à l'orchestration de web services.

Nous avons également vu que les web services pouvaient jouer un rôle pivot dans la migration d'application, notamment si l'on adoptait une démarche de migration « verticale ».

Néanmoins nous souhaitons attirer l'attention sur une autre utilisation des web services, qui est celle de la mutualisation applicative

7.1.1 Mutualisation applicative

La mutualisation applicative par services web consiste simplement à centraliser un certain nombre de fonctionnalités usuellement redondantes dans les applications sous formes de web services « centraux » et accessibles par l'ensemble de ces applications.

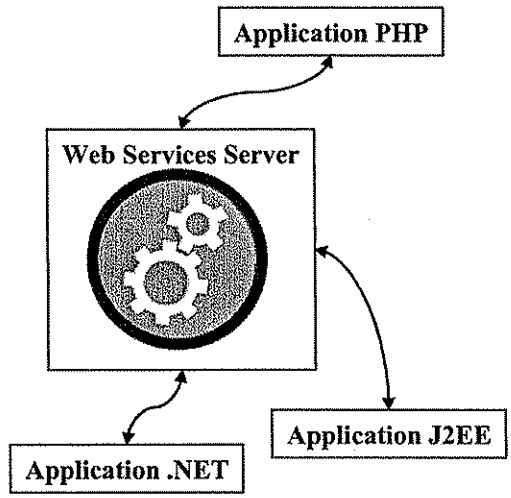


Figure 30 - Mutualisation applicative par services web

Les avantages de cette approche sont multiples :

- L'investissement sur des fonctionnalités est centralisé, cela permet d'investir plus, notamment sur des phases de test (debuggage des fonctionnalités)
- Les services offerts sont indépendants des choix technologiques des différentes applications appelantes, qui peuvent s'appuyer indifféremment sur du .NET, du J2EE ou du PHP
- Les services web correspondant peuvent être hébergés sur des plateformes centralisées, sécurisées et offrant une haute disponibilité

- Il est possible de monitorer la consommation globale des différents services et de suivre la demande réelle

7.1.2 Limites

Il faut toutefois garder en tête les limites d'utilisation de ce type de mutualisation. On veillera en particulier à ce que les aspects de performance soient pris en compte (le dialogue par protocole SOAP sur des réseaux à faible débit peut être pénalisant) ; on surveillera également la capacité transactionnelle des services proposés.

7.1.3 Importance des guidelines

Il est essentiel que le développement de ces services mutualisés respecte des règles de conception, de sécurité, de performance et de maintenabilité. Ces services sont en effet appelés à être utilisés de manière intensive, et toute erreur de conception peut s'avérer fortement pénalisante.

La constitution de guidelines de conception et de développement de services web, adaptés au contexte de l'entreprise, aide à garantir la qualité de ces services mutualisés. On veillera en particulier à définir des règles quand à la granularité des services, et leur typologie (services techniques, services métiers...).

7.2 WEB SERVICES ET .NET

L'implémentation de web services avec la plate-forme .NET est en fait assez simple à mettre en œuvre. Sans rentrer dans les détails d'implémentation, il suffit d'exposer les méthodes des objets avec l'attribut « Webmethod » pour que celles-ci soient accessibles par le biais d'un client web service. La complexité de la tuyauterie est masquée pour le développeur.

Ce travail peut être réalisé dans la couche « Interfaces de services » que nous avons présentée dans la section architecture de ce document.

Réciproquement, l'accès à des services web existants (qu'ils reposent sur une technologie Microsoft ou autre) se fait simplement, le code nécessaire à l'appel aux services distant étant généré par les outils de développement Microsoft (génération du proxy).

Notons toutefois que Microsoft propose une alternative pour le développement distribué, moins « marketing » que les web services mais tout aussi intéressante, .NET Remoting, l'héritier de DCOM.
.NET Remoting permet de manipuler simplement des objets distants, en s'appuyant sur des protocoles plus ou moins riches (http est ainsi supporté).

Ajoutons enfin qu'il est possible de manipuler facilement les entêtes des messages SOAP (SOAPHeaders) notamment pour étendre les

fonctionnalités des web services. C'est d'ailleurs le mécanisme utilisé par Microsoft pour implémenter GXA, comme nous allons le voir dans le paragraphe qui suit.

