La plate-forme .Net
Introduction

Michel RIVEILL

Université de Nice – Sophia Antipolis

riveill@unice.fr - http://www.polytech.unice.fr/~riveill

.NET

- Introduction
  - La plate-forme
  - Le C.L.R (Common Language Runtime)
- Le langage C# (et le CLR)
  - Assemblies et module
  - Types références et valeurs
  - Réflexion et attributs
  - Delegates et Events
- Les applications Web
  - L'accès aux données
  - Les services Web
  - Les pages dynamiques
- Les services techniques
  - La sécurité
  - Les transactions
- .Net 2.0, 3.0, 3.5
- Evaluation
  - Le multi-langage dans .Net
  - J2EE vs .Net
- Pour aller plus loin

© 2007, M. Riveill

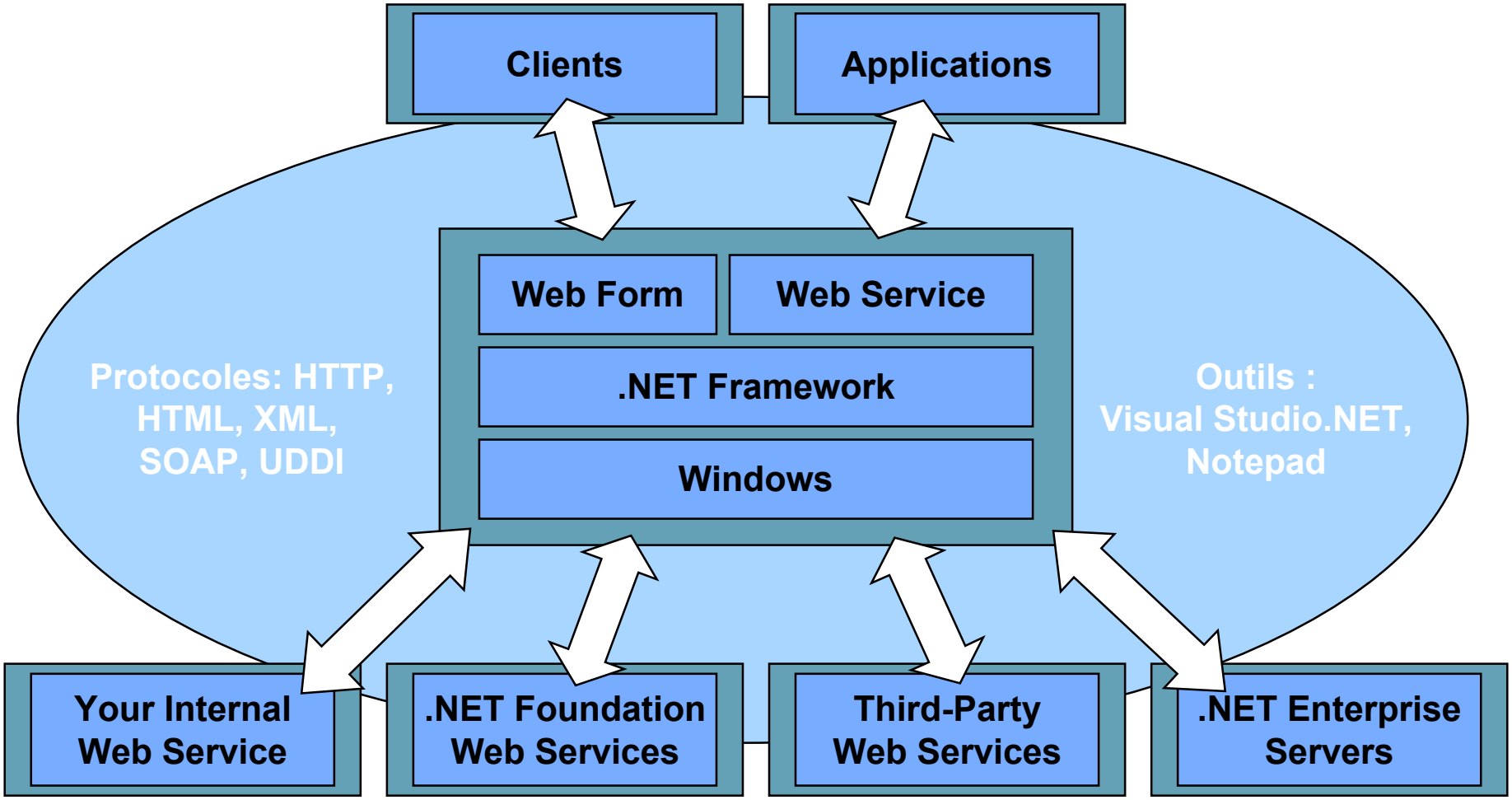
Première partie

.NET, C'est quoi ?

- Une vision (celle de Bill & Co) de l'évolution des technologies du Web
  - Les applications sont des services auxquels on peut s'abonner et qui peuvent être référencés
    - Services offerts par des sites web
  - De nouveaux terminaux complètent les PCs
    - IHMs adaptables et personnalisables
  - Respect des standards (issus du Web)
    - C'est nouveau pour Microsoft
- Une plate-forme qui supporte cette vision
  - .NET Framework et Visual Studio.NET
  - .NET Enterprise Servers – passerelle avec le système d'exploitation Windows
    - Base de données, messages, Intégration, proxy, sécurité, mobilité, gestion du contenu, orchestration des services, …
  - .NET Building Block Services
    - .NET My Services – des services fournis par Microsoft
      - Passport (authentification)
      - Hailstorm (profils utilisateurs)
      - .NET Alerts (service d'alerte)
  - Objectif : rendre la construction d'applications Web aisées
    - Par assemblage de Web services

© 2007, M. Riveill

La plate-forme .NET

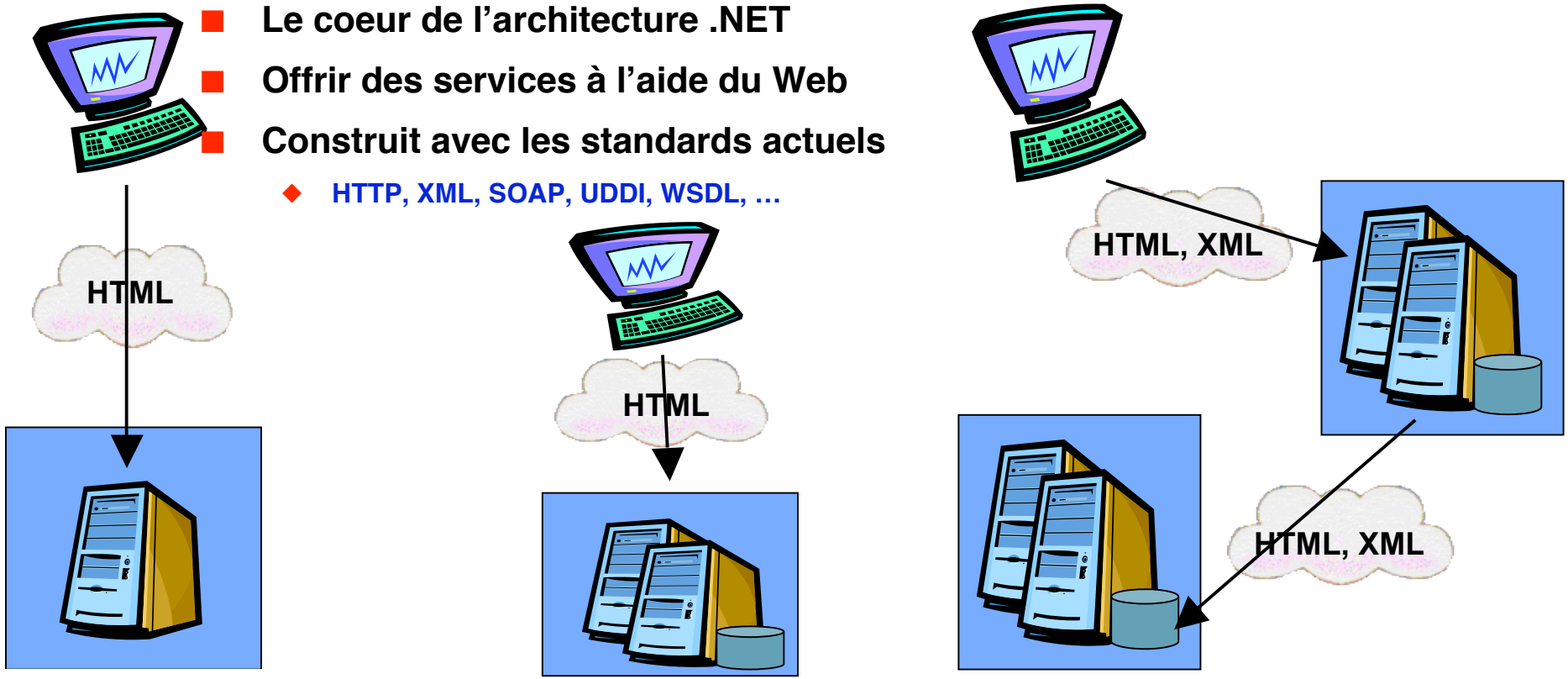


© 2007, M. Riveill

Services Web

Une évolution du Web

- Des composants applicatifs accessibles à partir des standards du Web
- Le coeur de l'architecture .NET
- Offrir des services à l'aide du Web
- Construit avec les standards actuels
  - HTTP, XML, SOAP, UDDI, WSDL, …



Generation 1
Static HTML

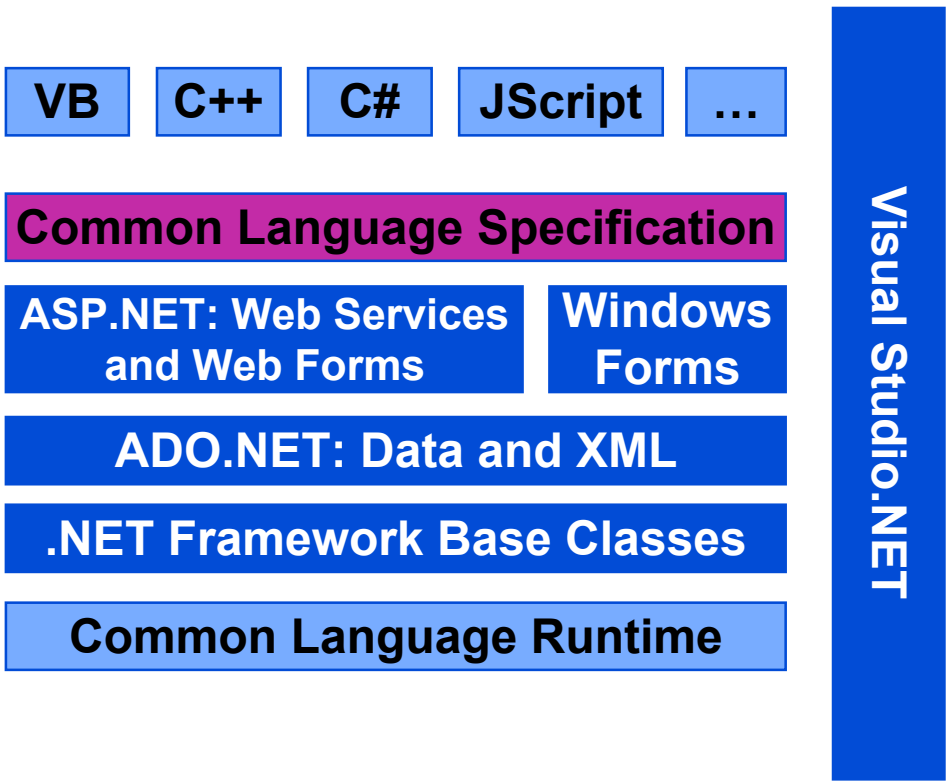
Generation 2
Web Applications

Generation 3
Web Services

© 2007, M. Riveill

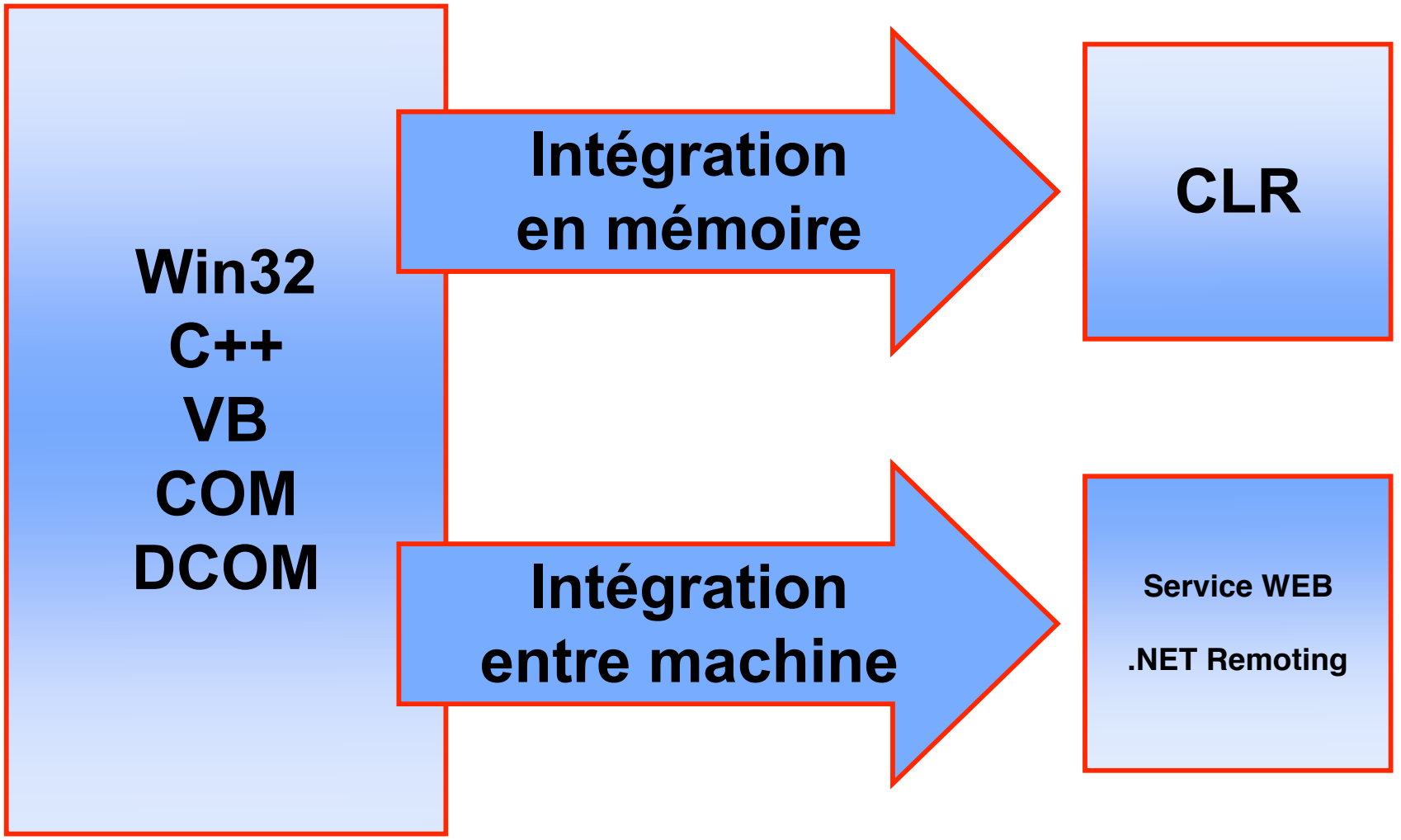
Le framework .NET

- Un ensemble de technologies pour développer et utiliser des composants :
  - Formulaires Web
  - Services Web
  - Applications Windows
- Des outils pour développer des applications
  - Développement
  - Mise au point
  - Déploiement
  - Maintenance

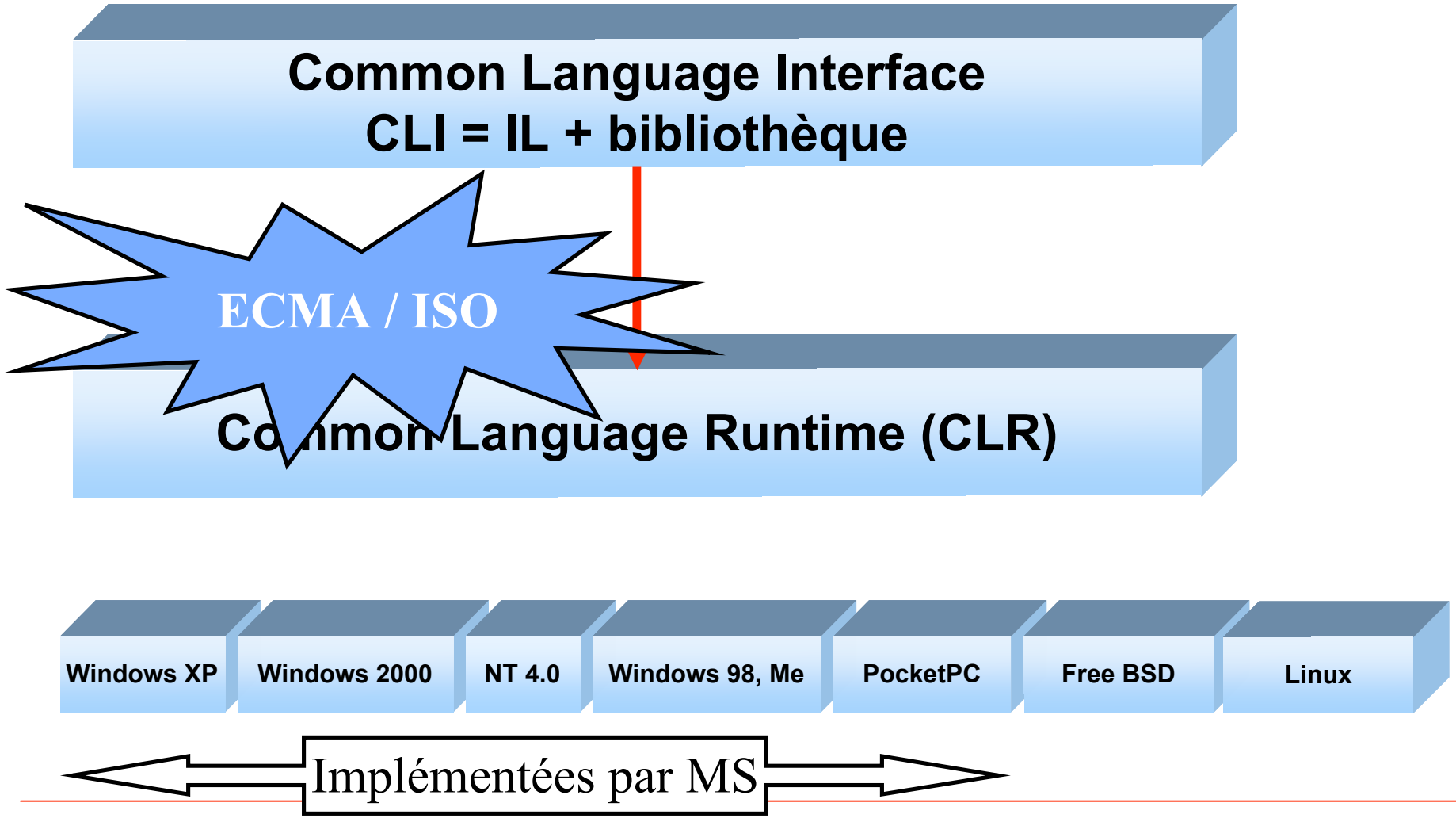


© 2007, M. Riveill

La plateforme .NET

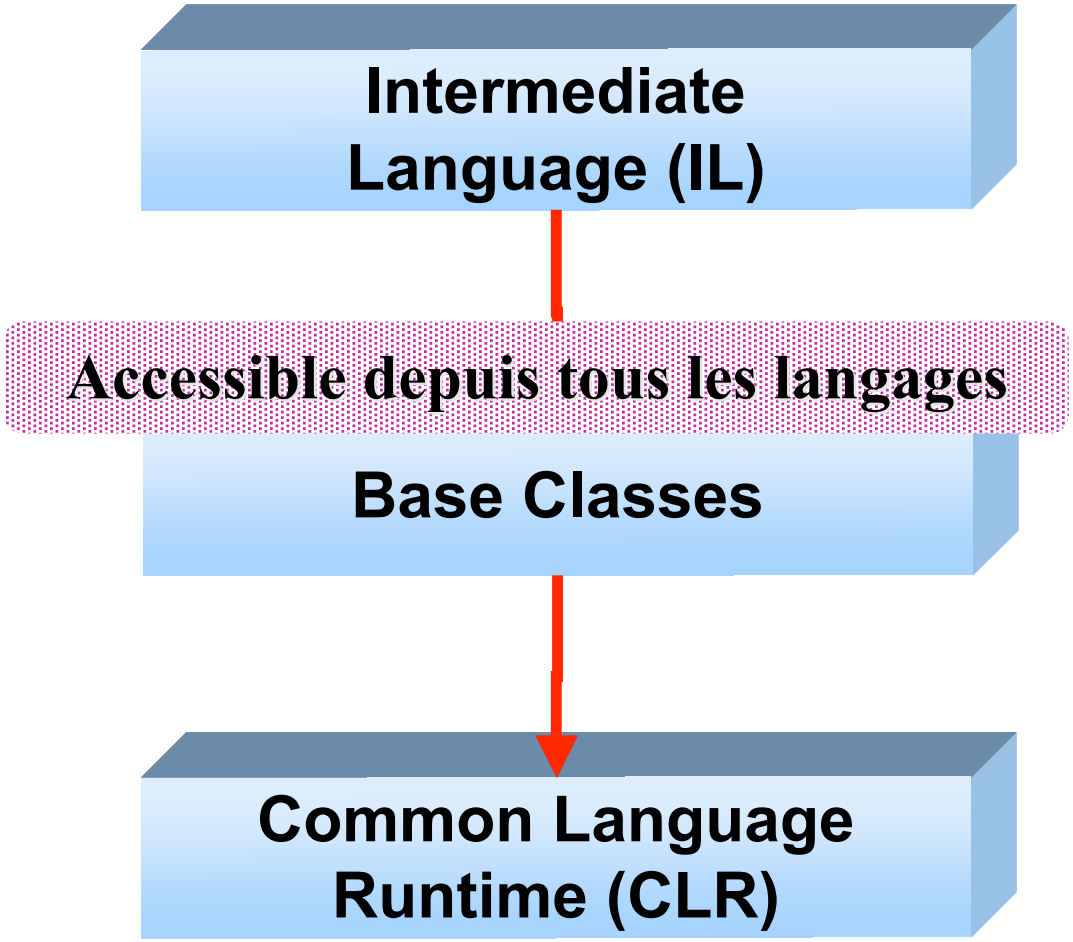


La plateforme .Net



© 2007, M. Riveill

Le .Net Framework



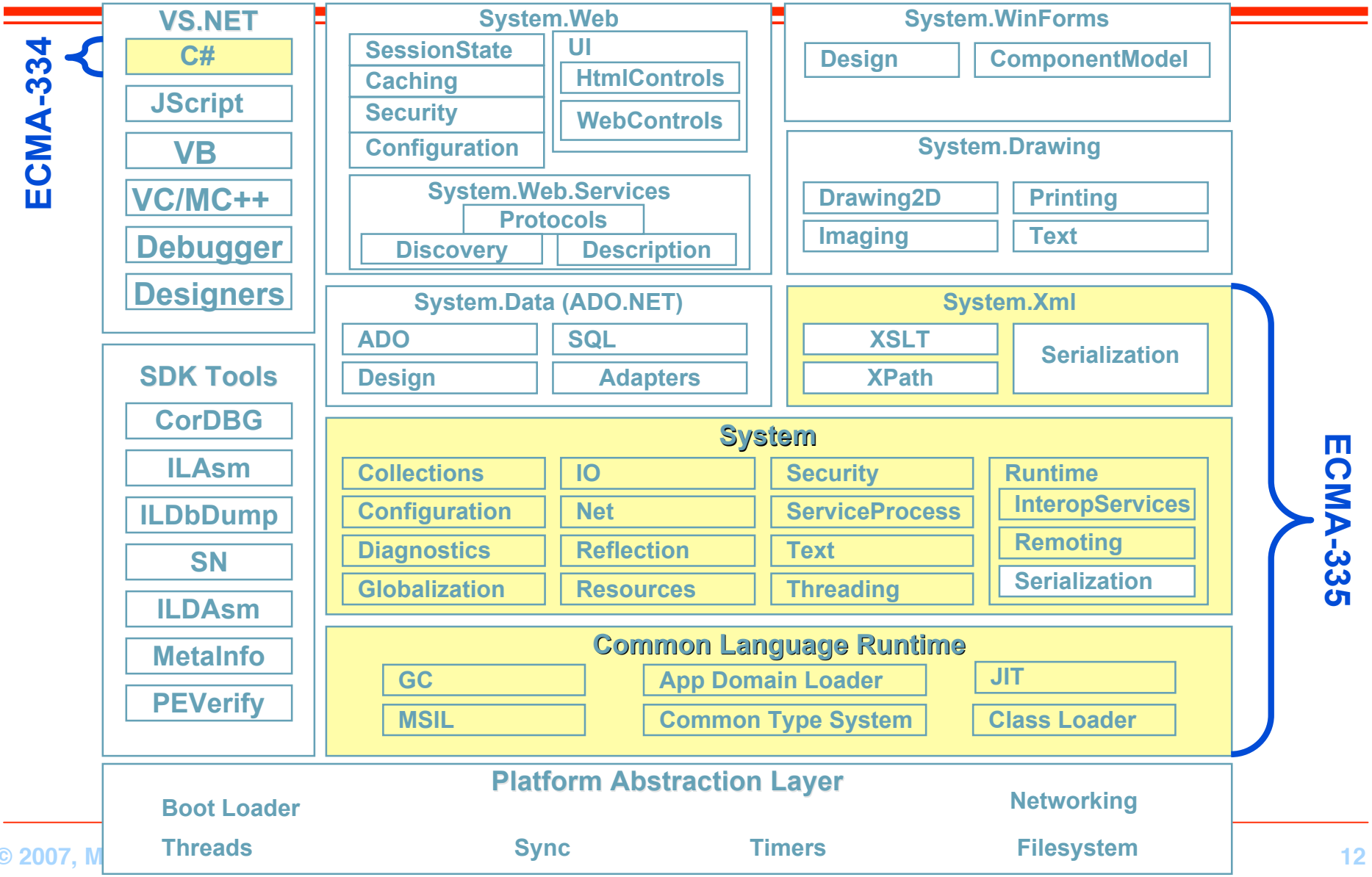
Entre le langage intermédiaire (IL) et le machine virtuelle (CLR) se trouvent les classes du .Net Framework

Ces classes sont accessibles par tous les langages puisque accessibles depuis l'IL

Une partie de ces classes ont été intégrées dans les spécifications
du CLR
- Modèle d'objet commun
- Types unifiés
- Meta-Data
- Modèle d'exception commun

© 2007, M. Riveill

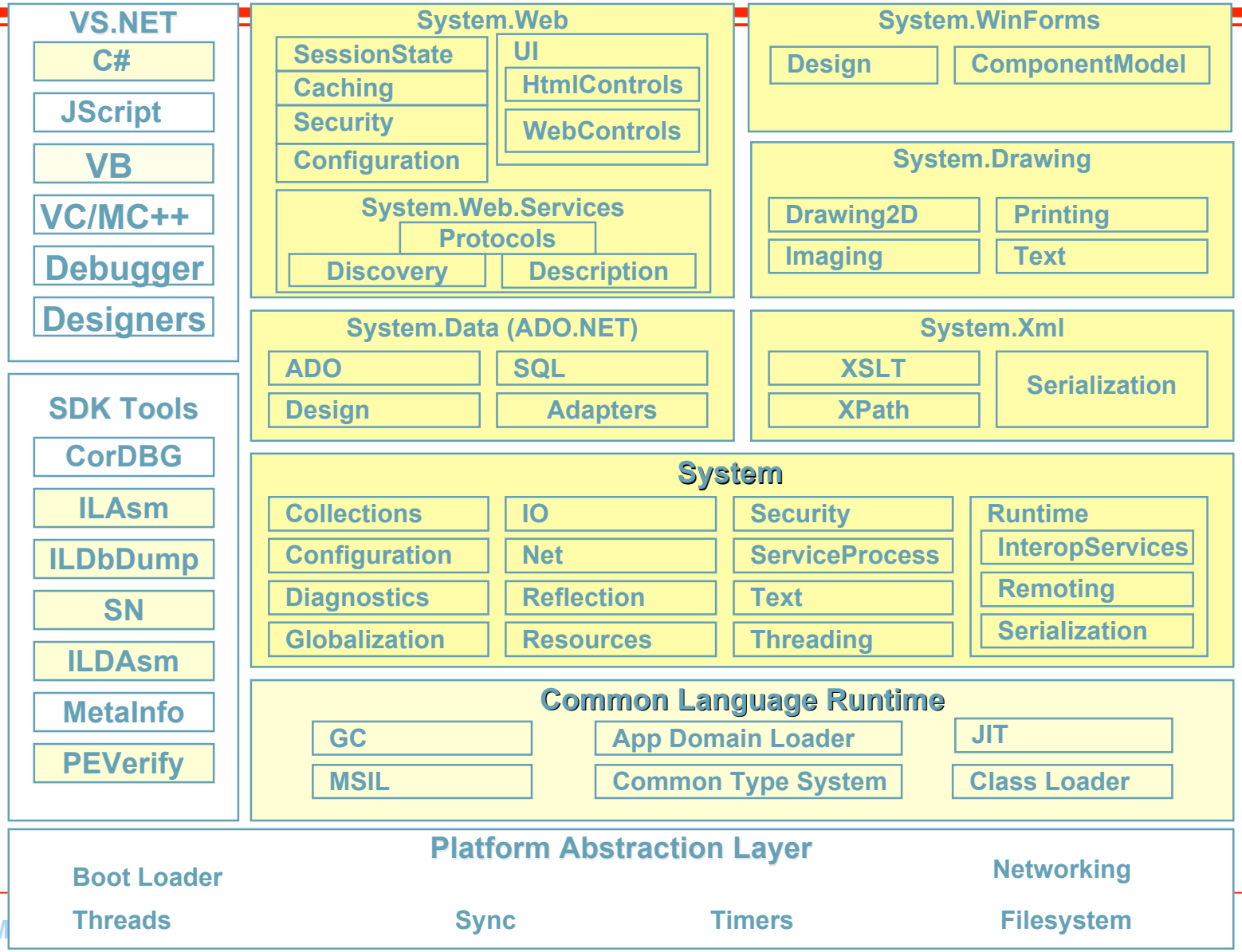
CLI Standards



© 2007, M

Le framework .NET

Les classes du framework



© 2007, M

Le CLR et ses implémentations

- **Framework .Net est suffisant pour programmer, c'est la version officielle**
  - pas accès aux sources
  - Il faut aussi prendre le SDK
  - Webmatrix permet d'avoir une plate-forme de développement gratuite
    - Intègre serveur web et base de données
  - Visual Studio .Net, SQL serveur
    - Disponibles pour les étudiants de Polytech'Nice-Sophia (licence MSDN AA)

- **ROTOR est une implémentation Microsoft de la technologie .Net**
  - http://msdn.microsoft.com/sscli
  - implémentation complète de standard ISO/ECMA (c'est un sur ensemble de la norme)
  - Fonctionne avec FreeBSD, Mac OS X et Windows XP (licence de type BSD)
  - 100% du code source est disponible
  - Il n'y a pas d'implémentation correcte de ROTOR pour Linux

© 2007, M. Riveill

CLR et les implémentations non Microsoft en 2003

- Ximian : MONO est un projet indépendant
  - http://www.go-mono.org
  - compatibilité avec les produits .Net
  - fonctionne sur Linux (licence GPL)
  - actuellement : C#, JIT très efficace, pas d'IDE
- DotGNU : Portable .Net
  - http://www.southern-storm.com.au/portable_net.html
  - implémentation du standard ISO/ECMA
  - fonctionne sous GNU/Linux, c'est un projet GNU
    - mais aussi sous Windows, Solaris, NetBSD, FreeBSD, and MacOS X.
    - le runtime a été testé sur les processeurs x86, PowerPC, ARM, Sparc, PARISC, s309, Alpha, and IA-64
  - Actuellement : C#, pas de JIT, de nombreux outils
- Il y en a d'autre…

