1

Introduction et installation

1. Introduction et installation

Introduction à Visual Basic

Voilà à peine 10 ans, la conception d'un logiciel s'exécutant sous Microsoft® Windows 3.x demeurait très complexe et consommateur de ressources et de temps. Microsoft® a complètement changé cette conception en introduisant Visual Basic 1.0 en mai 1991. Désormais, les programmeurs peuvent concevoir des applications pour Windows à l'aide d'un minimum de temps et d'énergie. Voici, pour votre plaisir, un petit historique de cet outil de développement.

Visual Basic 1.0 a vu le jour en 1991 12 mois après le début du développement de cet outil sous le nom de code *« Thunder »*. Cette première version était fortement limité en fonctionnalités mais permettait pour la première fois de développer rapidement des applications s'exécutant sous Windows 3.x. L'arrivée de Visual Basic a eu un impact dépassant les attentes de ses concepteurs : la conception d'applications pour Windows a monté en flèche.

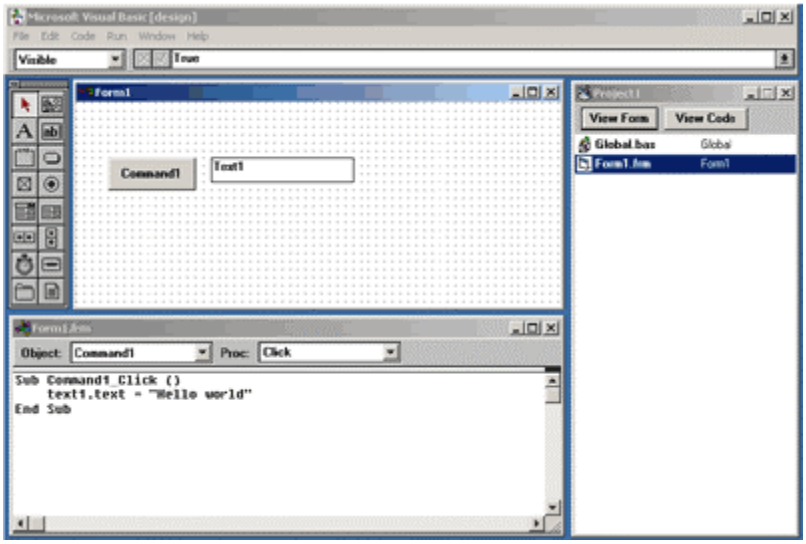


Figure 1.1 – Visual Basic 1.0

En novembre 1992, Microsoft® annonçait la mise en marché de Visual Basic 2.0. Cette version introduisait l'éditeur reconnaissant et colorant les éléments du code, la fenêtre de propriétés, des outils de debug plus élaborés, des outils permettant la connexion aux bases de données via ODBC et un support complet pour la création d'interfaces multi-documents (MDI) alors que Visual Basic 2.0 ne supportait pas lui-même les multi-documents!

Six mois seulement après la sortie de la version 2.0, Microsoft® annonçait la mise en marché de Visual Basic 3.0. Cette version incorporait de solides composants permettant la création d'applications orientées vers les bases de données en incorporant la suite d'objets DAO *(Data Access Objects)* permettant la connexion au moteur de données d'Access 1.1, différents contrôles permettant l'affichage des données sous différentes formes ainsi qu'un support pour piloter l'utilitaire de création de rapports Crystal Report®.