7.3 ETENDRE LES POSSIBILITES DES WEB SERVICES : GXA ET WSE

Si la mise en œuvre de web services semble simple, il faut toutefois y regarder à deux fois car l'implémentation de services intégrant des aspects sécurité, routage et transactions reste complexe.

7.3.1 GXA : Global XML Web Services Architecture

Pour essayer de clarifier les choses à ce sujet, IBM, Microsoft ont travaillé – avec d'autres – sur la mise en place d'une architecture assez générique permettant d'intégrer ces aspects et ouvrant la porte à la réalisation d'applications distribuées professionnelles, appuyées sur les Web Services : GXA. Les grands principes de GXA sont les suivants :

- **Décentralisation complète** : les web services n'ont pas vocation à être gérés par un point centralisateur, ils ont une nature complètement distribuée et ce sont plutôt les règles de namespaces et d'URI qui joueront le rôle de fédérateur
- **Modularité** : les protocoles GXA peuvent être utilisés ensemble, « en bloc » mais également pièce par pièce.
- **Modèles de données fondés sur XML**
- **Neutralité vis-à-vis de la couche transport** : GXA ne dépend en particulier pas de HTTP
- **Neutralité vis-à-vis du domaine d'application** : GXA fournit une base mais peut être décliné et adapté en fonction des contextes.

Les principales fonctionnalités offertes par GXA se regroupent autour de deux protocoles : WS-Routing et WS-Security. Le premier donne la possibilité de créer au dessus des infrastructures et des protocoles existants une sorte de « Réseau virtuel » utilisable par les web services. WS-Routing permet notamment à deux points de communiquer alors qu'ils n'ont aucun protocole sous-jacent commun. Le deuxième, WS-Security, permet d'encapsuler la sécurité au sein des messages SOAP en s'affranchissant ainsi du niveau de sécurité des réseaux et protocoles sous-jacents.

A ces protocoles essentiels il faut ajouter DIME et WS-Attachements, qui permettent de prendre en charge la gestion de « pièces attachées » au sein des messages échangés par les web services. Enfin notons également l'émergence de protocoles complémentaires comme WS-Coordination (gestion de la coordination au niveau des applications distribuées).

7.3.2 Les extensions WSE de Microsoft

Microsoft a très rapidement proposé une extension au framework .NET, permettant notamment de prendre en charge les protocoles GXA au sein des applications .NET, et ce sans révolution sur la manière d'implémenter les web services. WSE inclut donc WS-Routing, WS-Security, DIME et WS-Attachements.Le principe de mise en œuvre de WSE est extrêmement simple. Il consiste à appliquer des « filtres » en entrée ou en sortie aux messages SOAP qui sont échangés par les web services.

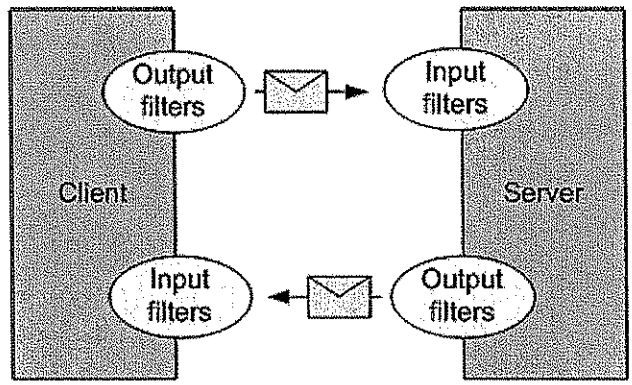


Figure 31 - Principe d'implémentation de WSE - Source : Microsoft

Le développeur dispose donc de quelques objets, simples à manipuler, qui lui donnent la possibilité d'ajouter des filtres aux messages en partance, ou au contraire d'appliquer des filtres de lecture en réception, selon un « pipeline de filtres » :