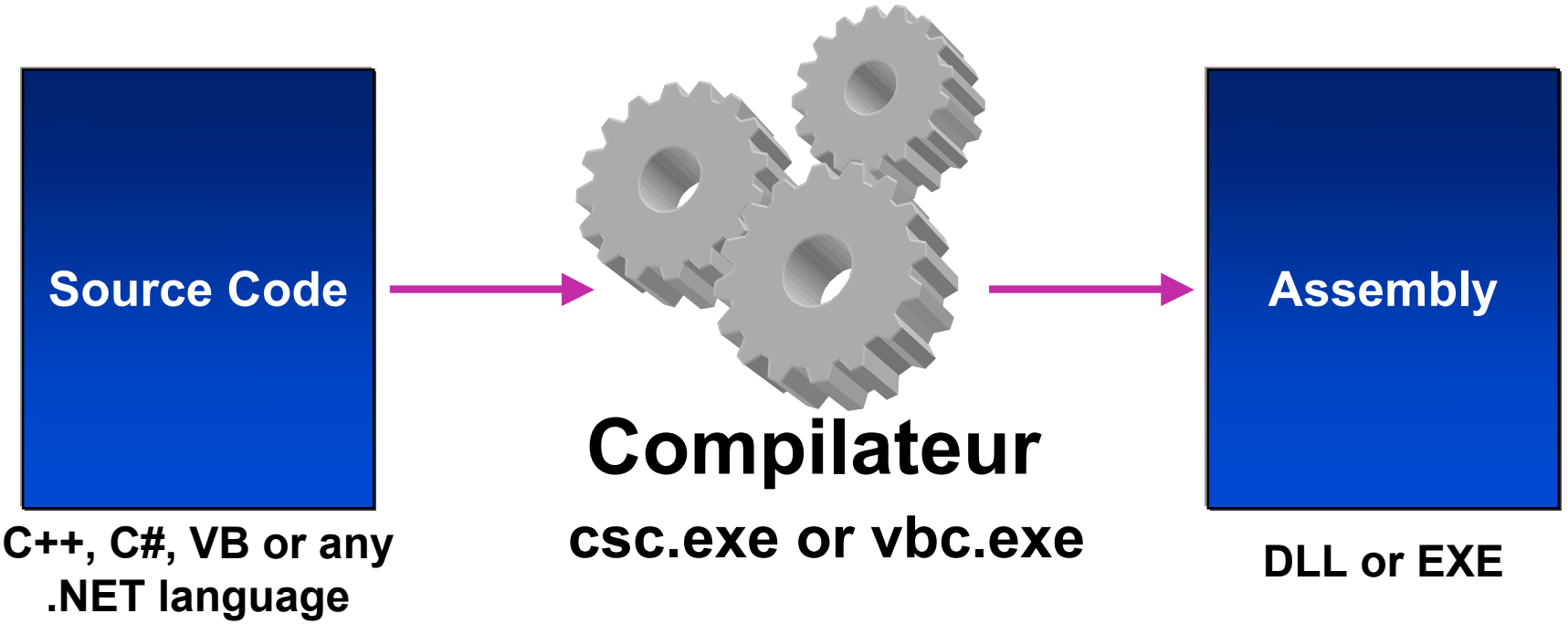
© 2007, M. Riveill

Common Language Runtime
Buts

- **Développement de services**
  - Permettre une interopérabilité entre les langages
  - Augmenter la productivité
    - chacun écrit dans le langage qu'il maîtrise, le plus adapté, …
- **Services déploiement**
  - Simple, sûr
  - Gestion des versions – NO MORE 'DLL HELL'
- **Services à l'exécution**
  - Performance
  - Extensibilité
  - Sûreté et disponibilité
  - Sécurité

© 2007, M. Riveill

Common Language Runtime
Compilation



© 2007, M. Riveill

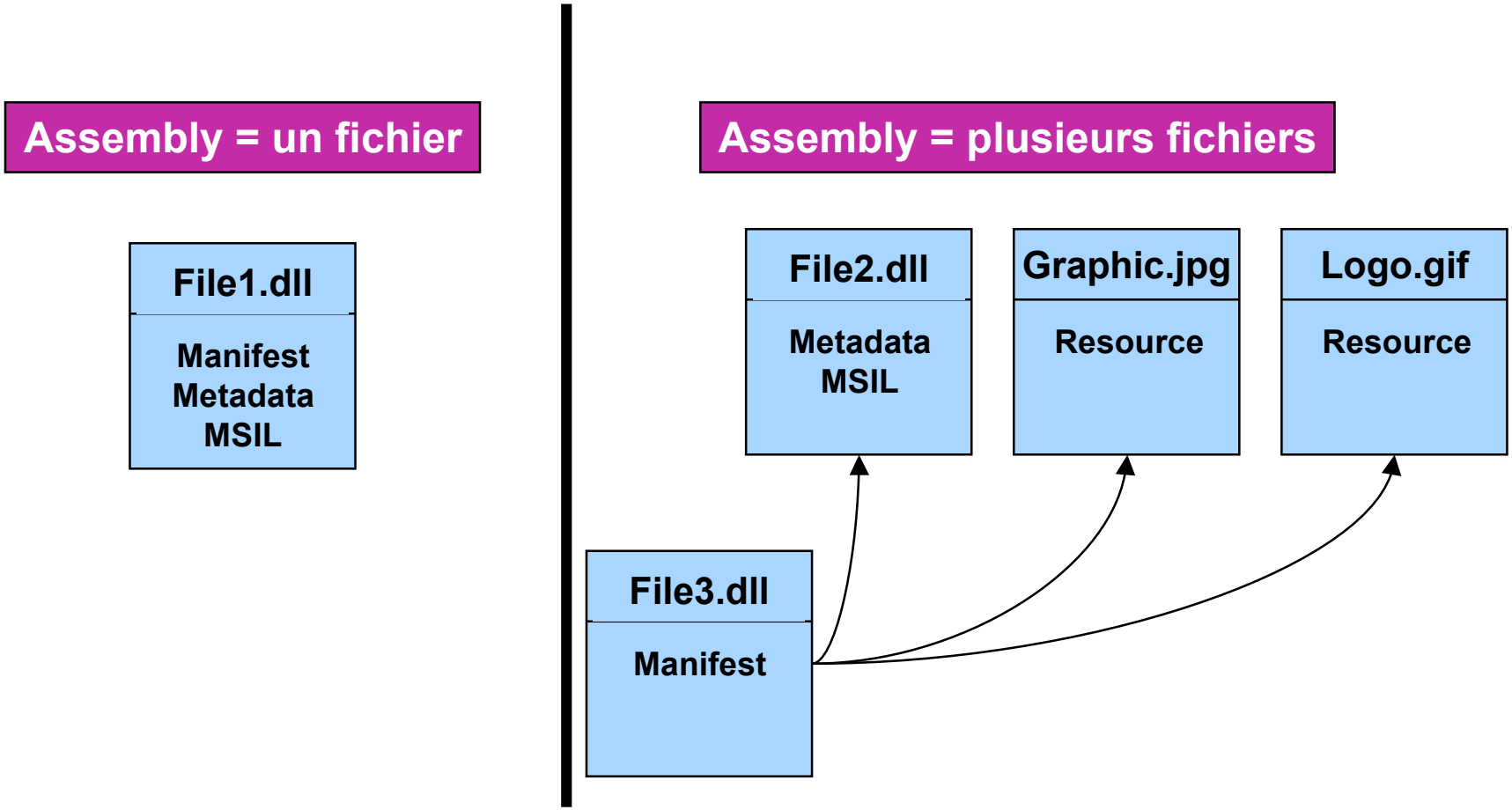
Common Language Runtime
Assemblages - Assemblies

- **Assembly : archive**
  - Unité logique de déploiement (composants de la machine virtuelle)
  - Contient Manifest, Metadata Type, code IL et autres ressources
- **Manifest : fichier de description de l'assemblage**
  - Metadata à propos des composants présent dans un assembly (version, types, dépendances, etc.)
- **Metadata Type**
  - Définition complète de tous les types présent dans l'assembly : attributs, méthodes, paramètres, ressources…
- **code IL : langage intermédiaire typé**
  - Tous les langages sont compilés en IL (managed code)
  - IL est toujours compilé en code natif avant exécution (JIT compiler)
- **Ressources**
  - .bmp, .jpg

© 2007, M. Riveill

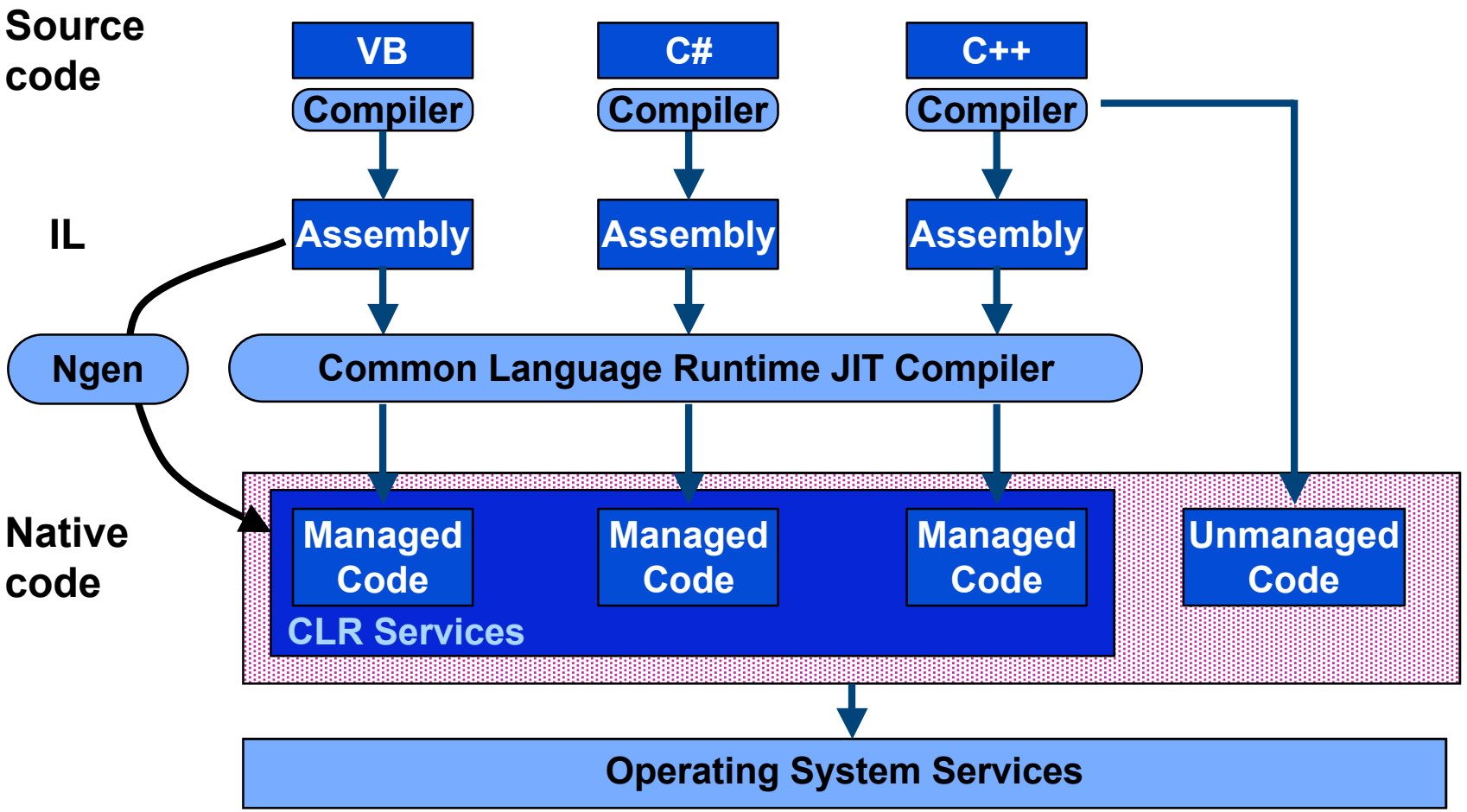
Assemblies

Components of an Assembly



© 2007, M. Riveill

Common Language Runtime
Modèle d'exécution



© 2007, M. Riveill

Common Language Runtime Services

- Gestion du Code
- Conversion du code MSIL en natif
- Chargement et exécution du ‘managed code’
- Création et gestion des metadata
- Contrôle des types
- Insertion et exécution des politiques de sécurité
- Gestion mémoire
- Gestion multi-langage des exceptions
- Interopérabilité entre les objets du framework .NET et les objets COM et les DLLs Win32
- Liaison tardive
- Services pour les développeurs (profiling, debugging, etc.)

© 2007, M. Riveill

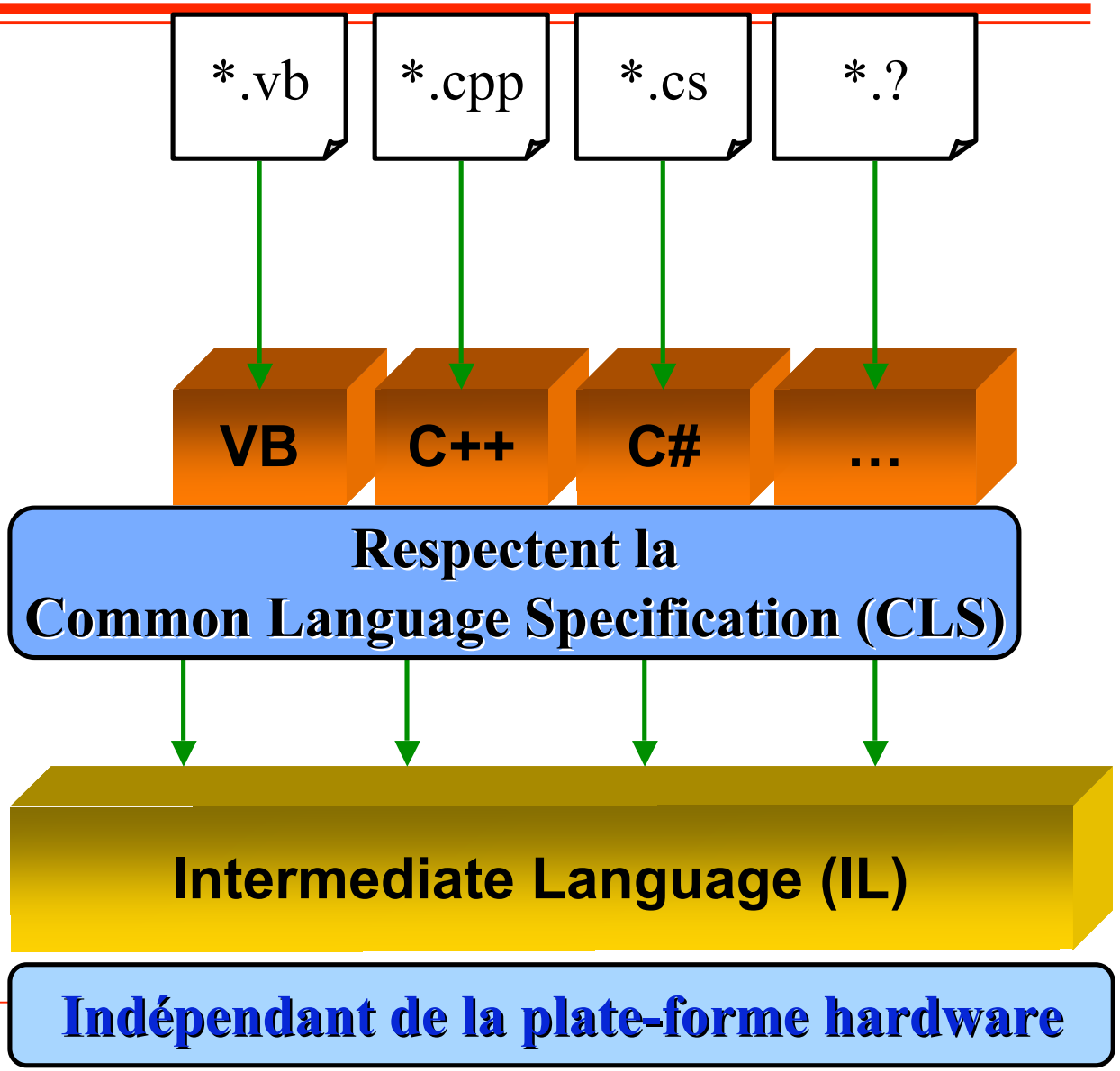
Common Language Runtime

Alias		
Visual Basic.Net	Visual C#.Net	.NET Runtime type structure
Boolean	bool	System.Boolean
Byte	byte	System.Byte
Char	char	System.Char
Date		System.DateTime
Decimal	decimal	System.Decimal
Double	double	System.Double
Integer	int	System.Int32
Long	long	System.Int64
Object	object	System.Object
Short	short	System.Int16
Single	float	System.Float32
String	string	System.String

© 2007, M. Riveill

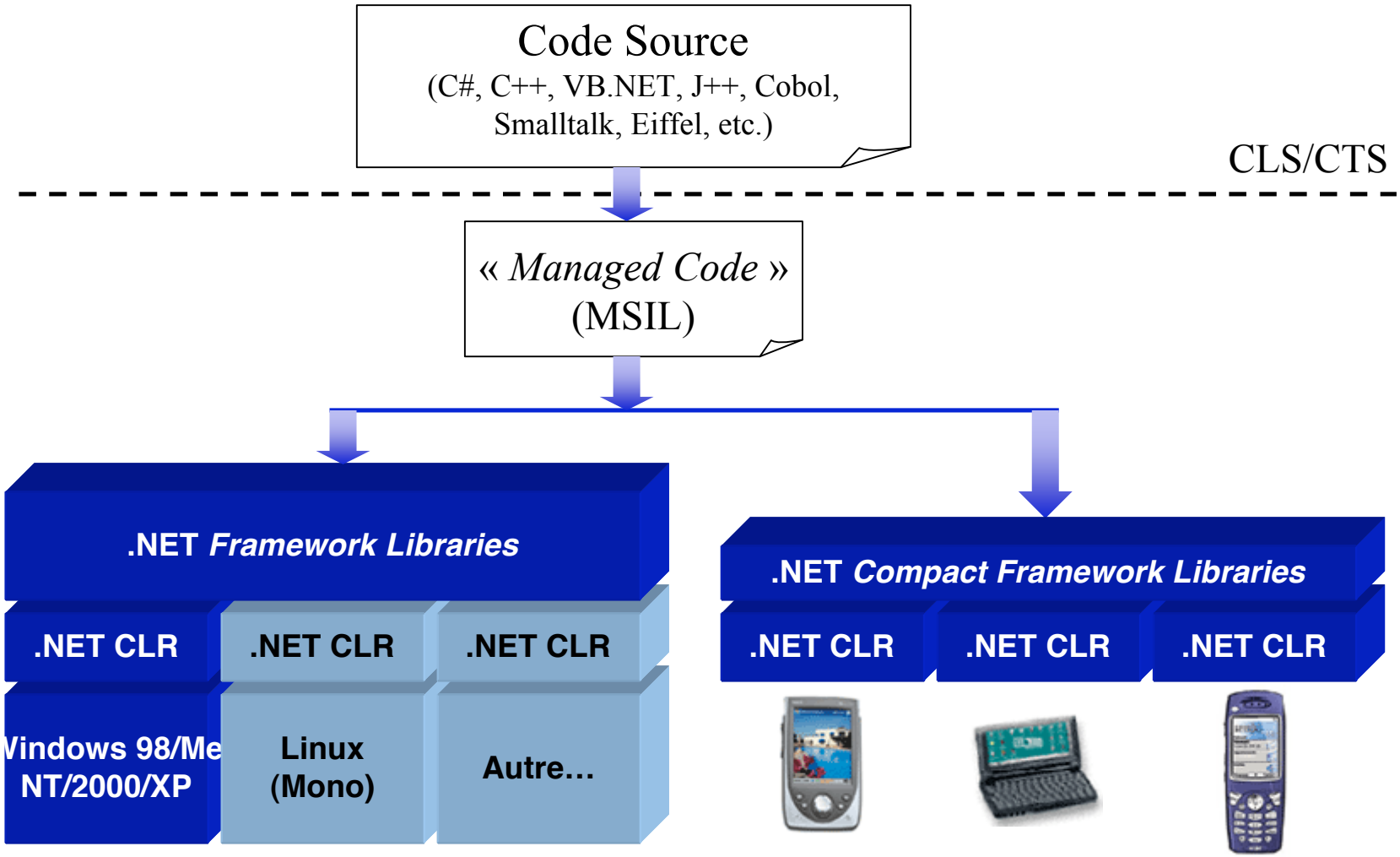
Les langages sur .Net

- Perl
- Python
- Cobol
- Haskell
- ML
- Jscript
- Ada
- APL
- Eiffel
- Pascal
- Fortran
- Managed C++
- Visual Basic
- C#
- SmallTalk
- Oberon
- Scheme
- Mercury
- Oz
- Objective Caml
- J#
- …



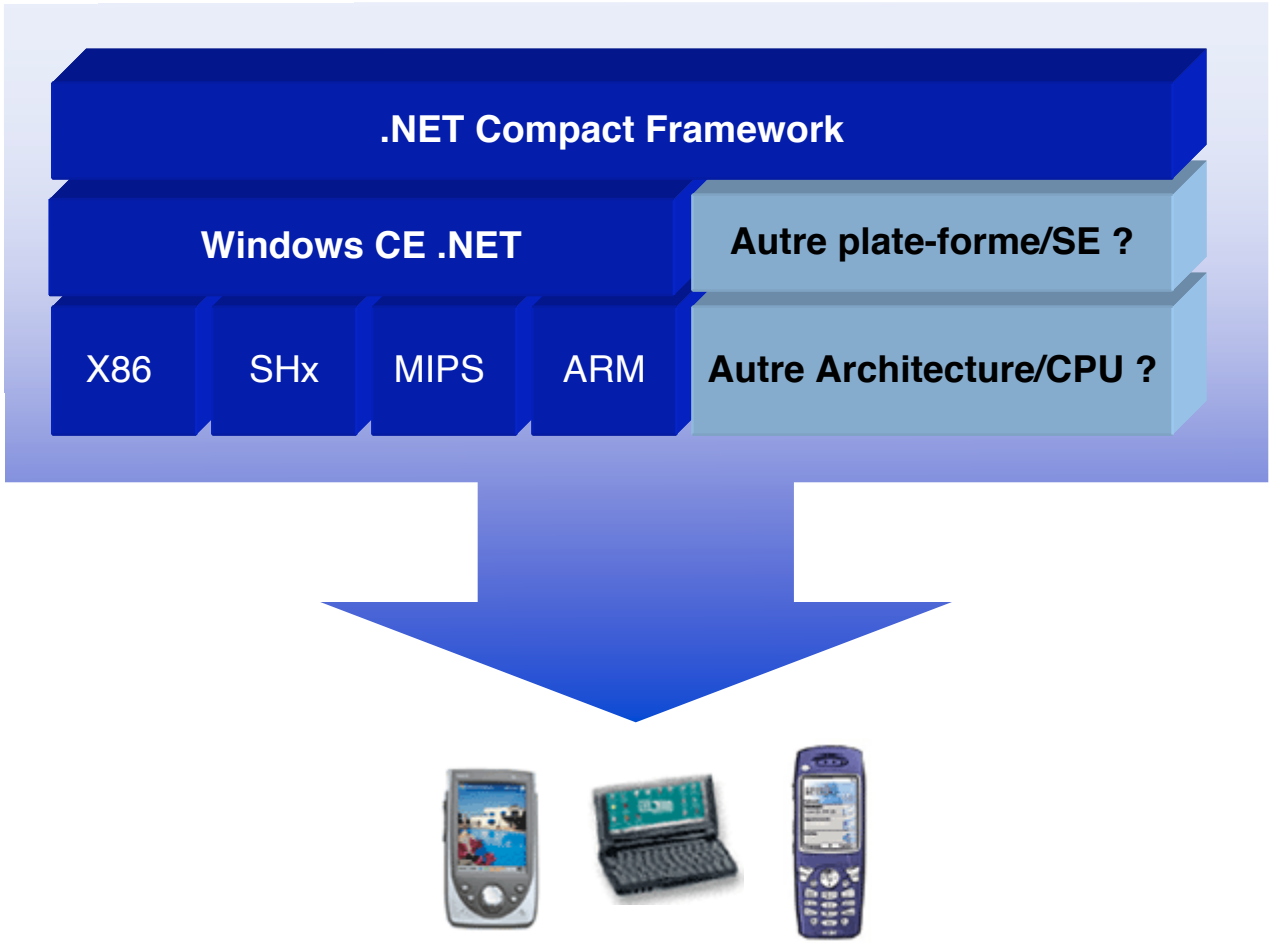
© 2007, M. Riveill

La plate-forme .NET



© 2007, M. Riveill

.NET *Compact Framework*



http://morpheus.developpez.com/cfdotnet/

© 2007, M. Riveill

Compact Framework .NET

- **Même philosophie que le Framework .Net**
  - Sous-ensemble du Framework .NET et du Common Language Runtime.
  - Code intermédiaire, JIT compiler, possibilité de code natif
    - Exécution de programmes indépendants du matériel et des systèmes d'exploitation
  - Prise en charge des principaux protocoles réseaux et connexion avec les services Web
- **Mais quelques spécificités**
  - Contient des classes spécialement conçues à son intention
  - Optimisé pour les dispositifs ayant :
    - Alimentation par batterie
    - Espace de travail compris entre 128 Ko et 1 Mo en RAM dynamique
    - Disque dur facultatif
- **En conséquence**
  - Une application Compact Framework .NET peut s'exécuter sur le Framework .NET si :
    - Les assemblys ne sont pas signés (pas le même mécanisme)
    - Pas d'utilisation des classes spécifiques (exemple IHM pocket PC)

© 2007, M. Riveill

Quelques singularités

- COM interop et fonctions de rappel:
  - **Le Compact Framework .NET n'assure pas l'interopérabilité avec les objets COM, mais vous pouvez utiliser PInvoke (*platform invoke*) pour accéder à des fonctions DLL natives qui, à leur tour, peuvent appeler des objets COM.**
- Répertoire en cours:
  - **La fonctionnalité de répertoire en cours est absente sur les périphériques qui exécutent Windows CE .NET ; par conséquent, le .NET Compact Framework ne prend pas en charge les méthodes** Directory.GetCurrentDirectory **et** Directory.SetCurrentDirectory**.**
- Données:
  - **Le Compact Framework .NET fournit une implémentation de sous-ensemble de ADO.NET et inclut le fournisseur de données SQL Server CE .NET. L'espace de noms System.Data.OleDb n'est pas pris en charge.**
- Tableaux:
  - **Au contraire de certains langages, le *Common Language Runtime* ne prend pas en charge les limites inférieures autres que zéro et lève une exception MissingMethodException si le premier élément d'un tableau n'est pas zéro.**
- ASP.NET:
  - **Le Compact Framework .NET est essentiellement une plate-forme de client élaboré et ne prend pas en charge ASP.NET.**
- Assemblys et GAC (*Global Assembly Cache*) :
  - **Dans sa version actuelle, le .NET Compact Framework ne prend pas en charge les assemblys composés de plusieurs modules, mais prend en charge les assemblys satellites.**
- Classes:
  - **Le .NET Compact Framework prend en charge un sous-ensemble de la Bibliothèque de classes .NET Framework.**
- Types de données et précision en virgule flottante:
  - **Seule Math.Round(double a) est prise en charge ; Math.Round(double a, double b) n'est pas prise en charge.**
- Délégués:
  - **Les délégués asynchrones, en particulier les méthodes** BeginInvoke **et** EndInvoke**, ne sont pas pris en charge.**
- Événements:
  - **Le .NET Compact Framework prend en charge les événements** GotFocus **et** LostFocus**, mais non les événements** Activated **et** Deactivated**.**
- Installation et fichiers CAB:
  - **Vous pouvez utiliser des fichiers CAB et créer des applications MSI pour distribuer vos applications.**
- Test des performances:
  - **Le .NET Compact Framework ne prend en charge ni le profilage de code, ni le fichier Perfmon.exe du Moniteur système.**
- Réflexion:
  - **Le .NET Framework ne prend pas en charge l'espace de noms System.Reflection.Emit.**
- Sérialisation:
  - **Pour des raisons de taille et de performances, le .NET Compact Framework ne prend en charge ni la sérialisation binaire à l'aide de BinaryFormatter, ni la sérialisation SOAP à l'aide de SoapFormatter.**
- Minuteries:
  - **Les méthodes** Timer.Start **et** Timer.Stop **ne sont pas prises en charge, mais vous pouvez démarrer et arrêter une minuterie en affectant à la propriété** Timer.Enabled **la valeur** true **or** false**.**

© 2007, M. Riveill

Deuxième partie

Le langage C# (et le CLR)

Assemblies et module
Types références et valeurs
Delegates et Events
Réflexion et attributs

Languages
C#

- Nouveau langage créé pour .NET
- Proche de C++ et de Java
- Concepts clés :
  - Composants orientés
  - Tout est objet
- Soumis à l'ECMA pour standardisation
- Utilise les classes du framework .NET

- Mais surtout :
  - C# est la fidèle reproduction des concepts de la machine virtuelle de .Net, le CLR

© 2007, M. Riveill

C#

- Ce n'est pas une présentation exhaustive de C#
  - Nous n'allons pas regarder ☹
    - Espace des noms, différents de Java
    - Instructions, pas grand-chose de très original
    - La gestion des threads et la synchronisation
    - L'accès aux objets distants, l'activation (Remoting vs RMI)
    - Opérateurs, proche de C++
    - Types génériques (disponible 2.0)
    - Commentaires
    - Génération de documentation (XML vs HTML JavaDoc)
    - …
  - Nous allons regarder ☺
    - Assemblies et module, signature, version
    - Types références et valeurs, boxing (disponible Java 1.5)
    - Delegates et Events
    - Réflexion et attributs

© 2007, M. Riveill

Assemblies et modules

- .Net
  - La CLR impose que tous les types appartiennent à une assembly
  - L'assembly est l'unité minimale de déploiement de code sur la plateforme .NET
  - Le module est la plus petite unité compilable:
    - Peut contenir des meta-data, du code et des ressources
- C#
  - Compilateur : csc.exe
  - Peut émettre des :
    - Modules => /target:module
    - DLL => /target:library
    - Exécutables console => /target:exe
    - Exécutables windows => /target:winexe
  - pour examiner des assemblies : ildasm.exe

© 2007, M. Riveill

Mon premier assembly : Hello World

```
using System;

class Bonjour {
        public void Print () {
                Console.WriteLine ("Bonjour !!!...");
        }
}

class Hello {
        static void Main( ) {
                Bonjour b = new Bonjour();
                Console.WriteLine("Hello world");
                // taper ‘enter’ pour finir
                Console.ReadLine();
                b.Print();
        }
}
```

```
csc /target:exe HelloWorld.cs
```

© 2007, M. Riveill

Le même en deux assemblies

```
using System;
namespace BonjourNS {
        public  class Bonjour  {
                public void Print () {
                        Console.WriteLine ("Bonjour !!!...");
}}}
```

```
using System;
using BonjourNS;
namespace Hello {
        class Hello {
                static void Main( ) {
                        Bonjour b = new Bonjour();
                        Console.WriteLine("Hello world");
                        Console.ReadLine();
                        b.Print();
}}}
```

```
csc /target:library Bonjour.cs
csc /target:exe /reference:Bonjour.dll HelloWorld.cs
```

Génération d'un module

- Pas de génération de manifeste de l'assembly :
  - /target:module.
  - Par défaut, le fichier de sortie porte l'extension .netmodule.
- Conséquence
  - Ce fichier dépourvu d'un manifeste de l'assembly ne peut pas être chargé par le CLR
  - Il peut être incorporé dans le manifeste d'un assembly au moyen de /addmodule.
- Si plusieurs modules sont créés dans une seule compilation :
  - les types internal d'un module unique seront accessibles aux autres modules dans la compilation.
  - Lorsque le code d'un module référence des types internal dans un autre module, les deux modules doivent être incorporés dans un manifeste de l'assembly au moyen de /addmodule.