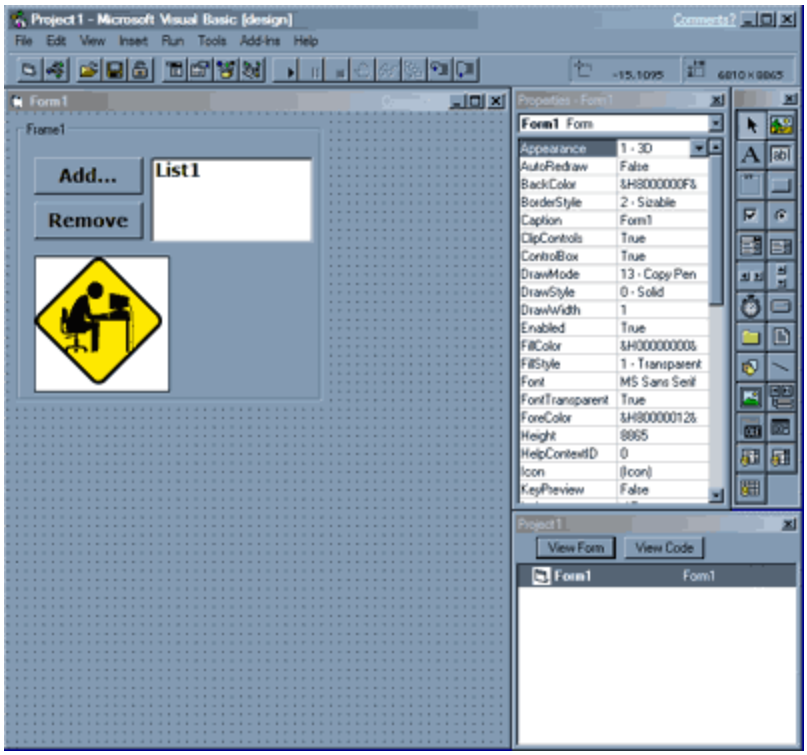


Figure 1.2 – Visual Basic 4.0

L'apparition de Windows 95 et de Windows NT provoqua le besoin d'une version de Visual Basic permettant de créer des applications en 32-bits et propre au nouvel aspect et aux nouvelles fonctionnalités de ces systèmes d'exploitation. En septembre 1995, la version 32-bits de Visual Basic 4.0 fut lancée. Cette nouvelle version fut la seule permettant à la fois la création d'applications 16-bits destinées à Windows 3.x et la création d'applications 32-bits destinées à Windows 95 et Windows NT. Visual Basic 4.0 permettait entre autres l'automation, le contrôle de données distantes et le contrôle des versions des codes à l'aide Visual SourceSafe.

Visual Basic 5.0 et Visual Basic 6.0 annoncés en mars 1997 et juin 1998 ajoutent encore aux fonctionnalités de ses précédents. À l'aide de son modèle ADO *(ActiveX Data Objects)*, la version 6.0 de Visual Basic permet entre autres un accès aux données très simple et polyvalent en plus d'offrir un concepteur intégré de rapports *(DataReport)*.

Par contre, à ce stade, la structure de Visual Basic semble avoir atteint un plafond puisqu'aucune fonctionnalité révolutionnaire ne sépare la version 6.0 de la version 4.0 si ce n'est que des ajouts de composants et l'amélioration du produit existant.

Visual Studio .NET est alors annoncé pour février 2002 et promet un Visual Basic complètement revampé et rempli de surprises.

À mon avis, la promesse a été tenue.

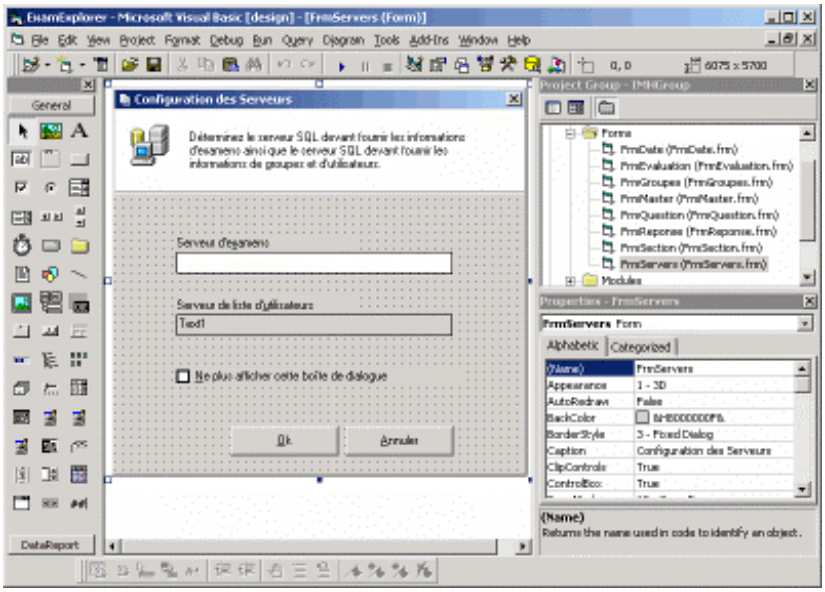


Figure 1.3 – Visual Basic 6.0

Dans les versions antérieures de Visual Basic pouvait être acquis séparément des autres langages fournis par Microsoft® mais la suite Visual Studio incluait l'ensemble de ceux-ci : Visual Basic, C++, Visual Fox Pro, Visual InterDev ainsi qu'une série d'utilitaires un peu comme Microsoft® Office intègre l'ensemble des outils de bureautique. Dès la version 6.0 de Visual Studio, on pouvait sentir que Microsoft® effectuait des tests au niveau de l'intégration de l'ensemble de ses langages au sein d'un même environnement de développement puisque Visual InterDev et Visual J++ partageaient le même environnement ainsi que d'autre utilitaires permettant entre autres la création de logiciels d'installation *(setup)*.