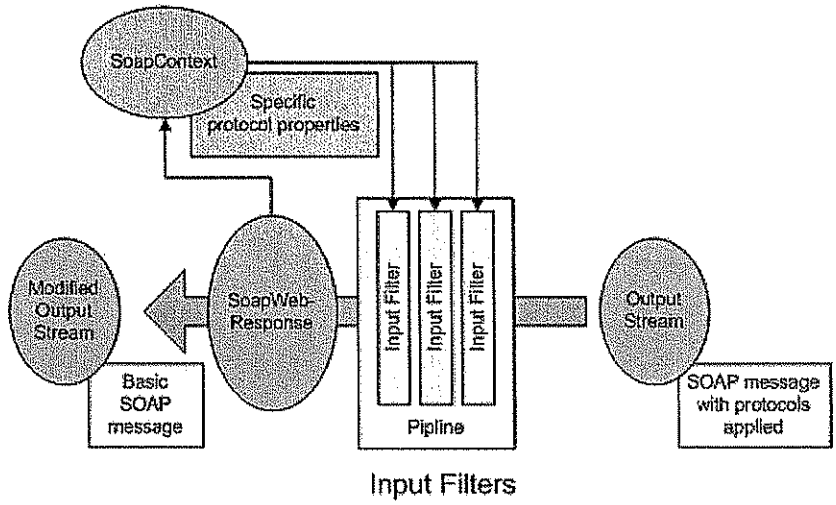


Figure 32 - Utilisation d'un pipeline de filtres pour "décoder un message" - Source : Microsoft

7.4 L'EAI

L'intégration des applications, au même titre que les services web , reste l'un des sujets de préoccupation majeurs.

Dans ce contexte, l'offre de Microsoft, Biztalk, va permettre de jouer plusieurs rôles :

- Un rôle de pivot entre applications : Biztalk va faciliter les échanges de données non seulement entre applications écrites sur mesure, mais également entre briques progicielles, Microsoft ou non. L'intégration de Biztalk et de Commerce Server permettra ainsi de gérer la synchronisation du catalogue produit de Commerce Server avec des progiciels de gestion.
- Un rôle d'orchestration applicative : Biztalk va pouvoir gérer le workflow de composants métiers, qu'ils soient sous forme d'objets COM+ ou de services Web (.NET ou J2EE). Le workflow pourra être constitué visuellement sous l'équivalent Biztalk de Visio, chacune des actions dessinées dans le workflow étant mappée avec un service web ou un composant applicatif.

En particulier, Biztalk couplé à WSE permet de jouer un rôle pivot dans l'orchestration de web services.

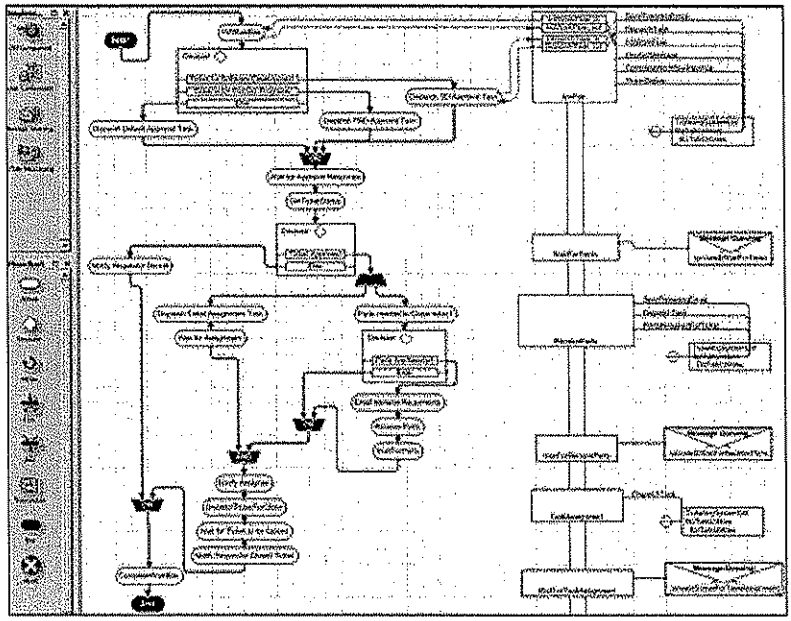


Figure 33 - Orchestration applicative avec BizTalk Server

Biztalk peut adresser à la fois des besoins d'échange de données et d'orchestration d'applications, avec une tarification licence très compétitive.
En revanche, l'offre de Microsoft ne dispose pas actuellement de connecteurs intégrés permettant d'aller puiser les données au cœur des progiciels ou des bases de données. Ces fonctions d'extraction devront être prises en charge par des connecteurs spécifiques, tels que ceux proposés par Information Builders, ou des extracteurs polyvalents type DataExchanger, de Cross Database Technlogy.