© 2007, M. Riveill

Le même en un seul assembly mais deux fichiers

```
using System;
namespace BonjourNS {
        internal  class Bonjour  {
                public void Print () {
                        Console.WriteLine ("Bonjour !!!...");
}}}
```

```
using System;
using BonjourNS;
namespace Hello {
        class Hello {
                static void Main( ) {
                        Bonjour b = new Bonjour();
                        Console.WriteLine("Hello world");
                        Console.ReadLine();
                        b.Print();
}}}
```

```
csc /target:Module Bonjour.cs
csc /target:exe /addmodule:Bonjour.netmodule HelloWorld.cs
```

© 2007,

Assemblies et modules
Portée

- La portée des types est spécifiée dans l'assembly elle-même
  - public indique que le type est vu depuis l'extérieur de l'assembly
  - internal indique que le type n'est vu que des autres types de cette même assembly
  - private indique que le type n'est vu que de l'intérieur du type qui le déclare
  - static indique qu'il s'agit d'une variable commune à toutes les instances

© 2007, M. Riveill

Assemblies et modules

Signature

- La signature d'une assembly est basée sur un couple Clé publique/Clé privée
- Une assembly signée ne peut être altérée sans devoir être re-signée à nouveau
- sn.exe est l'outil permettant de gérer cela
- AssemblyKeyFile et AssemblyDelaySign sont les attributs à utiliser

© 2007, M. Riveill

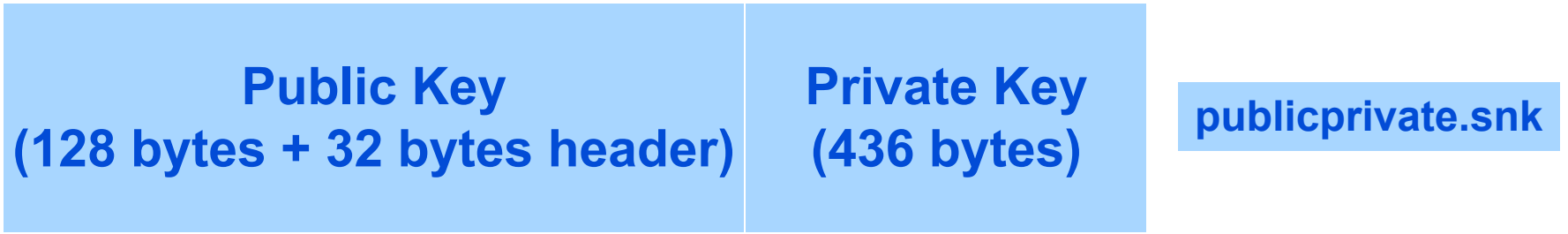
Pourquoi signer un assembly ?

- **Signer permet d'obtenir un nom fort :**
  - un nom unique
    - Nul ne peut générer le même nom d'assembly que le vôtre, car un assembly généré avec une clé privée possède un nom différent de celui d'un assembly généré avec une autre clé privée.
  - Les noms forts protègent l'enregistrement en ligne de la version d'un assembly.
    - Un nom fort peut garantir que nul ne peut produire une version ultérieure de votre assembly.
    - Les utilisateurs peuvent être assurés qu'une version de l'assembly qu'ils chargent provient du même éditeur qui a créé la version avec laquelle l'application a été générée.
  - Les noms forts assurent un solide contrôle d'intégrité.
    - Le contenu de l'assembly n'a pas été modifié depuis sa génération.
- **ATTENTION :**
  - les noms forts n'est pas une signature numérique, ni un certificat

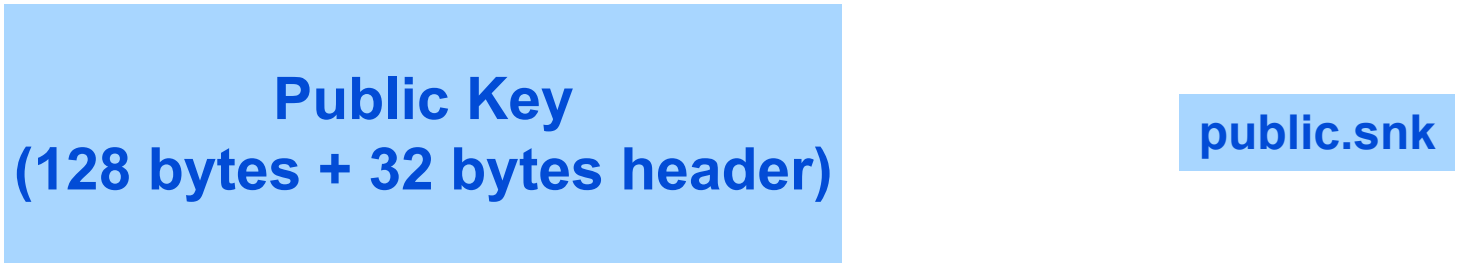
© 2007, M. Riveill

Les clés

sn.exe -k publicprivate.snk



sn.exe -p publicprivate.snk public.snk



© 2007, M. Riveill

Signature d'un assembly

MaLib.cs

```
using System.Reflection;

[assembly: AssemblyKeyFile("fichierdecles.snk")]
[assembly: AssemblyDelaySign(false)]
```

csc.exe /t:library MaLib.cs

MaLib.dll

PE/COFF Header	Public Key (128 bytes + 32 bytes header)	Signature	Code
	CLR Metadata		

fichierdecles.snk

Public Key (128 bytes + 32 bytes header)

Private Key (436 bytes)

csc.exe /t:exe /r:MaLib.dll MonApp.cs

MonApp.exe

PE/COFF Header	MaLib Public Key Token (8 bytes) Assembly reference	Code
	CLR Metadata	

Signature retardée d'un assembly

MaLib.cs

```
using System.Reflection;

[assembly: AssemblyKeyFile("public.snk")]
[assembly: AssemblyDelaySign(true)]
```

public.snk

Public Key (128 bytes + 32 bytes header)

csc.exe /t:library MaLib.cs

MaLib.dll

PE/COFF Header

Public Key (128 bytes + 32 bytes header)

Espace libre

CLR Metadata

Code

fichierdecles.snk

sn.exe -R MaLib.dll fichierdecles.snk

MaLib.dll

PE/COFF Header

Public Key (128 bytes + 32 bytes header)

Signature

CLR Metadata

Code

Public Key (128 bytes + 32 bytes header)

Private Key (436 bytes)

© 20

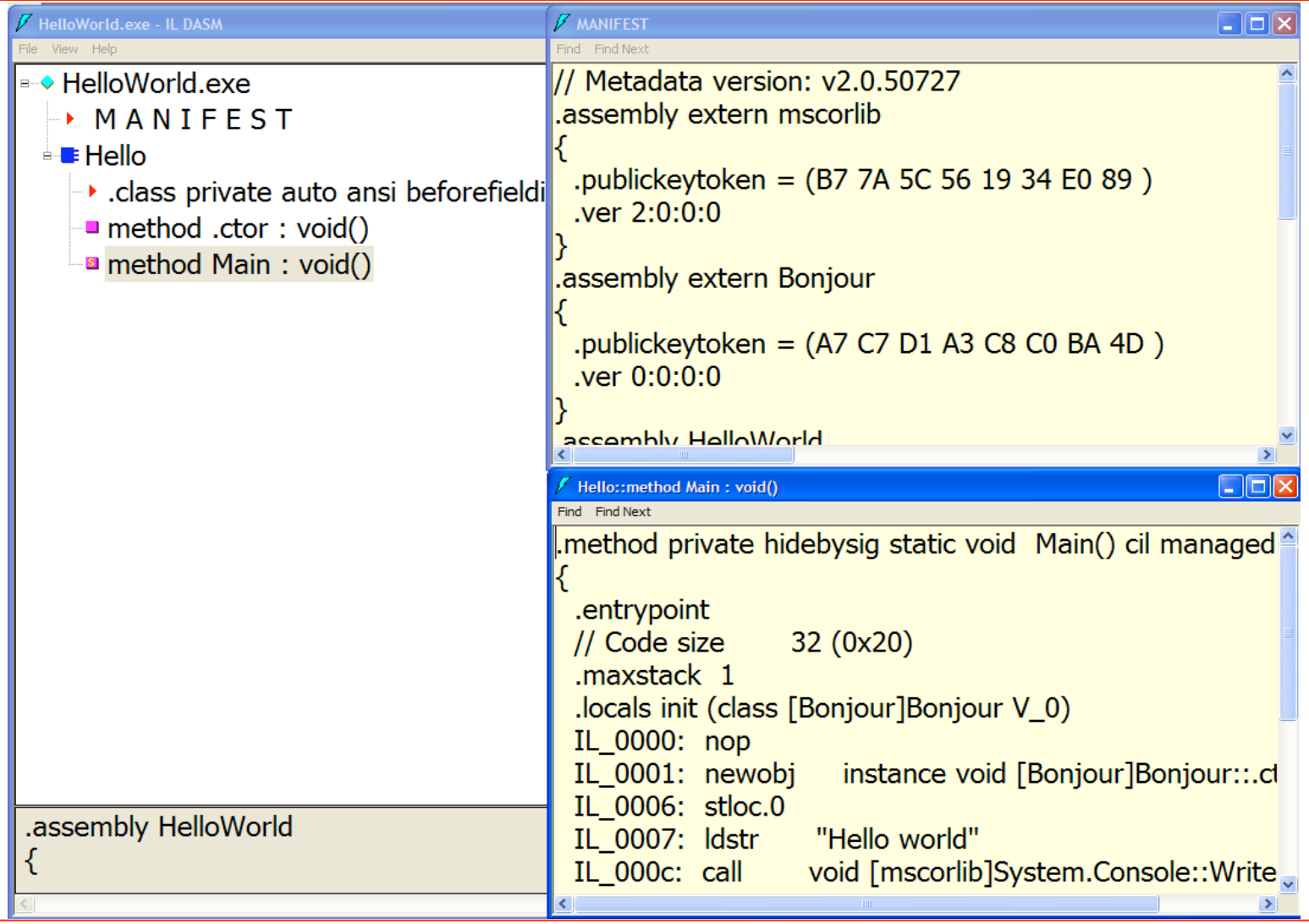
Signature retardée

- Retarder la signature permet de séparer les processus de développement de ceux de la signature

- Pour suspendre la vérification de la signature sur une assembly retardée:

```
sn -Vr MonAssembly.dll
```

Examinateur de code : Ildasm.exe



© 2007, M. Riveill

Déploiement d'un assembly

- GAC : 'Global Assembly Cache'
  - Cache qui contient tous les assemblies publiés
    - `C:\WINDOWS\assembly`
  - Permet le partage des assemblies entre application
- Ou déployer
  - Un répertoire quelconque de la station
    - Pas de gestion version
  - Dans le GAC
    - Partage de l'assembly entre toutes les applications
    - Gestion de version
- Comment publier un assembly dans le GAC
  - Pour être publier un assembly doit avoir un nom fort
  - `gacutil /i nom_de_la_dll.dll`
    - `/u` pour désinstaller
    - /l pour lister le contenu du GAC
  - Ou à l'aise de l'explorateur Windows

© 2007, M. Rivelli

Gestions de versions

- **Plusieurs versions d'un même assembly peuvent être exécutées simultanément**
  - Exécution 'côte à côte'('side by side')
- **Le multi-version nécessite**
  1. Donner un numéro de version
     - Utilisation d'un attribut : `<Assembly: AssemblyVersion ("1.0.1.0")>`
     - Majeure.Mineure.Génération.Révision
  2. Donner un 'nom fort' ('strong name')
  3. Publier l'assembly dans la gac ('Global Assembly Cache')

© 2007, M. Riveill

Assemblies et modules

Fichier de configuration

- Fichier XML permettant de spécifier un certain nombre de paramètres à l'assembly
- Créer un fichier à extension .config

```
MonAssembly.dll
MonClient.exe

=> MonClient.exe.config
```

Assemblies et modules

Redirection

- On peut rediriger un client vers une assembly spécifique même si il avait été conçu pour une autre à l'origine
- On ne peut déployer qu'une amorce pour son application et télécharger le reste des assemblies dynamiquement
- Utiliser le fichier de configuration XML

© 2007, M. Riveill

Assemblies et modules

Redirection: bindingRedirect

```
<?xml version="1.0" ?>
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity
                name="MonAssembly"  publicKeyToken="32ab4ba45e0a69a1" />
            <bindingRedirect oldVersion="1.0.0.0" newVersion="2.0.0.0"/>
         </dependentAssembly>
      </assemblyBinding>
   </runtime>
</configuration>
```

```
<?xml version="1.0" ?>
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity
                name="MonAssembly"  publicKeyToken="32ab4ba45e0a69a1" />
            <bindingRedirect oldVersion="1.0.*-1.6.*" newVersion="2.0.0.0"/>
         </dependentAssembly>
      </assemblyBinding>
   </runtime>
</configuration>
```

© 2007

Assemblies et modules

Téléchargement: codeBase

```
<?xml version="1.0" ?>
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity
                name="MonAssembly"  publicKeyToken="32ab4ba45e0a69a1" />

            <bindingRedirect oldVersion="1.0.0.0" newVersion="2.0.0.0"/>

            <codeBase version="2.0.0.0"href="http://.../MonAssembly.dll"/>
            <codeBase version="2.0.0.0"href="ftp://.../MonAssembly.dll"/>
            <codeBase version="2.0.0.0"href="file://.../MonAssembly.dll"/>

         </dependentAssembly>
      </assemblyBinding>
   </runtime>
</configuration>
```

Assemblies et modules

Conclusion

- L'assembly est le composant de la CLR
- Les types n'appartiennent qu'à une et une seule assembly
- Les assemblies peuvent être publiés dans un répertoire partagé (le gac)

© 2007, M. Riveill

Demo

‘Hello world’

Désassembleur

Signature d’un assembly

Publication - Gestion de version

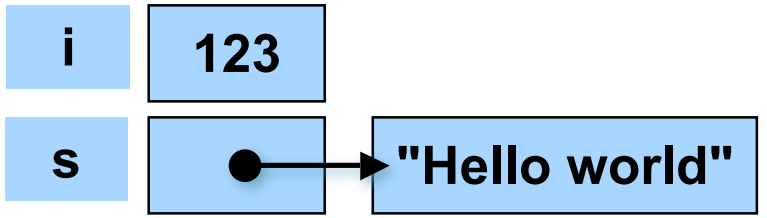
Types

- Un programme C# est une collection de type
  - Classes, structures, énumérations, interfaces, delegates, tableaux
- Chaque programmeur peut créer ses propres types
  - types prédéfinis : int, byte, char, string, object, …
- Les types sont :
  - Des données membres : champs, constantes, tableaux, évènements
  - Des fonctions membres : Méthodes, opérateurs, constructeurs, destructeurs, propriétés, indexeurs
  - D'autres types : classes, structures, énumérations, interfaces, delegates
- Les types peuvent être instancié…
  - …puis utilisés : appel de méthode, get et set de propriétés, etc.
- Les types peuvent être convertis, implicitement ou explicitement
- Les types sont organisé en espace de noms, fichiers et assemblies
  - l'ensemble forme une hiérarchie
- Il y a 2 catégories de types : valeur et référence

Types
Un seul système de type

- **Type valeur**
  - Contient des données
  - Ne peut être 'null'
  - Primitives `int i; float x;`
  - Enumérations `enum State { Off, On }`
  - Structures `struct Point {int x,y;}`
- **Type référence**
  - Contient des références vers des objets
  - Peut être 'null'
  - Racine `object`
  - Chaîne `string`
  - Classes `class Foo: Bar, IFoo {...}`
  - Interfaces `interface IFoo: IBar {...}`
  - Tableau `string[ ] a = new string[10];`
  - Delegates `delegate void Empty();`

```
int i = 123;
string s = "Hello world";
```



Types
Un seul système de type

	Value (Struct)	Reference (Class)
Variable holds	**Actual value**	**Memory location**
Allocated on	**Stack, member**	**Heap**
Nullability	**Always has value**	**May be null**
Default value	**0**	**null**
Aliasing (in a scope)	**No**	**Yes**
Assignment means	**Copy data**	**Copy reference**

- **Bénéfice des types valeurs**
  - Pas d'allocation dans le tas, moins de travail pour le GC
  - Meilleure utilisation de la mémoire
  - Moins de référence indirecte
  - Un seul système de type
    - Pas de dichotomie type primitif/objet

© 2007, M. Riveill

Types
Conversions

- Conversion implicite
  - Fonctionne automatiquement et sans possibilité d'échec
  - Pas de perte de précision/d'information
- Conversion explicite
  - Nécessite un 'cast'
  - Peut être refusée
  - De l'information/précision peut être perdue
- Conversion implicite et explicite sont précisées par le programmeur

```
int x = 123456;
long y = x;                          // implicit
short z = (short)x;                  // explicit

double d = 1.23456789012345;
float f = (float)d;                  // explicit
long l = (long)d;                    // explicit
```

© 2007, M. Riveill

Types
Un système de type unifié

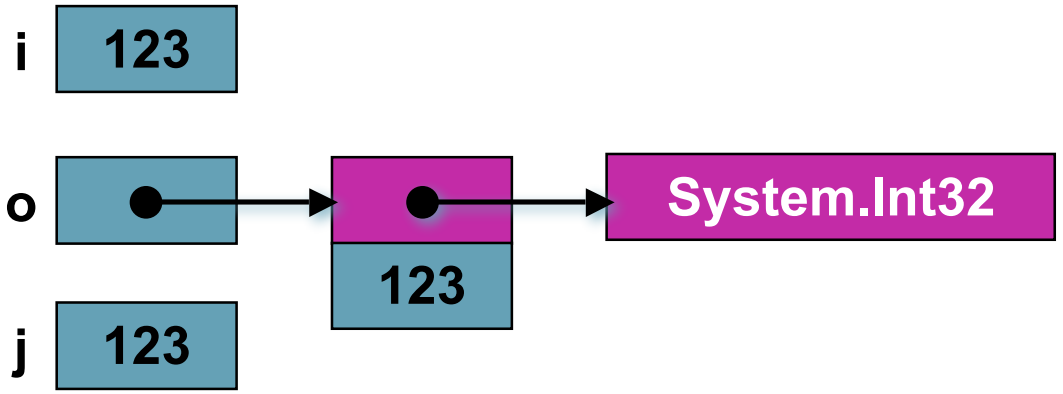
- Question : Comment peut-on traiter les valeur et les références de la même manière ?
  - int (type valeur) ←→ objet (type référence) ?
- Réponse : Boxing!
  - uniquement pour les types valeurs
  - Dans version 1.5 de java (avec la généricité)
- Boxing
  - Copie un type valeur dans un type référence (objet)
  - Chaque type valeur à une correspondance “cachée” dans un type référence
  - Une copie d’un type référence est interprétée comme une copie du type valeur
    - Les types valeurs n’ont jamais de synonyme (d’alias)
  - Conversion implicite
- Unboxing
  - Copie un type référence dans un type valeur
  - Nécessite une conversion explicite
    - Peut échouer

© 2007, M. Riveill

Types
Un système de type unifié

- Bénéfice du boxing,
  - Permet le polymorphisme pour tous les types (y compris les types valeurs)
  - Les classes 'collections' fonctionnent avec tous les types
  - Elimine la nécessité de construire manuellement des classes d'encapsulation
  - On trouve plein d'exemple d'utilisation dans le Framework .NET
- Désavantage du boxing
  - Performance
  - L'utilisation de la généricité a remplacé la nécessité du boxing
    - C++ templates – C# intrinsics - Java generics

```
int i = 123;

object o = i;

int j = (int)o;
```



© 2007, M. Riveill

Types

Méthodes définies sur Object

- public bool Equals(object)
  - Prend en charge la comparaison entre objet
- protected void Finalize()
  - ‘nettoyage’ avant de récupérer un objet
- public int GetHashCode()
  - Appelé pour insérer l’objet dans une table de hachage
- public System.Type GetType()
  - Donne le type de l’objet
- protected object MemberwiseClone()
  - Construit un clone de l’objet
- public string ToString()
  - Construit une chaîne de caractère qui décrit une instance de la classe

© 2007, M. Riveill

Types : classes et structures

this

- Le mot clé this est une variable prédéfinie accessible dans chaque fonction membre non statique
  - Utilisé pour supprimer les ambiguïtés lors de l'accès aux données ou aux fonctions membres

```
class Person {
  string name;
  public Person(string name) {
    this.name = name;
  }
  public void Introduce(Person p) {
    if (p != this)
      Console.WriteLine("Hi, I'm " + name);
  }
}
```

© 2007, M. Riveill

Types : classes et structures

base

- Le mot clé base est utilisé pour accéder aux membres de classes masqués par un nom similaire de la classe courante
- La surcharge d'une méthode doit être explicitée (override)

```
class Shape {
  int x, y;
  public override string ToString() {
    return "x=" + x + ",y=" + y;
  }
}
class Circle : Shape {
  int r;
  public override string ToString() {
    return base.ToString() + ",r=" + r;
  }
}
```

© 2007, M. Riveill

Classes et Structures
Constantes

- Une constante est une donnée membre qui est évaluée à la compilation
  - Elle est implicitement statique
  - i.e. Math.PI

```
public class MyClass {
  public const string version = “1.0.0”;
  public const string s1 = “abc” + “def”;
  public const int i3 = 1 + 2;
  public const double PI_I3 = i3 * Math.PI;
    //ERROR
  public const double s = Math.Sin(Math.PI);
  ...
}
```

© 2007, M. Riveill

Classes et Structures

Champs non modifiable ('read-only')

- Similaire à une constante, mais initialisé à l'exécution
  - Une fois initiatisé, il ne plus être modifié
- Différents à une constante
  - Initialisé à l'exécution (vs. à la compilation)
    - Il n'est pas nécessaire de re-compiler les client de la classe/struct qui le défini
  - Il peut être statique (et partagé par les différentes instances) ou propre à chaque instance

```
public class MyClass {
  public static readonly double d1 = Math.Sin(Math.PI);
  public readonly string s1;
  public MyClass(string s) { s1 = s; }
}
```

© 2007, M. Riveill

Classes et Structures

Propriétés

- Une propriété est un champ virtuel
- Ressemble à un champ, mais est implémenté par du code
- Peut être read-only, write-only ou read/write

```
public class Button: Control {
   private string caption;
   public string Caption {
      get { return caption; }
      set { caption = value;
            Repaint(); }
   }
}
```

```
Button b = new Button();
b.Caption = "OK";
String s = b.Caption;
```

© 2007, M. Riveill

Classes et Structures

Indexeurs

- Un indexeur permet à une instance d'être un tableau virtuel
- Peut être surchargé (i.e. indexé par int et par string)
- Peut être read-only, write-only ou read/write

```
public class ListBox: Control {
   private string[] items;
   public string this[int index] {
      get { return items[index]; }
      set { items[index] = value;
            Repaint(); }
   }
}
```

```
ListBox listBox = new ListBox();
listBox[0] = "hello";
Console.WriteLine(listBox[0]);
```

© 2007, M. Riveill

Types
Delegates

- Un délégué (delegate) est un type référence qui défini la signature d'une méthode
- Quand il est instancié, un délégué peut faire référence à une ou plusieurs méthodes
  - De manière intuitive :
    - delegate = un pointeur sur une fonction dans le modèle objet
- Sert de base pour la gestion des évènements

Delegates

- Un ‘delegate’ est une référence sur une signature de méthode
- Une instance de ‘delegate’ défini une ou plusieurs méthodes
  - Les méthodes peuvent être statique ou non
  - Les méthodes peuvent retourner une valeur
- Permet le polymorphisme sur les fonctions/méthodes
- La base pour la programmation par évènement

```
delegate double Del(double x);          // Declare

static void DemoDelegates() {
  Del delInst = new Del(Math.Sin);      // Instantiate
  double x = delInst(1.0);              // Invoke
}
```

© 2007, M. Riveill

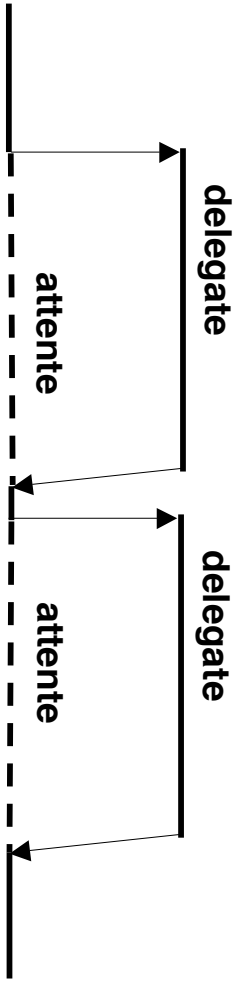
Delegates multicast

- Un delegate peut appeler plusieurs méthodes
  - Les delegates non multicast peuvent contenir des fonctions
  - Les delegates multicast ne contiennent pas de fonction
    - le paramètre de retour est void
- Chaque delegate possède sa propre liste d'appel
  - Les méthodes sont appelées séquentiellement, dans leur ordre d'ajout
- Opérateurs += et -=
  - Utilisés pour insérer supprimer une méthode à la liste du delegate
  - Fonctionnent correctement avec les threads

© 2007, M. Riveill

Delegates

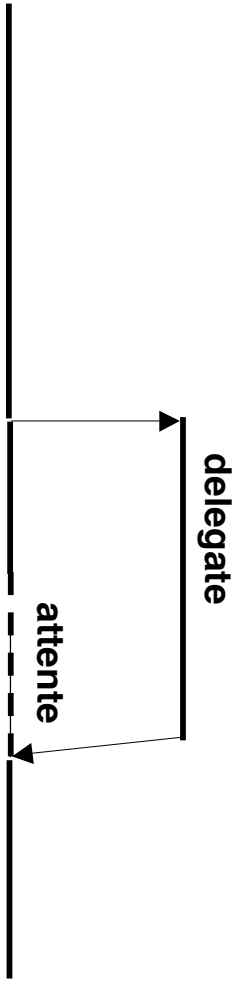
Multicast Delegates




```
delegate void SomeEvent(int x, int y);

static void Foo1(int x, int y) {
  Console.WriteLine("Foo1");
}

static void Foo2(int x, int y) {
  Console.WriteLine("Foo2");
}

public static void Main() {
  SomeEvent func = new SomeEvent(Foo1);
  func += new SomeEvent(Foo2);
  func(1,2);              // Foo1
                                  // et Foo2 sont appelés

  func -= new SomeEvent(Foo1);
  func(2,3);                      // Seul Foo2 est appelé
}
```

© 2007, M. Riveill

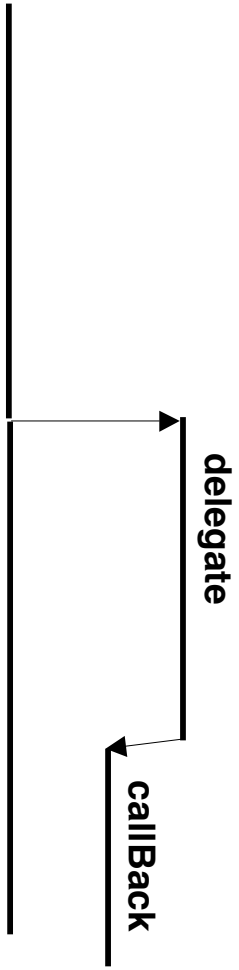
Delegate
Appel asynchrone




```
int resultat;
IAsyncResult asyncResult;

// création du delegate
monDelegate = new …

// appel du delegate
asyncResult = monDelegate.BeginInvoke(null, null);

// suite de l'activité

// attente de la fin d'exécution du delegate
resultat = monDelegate.EndInvoke (asyncResult);

Console.WriteLine ("Valeur de retour du delegate {0}", resultat);
```

© 2007, M. Riveill

Delegate
Appel asynchrone




```
// code du callback
public void callBack (IAsyncResult asyncResult) {
      delegateAppele = asyncResult.AsyncState;
      int resultat = delegateAppele.EndInvoke (asyncResult);
      Console.WriteLine ("Valeur de retour du delegate {0}", resultat);
}

// création du callback
AsyncCallback asyncCallback = new AsyncCallback (callBack);

// création du delegate
monDelegate = new …

// appel du delegate
IAsyncResult asyncResult = monDelegate.BeginInvoke(asyncCallback,
      monDelegate);

// suite de l'activité
```

© 2007, M. Riveill

Thread
Déclaration

```
public class ActEnPar {
  public ActEnPar () {
    thread = new Thread (new
    ThreadStart (Activite));
  }
  public void DemarrageFork () {
    thread.Start();
  }
  public void AttenteJoin () {
    thread.Join();
  }
  public void Activite() {
    // code de l'activité
  }
}
```

- En C#, utilisation d'un delegate (qui ne prend aucun paramètre et rend aucune valeur)

```
public class ActEnPar implements
    Runnable {
  public ActEnPar () {
    thread = new Thread(this);
  }
  public void demarrageFork () {
    thread.start();
  }
  public void attenteJoin () {
    thread.join();
  }
  public void run() {
     // code de l'activité
  }
}
```

Thread
Utilisation

```
class ClasseUtilisatrice {
  public static void Main(string[]
    args) {
     ActEnPar act = new ActEnPar ();
     act.DemarrageFork ();
     // D'autres activité peuvent
    être démarrées ici

     act.AttenteJoin ();
  }
}
```

```
public class ClasseUtilisatrice {
  public static void main(String[]
    args) {
     ActEnPar act = new ActEnPar ();
     act.demarrageFork ();
     // D'autres activité peuvent
    être démarrées ici

     act.attenteJoin ();
  }
}
```

Evènements

- La programmation par évènement permet à un objet de prévenir un ensemble d'objets en attente de la production d'un évènement
  - Modèle 'publish-subscribe'
  - Basé sur un mécanisme de 'call-back'
  - Le programmeur se concentre sur les réactions à exécuter et non pas sur le lien évènement-réaction
  - Code plus lisible

© 2007, M. Riveill

Evènements

- Les évènement sont très utilisés lors de la construction d'IHM
  - L'utilisateur fait une action (click sur un bouton, déplace la souris, change une valeur, etc.) et le programme réagit
  - Il existe de nombreuses autres utilisation, i.e.
    - Évènements temporels
    - Gestion des exécutions asynchrones (prévenir qu'un email est arrivé, qu'une session Web a démarré)
  - Modèle de programmation fréquent
    - WinForm
    - Pages ASP
    - Les environnements de développement proposent de ne programmer que la réaction à des évènements connus

© 2007, M. Riveill

Evènement

- C# supporte les évènements
  - L'émetteur de l'événement
    - défini le délégué
    - il sera exécuté à chaque occurrence de l'événement
  - Les clients de l'événement
    - enregistrent le traitement à associer à l'évènement
      - Appel de += (ou de -=) sur les délégués
    - ne peuvent pas créer les évènements
  - Utilisation de délégués multicast
    - plusieurs objets peuvent enregistrer une réaction pour le même évènement

© 2007, M. Riveill

Evènements

Exemple coté composant (émetteur)

```
public class Button {
  // Déclarer le delegate
  public delegate void SampleEventHandler (object sender,
                                           EventArgs e);

  // Déclarer l'événement
  public event SampleEventHandler click;

  // encapsuler l'événement dans une méthode virtuelle
  // afin que les classes dérivées puissent lever
  // l'événement et demander l'éxécution de OnClick
  protected virtual void OnClick(EventArgs e) {
    if (click != null) {
      // exécuter les réactions enregistrées
      click(this, e);
    }
  }
}
```

Evènements

Exemple : coté utilisateur de l'évènement

```
public class MyForm: Form {
  // variable de la classe précédente
  Button okButton;

  // Définition de la réaction
  static void OkClicked (object sender,
                         EventArgs e) {
    ShowMessage("You pressed the OK button");
  }

  public MyForm() {
    // création du 'button' si nécessaire
    okButton = new Button(...);
    okButton.Caption = "OK";
    // enregistrement de la réaction
    okButton.Click +=
       new EventHandler(OkClicked);
  }
}
```

© 2007, M. Riveill

Evénement – un autre exemple

- **Définir éventuellement un événement**

```
public class EventFeuArgs: EventArgs {
    public string piece;            // le lieu de l'incendie
    public int force;               // la puissance du feu

    // constructeur
    public EventFeuArgs(string piece, int force) {
            this.piece = piece;
            this.force = force;
    }
}
```

© 2007, M. Riveill

Evénement - exemple

- **Définir la classe qui génère l'évènement et appelle le délégué associé (contiendra l'ensemble des réactions)**

```
public class Alarme {
```