Visual Basic 7.0, communément appelé Visual Basic.NET s'intègre ainsi aux autres outils de la suite Visual Studio au sein d'un même interface. Ce nouvel environnement de développement est une addition des meilleurs éléments de chacun des environnements des outils précédents. On y retrouvera donc des aspects typiques à Visual InterDev 6.0, à Visual C++ 6.0, Visual Basic 6.0 etc. Microsoft® en a profité pour lancer un nouveau langage permettant d'exploiter au maximum cette intégration des environnements de développement, le C#, langage issu d'un mélange entre les forces de C++ et celles de Visual Basic.

Chacun des outils de développement de Visual Studio.NET partagent le même modèle d'objets communément appelé la plate-forme .NET ainsi que le même compilateur. La plate-forme .NET est l'ensemble des composants et services mis à la disposition du programmeur afin de lui simplifier le développement. La plate-forme .NET encapsule une série d'objets hiérarchisés au sein d'un modèle. Chacun de ces objets permettent d'accéder aux fonctionnalités du système d'exploitation, de l'internet, etc. La plate-forme .NET contient l'ensemble des éléments requis pour programmer à l'aide de Visual Basic.NET.

Par sa refonte majeure, Visual Basic.NET amène une multitudes de nouvelles fonctionnalités en comparaison aux versions antérieures. Il permet la création de services NT, la création d'applications en mode console DOS, la création de services Web, la gestion simplifiée des serveurs NT, la gestion complète des concepts orientés-objets tels l'héritage et le polymorphisme, etc. Toute cette nouvelle puissance est aisément explicable par le fait que Visual Basic partage désormais le même modèle d'objet et le même compilateur que ses confrères : la plate-forme .NET est à l'origine de ces grands gains de fonctionnalités.

Note à propos des symboles et notations utilisées

Voici une description des symboles et notations spéciales utilisées au sein de cette ouvrage :

`Objet`	Utilisé au sein du texte normal, indique le nom d'une variable, d'une fonction ou d'un objet.
`Code`	Indique un bloc de code qui doit être tapé à l'intérieur de l'éditeur de code de Visual Basic.NET.
▼	Utilisé au sein d'un bloc de code, indique que le code se continue sur la ligne suivante au sein du document seulement puisque la largeur de la feuille de papier en restreint l'inscription sur une seule et même ligne.
	Indique une note spéciale généralement à l'attention des utilisateurs des versions antérieures de Visual Basic.

Initiation à la programmation orientée-objets

Visual Basic.NET est désormais un langage pleinement orientée-objets et supporte donc l'héritage et le polymorphisme en plus d'emprunter diverses fonctionnalités du C++ telles la surcharge de fonctions et la gestion structurée des erreurs. Mais avant de commencer, certains voudront connaître de quoi retourne au juste la programmation orientée-objets (POO).

La programmation orientée-objets place l'objet au milieu de son concept. L'objet réuni aux autres objets l'entourant et aux liens les unissant permettent à la programmation orientée-objets de modéliser notre univers le plus fidèlement possible.

Toute l'histoire commence par le concept de l'encapsulation qui stipule que tout objet doit conserver confidentiel les détails de son fonctionnement interne et ne rendre disponible publiquement que des fonctionnalités intuitives et simples d'utilisation. Ainsi, si pour calculer le salaire hebdomadaire d'un employé je dois multiplier le nombre d'heures qu'il a travaillé par son salaire horaire puis soustraire une multitudes d'impôts, de taxes et de rentes de toutes sortes, un objet `Employé` n'aurait qu'à rendre public la fonctionnalité `CalculerSalaireHebdo` :

Programmation procédurale

```
Dim SalH As Single, NbrH As Integer
Dim Total As Single, X As Single
SalH = 22.5
NbrH = 40
Total = SalH * NbrH
X = Total - (Total * .025)
X = X - (Total * .08)

MsgBox "Jack a gagné " & X & "$."
```

Programmation orientée-objets

```
Dim Jack As New Employe, X As Single
Jack.SalaireHoraire = 22.5
Jack.NbrHeures = 40
X = Jack.CalculerSalaireHebdo()

MsgBox ("Jack a gagné " & X & "$.")
```

On remarque que l'objet `Employe` utilisé au sein de la programmation orientée-objets possède des fonctionnalités intuitives et qu'il encapsule les détails du calcul du salaire hebdomadaire évitant ainsi les risques d'erreurs et la répétition du code.

Un objet, communément appelé *classe*, exposera donc au programmeur une série de fonctionnalités et en dissimulera son fonctionnement interne. Ces fonctionnalités sont regroupés au sein de trois grandes catégories :

Propriétés Caractéristique de l'objet ou comportement qu'il doit adopter. Une voiture est rouge, possède quatre roues, possède une consommation d'essence, etc.