8 ETUDES DE CAS

8.1 INTRODUCTION

Nous avons souhaité présenter dans cette dernière partie des études de cas concrètes. Depuis le lancement du framework et des serveurs .NET, nous avons réalisé bon nombre de missions auprès de grands comptes autour de ces architectures. Nous avons sélectionné quelques cas typiques et représentatifs des enjeux. Car au-delà de la théorie, la pratique et le retour du terrain sont les seuls éléments sur lesquels il est possible de s'appuyer. Les études de cas sélectionnées ici présentent une mise en œuvre du framework .NET.

8.2 MIGRATION D'UNE APPLICATION VISUAL BASIC VERS .NET

Contexte : *le client dispose d'un fort existant client-serveur Visual Basic 6. La base de données est une base Oracle. Les équipes de développement ont une bonne connaissance du développement VB mais ne maîtrisent pas le développement objet. L'objectif est de migrer progressivement le parc applicatif vers .NET.*

Mission de Business Interactif : définir une architecture cible pour la migration, réaliser au forfait la migration et assurer le transfert technologique des équipes.

Options technologiques retenues : orientation vers une solution de type WebForm. Les contraintes de déploiement de l'application client-serveur étaient assez fortes, le passage à une architecture trois-tiers permet de les faire disparaître. En outre l'application doit être conçue pour évoluer vers du multi-terminal, et le client léger est le plus adapté.

Le langage retenu est finalement VB.NET, le client souhaite minimiser le travail d'acquisition de ses équipes. Le versionning est géré sous Visual Source Safe. Le driver Microsoft permettant un accès natif à Oracle est sélectionné.

Pour gérer la persistance relationnelle-objet, le framework Objectspaces n'est pas utilisé (encore en version beta au moment du projet), c'est finalement le framework de persistance .NET développé par Business Interactif qui est utilisé et adapté au contexte.

Un certain nombre de composants visuels permettant de créer une interface utilisateur très proche de l'environnement Windows dernière génération sont utilisés (non fournis avec Visual Studio).

C'est finalement un scénario de migration verticale qui est retenu (migration d'un ensemble de fonctionnalités). Les ponts avec les autres fonctionnalités existantes en VB sont implémentées sous forme de web services (client côté .NET, serveur de web services côté VB).

Difficultés rencontrées : l'interface web a obligé à certains compromis par rapport au client-serveur. Certains rendus visuels ont du être adaptés, les composants correspondants n'étant pas disponibles chez les éditeurs tiers. Les fonctionnalités de post-back ont été utilisées de manière limitée pour des questions de performance. Des opérations liées à la manipulation d'objets.

Autre remarques : l'utilisation « fine » du cache a permis de monter fortement les performances. VB.NET a confirmé ses limites par rapport à C#, notamment en termes de manipulation des types objet.

8.3 NOUVELLE APPLICATION « CLIENT-SERVEUR .NET »

Contexte : *le client souhaite réaliser une application de supply chain permettant de gérer la préparation de commandes clients (depuis la commande fournisseur jusqu'à l'expédition).*

Mission de Business Interactif : réaliser l'application et les services NT associés pour gérer les flux vers et depuis un site de e-commerce et des prestataires de livraison. Réalisation sur des délais très courts.

Options technologiques retenues : La richesse des interfaces utilisateur et l'absence de problèmes de déploiement a orienté le choix vers un développement Winforms, en C#. La gestion des états (rapports, impressions) est réalisée avec Crystal Report .NET. La base de données est SQL Server 2000, la gestion de version est réalisée avec CVS.