- **Définir la signature de la fonction associée à l'événement comme un délégué**

```
    public delegate void EventFeuHandler(object sender, EventFeuArgs feu);
```

- **Définir la variable qui contiendra les délégués**

```
    public EventFeuHandler EventFeu;
```

- **Définir la logique d'émission par appel du delegué**

```
    public void ActivationAlarme(string piece, int force) {
      EventFeuArgs feuArgs = new EventFeuArgs(piece, force);

      EventFeu(this, feuArgs);
    }
}
```

© 2007, M. Riveill

Evènement - exemple

- **Définir la classe qui propose un traitement à l'évènement**

```
class Secours {
```

- **Définir le traitement à associer à l'évènement**

```
  void EteindreFeu(object sender, EventFeuArgs feu) {
    Console.WriteLine("Feu dans piece {0}.",  feu.piece);
    if (feu.force < 5)
      Console.WriteLine("Utiliser un instincteur");
    else
      Console.WriteLine("Appeler les pompiers);
  }
```

- **S'abonner à l'évènement**

```
  // constructeur
  public Secours(Alarme alarme) {
    alarme.EventFeu += new Alarme.EventFeuHandler(EteindreFeu);
  }
}
```

© 2007, M. Riveill

Evènement – exemple

- **Utilisation**

```
public class Test {
  public static void Main () {
  // Créer une instance de la classe qui lève les alarmes
  Alarme monAlarme = new Alarme();
  // Créer une instance de la classe Secours
  Secours incident = new Secours(monAlarme);
  // Utiliser la classe pour émettre quelque évènements
  monAlarme.ActivationAlarme("Cuisine", 3);
  // Créer une autre instance de la classe Secours
  AutreSecours bis = new AutreSecours(nouvelleAlarme);
  }
}
```

© 2007, M. Riveill

Demo

Delegates : multicast et appel asynchrone

Evènements

Les attributs
Etendre les informations des types

- Les attributs permettent:
  - d'associer des informations « utilisateurs » aux types (version, URL de la documentation, COM ProgID, …)
  - de contrôler l'exécution (serialization, mapping XML, remoting, activation transaction, …)
- La CLR propose un certain nombre d'attributs en standard
- Décoration du code
  - Assembly, module, type, membre, paramètres et valeur de retour

© 2007, M. Riveill

Les attributs

- Les attributs sont
  - Intégrés au code sans modification de la syntaxe d'origine
  - Extensibles
    - Ajout d'informations utilisateurs
  - Sûr
    - le type des arguments est vérifié lors de la compilation
  - Pris en charge par le framework .NET
  - Commun à tous les langages de programmation
    - Sauvegardé dans l'assembly metadata
  - Utilisés dans tous le framework .Net
    - XML, services Web, sécurité, serialisation, modèle de composant, intéroperabilité avec COM, configuration du code…
- Les attributs peuvent être
  - Examiné à l'exécution en utilisant la réflexivité

© 2007, M. Riveill

La réflexion / introspection

- **Le CLR supporte la réflexivité**
  - Permet d'inspecter dynamiquement le contenu d'un assembly
  - De créer des instances de types
  - Principales méthodes
    - Assembly.GetTypes
      - Renvoie un tableau de 'System.Type' contenant tous les types définis dans l'assemblage
    - Assembly.GetMembers
      - Renvoie un tableau de 'System.Reflection.MemberInfo' contenant les signatures de tous les membres de chaque type de l'assemblage
    - Activator.CreateInstance
      - Crée dynamiquement une instance du type spécifié
    - Type.InvokeMember
      - Tente d'invoquer le membre spécifié sur le type

© 2007, M. Riveill

Introspection

Exemple

```
// sur les types
Assembly monAssembly = Assembly.LoadFrom(''aFiles'');
Foreach (Type type in monAssembly.GetTypes()) {
    Console.WriteLine(''Type présent dans l'asssembly : '' + type.Name);
}
// sur les membres
aType anInstance = new aType ();
// Employe employe = new Employe();
Foreach (MemberInfo memberInfo in anInstance.GetTypes().GetMembers()) {
    Console.WriteLine(''{0}'', memberInfo.Name);
}
```

```
public class Employe {

    public int Id;
    public decimal Salaire;
    public bool Cdi;

    public string Fonction(int A) { return("OK"); }

    public int Propriete {
        get { return(2); }
        set { }
    }
}
```

```
Id
Salaire
Cdi
GetHashCode
Equals
ToString
Fonction
get_Propriete
set_Propriete
GetType
.ctor
Propriete
```

Attributs

Etendre les informations des types

- **Syntaxe:**

```
using System;

[assembly: AssemblyVersion("1.0.*")]

[MonAttribut1()]
public class MaClasse {

    [MonAttribut2(True)]
    public int MonMembre = 0;

    [MonAttribut3("Valeur", false)]
    public void MaMethode() {

    }

    [MonAttribut4()]
    public string MaPropriete {
        get {
            return(String.Empty);
        }
    }
}
```

Attribut sur une assembly

Attribut sur une classe

Attribut sur un membre

Attribut sur une méthode

Attribut sur une propriété

© 2007, M. Riveill

Attributs

Exemple : Sérialisation

- La sérialisation (binaire ou XML) des types est automatiquement prise en charge par la CLR
- On peut intervenir sur le comportement en positionnant un certain nombre d'attributs

```
[Serializable()]
public class MaClasse {
    public MaClasse() { }

    [NonSerialized()]
    public int MonMembre = 0;

    private string maPropriete = String.Empty;

    public string MaPropriete {
        get { return(this.maPropriete); }
        set { this.maPropriete = value; }
    }
}
```

© 2007, M. Riveill

Attributs

Exemple : Sérialisation

```
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;



MaClasse maClasse = new MaClasse();
maClasse.MaPropriete = "MaValeur";
maClasse.MonMembre = 1000;

// On sérialise et on sauvergarde dans un fichier
Stream stream = File.Open("Demo.bin", FileMode.Create);
BinaryFormatter formatter = new BinaryFormatter();
formatter.Serialize(stream, maClasse);
stream.Close();



// On désérialise depuis un fichier
stream = File.Open("Demo.bin", FileMode.Open);
formatter = new BinaryFormatter();
MaClasse obj = (MaClasse)formatter.Deserialize(stream);
stream.Close();
```

© 2007, M. Riveill

Attributs

Exemple: Sérialisation

```
// MaPropiete a bien été sérialisée
Console.WriteLine("maClasse.MaPropriete = {0}", obj.MaPropriete);

// MonMembre n'a pas été sérialisé
Console.WriteLine("maClasse.MonMembre = {0}", obj.MonMembre);
```

```
maClasse.MaPropriete = MaValeur
maClasse.MonMembre = 0
```

Attributs

Custom attributes

- On peut étendre le .NET Framework avec ses propres attributs
- Exemple :
  - Construire un attribut permettant d'indiquer l'auteur d'une classe et sa version

```
[MonAttribut ("Moi", 3]
public class AccesAuxDonnees {
    // code de la classe
}
```

© 2007, M. Riveill

La reflection et les attributs

Custom attributes

```
[AttributeUsage (AttributeTargets.Class)]
public class MonAttributAttribute : Attribute {
        private string codeCreePar = String.Empty;
        public string CodeCreePar {
                get { return (this.codeCreePar); }
                set { this.codeCreePar = value; }
        }
        private int codeVersion = 0;
        public int CodeVersion {
                get { return (this.codeVersion); }
                set { this.codeVersion = value; }
        }
        public MonAttributAttribute () {}
        public MonAttributAttribute (string owner, int version) {
                this.codeCreePar = owner;
                this.codeVersion = version;
        }
}
```

© 2007, M. Riveill

La reflection et les attributs

Custom attributes

```
static void Main(string[] args) {
      AccesAuxDonnees anInstance … ;

      MonAttributAttribute monAttribut =
            (MonAttributAttribute) Attribute.GetCustomAttribute
                  (type, anInstance.GetTypes(), false);

      Console.WriteLine ("Code crée par : " + customAttribute.Owner);
      Console.WriteLine ("Version de code : " + customAttribute.Version);
}
```

La reflection et les attributs

Conclusion

- Les informations sur les types sont facilement accessibles
- Permettent d'effectuer des opérations en runtime
- Permettent d'effectuer des opérations au moment du développement
- Extensibles grâce aux Custom Attributes

© 2007, M. Riveill

Demo

Attributs et réflexivité

Troisième partie

Les applications Web et les services techniques

L'accès aux données (ADO.NET)

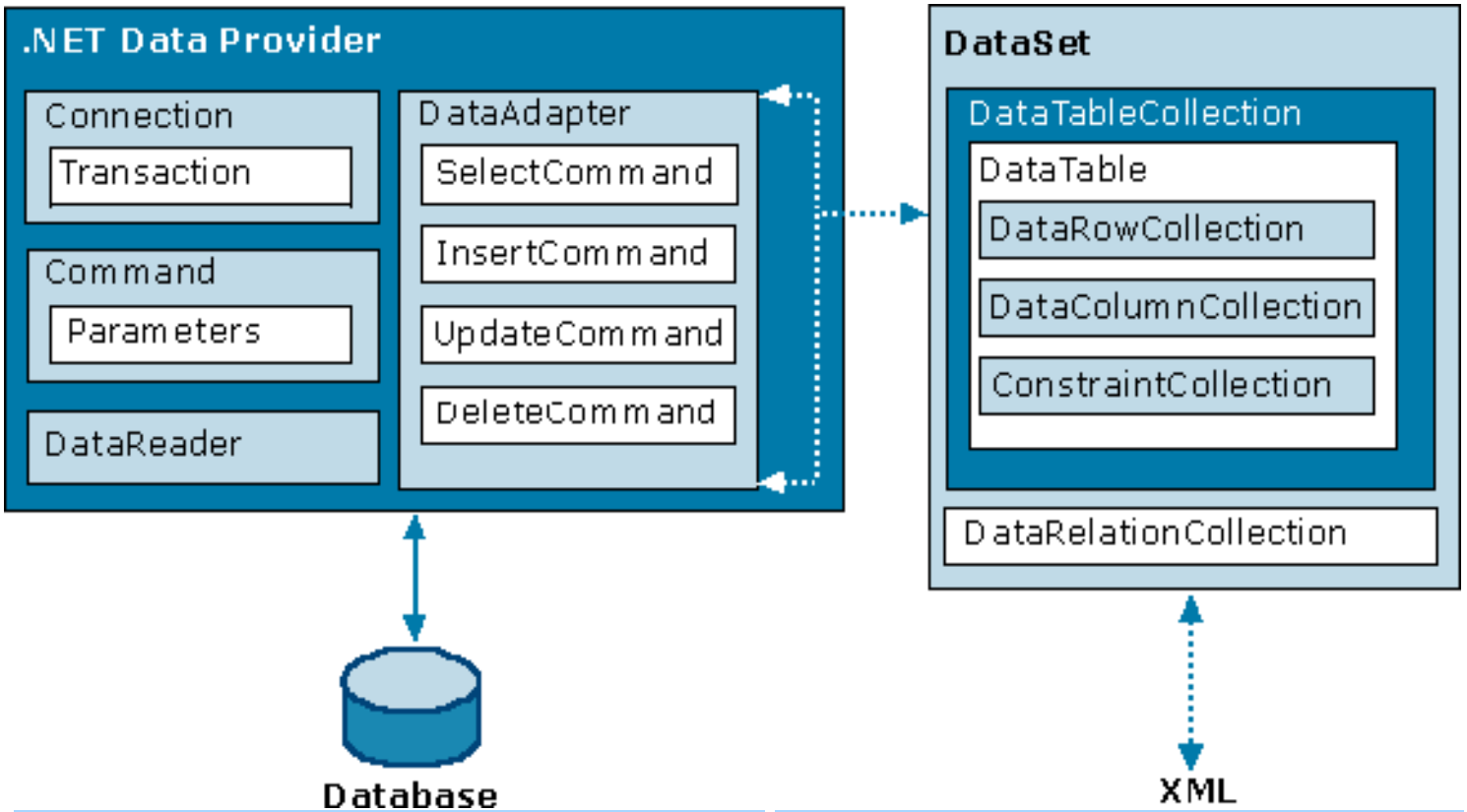
Les services Web (ASMX)

Les pages dynamiques (ASP.NET)

La sécurité

Les transaction

L'accès aux données :
une architecture à deux niveaux



- Mode connecté :
  - **Accès physique à la base lors de toutes consultations / modifcations**
- Mode déconnecté :
  - **Le Dataset est le cache des données de la base**

© 2007, M. Riveill

Mode connecté

- **Extraction des données en mode connecté à l'aide de DataReader**
  - Extraction d'un flux de données en lecture seule
  - Les résultats sont retournés pendant que la requête s'exécute
  - Ils sont stockés dans la mémoire tampon de réseau sur le client
  - Demande de leurs disponibilités par la méthode Read de DataReader.

  - Le mode connecté se révèle être un bon choix lors de l'extraction de grandes quantités de données
    - Extraction des données dès qu'elles sont disponibles
      - On n'attend pas que tous les résultats de la requête soient retournés,
    - Pas de mises en cache dans la mémoire.

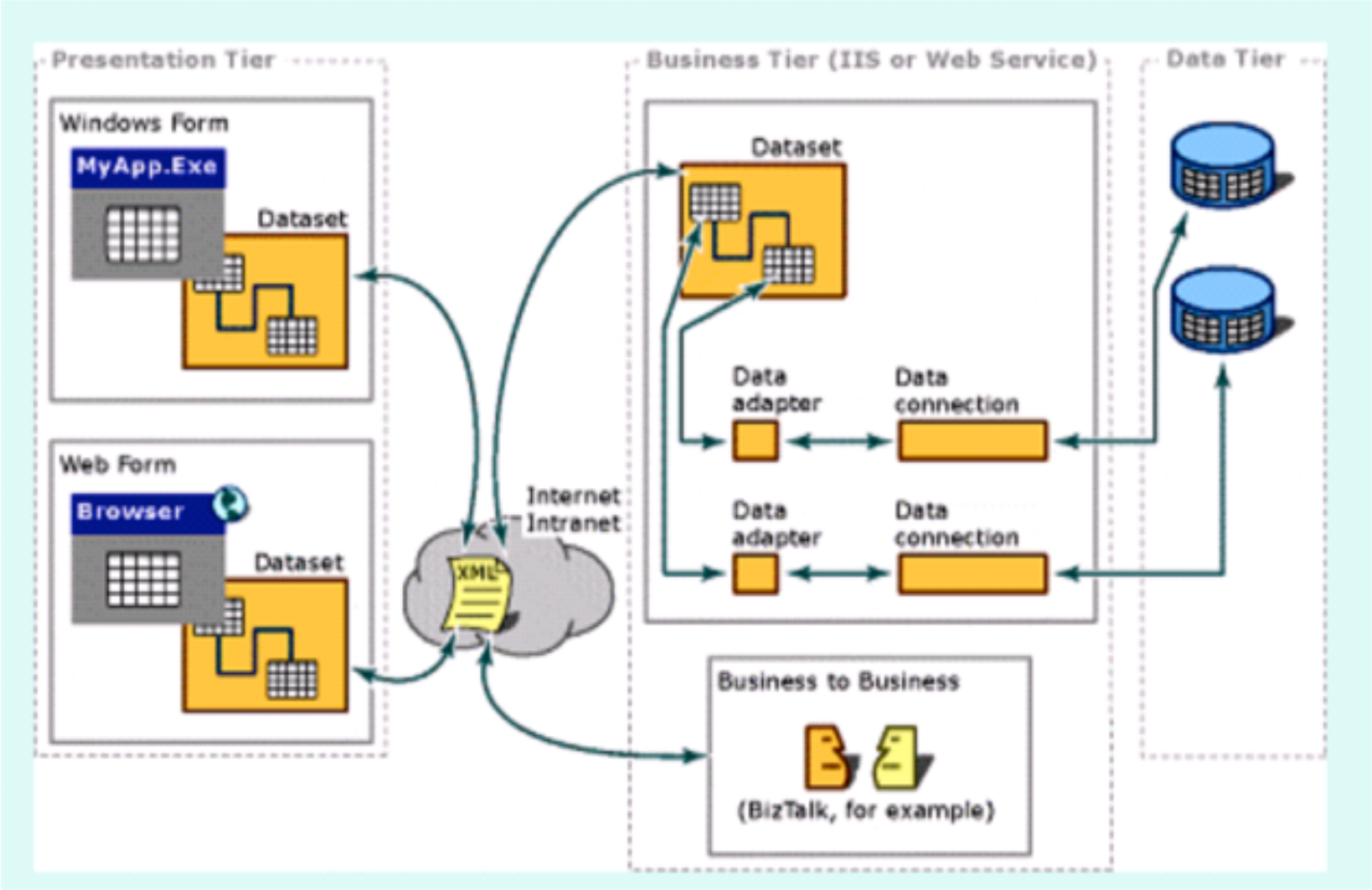
© 2007, M. Riveill

Mode déconnecté

- Les données sont “cachées” dans les DataSets
  - Limite les allers retour vers le serveur
    - Lecture / Ecriture sur le DataSet qui est un cache des données de la base
  - Notion de “virtual data store“
- Un DataSet est structuré comme la base de données
  - Contient une ou plusieurs “tables”
  - DataAdapters assurent le transfert (bi-directionnel) des données entre DataSet et la base de données
    - Ils chargent les données depuis la base de données dans le DataSet
    - Ils répercutent les modifications du DataSet vers la base de données sur demande

© 2007, M. Riveill

Architecture 3 parties



Demo

Accès aux données

Mode connecté (DataReader) et non connecté (DataSet)

Les services Web

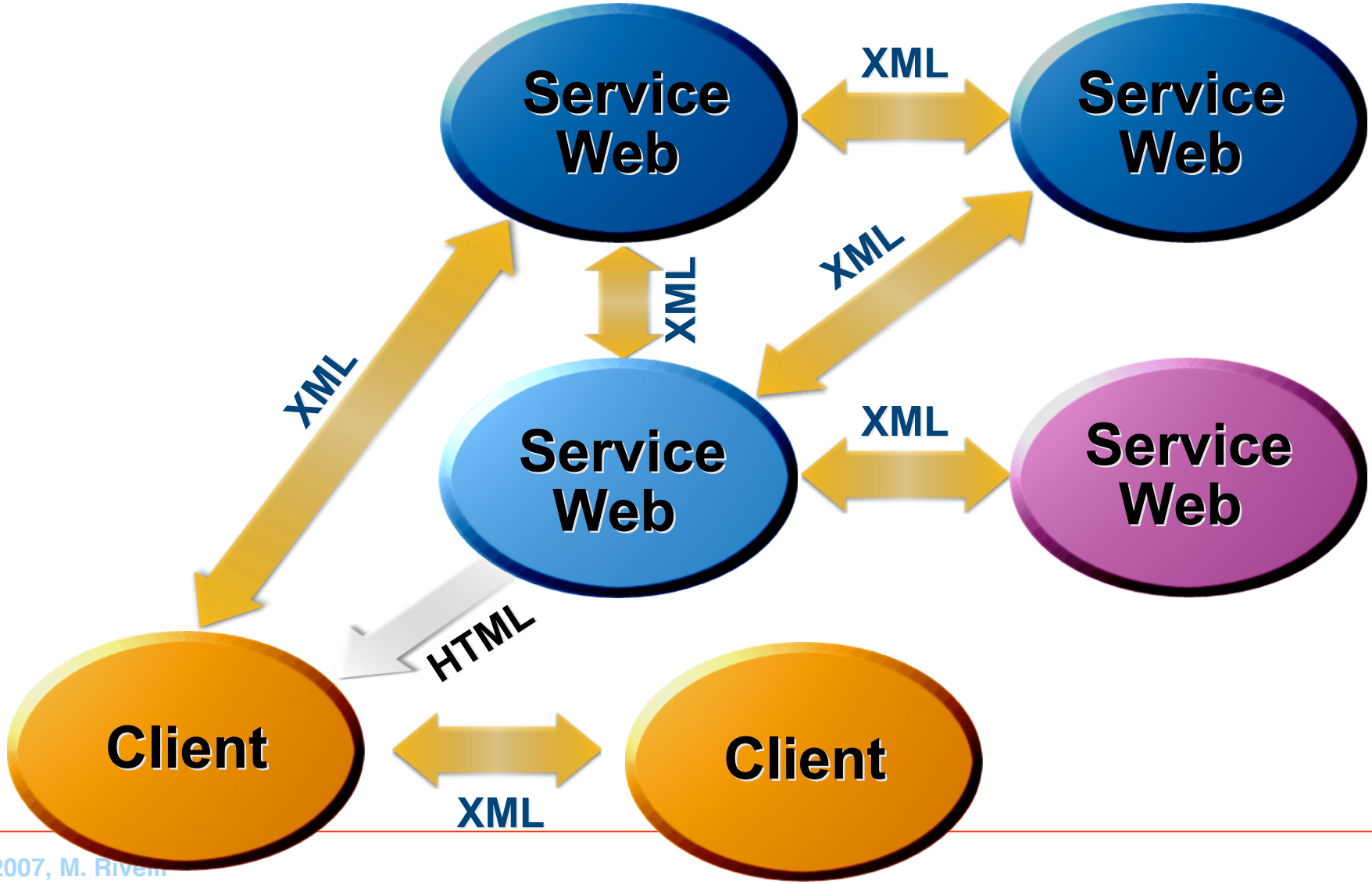
La base de la «vision» .NET

- Un service Web est une « unité logique applicative » accessible en utilisant les protocoles standard d'Internet
- Caractéristiques:
  - Réutilisable
  - Indépendamment de
    - la plate-forme (UNIX, Windows, …)
    - l'implémentation (VB, C#, Java, …)
    - l'architecture sous-jacente (.NET, J2EE, …)
- Ce sont des «librairies» fournissant des données et des services à d'autres applications
- Combinent les meilleurs aspects du développement à base de composant et du Web

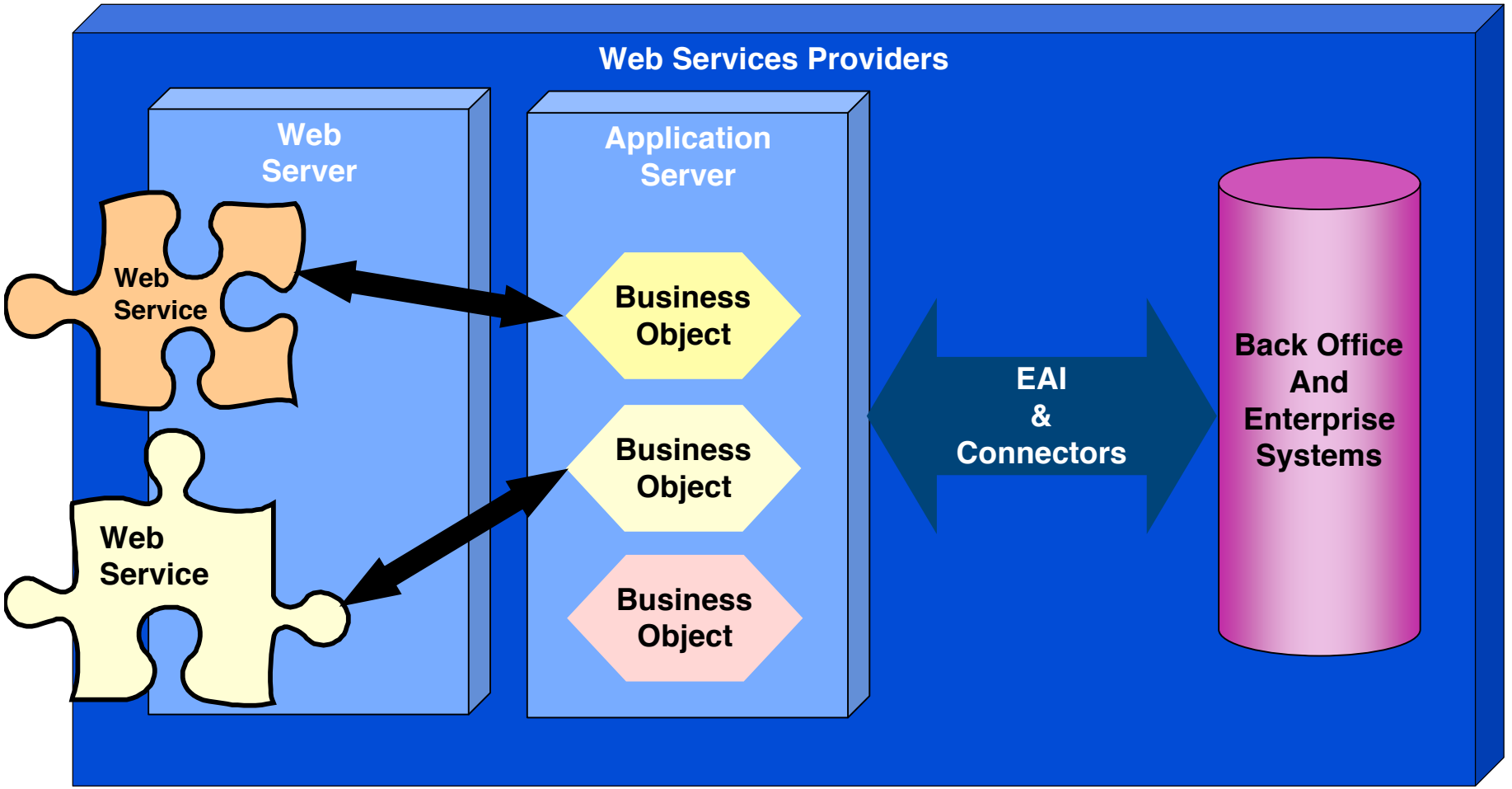
© 2007, M. Riveill

Les Services Web

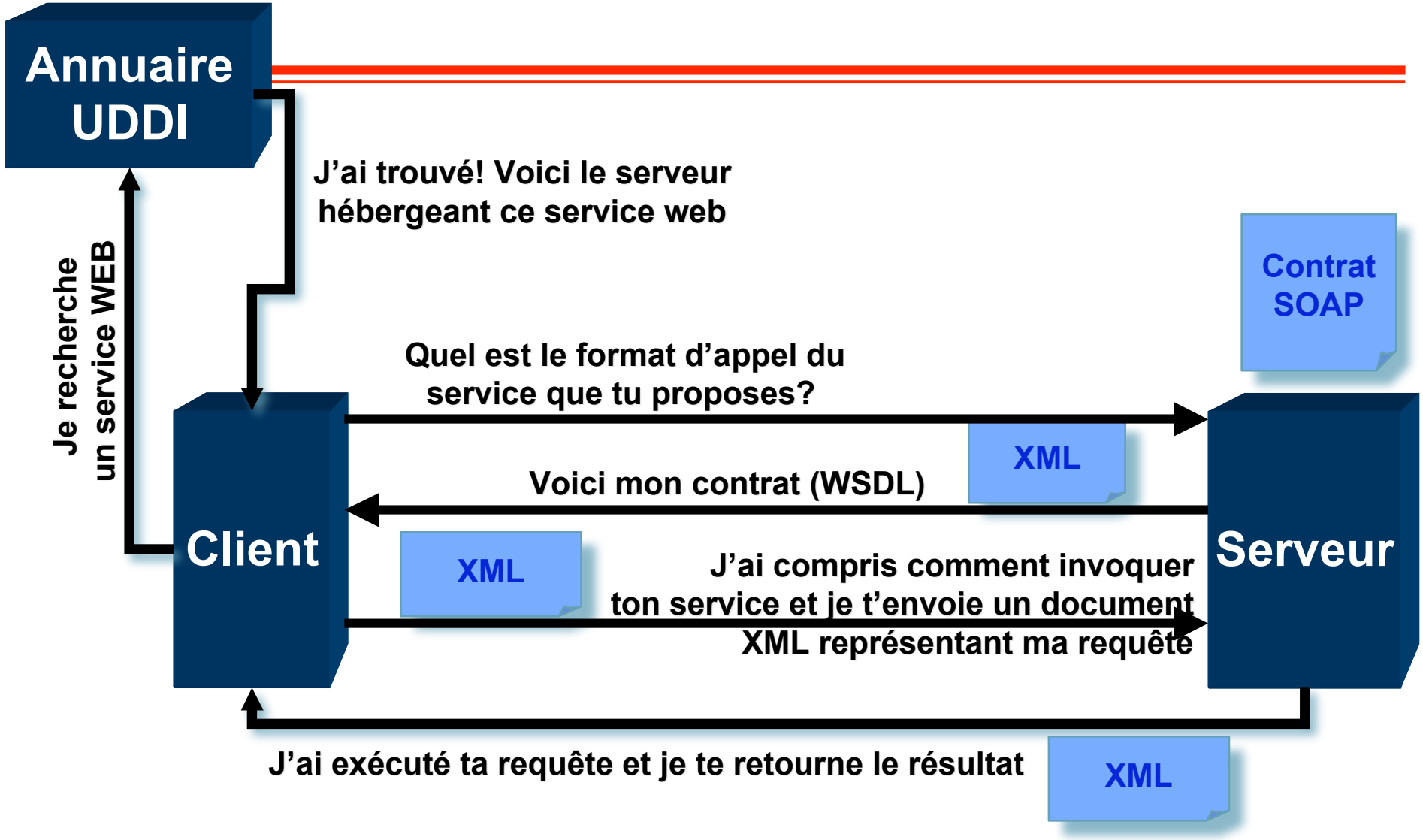
Au-delà de la navigation



Accès Universel aux services



© 2007, M. Riveill



© 2007, M. Riveill

Développer un service Web

- Dans deux fichiers (code behind)

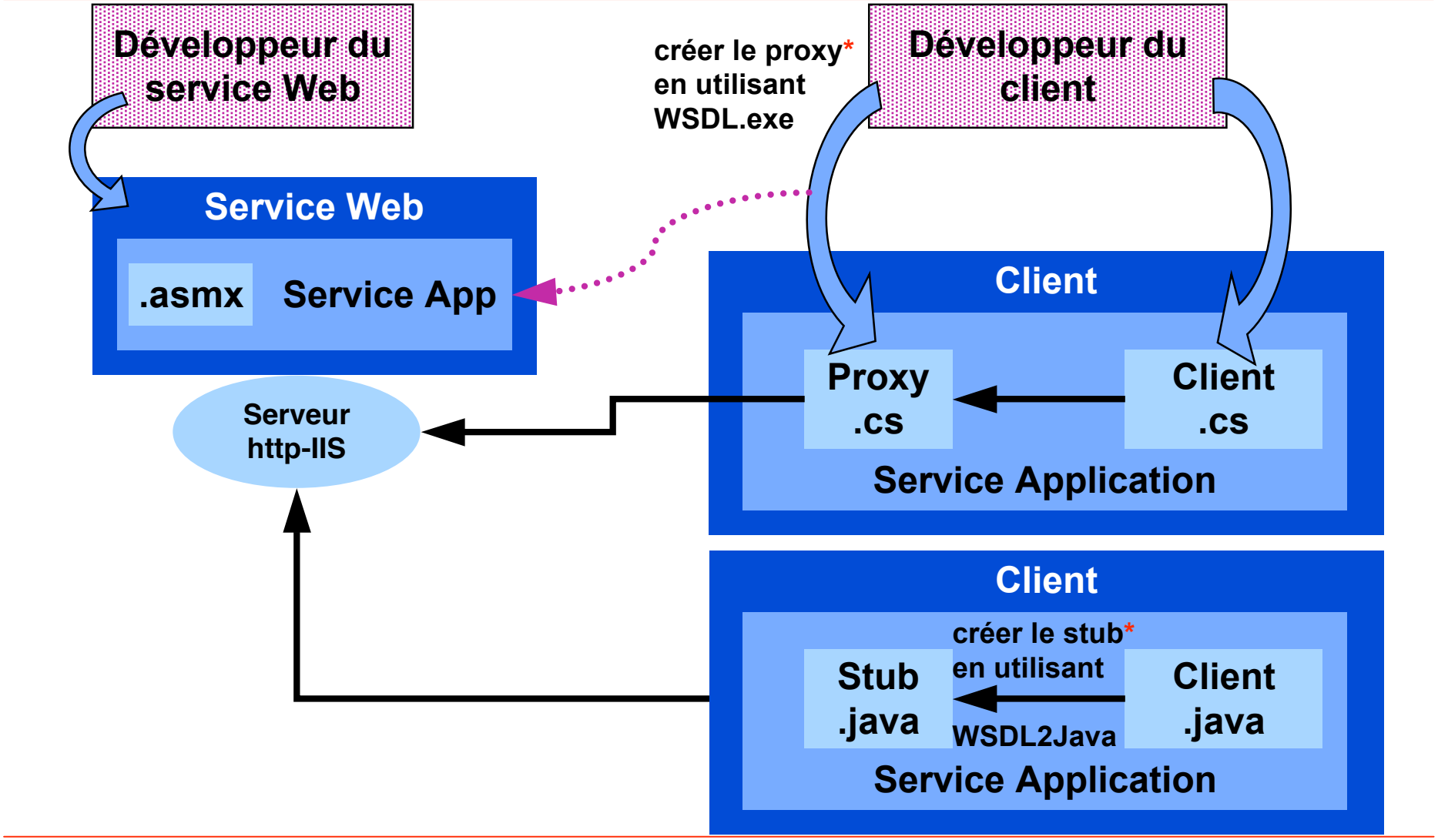
```
<%@ WebService Language="c#" Codebehind="MyWebService.cs"
Class="FirstWebService.MathService" %>
```