Méthodes Fonction exécutable par l'objet. On peut demander à la voiture de rouler, klaxonner, freiner, etc.

Événements Soulevé par l'objet afin de nous notifier un événement. La voiture a étouffée, a percuté un poteau, a manquée d'essence, etc.

Différents autres concepts tels l'héritage et le polymorphisme qui accompagnent le concept de programmation orientée-objets seront explorés en temps venus.

Aperçu de la plate-forme .NET

La plate-forme .NET est l'ensemble des composants et services mis à la disposition du programmeur afin de lui simplifier le développement. La plate-forme .NET encapsule une série d'objets hiérarchisés au sein d'un modèle. Chacun de ces objets permettent d'accéder aux fonctionnalités du système d'exploitation, de l'internet, etc. La plate-forme .NET contient l'ensemble des éléments requis pour programmer à l'aide de Visual Basic.NET.

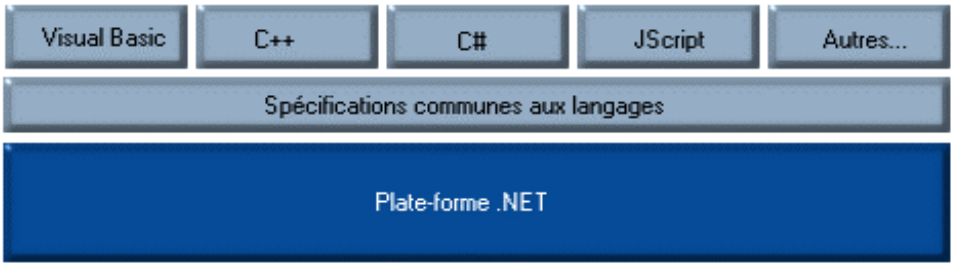


Figure 1.4 – Structure globale de la plate-forme .NET

La plate-forme .NET est neutre et n'est aucunement spécifique à aucun langage. Elle encapsule seulement une pluralité de fonctionnalités utilisables par différents langages. Ainsi, les langages Visual Basic, C++, C# et Jscript utilisent le même modèle d'objets que constitue la plate-forme .NET. Par exemple, si je désire afficher un texte à l'écran en mode console, j'utiliserai la fonction

```
Console.WriteLine( )
```

Maintenant, si je désire exécuter cette fonction en Visual Basic, mon code ressemblera à ceci :

```
Dim St As String =  "Bonjour le monde"
Console.WriteLine( St )
```

tandis que si je désire exécuter la même fonction en C#, mon code ressemblera à ceci :

```
String St = "Bonjour le monde" ;
Console.WriteLine( St );
```

Cependant, notez que le nom de la fonction et son utilisation demeure la même d'un langage de programmation à un autre. Ce comportement procure à la plate-forme .NET une force inestimable : elle est aisément programmable à l'aide de la majorité des langages existants à ce jour. Ainsi, la plate-forme .NET peut tolérer au-dessus d'une vingtaine de langages différents en incluant le COBOL, le Pascal, le Perl, le Python et le SmallTalk pour n'en nommer que quelques-uns.

Un autre avantage de l'indépendance de la plate-forme .NET face aux différents langages est que, tout programme écrit, peu importe le langage utilisé, sera dépendant d'une même série de fichiers systèmes nécessaires à l'exécution de ce programme. Auparavant, chacun des langages possédaient leurs fichiers dépendants qui devaient être installés sur la machine client avant de pouvoir être exécuté. Avec .NET, l'ensemble des langages partagent les mêmes fichiers qui sont inclus automatiquement au sein de Windows XP et le seront dans les versions ultérieures de Windows.

Finalement, la plate-forme .NET encapsule l'ensemble des fonctionnalités jadis disponibles individuellement au sein des outils de développement des versions 6.0 et antérieures de Visual Studio. La plate-forme .NET prévoit donc pour l'ensemble des langages .NET les fonctionnalités de la création de formulaires de Visual Basic 6.0, les fonctionnalités orientée-objets de Visual C++ 6.0 et les fonctionnalités Internet d'InterDev 6.0 réunis au sein d'un seul élément. Cette unification des technologies profite évidemment à Visual Basic : le programmeur VB peut désormais aisément créer des services NT, créer des sites web interactifs et transactionnels sans jamais quitter son langage préféré!