Difficultés rencontrées : certains composants de rendu graphique évolués ne sont pas disponibles (le parc de composants est encore limité), nécessité de les développer. Ainsi le datagrid des winforms ne permet pas un paramétrage visuel avancé. Même chose pour des composants comme le FTP, non disponible en .NET natif (d'où réutilisation d'un objet COM+).

Autres remarques : le développement complètement objet a permis une réutilisation maximale et une productivité très forte, nécessaire pour tenir les délais.

8.4 PROJET DE PORTAIL : ASP.NET, C#, BIZTALK ET WEB SERVICES

Contexte : *le client souhaite réaliser un portail applicatif permettant d'afficher dans des portlets des flux d'informations issues de différentes applications et progiciels. La base de données est Oracle.*

Mission de Business Interactif : Cadrage des besoins, définition de l'architecture, réalisation du portail et de l'orchestration des flux.

Options technologiques retenues : le portail est réalisé en C#, il est conçu pour pouvoir supporter une mise en cluster. Un modèle événementiel est développé pour gérer la communication entre portlets.

Les flux en provenance des différentes applications sont gérés par Biztalk. Une orchestration de web services est implémentée sous biztalk, les web services étant réalisés en C#. Ce sont ces web services qui ont notamment pour charge d'utiliser les flux XML en provenance des différentes applications et progiciels.

Difficultés rencontrées / remarques : la richesse de l'architecture et son avance technologique ont nécessité une gestion des risques technologiques très amont, via maquettes et prototypes. Ces prototypes ont permis de s'assurer très tôt de la faisabilité de l'ensemble de la chaîne.

Sujet 1 :

Qu'est ce qu'un thread ?

Sujet 2 :

Qu'est-ce que la sérialisation en Java ?

Sujet 3 :

Qu'est-ce que la normalisation d'un schéma relationnel ?

Sujet 4 :

Qu'est-ce que le processus unifié ?

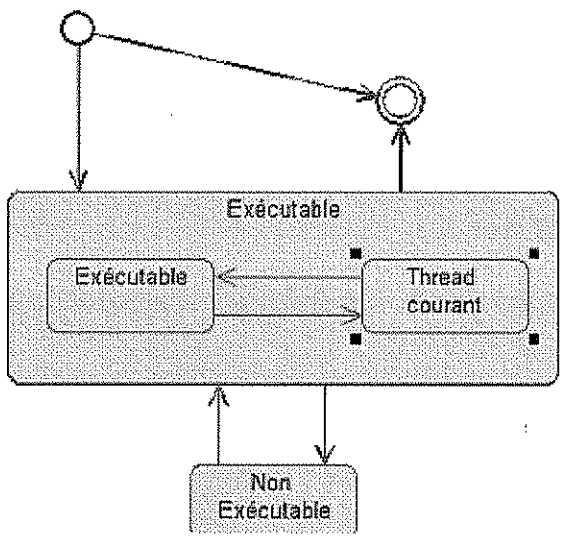
Qu'est ce qu'un thread ?

Un système multi-tâches est capable d'exécuter plusieurs programmes en parallèle sur une même machine. Dans la quasi totalité des cas il n'y a jamais deux programmes différents qui s'exécutent au même instant.

La plupart des systèmes d'exploitation sont équipés d'un ordonnançeur de tâches. Ce composant logiciel a pour mission de donner à tour de rôle le processeur aux programmes (on dit au processus) en cours d'exécution.

La meilleure traduction française de thread semble être une tâche.

Un thread peut passer par différents états, tout au long de son cycle de vie. Le diagramme suivant illustre
ces différents stades ainsi que les différentes transitions possibles. Nous reprendrons ensuite en détail chaque point du diagramme, pour mieux comprendre chaque étape de vie d'un thread.



Les threads en Java

Vous avez, bien entendu, la possibilité en Java de créer des threads. En effet, quand vous allez lancer un programme Java (par l'intermédiaire de la JVM) vous lancez en fait un processus. Ce processus, de base, contient plusieurs threads : le thread principal (celui qui exécute votre code à partir du main) et d'autres pour, par exemple, la gestion