- Dans un seul fichier

```
<%@ WebService Language="C#" Class="MathService" %>

using System.Web.Services;
public class MathService : WebServices {
   [WebMethod]
   public int Add(int num1, int num2) {
      return num1 + num2;
   }
}
```

© 2007, M. Riveill

Utiliser un Service Web



* : un stub (Corba) est un proxy dans le monde MS

Génération du proxy / stub

- **C#**

```
wsdl URL_du_service_web?WSDL
```

- **Java…**

```
java org.apache.axis.wsdl.WSDL2Java 'URL_du_service_web?WSDL '
```

© 2007, M. Riveill

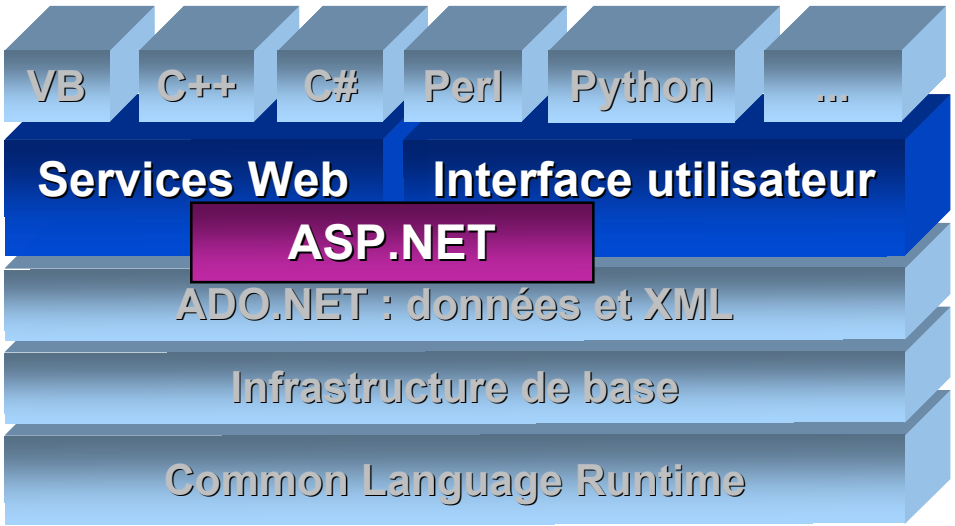
Demo

Service Web

Les pages dynamiques

ASP.Net

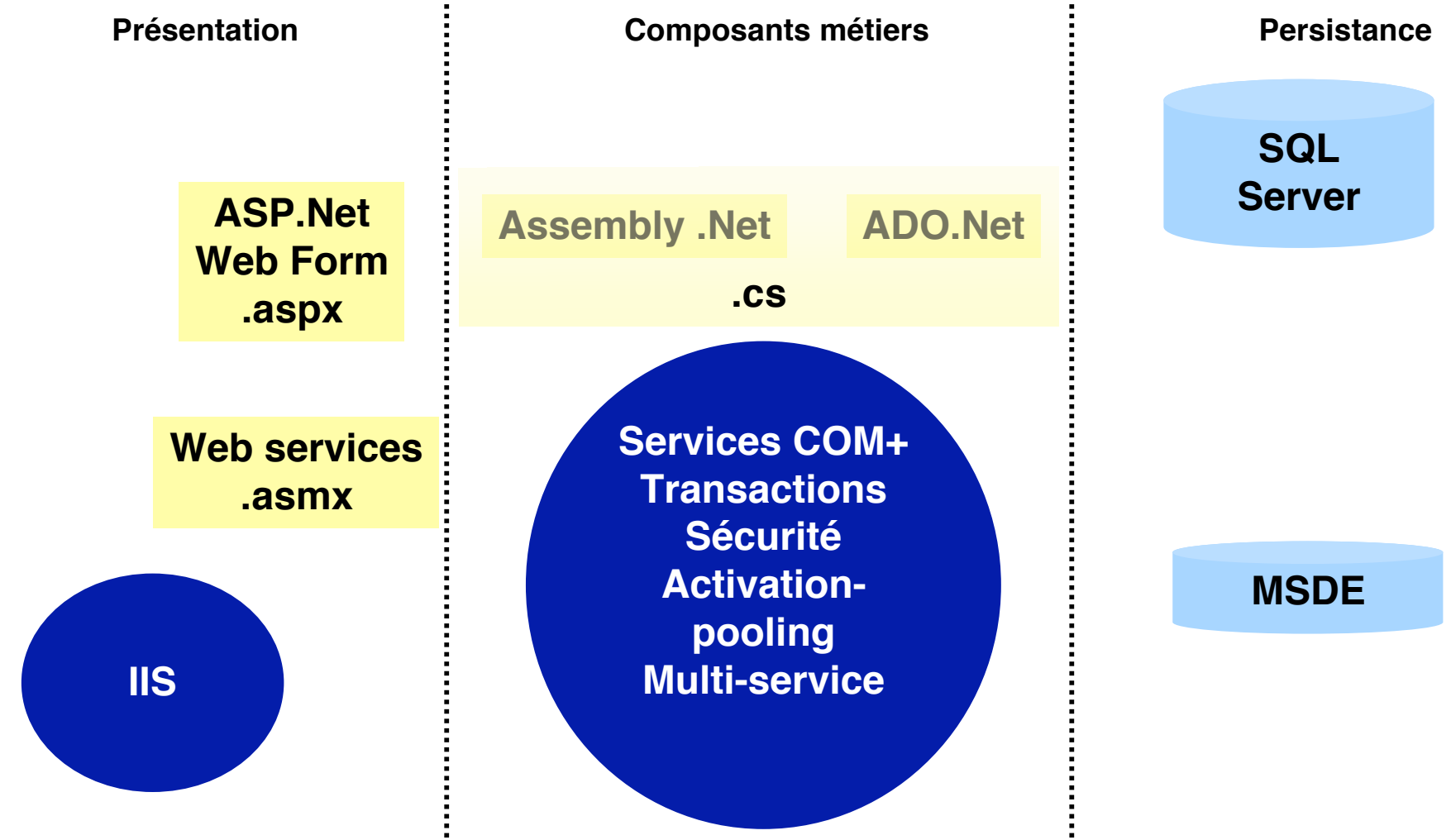
- Fait partie de .NET Framework
- Construit, déploie et exécute des applications et des services Web
- Les applications Web Visual Studio .NET prennent directement en charge ASP.NET



© 2007, M. Riveill

Architecture 3 parties

vocabulaire



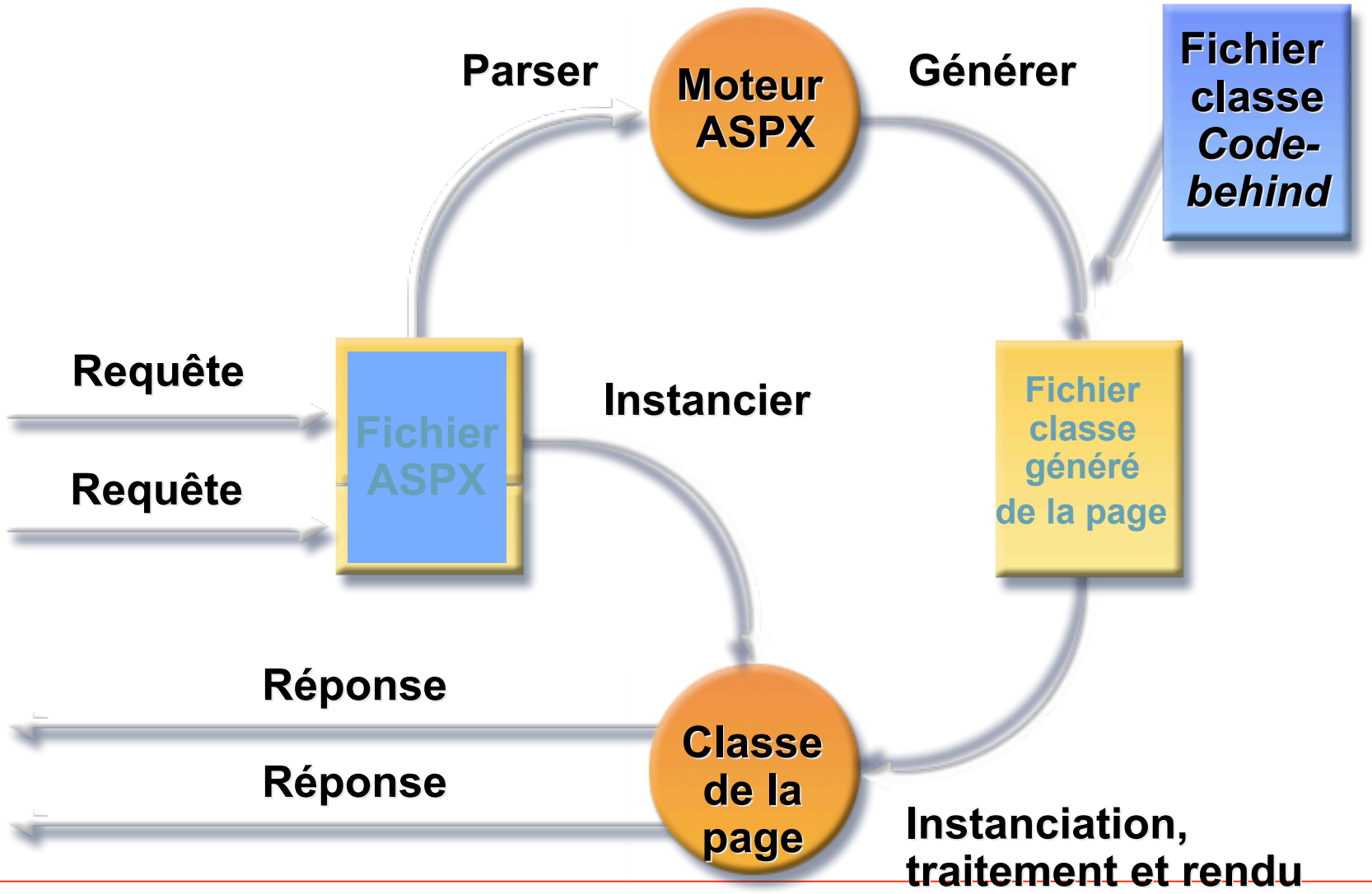
© 2007, M. Riveill

Web Forms
Infrastructure de page ASP.NET

- Objectif : développer une page Web comme on développe un écran client riche.
  - Unification du modèle de programmation
    - Modèle d'exécution événementiel basé sur des contrôles
    - Modèle évènementiel et DOM (modèle de document) projeté sur le serveur
  - Génération dynamique du code HTML correspondant au navigateur
    - System.Web.UI.WebControls
- Exécution sur le serveur IIS parallèlement aux applications ASP existantes
  - Exécuté via CLR en tant que code natif
- Visual Basic, C#, JScript

© 2007, M. Riveill

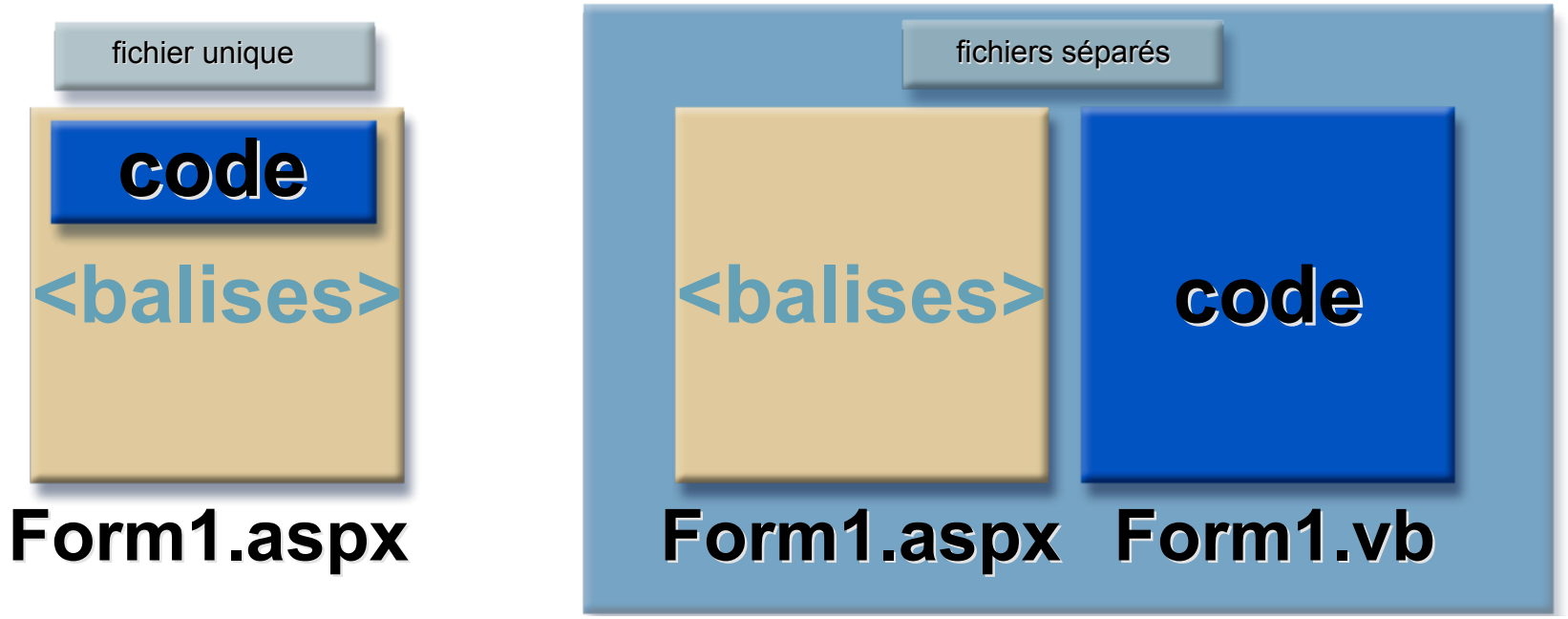
Compilation à la volée



© 2007, M. Riveill

Web Forms

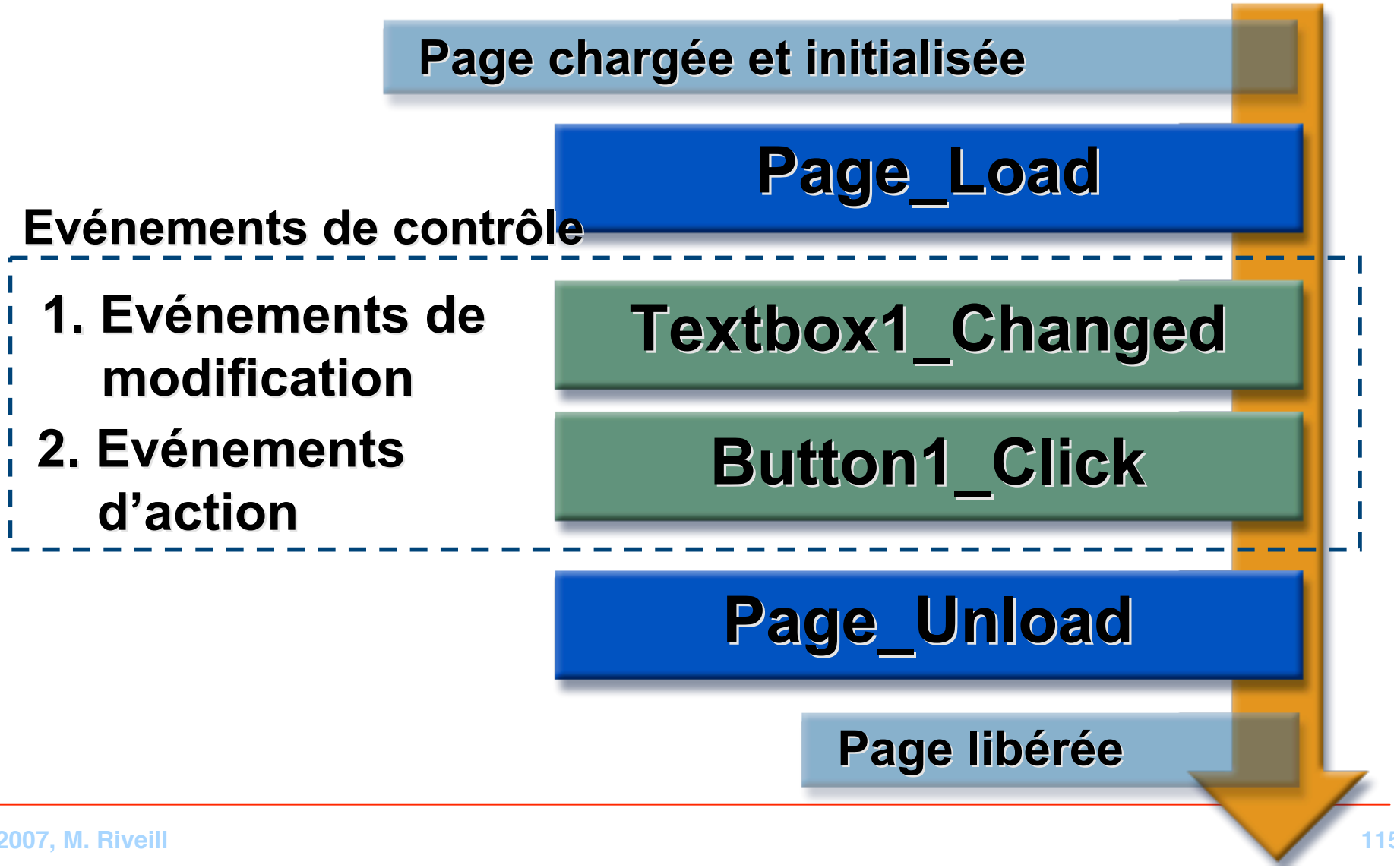
- Un formulaire Web combine des balises déclaratives (HTML, directives ASP, contrôles serveur et texte statique) avec du code
- Toutefois, le code et les balises sont bien séparés



© 2007, M. Riveill

Web Forms

Cycle de vie d'une page



© 2007, M. Riveill

- 45 contrôles intégrés
  - Génère du HTML 3.2
  - On retrouve les contrôles usuels
    - textbox, checkbox, radio, button
  - Déclenchent des événements vers le serveur
    - Textbox1_ Changed, Button1_Click
  - Déclanchent des évènements liés au cycle de vie de la page
    - Page_Load, Page_Unload

© 2007, M. Riveill

Applications ASP.NET

Variables d'exécution

- Un état de session est associé à chaque processus ASP.NET
  - InProc : variables de session conventionnelles. Stockées en mémoire sur le serveur Web
  - StateServer : les sessions sont stockées sur un serveur externe, en mémoire
  - SqlServer : les sessions sont stockées dans une base de données SQL
    - Améliore grandement la fiabilité
    - L'état de session peut survivre à un crash ou à un redémarrage
- Possibilité de créer des variables globales liées à l'application
  - Commune à toutes les instances de l'application (exemple verrouillage d'une section critique)

```
Application.Lock
Application("CompteurGlobal") = NouvelleValeur
Application.Unlock
```

- Possibilité de créer des variables propres à chaque utilisateur
  - Disponibles pour toutes les pages du site

```
Session("UserID") = 5
UserID = Session("UserID")
```

© 2007, M. Riveill

Applications ASP.NET

Configuration de l'état de session

- Mode
  - InProc : variables de session conventionnelles. Stockées en mémoire sur le serveur Web
  - StateServer : les sessions sont stockées sur un serveur externe, en mémoire
  - SqlServer : les sessions sont stockées dans une base de données SQL
    - Améliore grandement la fiabilité
      - L'état de session peut survivre à un crash ou à un redémarrage
- Permet le déploiement de fermes Web
  - Les états de session peuvent être partagées à travers une ferme de serveurs Web ASP .NET
  - Les applications ne sont plus reliées à une machine particulière
- Cookieless
  - Détermine si des sessions sans cookies doivent être utilisées
  - Les valeurs possibles sont true et false

© 2007, M. Riveill

Applications ASP.NET

Configuration de l'état de session

- Possibilité d'utiliser les cookies ou non
- Compatible avec l'utilisation de fermes Web
- Le comportement et la configuration est décrite
  - Dans le fichier Web.Config

```
<sessionState

        mode="sqlserver"
        cookieless="false"
        timeout="20"
        sqlConnectionString="data source=127.0.0.1;
                                        user id=sa;password=""
        stateConnectionString="tcpip=127.0.0.1:42424"

/>
```

Demo

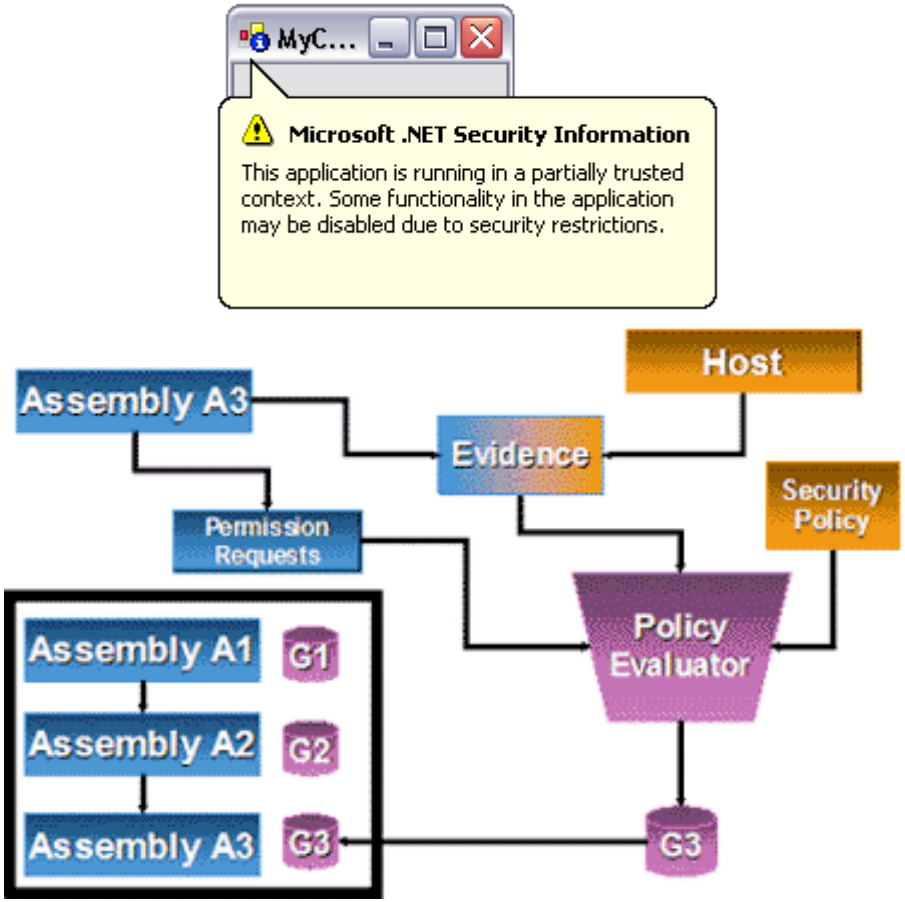
Pages ASP.Net

Sécurité

Modèle de sécurité - .NET

Deux types de sécurité

- **Sécurité d'accès au code**
  - Basé sur les politiques de sécurité et sur des *évidences.*
- **Sécurité basée sur les rôles**
  - Basé sur l'identité de l'utilisateur. (Compte Windows ou autre)



© 2007, M. Riveill

Sécurité d'accès au code

1ère étape : Obtention des preuves

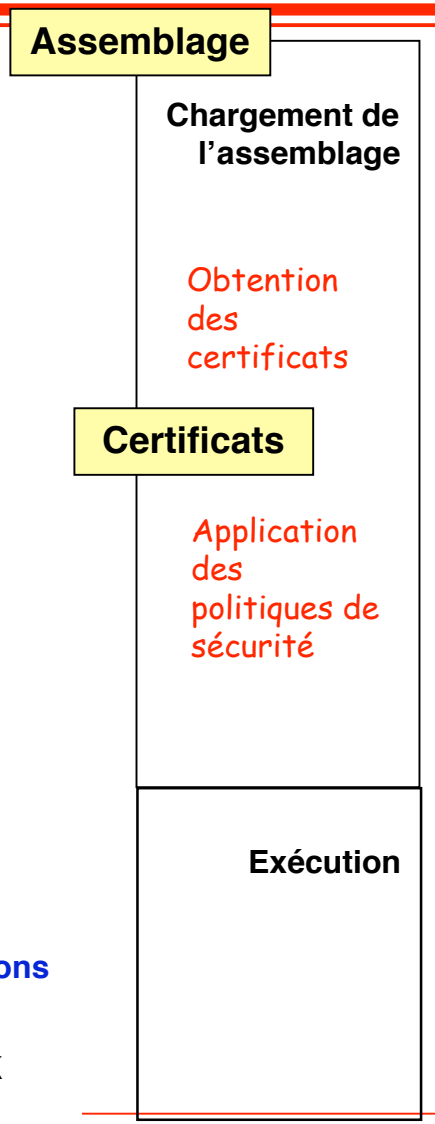
- Chaque fois qu'un assembly est chargé dynamiquement
  - l'environnement doit produire un certificat
  - Ce certificat permet d'identifier la politique de sécurité qui sera appliquée
- Voici quelques formes de certificat
  - A chaque certificat correspond une classe du framework qui le décrit
  - System.Security.Policy.Site : le site d'où provient le composant
  - System.Security.Policy.Url : une URL qui identifie la ressource
  - System.Security.Policy.Zone : même concept que les zones d'Internet Explorer.
  - System.Security.Policy.Publisher : certificat déposé de l'assembly
  - System.Security.Policy.StrongName : Strong Name: généré à partir de sn.exe (Composants .Net)
  - System.Security.Policy.Hash : la valeur de hashage d'un assembly (généré par SHA1)
  - System.Policy.ApplicationDirectory : le répertoire qui contient le code

© 2007, M. Riveill

Sécurité d'accès au code

2ème étape : Application des politiques de sécurité

- 3 groupes de permissions
  - Permissions d'accès aux ressources
    - Si l'objet est une ressource de tel type alors je peux faire cela
      - System.Security.Permission.FileIOPermission permet le contrôle des accès aux fichiers et répertoires
  - Permissions associées aux certificats
    - Si l'objet a ce certificat alors je peux faire cela
  - Meta permission
    - Permissions accordées au gestionnaire de sécurité
- Application des politiques de sécurité
  - Obtenir un ensemble de permissions en fonction des certificats
  - 4 politiques de sécurité
    - Entreprise
    - Machine
      - Configuré par l'administrateur de la machine
    - Utilisateur
      - Configuré par l'utilisateur
      - S'applique au code géré par les processus de l'utilisateur
    - Domaine d'application
      - Configuré par l'application
  - les permissions accordées à un assembly sont l'intersection des permissions accordées pour chaque politique
- Se configure dans : Configuration Microsoft .NET Framework
  - Dans le répertoire : Outils d'administration

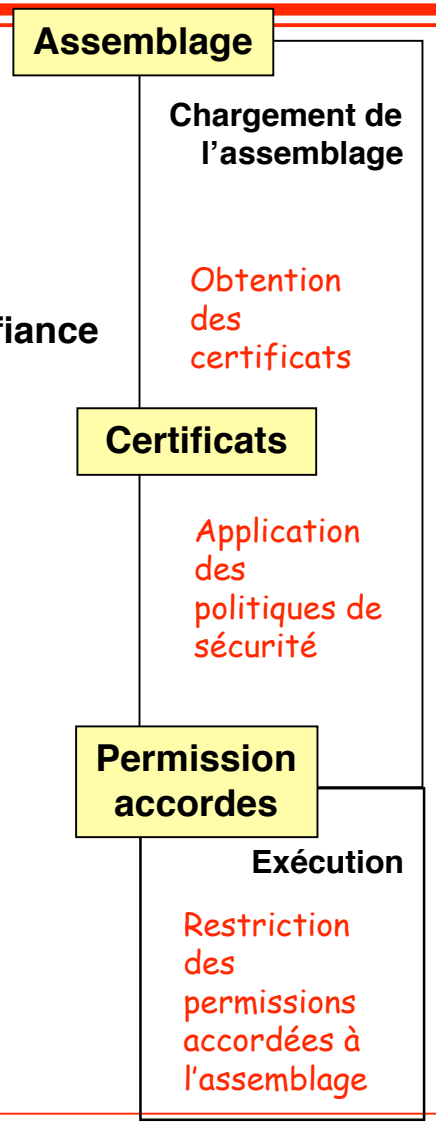


© 2007, M. Riveill

Sécurité d'accès au code

3ème étape : restriction des permissions à l'exécution

- Si les droits sont suffisants, alors
  - L'objet est couplé
  - Le code est exécuté
- Il est possible temporairement de retreindre des droit
  - Exemple : garantir que du code tiers, dont on a moyennement confiance accède à des zones sensibles
  - Le composant B accède à une ressource O
  - B a les droits pour accéder à O
  - Le composant A appelle le composant B
  - A n'a pas les droits pour accéder à O
  - Est-ce que A peut néanmoins y accéder via B ?