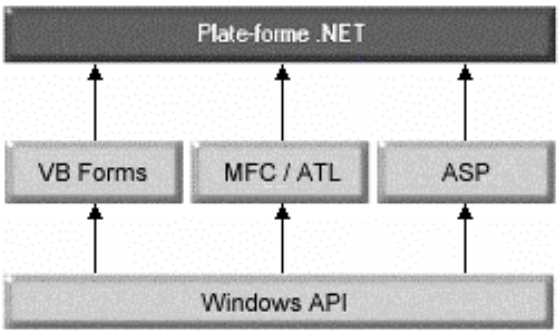


Figure 1.5 – Unification des fonctionnalités

La plate-forme .NET est sous-divisée en sous-éléments comme illustré ci-dessous.

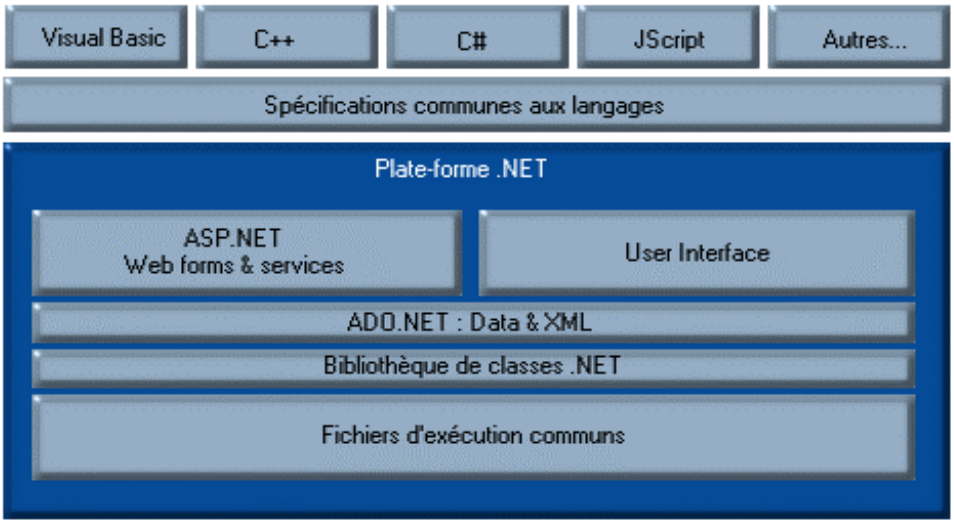


Figure 1.6 – Structure plus élaborée de la plate-forme .NET

Fichiers d'exécution communs *(Common Language Runtime)*
Cet élément expose différentes fonctionnalités communes à l'ensemble des langages .NET. Cet élément de l'architecture .NET encapsule les fonctionnalités de débogage, de la gestion de la mémoire lorsque des classes et objets sont chargées et déchargées par l'application, de compilation, de gestion des Threads, de gestion du Marshalling entre divers composants, etc. Nous reviendrons au moment adéquat sur ces principes mais notez que cet élément est l'élément de base pour toute application conçue à l'aide de Visual Studio.NET, tous langages confondus.

Bibliothèque de classes .NET *(.NET Framework Class Library)*
Cet élément expose l'ensemble des classes utilisables par les différents langages .NET. Ces classes sont regroupées de manière thématique et permettent la manipulation du texte, des périphériques, du système d'exploitation, de la sécurité, etc. Ce sont les classes que vous serez le plus souvent appelé à utiliser en Visual Basic.NET.

ADO.NET
Cet élément expose l'ensemble des objets ADO *(ActiveX Data Objects)* permettant l'accès et la manipulation de jeux d'enregistrements provenant de bases de données Access, SQL Server, Oracle ou autres. Chacun des jeux d'enregistrements est doublé d'un équivalent XML permettant aux données d'être transmise au sein d'un protocole standard de modélisation des données.

ASP.NET
Cet élément expose l'ensemble des classes permettant la conception de pages et de services Web. Les fonctionnalités de l'ASP *(Active Server Pages)* à concevoir des sites web interactifs et transactionnels y sont encapsulées et sont accessibles par l'ensemble des langages .NET sous forme de services Web *(Web Services)*.

Interface utilisateur *(User Interface)*
Cet élément expose l'ensemble des fonctionnalités permettant la conception d'interfaces Windows. Plusieurs fonctions qui n'étaient auparavant accessibles que par l'API sont intrinsèques à Visual Studio.NET et rendent donc Visual Basic plus puissant que jamais. Le modèle d'objets `System.Drawing` permet entre autres l'accès aux fonctionnalités GDI+ de dessin 2D et de manipulation graphique du texte.