Sécurité d'accès au code

Exemple 1

- **Exemple de code pour B**

```
PermissionSet PS = new PermissionSet(PermissionState.None);
// la ressource est le répertoire C:\Windows
PS.AddPermission(new FileIOPermission(FileIOPermissionAccess.AllAccess, @"C:\Windows"));
// la ressource est la base de registre
PS.AddPermission (new RegistryPermission(RegistryPermissionAccess.AllAccess,""));


// proposition 1 : j'empêche tout le monde à accéder à la ressource
PS.Deny(); // supprime les permissions contenues dans PS
    // ressource inaccessible y compris quelqu'un qui avait les droits
CodeAccessPermission.RevertDeny(); // rétabli les permissions
```

- **Peut aussi se programmer à l'aide d'un attribut**

```
[FileIOPermission(SecurityAction.Deny, @"C:\Windows"]
```

© 2007, M. Riveill

Sécurité d'accès au code

Exemple 1

- **Exemple de code pour B**

```
PermissionSet PS = new PermissionSet(PermissionState.None);
// la ressource est le répertoire C:\Windows
PS.AddPermission(new FileIOPermission(FileIOPermissionAccess.AllAccess, @"C:\Windows"));
// la ressource est la base de registre
PS.AddPermission (new RegistryPermission(RegistryPermissionAccess.AllAccess,""));


// proposition 2 : je me porte garant pour l'accès à la ressource
PS.Assert(); // supprime les permissions contenues dans PS
    // ressource accessible pour tous
    // (sous réserve que celui qui fasse la demande possède bien les droits)
CodeAccessPermission.RevertDeny(); // rétabli les permissions
```

- **Peut aussi se programmer à l'aide d'un attribut**

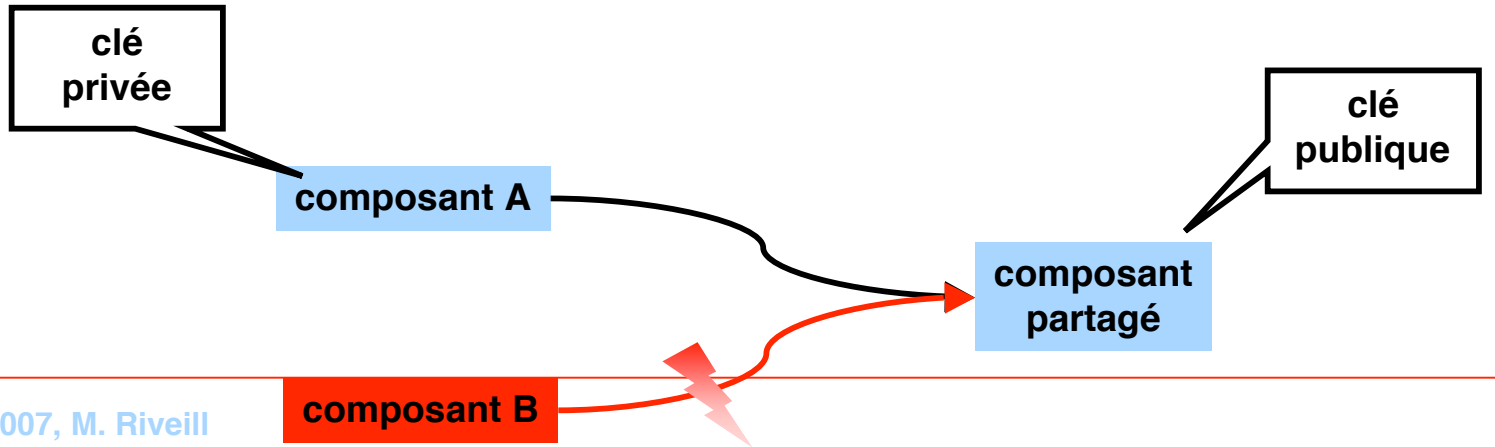
```
[FileIOPermission(SecurityAction.Assert, @"C:\Windows"]
```

© 2007, M. Riveill

Sécurité d'accès au code

Exemple 2

- Contrôler l'identité de l'appelant en se basant sur un certificat
  - Certificat utilisé : nom fort (strong name)
  - Dans composant partagé
    - On indique que seul celui qui possède la clé privée associée à la clé publique donnée peut appeler ce composant
  - Dans composant A qui connaît la clé privée
    - appel du composant partagé (en utilisant la clé privée ☺)
  - Dans composant B qui ne connaît pas la clé privée
    - appel du composant partagé (sans la clé privée ☹)

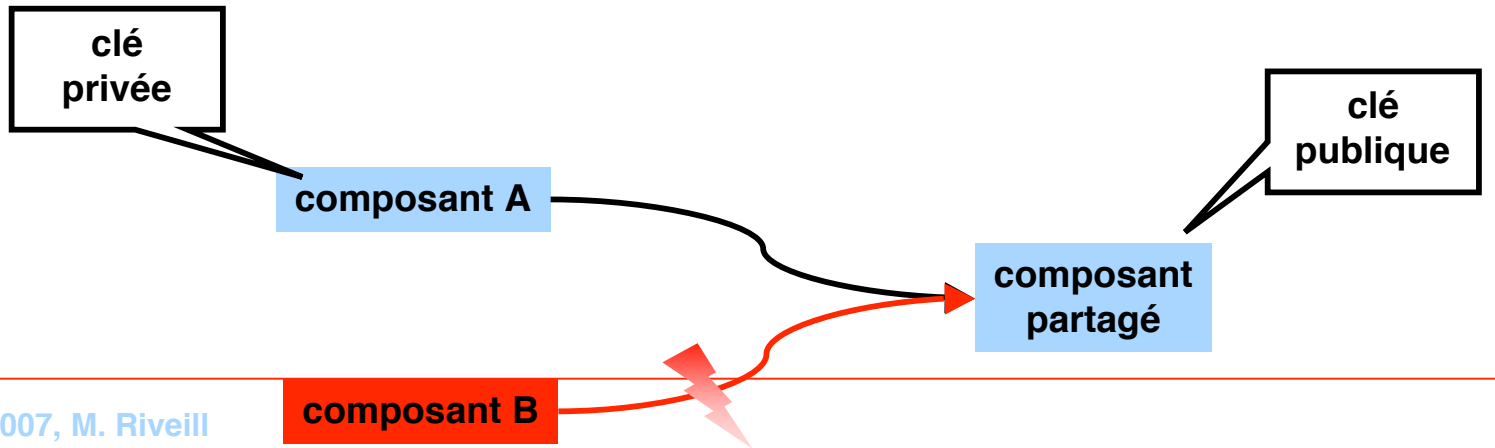


© 2007, M. Riveill

Sécurité d'accès au code

Code de l'exemple

```
// code du composant partagé
[StrongNameIdentityPermission(SecurityAction.Demand,
                              PublicKey = "la clé publique")]
public class SharedComponent {
  public static string GetInfo() {
    return "La composant 'privé' a été appelé.";
  }
}
```

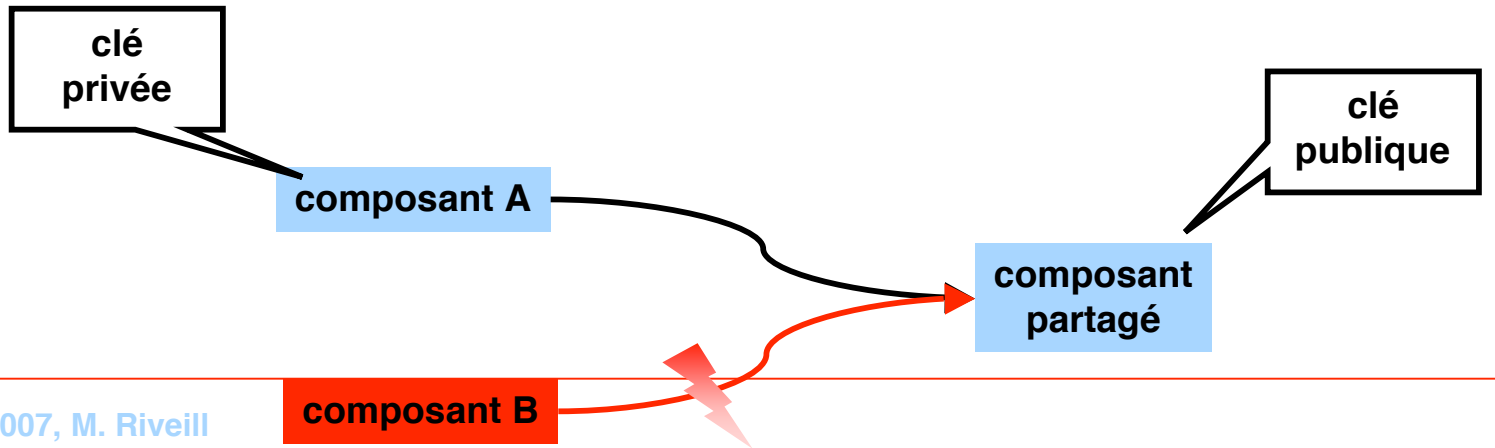


© 2007, M. Riveill

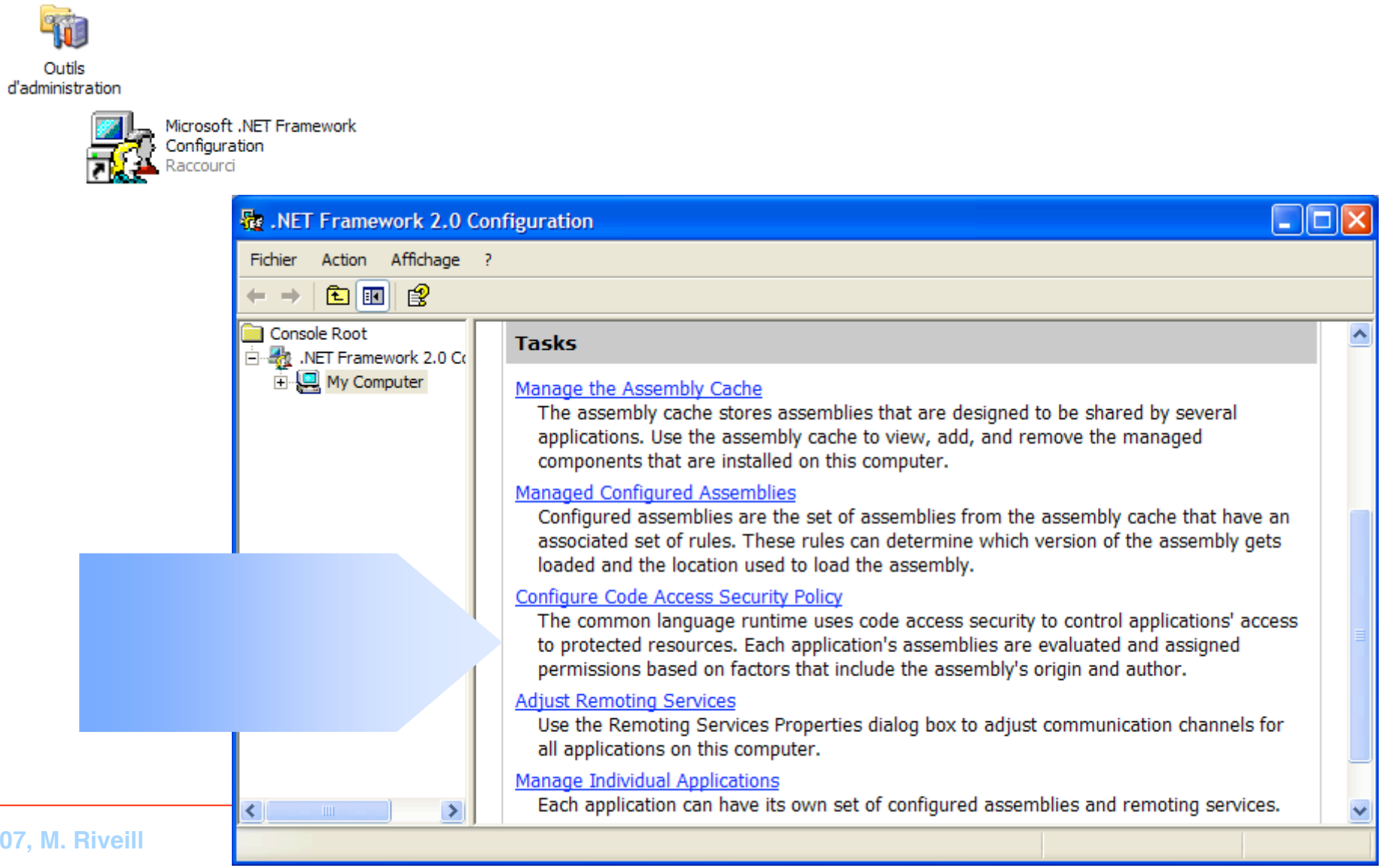
Sécurité d'accès au code

Code de l'exemple

```
// code du composant A ou B
[assembly:AssemblyKeyFile("keypair.dat")]
public class Caller {
  public static void Main() {
    try {
      Console.WriteLine(SharedComponent.GetInfo());
    } catch(Exception e) {
      Console.WriteLine("\nException occurred: {0}\n", e);
    }
  }
}
```



Sécurité d'accès au code
Configuration



© 2007, M. Riveill

Sécurité basée sur les rôles
Role-based security

- Comparable au mécanisme COM+
- Utilisateurs et rôles et peuvent être :
  - Des comptes/groupes Windows
  - Spécifiques à l'application
- Utilise l'objet 'PrincipalPermission'
  - Décrit un utilisateur et un rôle
  - Associé au thread courant

© 2007, M. Riveill

Sécurité basée sur les rôles

Exemple

```
public class aClass {
 public void AdminMethod () {
   if (Thread.CurrentPrincipal.IsInRole
              (@"BUILTIN\Administrateurs")) {
     // corps de la méthode
     // ici le principal est un administrateur
   }
 }
 public void GuestMethod () {
   if (Thread.CurrentPrincipal.IsInRole
              (@"BUILTIN\Utilisateur")) {
     // corps de la méthode
     // ici le principal est un utilisateur
   }
 }
}
```

1) Utilisation des rôles Windows

© 2007, M. Riveill

Sécurité basée sur les rôles

Exemple

```
public class aClass {
  [PrincipalPermission(SecurityAction.Demand, Role=@"BUILTIN\Administrateurs")]
  public void AdminMethod () {
      // corps de la méthode
  }
  [PrincipalPermission(SecurityAction.Demand, Role=@"BUILTIN\Utilisateurs")]
  public void GuestMethod () {
      // corps de la méthode
  }
}
public class IdentityCheck {
  public static void Main() {
    aClass o = new aClass;
    try {
      o.AdminMethod();
    } catch(Exception e) {
      if (e is SecurityException & ((SecurityException)e).PermissionType
                  == typeof(PrincipalPermission)) {
          // On n'avait pas de droit d'exécuter la méthode
    }
    ...
```

1bis) Utilisation des rôles Windows

© 2007, M. Riveill

Sécurité basée sur les rôles

Exemple

```
public class aClass {
  public void AdminMethod () {
    SecurityCallContext sec = SecurityCallContext.CurrentCall;
    if (sec.IsCallerInRole("Agent Administrateur")) {
      // corps de la méthode
      // Seul les 'principaux' avec le rôle 'Agent Administrateur'
      // peuvent exécuter cette méthode
    } else {
       throw new Exception ("Vous n'avez pas le droit");
    }
}
```

2) Utilisation des rôles utilisateur (liste d'accès)

Basé sur la classe ServicedComponent

© 2007, M. Riveill

Sécurité basée sur les rôles

Exemple : classe cliente

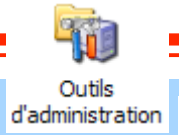
- **Quelque soit la manière de décrire le rôle :**

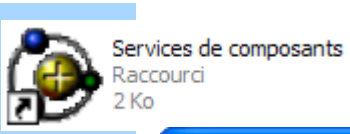
```
public static void Main() {
  aClass o = new aClass;

  try {
    o.AdminMethod();
  } catch(Exception e) {
    if (e is SecurityException
        & ((SecurityException)e).PermissionType
              == typeof(PrincipalPermission)) {
     // On n'avait pas de droit d'exécuter la méthode
  }
  …
}
```

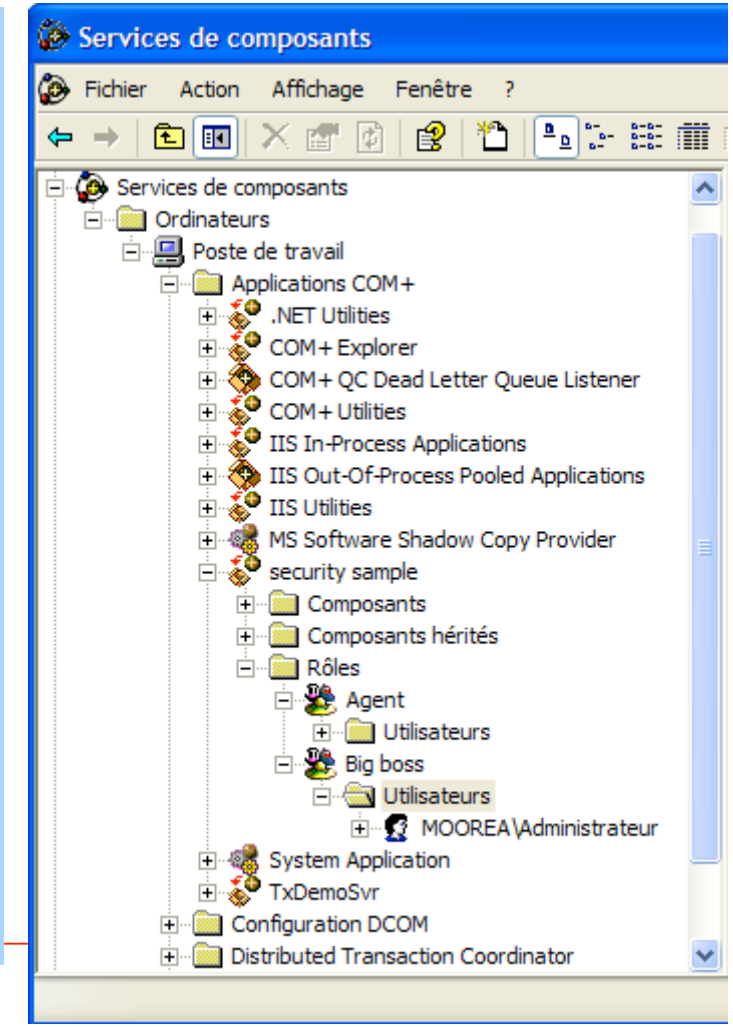
Sécurité basé sur les rôles

Configuration des rôles

```
using System;
using System.Reflection;
using System.EnterpriseServices;
using System.Security.Permissions;

[assembly:ApplicationName("security sample")]
[assembly:AssemblyKeyFile("keypair.dat")]
public class aClass : ServicedComponent {
  public void aclMethod () {
    SecurityCallContext sec =
       SecurityCallContext.CurrentCall;
    if (sec.IsCallerInRole("Big boss")) {
      // corps de la méthode
      // Seul les 'principaux' avec le rôle
      // adéquat peuvent exécuter cette méthode
    } else {
       throw new Exception ("Vous n'avez pas le droit");
    }
}
```



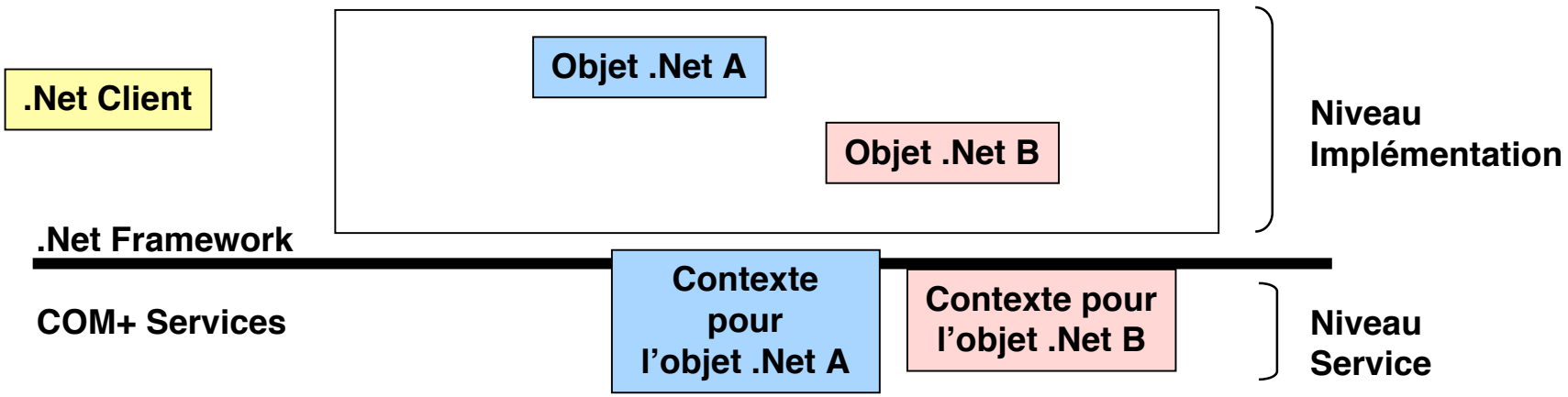




© 2007, M. Riveill

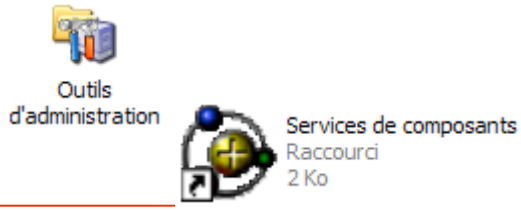
La classe ServicedComponent

- Services techniques supportés par Windows XP
  1. Liste d'accès et rôles
  2. Transactions
  - Autres services : Queue de messages, …
- Classe ServicedComposent fait le lien entre .Net et Windows



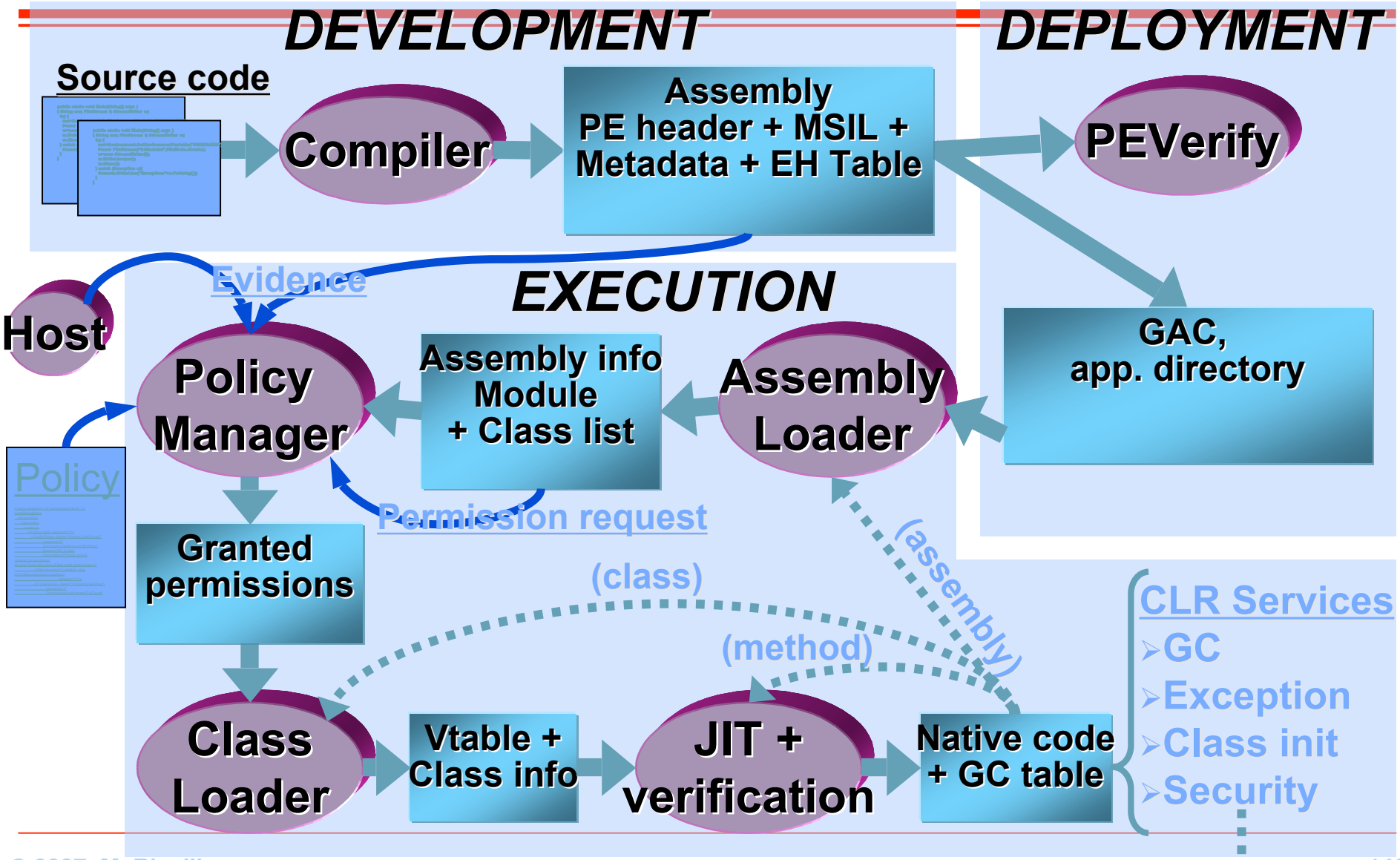
Sécurité basé sur les rôles
mode opératoire

- **Pour créer une classe .Net connue par COM+**
  1. **Faire dériver la classe ServicedComponent**
  2. **Donner un nom unique à l'assembly**
     - **sn -k TestApp.snk pour produire la paire de clé**
     - **Ajouter les clés à l'assembly**
       [assembly: AssemblyKeyFileAttribute("TestApp.snk")]
  3. **Enregistrer l'assembly qui contient la classe dans le catalogue COM+**
     - **Si le client de la classe est 'managé' (i.e. pris en charge par le common language runtime), l'enregistrement est effectué par le CLR**
     - **sinon anticiper l'usage de la classe par du code non 'managé', en utilisant le service .Net d'enregistrement (Regsvcs.exe)**
  4. **Configurer le service technique concerné dans COM+**



© 2007, M. Riveill

Modèle d'exécution .Net



© 2007, M. Riveill

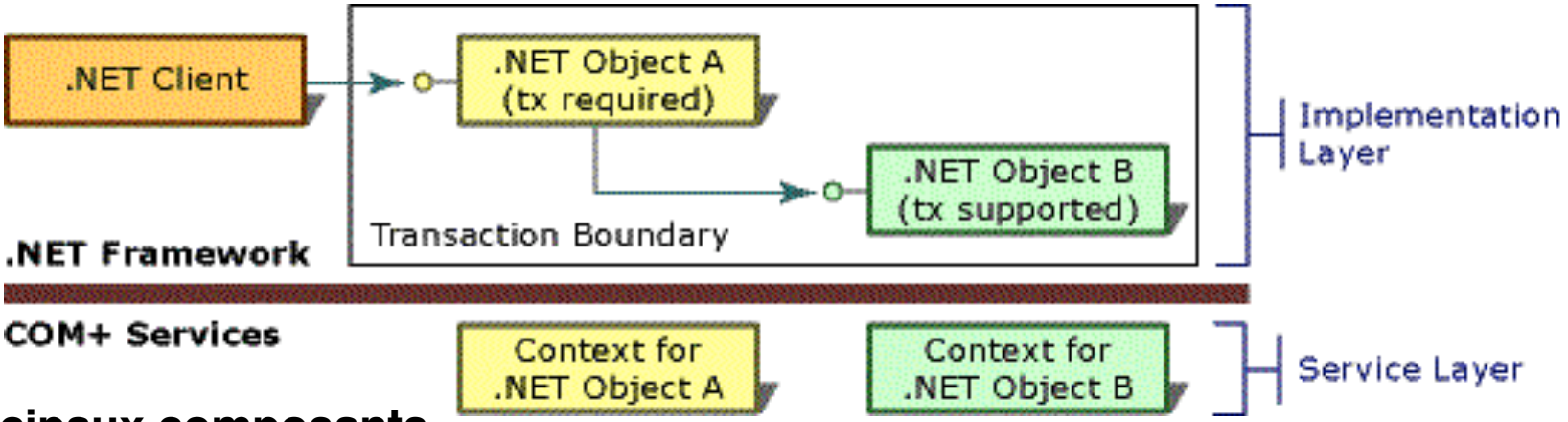
Demo

Sécurité

Transactions

Transactions

- **Elles sont prises en charge par COM+**
  - **MTS (Microsoft Transaction Server)**



- **Principaux composants**
  - **processus MTS**
    - **installation, création, cycle de vie, contexte d 'exécution, sécurité, activation juste à temps (just-in-time)**
  - **MS Distributed Transaction Coordinator (DTC)**
    - **service de gestion de transactions réparties orienté objet**
    - **gère aussi les transactions liées aux serveurs de bases de données**

© 2007, M. Riveill

Transactions

- **Toute instance d'une classe .Net peut participer automatiquement à une transaction**
  - Il faut la préparer
  - Chaque ressource accédée par une instance est 'enrollée' dans la transaction
  - Par exemple
    - Un objet utilise ADO.Net pour déposer une somme dans une base de donnée (gestion de compte bancaire),
    - Le gestionnaire de ressource détermine les différents objets qui font parti de la transaction,
    - Et inclue
      - Les objets .Net
      - La base de données
    - Dans la transaction

© 2007, M. Riveill

Transactions

- **Pour créer une classe 'transactionnelle'**
  1. Positionner l'attribut 'TransactionAttribute' à votre classe
  2. Faite la dériver de la classe ServicedComponent
  3. Donner un nom unique à l'assembly
     - sn -k TestApp.snk pour produire la paire de clé
     - Ajouter les clés à l'assembly
       [Visual Basic] <assembly: AssemblyKeyFileAttribute("TestApp.snk")>
       [C#] [assembly: AssemblyKeyFileAttribute("TestApp.snk")]
  4. Enregistrer l'assembly qui contient votre classe au catalogue COM+
     - Si le client de votre classe est 'managé' (i.e. prise en charge par le common language runtime), l'enregistrement est effectué par le CLR
     - sinon vous pouvez anticiper l'usage de votre classe par du code non 'managé', en utilisant le service .Net d'enregistrement (Regsvcs.exe)

- Idem que 'classe sécurisée'
  - Couplage .Net / COM+

© 2007, M. Riveill

Les attributs ‘Transaction’

Proche des attributs de J2EE

- **Disabled [Transaction(TransactionOption.Disabled)]**
  - Indique que cet objet ignore les transactions existantes (i.e. gérée au niveau de .Net)
  - Il peut appeler directement le Distributed Transaction Coordinator (DTC).
- **NotSupported [Transaction(TransactionOption.NotSupported)]**
  - Indique que cet objet s’exécute en dehors d’un contexte transactionnel.
  - Si le client est dans une transaction alors l’objet n’est pas inclus dans la transaction
- **Supported [Transaction(TransactionOption.Supported)]**
  - Indique que cet objet peut être inclue dans une transaction.
  - Si le client est dans une transaction alors l’objet est inclus dans la transaction ; sinon l’objet s’exécute hors transaction.
- **Required (default) [Transaction(TransactionOption.Required)]**
  - Indique que cet objet s’exécute toujours dans une transaction.
  - Si le client est dans une transaction alors l’objet est inclus dans la transaction ; sinon l’objet démarre une transaction.
- **RequiresNew [Transaction(TransactionOption.RequiresNew)]**
  - Indique que cet objet nécessite une nouvelle transaction.
  - Si le client est dans une transaction alors l’objet démarre une nouvelle transaction.

© 2007, M. Riveill

Transaction
Exemple : la classe

- **Contrôle de la transaction par programme**

```
[Transaction(TransactionOption.Required)]
public class Account : ServicedComponent {

public void Debit(int amount) {
   // effectue qq actions
   // éventuellement avec accès à une BD
   if (‘c’est bon’) {
     // Commit
     ContextUtil.SetComplete();
   } else {
     // Abort
     ContextUtil.SetAbort();
   }
  }
}
```

- **En utilisant un attribut**

```
[Transaction(TransactionOption.Required)]
public class Account : ServicedComponent {
  // en l’absence d’exception commit
  // sinon abort
  [AutoComplete]
  public void Debit(int amount) {
    // effectue qq actions
    // éventuellement avec accès à une BD
  }
}
```

© 2007, M. Riveill

Transaction
Exemple : main

```
using System;
using System.Runtime.CompilerServices;
using System.EnterpriseServices;
using System.Reflection;

// Nom de l'application pour COM+
[assembly: ApplicationName("TestApp")]
// Nom fort de l'assembly.
[assembly:AssemblyKeyFileAttribute("TestApp.snk")]
public class client {
  public static int Main() {
    Account accountX = new Account();
    accountX.Debit(100);
    return 0;
  }
}
```

© 2007, M. Riveill

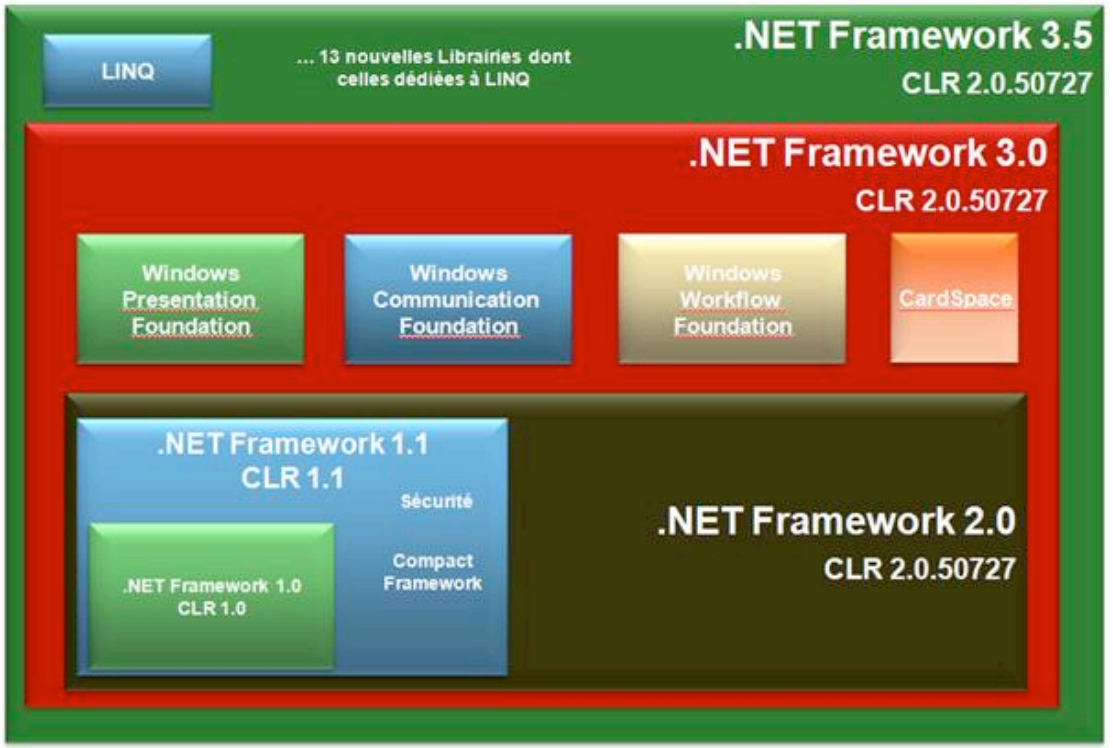
Demo

Transaction

.Net aujourd’hui

Le calendrier

- 2002
  - .NET Framework 1.0
- 2003
  - .NET Framework 1.1
- Fin 2005
  - .NET Framework 2.0
- Fin 2006
  - .NET Framework 3.0
- Fin 2007 (début 2008)
  - .NET Framework 3.5



© 2007, M. Riveill

.Net 2.0

- Des bugs ont été corrigés
- Pleins de nouvelles classes
  - ADO, ASP, WindowsForm
- Pleins de petites nouveautés
  - Généricité, delegate générique
  - Classe partielle
  - On signe à la compilation /keyfile, /delaysign
- Pleins de nouveaux outils
  - Resgen.exe : internationalisation des applications
  - Msbuild.exe : outil pour la construction d'une application
- Prise en compte des architectures 64 bits
- Mais rien de fondamental
  - Le détail ici : http://msdn.microsoft.com/fr-fr/library/t357fb32.aspx

© 2007, M. Riveill

.Net 3.0

- Sur ensemble du .NET Framework 2.0
  - Aucune modification du framework
  - Mais beaucoup de choses nouvelles
    - De l'anecdotique
    - Et du fondamental
- Comprend 4 briques fondamentales
  - Windows Presentation Foundation (WPF - Avalon)
  - Windows Communication Foundation (WCF - Indigo)
  - Windows Workflow Foundation (WF)
  - Windows CardSpace (WCF)
- Développé pour Windows Vista
  - Mais disponibles en tant que composants pour Windows XP/2003
  - Les APIs de .Net 3.0
    - seront reprise dans Vista
    - remplaceront celles de XP

© 2007, M. Riveill

Windows Presentation Foundation

WPF (Avalon)

- http://wpf.netfx3.com
- **Système d'interface graphique**
  - Unifie de l'interface utilisateur, documents, médias
  - Utilise un moteur de composition pour permettre l'affichage vectoriel
  - Modèle de programmation déclarative basé sur XAML (eXtensible Application Markup Language)
    - Séparation des rôles entre designer et développeur

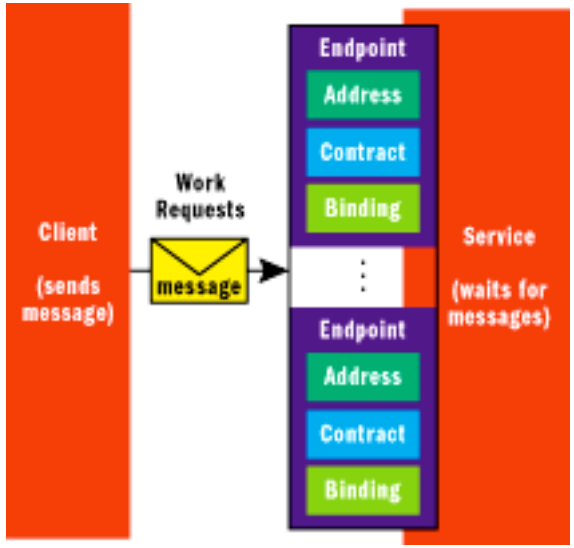
  - Faciliter le déploiement sur différentes plates-formes
    - WPF /E (Windows Presentation Foundation /EveryWhere)
      - PC, Mac, Pocket PC, Smartphone, etc.
    - Applications qui s'exécutent dans un navigateur web (en ligne)

© 2007, M. Riveill

Windows Communication Foundation

WCF (Indigo)

- http://wcf.netfx3.com/
- Unifie le développement des applications distribués
  - Multi-protocoles
  - Sûres
    - intérité, confidentialité, authentification autorisation, audit de la communication
  - Fiables
    - transactions, isolation, concurrence
- Approche orientées services
  - Services isolés
  - Indépendant de la localisation
  - Indépendant du transport, du protocole et du format
  - Indépendant des plateformes et de l'impléentation
  - Doit pouvoir fonctionner quel que soit les dispositifs de monté en charge mis en place (ferme de processeur)
  - Doit pouvoir fonctionner même lorsque l'ne des parties n'est pas disponible



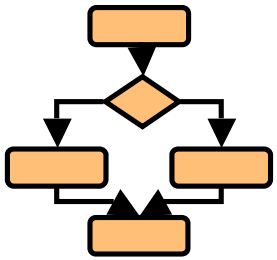
© 2007, M. Riveill

Windows Workflow Foundation

WF

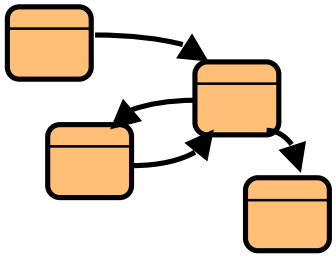
- http://wf.netfx3.com
- **Permet la programmation de workflow selon différents paradigmes**

Séquentiel



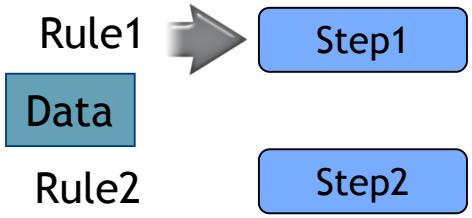
La structure définit le flux d'exécution

Automate à états



Les événements définissent le flux

A base de règles



Les données définissent le flux

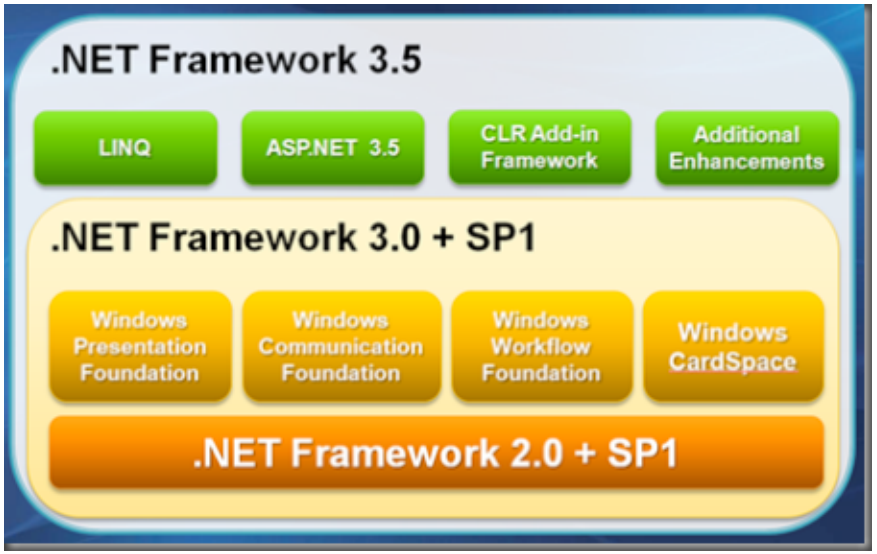
Windows CardSpace

WCS (Infocard)

- http://cardspace.netfx3.com
- Un meta système d'indentité
  - Carte d'identités virtuelles
  - Fait suite à Passport et reprend des éléments de WS-security, WS-Trust
  - Identité = ensemble d'affirmations
    - Affirmation = qq chose que qq dit de moi
      ou qq chose que je dis de moi
  - Chaque identité est représenté par une carte (logotype - RFC3709)
    - Un utilisateur peut avoir plusieurs identités / cartes
      - Une carte anonyme, émise par moi avec des informations génériques et fausses
      - Une carte personnelle, émise par moi mais avec des informations vraies
      - Une carte émise par un fournisseur d'identité réputé
      - Des cartes spécialisées (banques, réputation sur eBay, …)

.Net 3.5

- On continue l'enrichissement
  - Ou l'engraissement
  - On s'éloigne de la norme d'origine
- 3 nouveautés
  - LINQ (Language INtegrated Query)
  - Nouvelle version d'ASP
  - CLA Add-in framework
- Et d'autres améliorations plus mineures



© 2007, M. Riveill

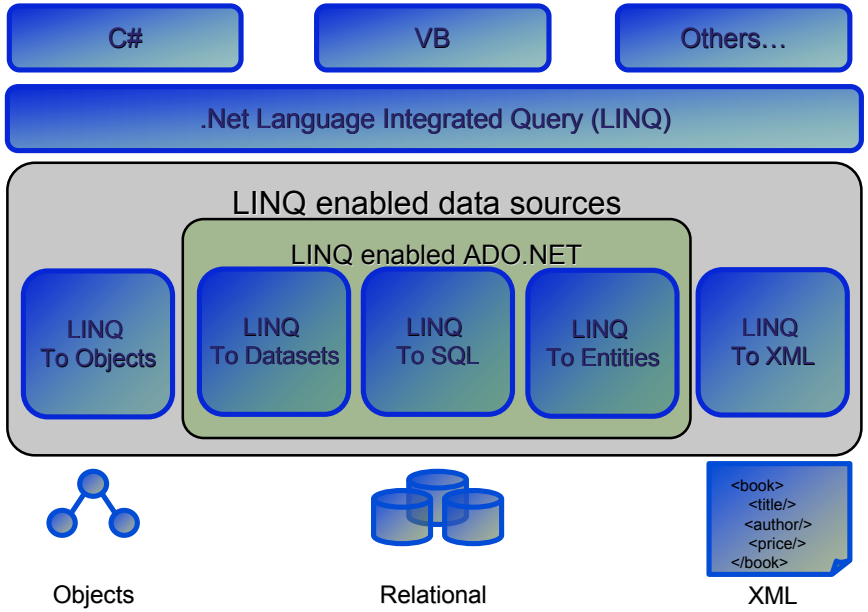
LINQ

Language INtegrated Query

- **Effectuer des requêtes sur des données de différents types**

```
var vieuxBasques = Personnes
    .Where(s => s.Age > 50)
    .Where(s => s.CP == "64")
    .OrderBy(s => s.Nom)
    .Select(s => s);
```

  - **Disponible pour différents langages de .Net (VB, C#)**
  - **Possibles pour les**
    - **Objets**
    - **Dataset**
    - **Base de données**
    - **Documents XML**
  - **Reprend les opérateurs usuels des bases de données**
    - `select, where, min, max, count, …`
    - **40 opérateurs**

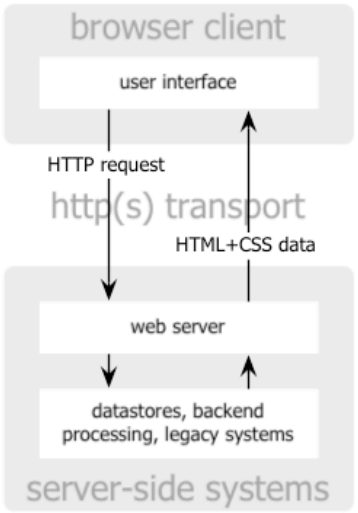


© 2007, M. Riveill

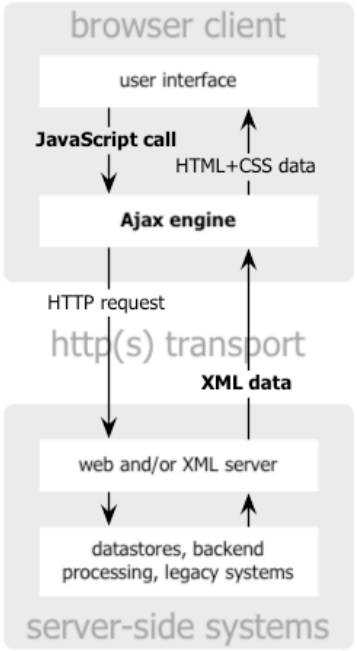
ASP.Net 3.5

ASP.Net AJAX

- **Complément à ASP.Net 2.0**
- **Intégration d'AJAX (Asynchronous JavaScript And XML)**
- **Utilise : HTML ou XHTML, les feuilles de styles CSS, JavaScript, le modèle objet de document (DOM), XML, XSLT, et l'objet XMLHttpReques**

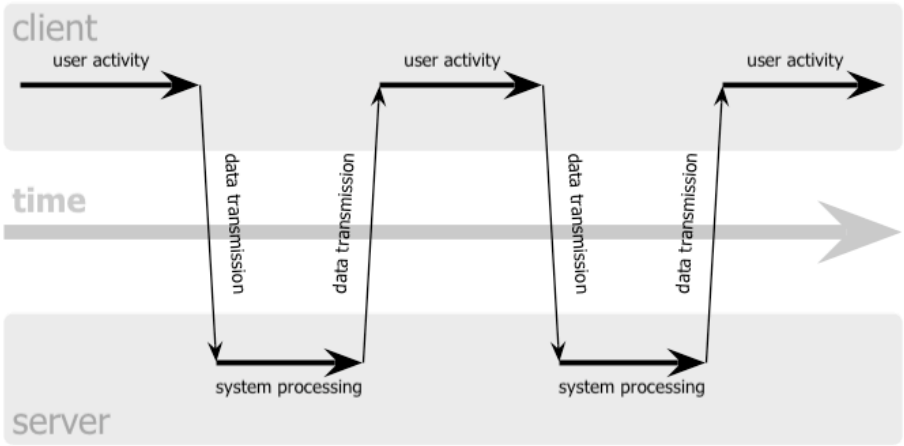


classic
web application model

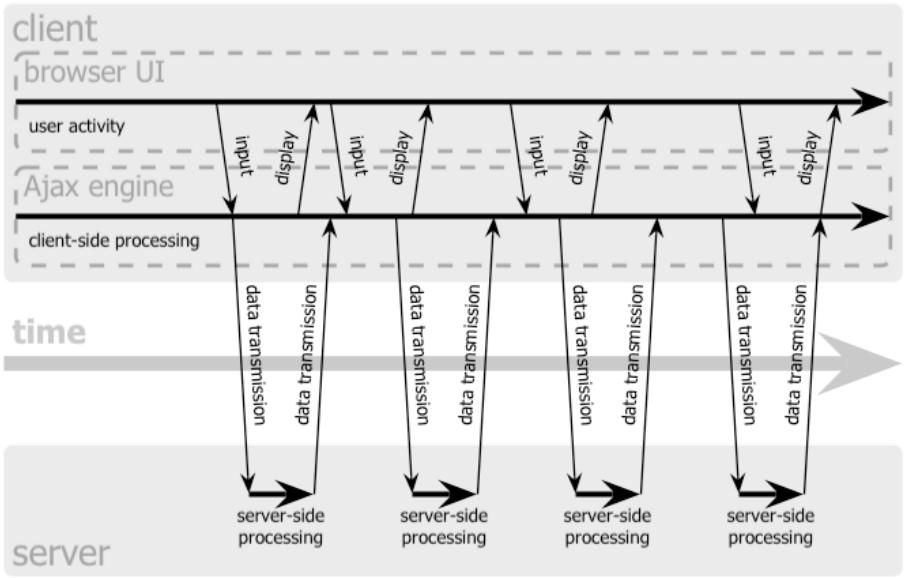


Ajax
web application model

classic web application model (synchronous)

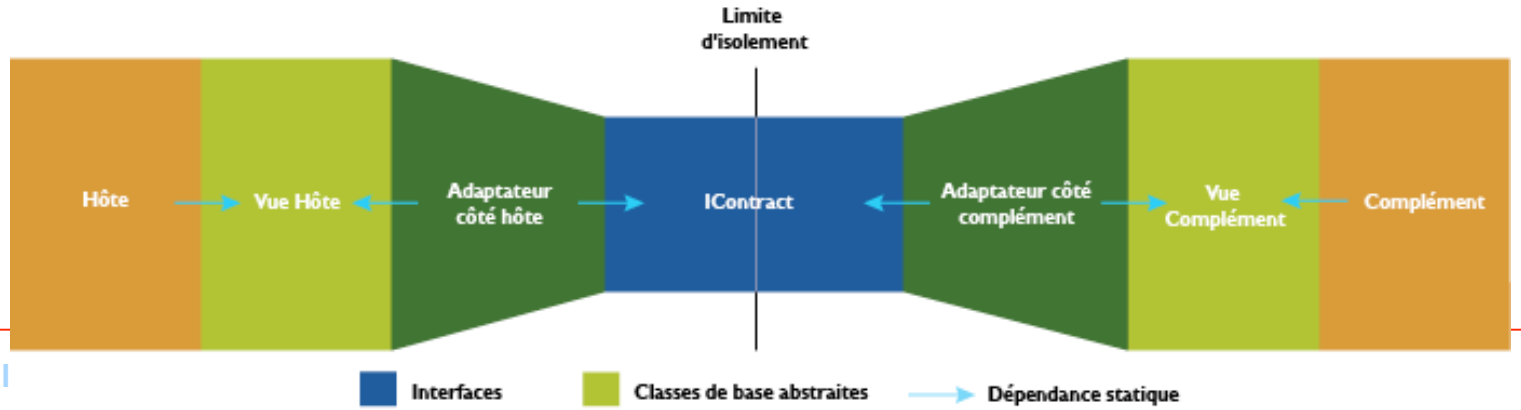


Ajax web application model (asynchronous)



CLR Add-IN framework

- Favoriser l'extension sûre d'une application
  - Nouvel espace de nom du CLR : 'System.AddIn'
- Principales fonctionnalités
  - Découvertes
  - Activation
  - Isolation
  - Déchargement
  - Sandbowing



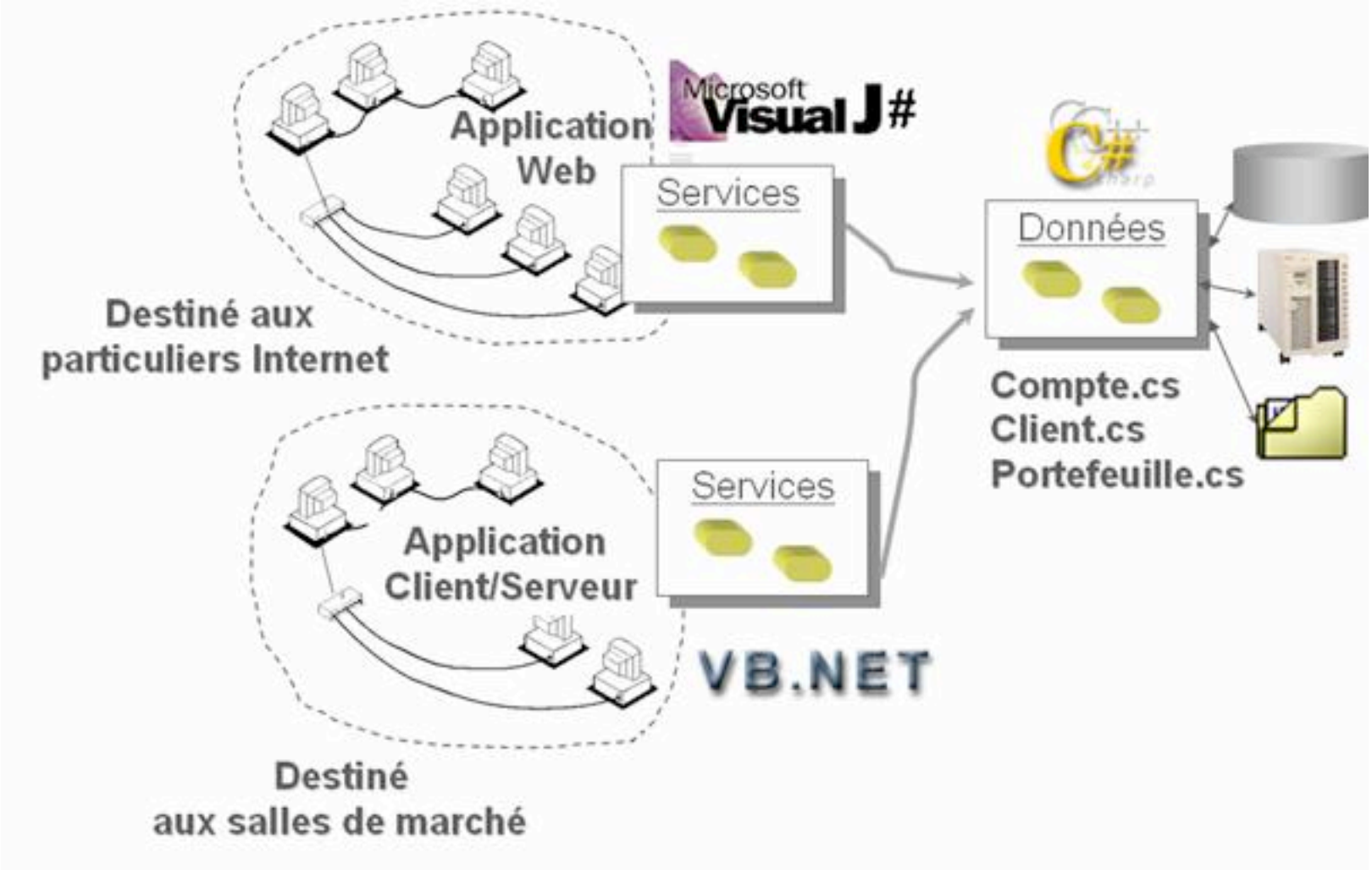
© 2007, M. Riveill

Evaluation

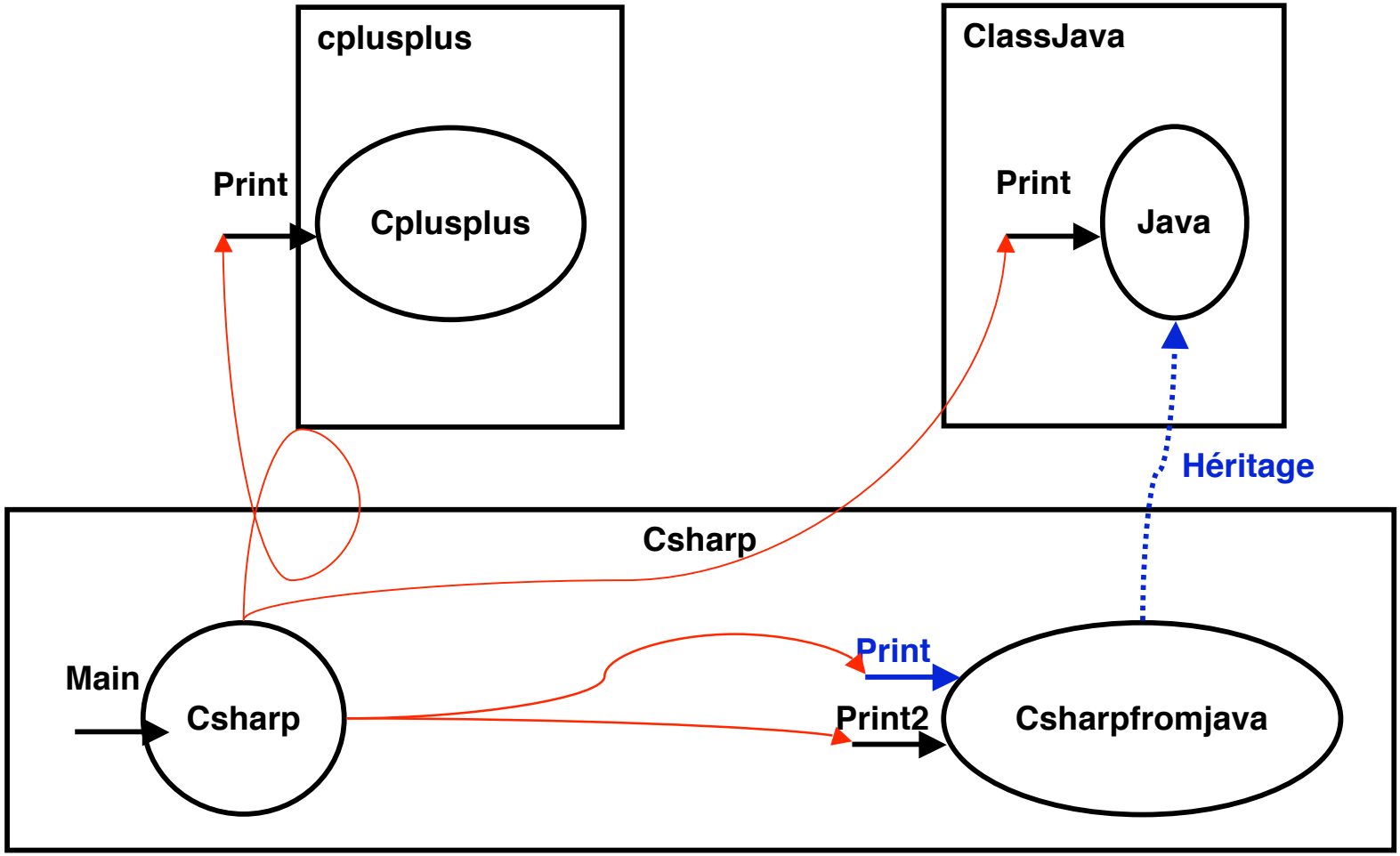
Le multi-langage

Comparaison avec le monde Java

Le multi-langage est une réalité



Le multi-langage est une réalité

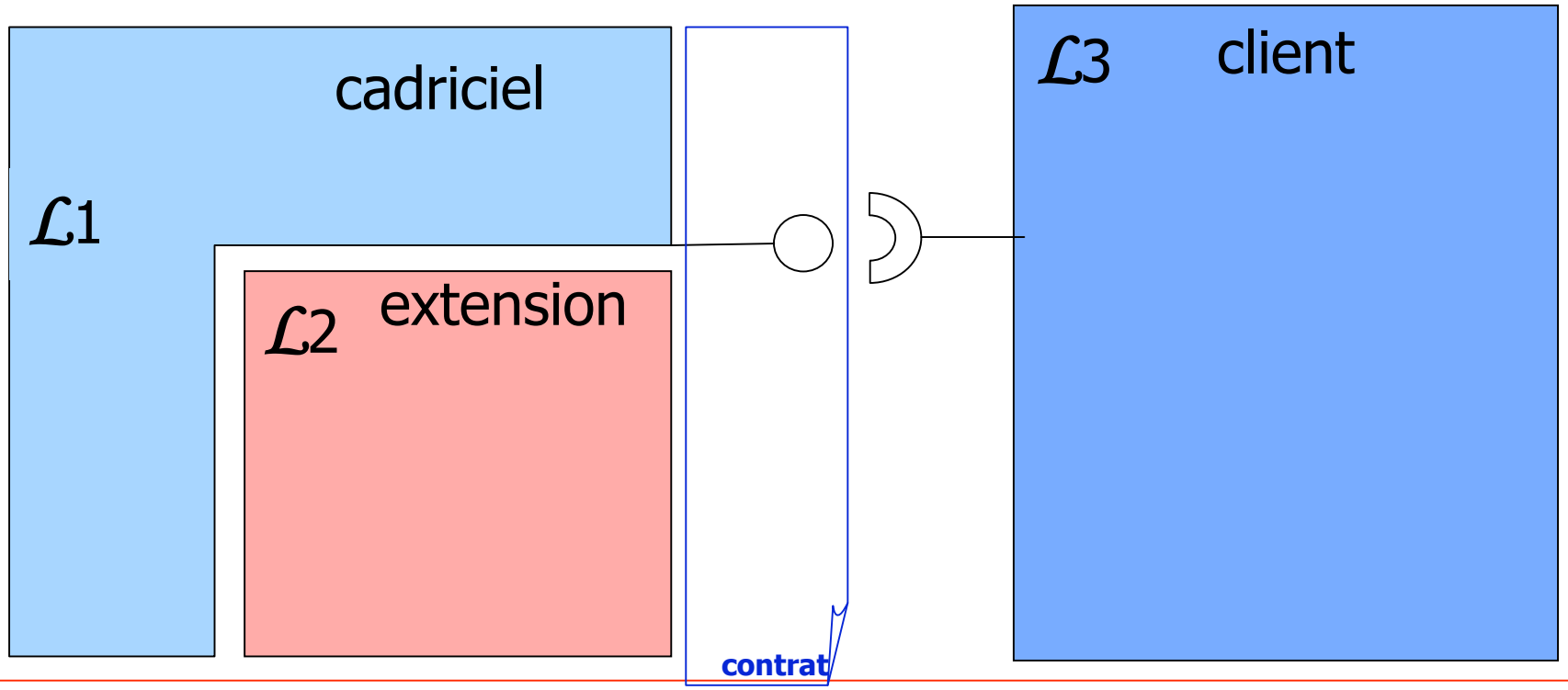


On peut écrire des classes qui héritent de classes écrites dans un autre langage

© 2007, M. Riveill

Mais…

- Peut-on assembler des composants sans se préoccuper du langage d'implémentation ?
- Est-ce que le comportement du client dépend du langage d'implémentation du serveur, du cadriciel ou de l'extension ?