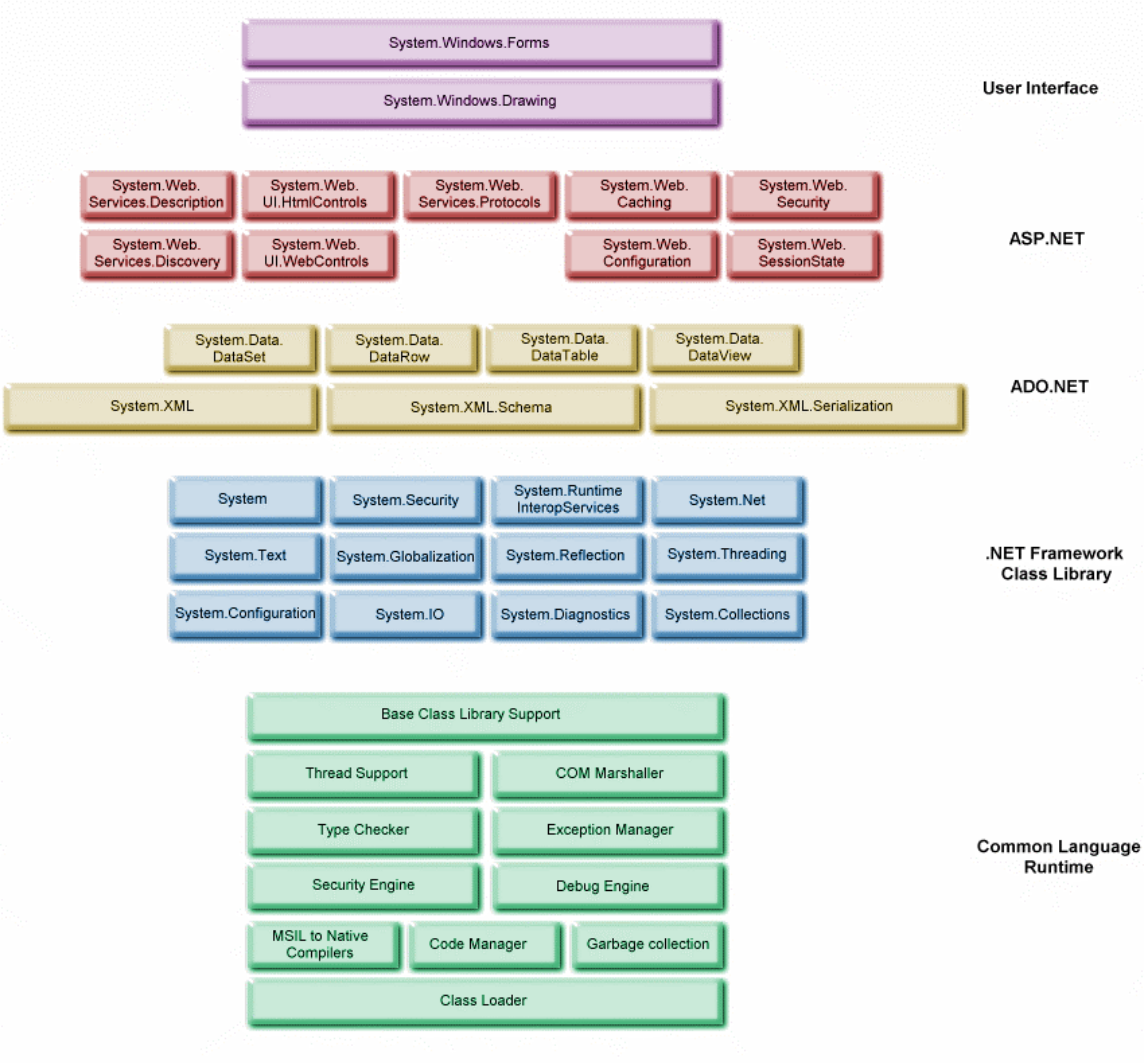


Figure 1.7 – Modèle plus complet de la plate-forme .NET

Installation de Visual Studio.NET

L'installation de Visual Studio.NET s'effectue en trois étapes :

1. la mise à jour de composants afin d'adapter correctement votre système pour qu'il puisse servir de plate-forme de développement .NET;

2. l'installation proprement dite de Visual Studio.NET tel que contenu sur votre CDRom ou DVDRom;

3. la mise à jour de votre installation à l'aide de Services Packs disponibles sur CDRom ou via l'Internet. Cette fonctionnalité vous permet de vous assurer de posséder les versions les plus récentes de tous les fichiers de l'installation.

Lorsque vous insérez votre CDRom ou votre DVDRom dans votre lecteur, l'installation de Visual Studio.NET s'exécute et la fenêtre suivante s'affiche :

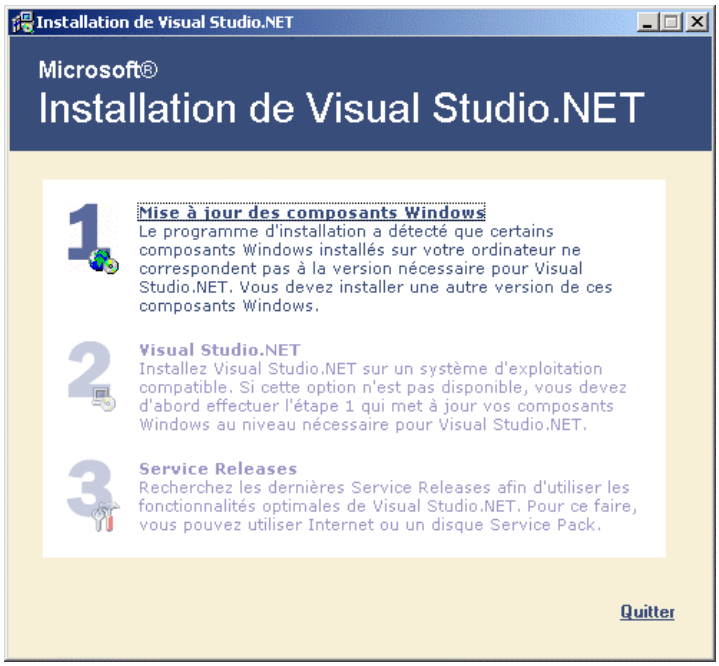


Figure 1.8 – Menu de l'installation de Visual Studio.NET

Lorsque Visual Studio.NET est installé sur un système d'exploitation antérieur à Windows XP, l'installation vous indique qu'une **mise à jour des composants Windows** est nécessaire. Cette mise à jour remplace différents composants Windows dont l'Internet Explorer, la machine virtuelle Java, etc. L'installation de Visual Studio.NET sous les versions antérieures à Windows XP peut nécessiter plusieurs redémarrage de l'ordinateur tandis que Windows XP ne nécessite aucun démarrage afin de compléter l'installation de Visual Studio.NET.

La mise à jour des composants Windows peut causer le dysfonctionnement d'une version antérieure de Visual Basic qui serait déjà installée sur le système. Il est préférable d'installer Visual Basic.NET sur un système d'exploitation différent si vous désirez conserver une version antérieure de Visual Basic.

Plusieurs redémarrages de l'ordinateur sont nécessaires lorsque que la mise à jour des composants est exécuté. Si vous installez sur un ordinateur exécutant Windows NT ou Windows 2000, l'installation peut ouvrir une session automatiquement à chaque redémarrage sans que vous n'ayez à vous authentifier.

Pour permettre à l'installation d'ouvrir automatiquement une session, cochez la case à cocher **Ouvrir une session automatiquement** et entrez votre mot de passe avant d'appuyer sur **Installer maintenant**.

Cette fonctionnalité est inutile sous Windows 95 / 98 et non nécessaire sous Windows XP puisque ce dernier ne nécessite aucun redémarrage de l'ordinateur.

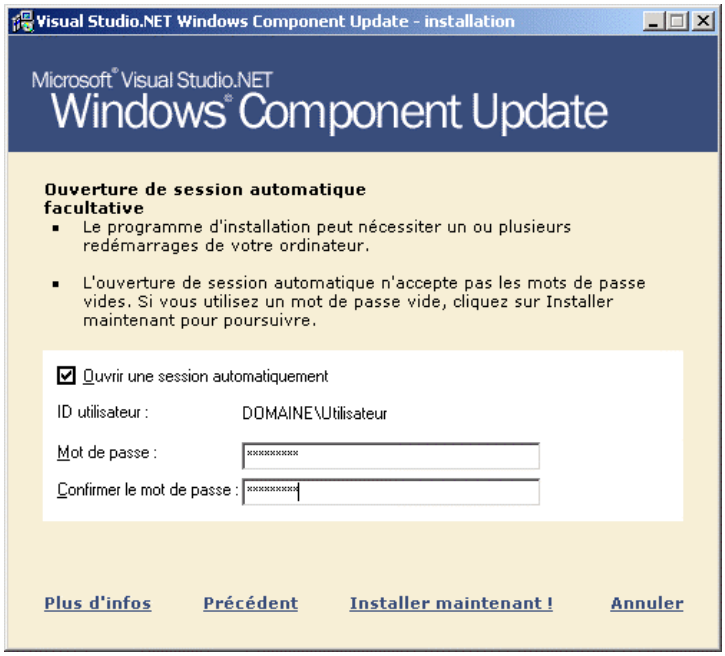


Figure 1.9 – Ouverture de session automatique facultative

Figure 1.10 – Mise à jour des composants en cours

L'installation proprement dite de Visual Studio.NET peut s'effectuer lorsque les mises à jour nécessaires ont été installées. Après avoir appuyé sur le second menu du menu principal, vous êtes invité à prendre connaissance du contrat de licence utilisateur, de préciser la clé du produit (numéro de série) le cas échéant ainsi que votre nom afin de personnaliser cette copie.

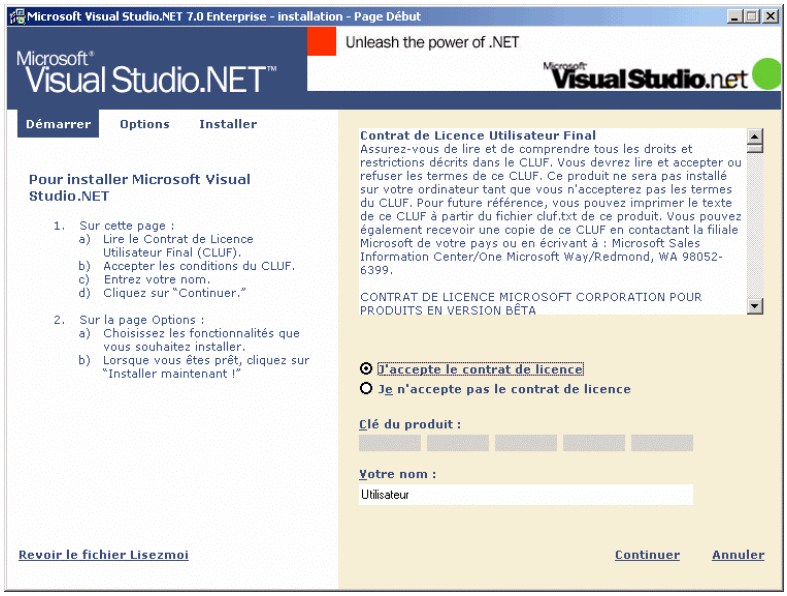
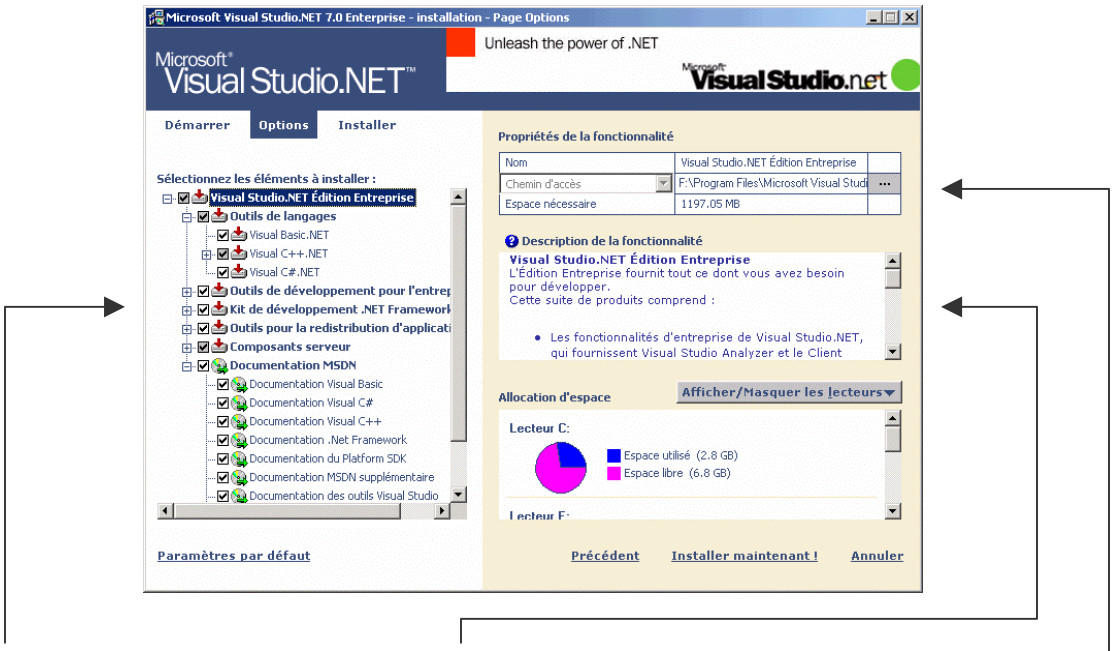


Figure 1.11 – Contrat de licence et clé du produit

Ensuite, vous serez invité à spécifier les produits à installer et la partition où ils devront s'installer.



Produits devant être installés Description du produit Destination de l'installation

Finalement, lorsque l'installation est terminée, la mise à jour à l'aide de Services Packs disponibles sur CDRom ou via l'Internet peut s'effectuer. Cette fonctionnalité vous permet de vous assurer de posséder les versions les plus récentes de tous les fichiers de l'installation. Visual Studio.NET est alors prêt à être utilisé.