© 2007, M. Riveill
Etude réalisées par Antoine.Beugnard@enst-bretagne.fr

Redéfinition / surcharge

- **Redéfinir**
  - Modifier une méthode (corps)
  - Spécialiser des arguments (covariance)
  - Généraliser des arguments (contravariance)
- **Surcharger**
  - Définir une *nouvelle* méthode en utilisant le même nom (ou le même symbole)

Mécanisme ou intention implicite
Interprétations délicates…

© 2007, M. Riveill

Une première expérience

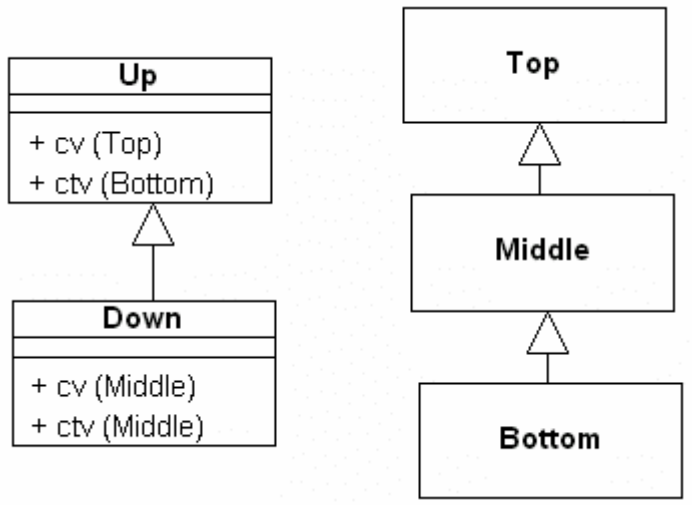
- **Héritage et surcharge ne sont pas identiques pour tous les langages**

- **Une hiérarchie de classe**

```
class Top
class Middle extends Top
class Bottom extends Middle
```

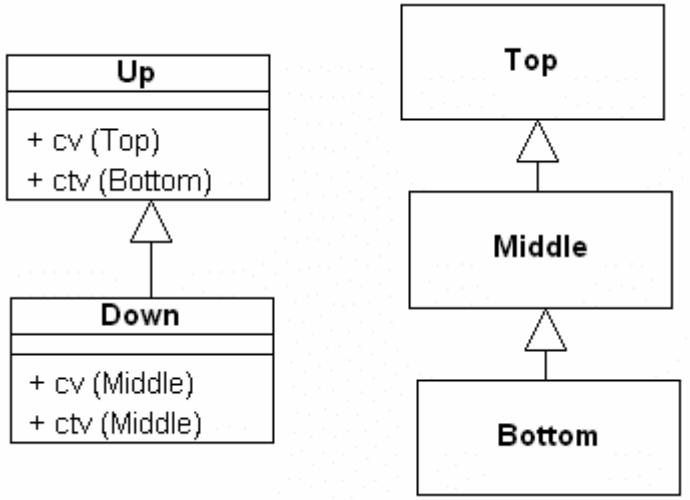
- **Une seconde hiérarchie de classe**

```
class Up
  // cv : redéfinition covariante
  method cv(Top t) print "Up"
  // ctv : redéfinition contravariante
  method ctv(Bottom b) print "Up"

class Down extends Up
  method cv(Middle m) print "Down"
  method ctv(Midle m) print "Down"
```



© 2007, M. Riveill

Une première expérience



- **Un programme**

```
Up u = new UpDown
d = new Down
Up ud = new Down

// puis tous les appels possibles…
u.cv(new Top); d.cv(new Top); ud.cv(new Top) ;
u.cv(new Middle) ; d.cv(new Middle) ; …
u.cv(new Bottom) ; …

u.ctv(new Top) ; …

// soit 18 appels…
```

© 2007, M. Riveill

La signature du langage C++

C++	u	d	ud
cv(Top)	Up	Error	Up
Cv(Middle)	Up	Down	Up
cv(Bottom)	Up	Down	Up
ctv(Top)	Error	Error	Error
ctv(Middle)	Error	Down	Error
ctv(Bottom)	Up	Down	Up

Commentaire : C++ .Net se comporte comme C++

© 2007, M. Riveill

La signature du langage C#

C#	u	d	ud
cv(Top)	Up	Up	Up
Cv(Middle)	Up	Down	Up
cv(Bottom)	Up	Down	Up
ctv(Top)	Error	Error	Error
ctv(Middle)	Error	Down	Error
ctv(Bottom)	Up	Down	Up

© 2007, M. Riveill

La signature du langage VB

VB	u	d	ud
cv(Top)	Up	Up	Up
Cv(Middle)	Up	Down	Up
cv(Bottom)	Up	Down	Up
ctv(Top)	Error	Error	Error
ctv(Middle)	Error	Down	Error
ctv(Bottom)	Up	Up	Up

© 2007, M. Riveill

Signature des 3 langages
C++ / C# / VB

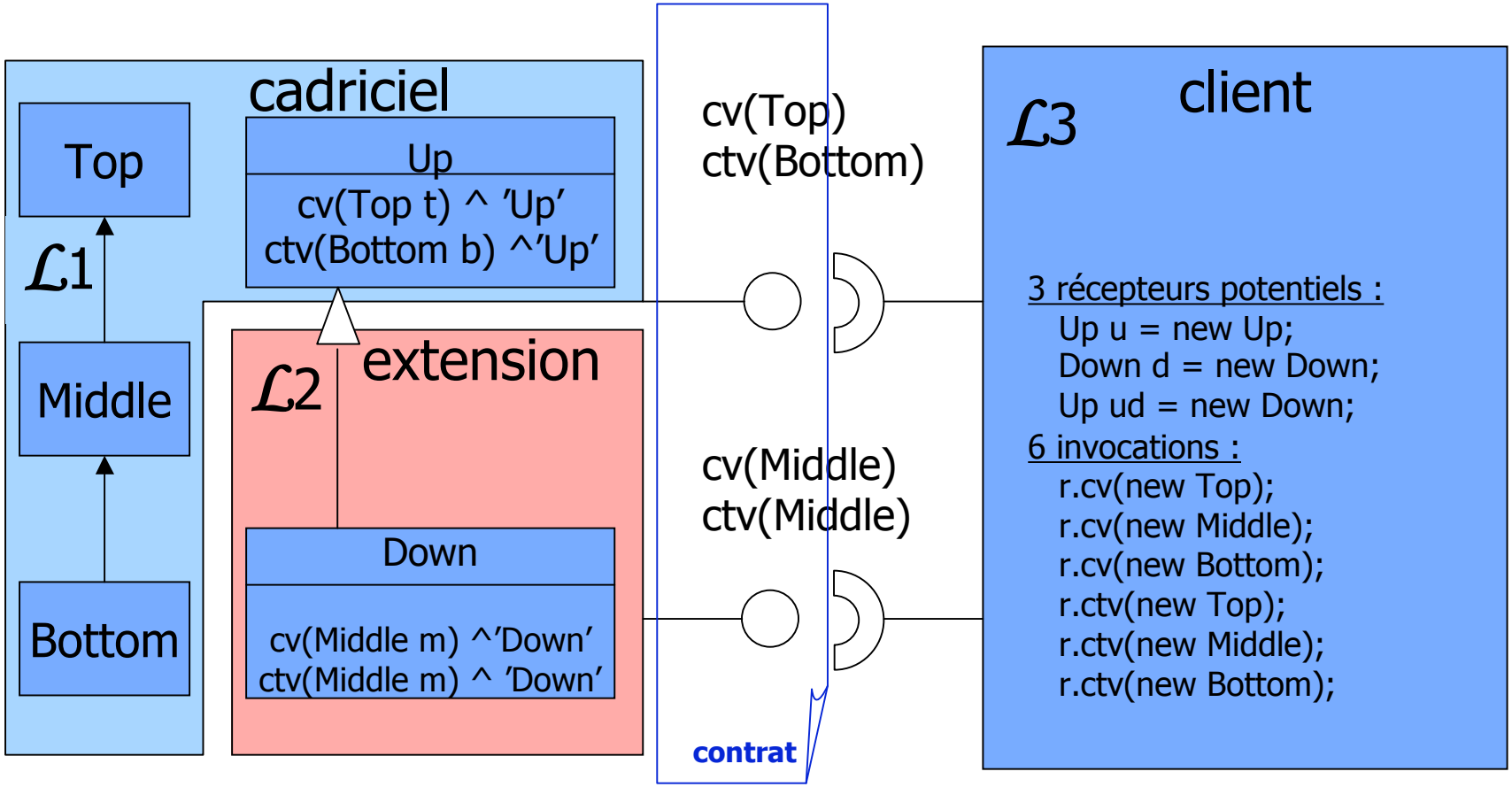
C++ / C# / VB	u	d	ud
cv(Top)	Up	Error I Up I Up	Up
Cv(Middle)	Up	Down	Up
cv(Bottom)	Up	Down	Up
ctv(Top)	Error	Error	Error
ctv(Middle)	Error	Down	Error
ctv(Bottom)	Up	Down I Down I Up	Up

L'étude a été menée pour de nombreux autres langages :
http://perso-info.enst-bretagne.fr/~beugnard/papiers/lb-sem.shtml

© 2007, M. Riveill

Une seconde expérience

L'expérience reprend le même principe que la précédente



© 2007, M. Riveill

La procédure de comparaison

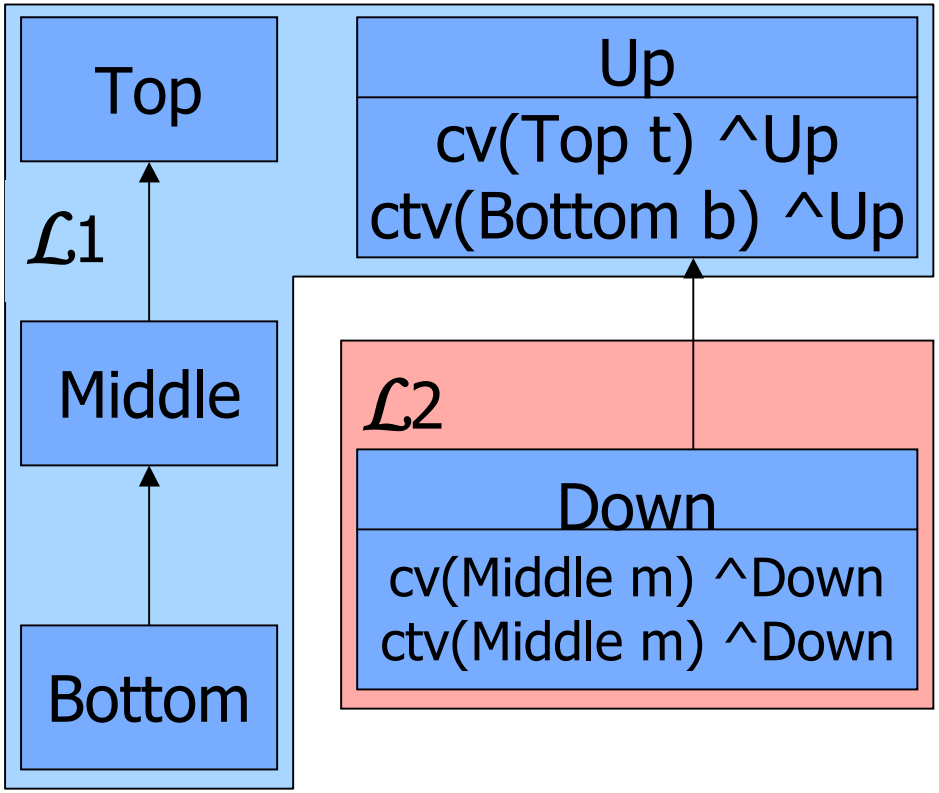
- **Utilisation de .NET**
  - Visual Basic, C# et C++

Langages de la même « famille »

- **Chacun est utilisé pour**
  - Le cadriciel
  - L'extension
  - Le client
- **Soit 27 assemblages de composants…**

© 2007, M. Riveill

La procédure de comparaison



3 récepteurs potentiels :

```
Up u = new Up;
Down d = new Down;
Up ud = new Down;
```

6 invocations :

```
r.cv(new Top);
r.cv(new Middle);
r.cv(new Bottom);
r.ctv(new Top);
r.ctv(new Middle);
r.ctv(new Bottom);
```

© 2007, M. Riveill

Client C# (L3)

L1 : cadriciel – L2 : extension – L3 : Client

L1 / L2	C#	VB	C++
C#	C#	C#	C++
VB	C#	C#	C++
C++	C#	C#	C++

© 2007, M. Riveill

Client Visual Basic (L3)

L1 : cadriciel – L2 : extension – L3 : Client

L1 L2	C#	VB	C++
C#	VB	VB	C++
VB	VB	VB	C++
C++	VB	C#	C++

© 2007, M. Riveill

Client C++ (L3)

L1 : cadriciel – L2 : extension – L3 : Client

L1 L2	C#	VB	C++
C#	Nouveau langage	Nouveau langage	Nouveau langage
VB	C++	C++	C++
C++	C++	C++	C++

© 2007, M. Riveill

Signature du nouveau langage

VB/C++ (???)	u	d	ud
cv(Top)	Up	Erreur	Up
Cv(Middle)	Up	Down	Up
cv(Bottom)	Up	Down	Up
ctv(Top)	Error	Error	Error
ctv(Middle)	Error	Down	Error
ctv(Bottom)	Up	Up	Up

Signature inconnue !

© 2007, M. Riveill

Synthèse

L1 : cadriciel – L2 : extension
L3 : Client (C#, VB, C++)

L1 L2	C#	VB	C++
C#	Client Client Nouveau langage	Client Client Nouveau langage	Cadriciel Cadriciel Nouveau langage
VB	Client	Client	Cadriciel
C++	Client	Client C# Client	Cadriciel

Attention : en .NET, les composants ne peuvent pas être considérés comme des boites noires…

© 2007, M. Riveill

Java vs .Net

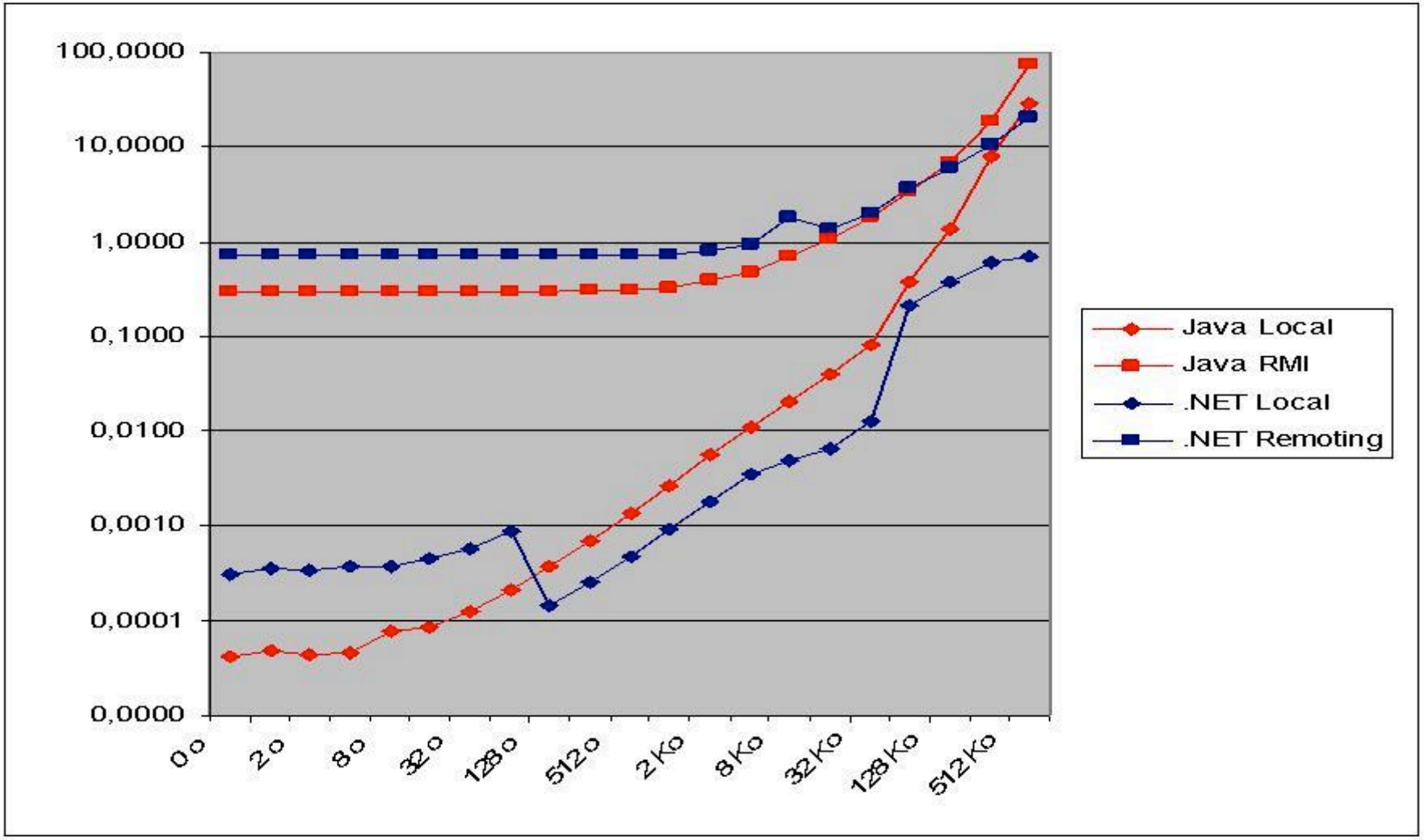
Performances comparatives

http://www.dotnetguru.org/articles/Comparatifs/benchJ2EEDotNET/J2EEvsNETBench.html

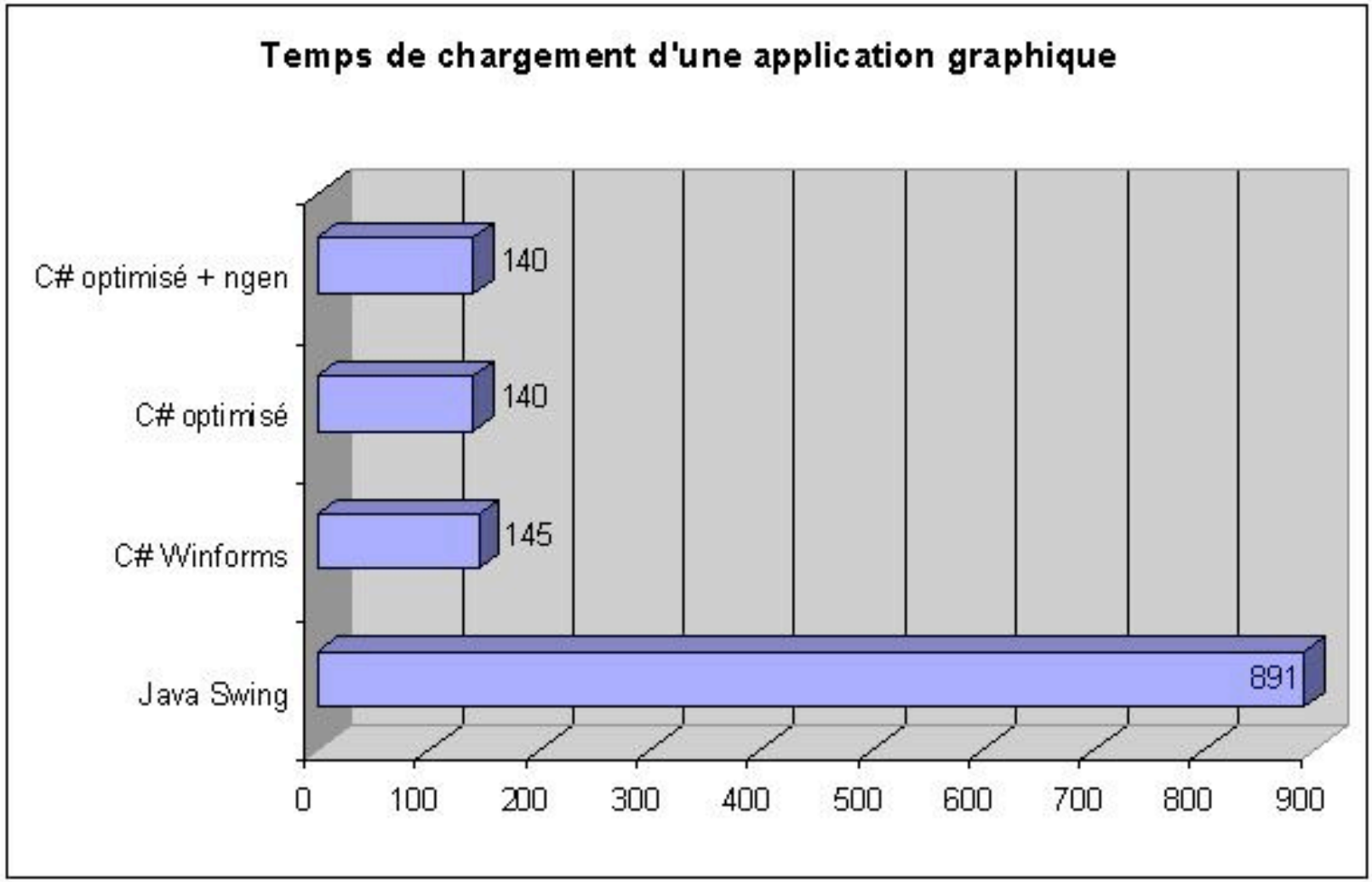
Historique

	1996	1998	2001	2005
Microsoft	Windows DNA *MTS (→ COM+)* *ASP* *ADO*		.NET Framework *CLR* *C#, VB.NET* *Nouvelle génération de COM+, ASP, ADO* *Web services / MyServices*	.Net 2.0 -Généricité
Java	Java *Java language* *Java VM* *J2SE*	J2EE *EJB* *JSP* *JDBC*		Java 1.5 -Boxing -Généricité

Appel de procédure

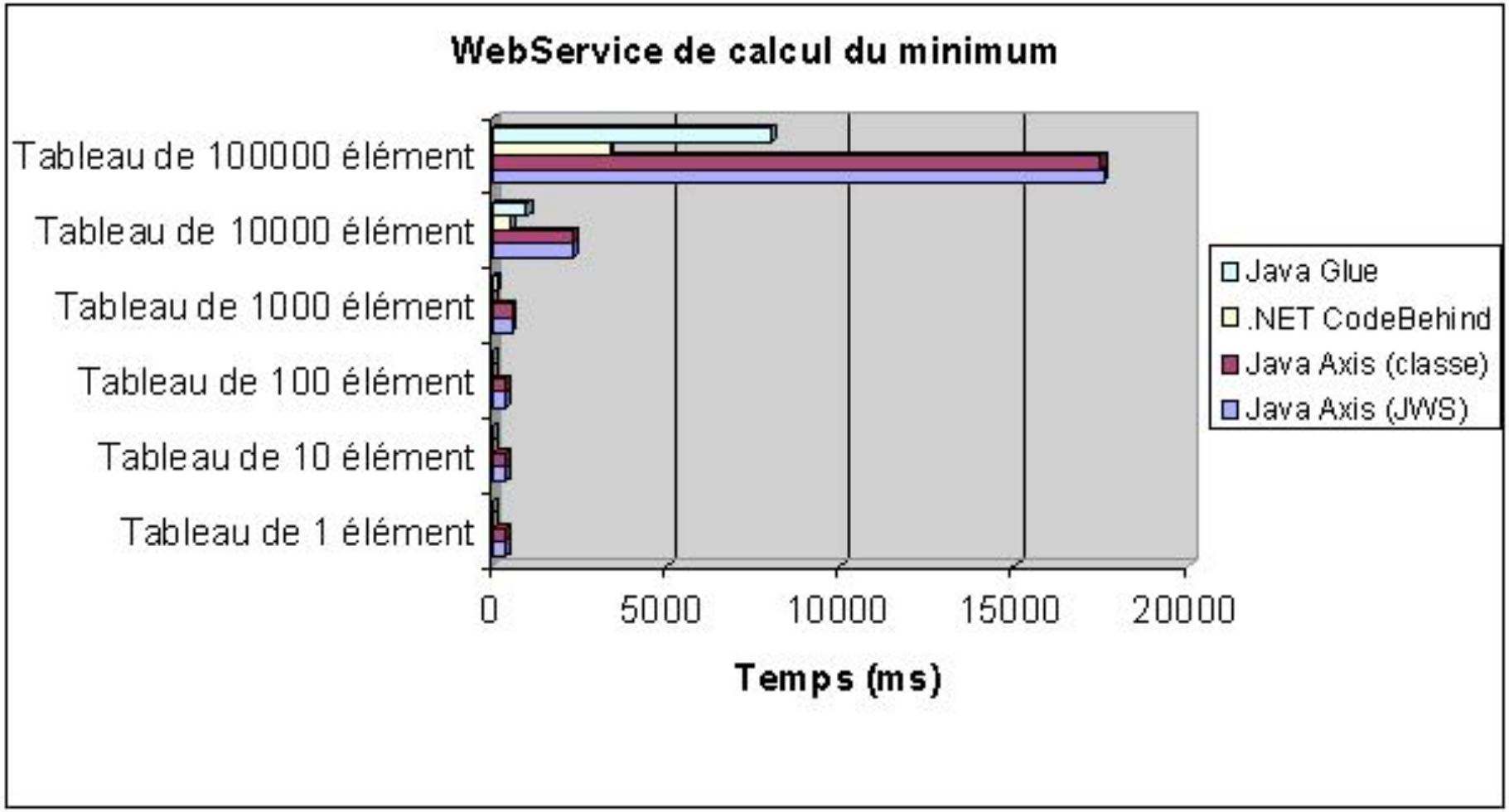


Application graphique



© 2007, M. Riveill

Services web



© 2007, M. Riveill

Quelques éléments de comparaison

	.Net	J2EE
Technologie	Propriétaire (norme ECMA / ISO)	Standard (contrôlée par JCA)
Vendeurs	Microsoft	30+
Plate-forme cible	Windows	Indépendant de la plate-forme
Langage de programmation	C# et les autres	Java
Terminaux léger	Windows CE/ .NET compact Framework	Java 2 Micro-Edition (J2ME)
Plate-forme (gratuite)	Framework .Net Cassini MSDE	J2EE – Jonas (par exemple) Tomcat MySQL

© 2007, M. Riveill

Quelques éléments de comparaison

	.Net	J2EE
Serveur d'application	Windows 2000/XP + Services applicatifs (IIS,NLB,MSMQ,COM+) + .NET Framework	IBM WebSphere Application Server BEA WebLogic Application Server
Environnement de développement (commerciaux)	Visual Studio.Net	IBM WebSphere Application Developper BEA CAJUN Borland JBuilder Oracle JDevelopper
Environnement de développement (gratuit)	WebMatrix	Eclipse

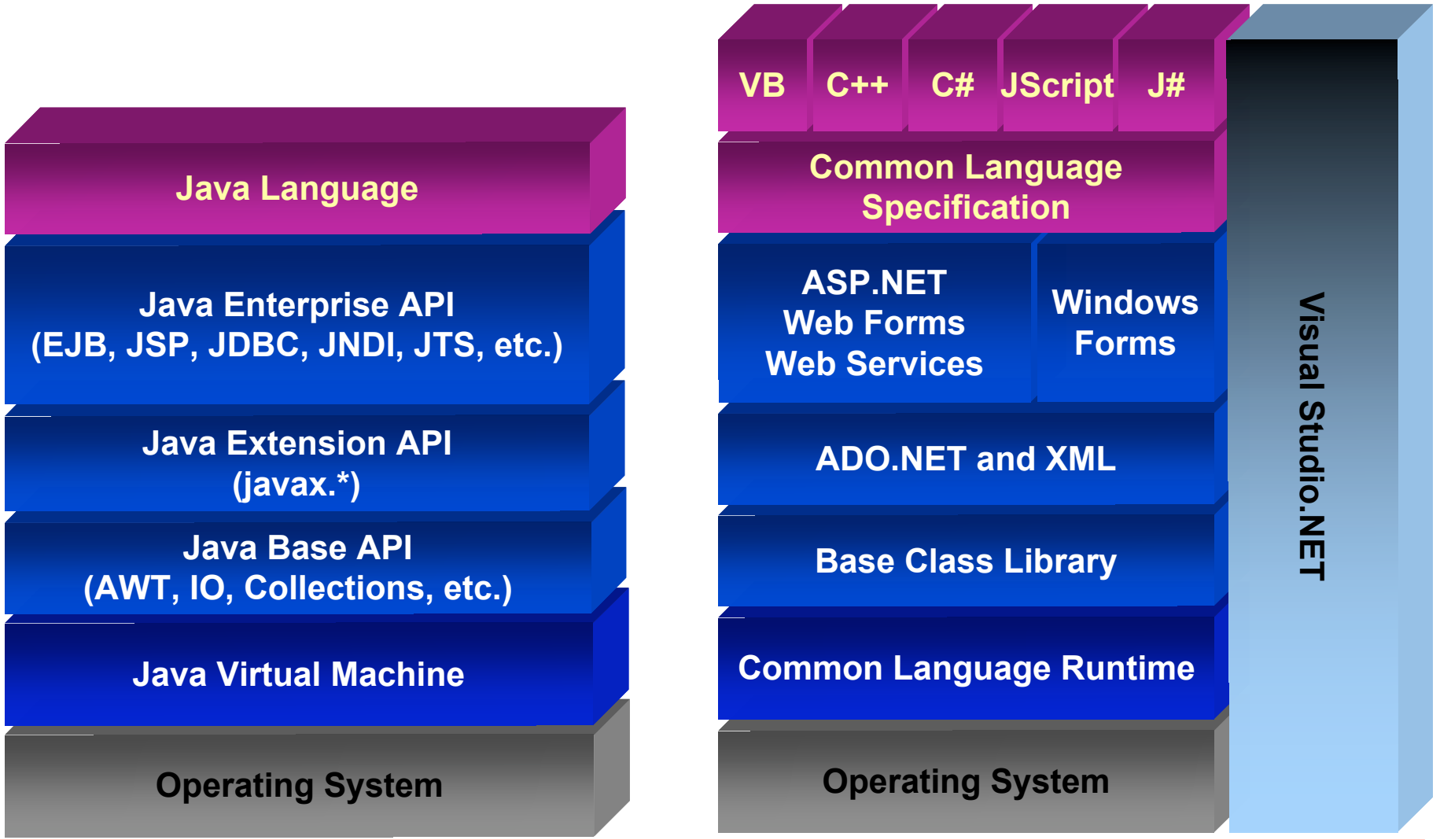
© 2007, M. Riveill

Autres éléments de comparaison

	.Net	J2EE
Langage interprété	MSIL (typé)	Java Bytecode
Environnement d'exécution	CLR	JVM/JRE
Client 'graphique'	MFC	Swing
Pages dynamiques	ASP.Net	JSP/Servlets
Composants métiers	Managed .Net component	EJBs
Intégration base de données	ADO.Net	EJB-SQL/JDBC
Intégration message	MSMQ	Msg EJBs / JMS
Intégration Web service	Oui	Oui
Intégration application	COM	Java Connector Architecture-JCA
Service miffleware (persistance, sécurité, transaction, …)	OUI (COM+)	OUI

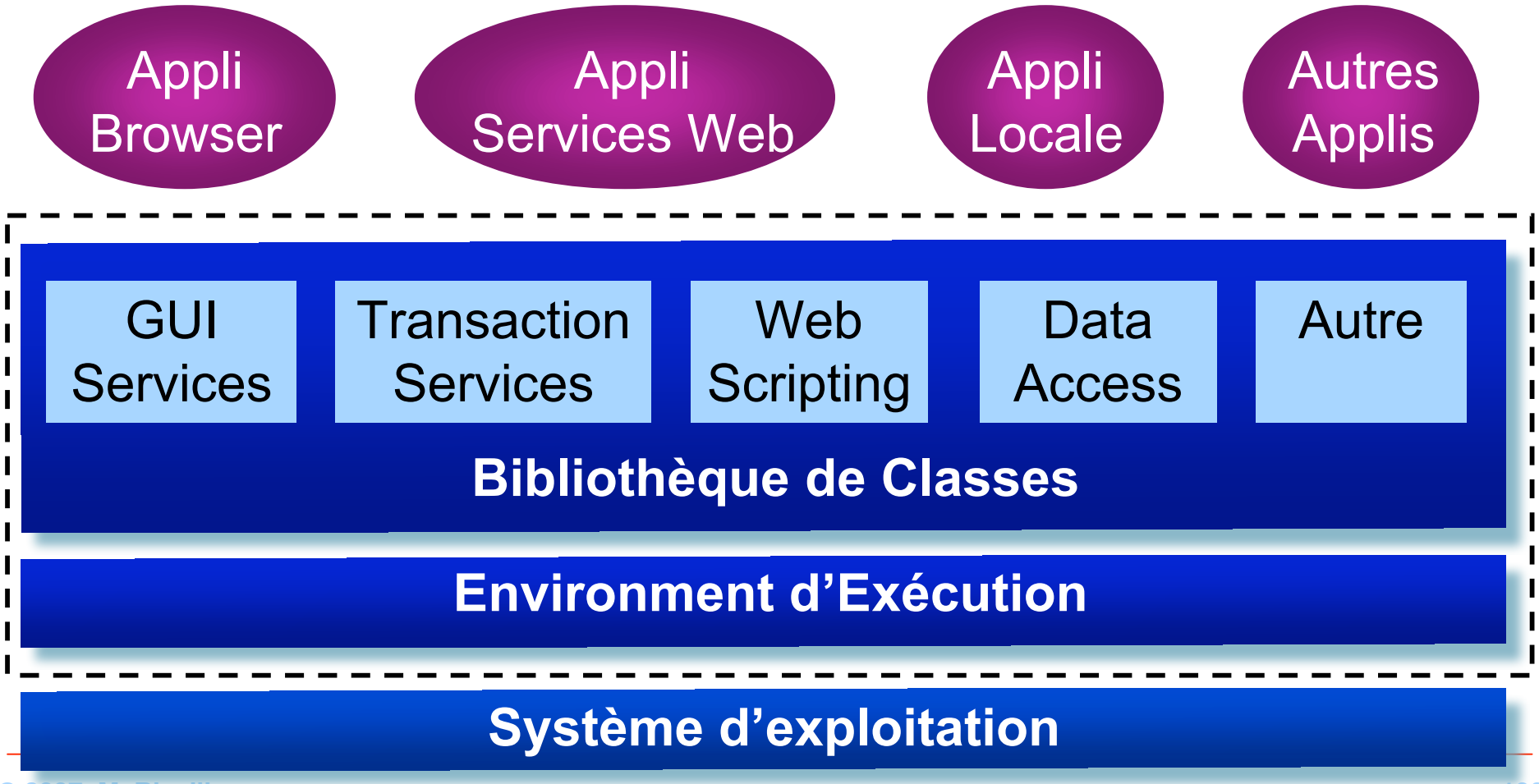
© 2007, M. Riveill

Les plates-formes



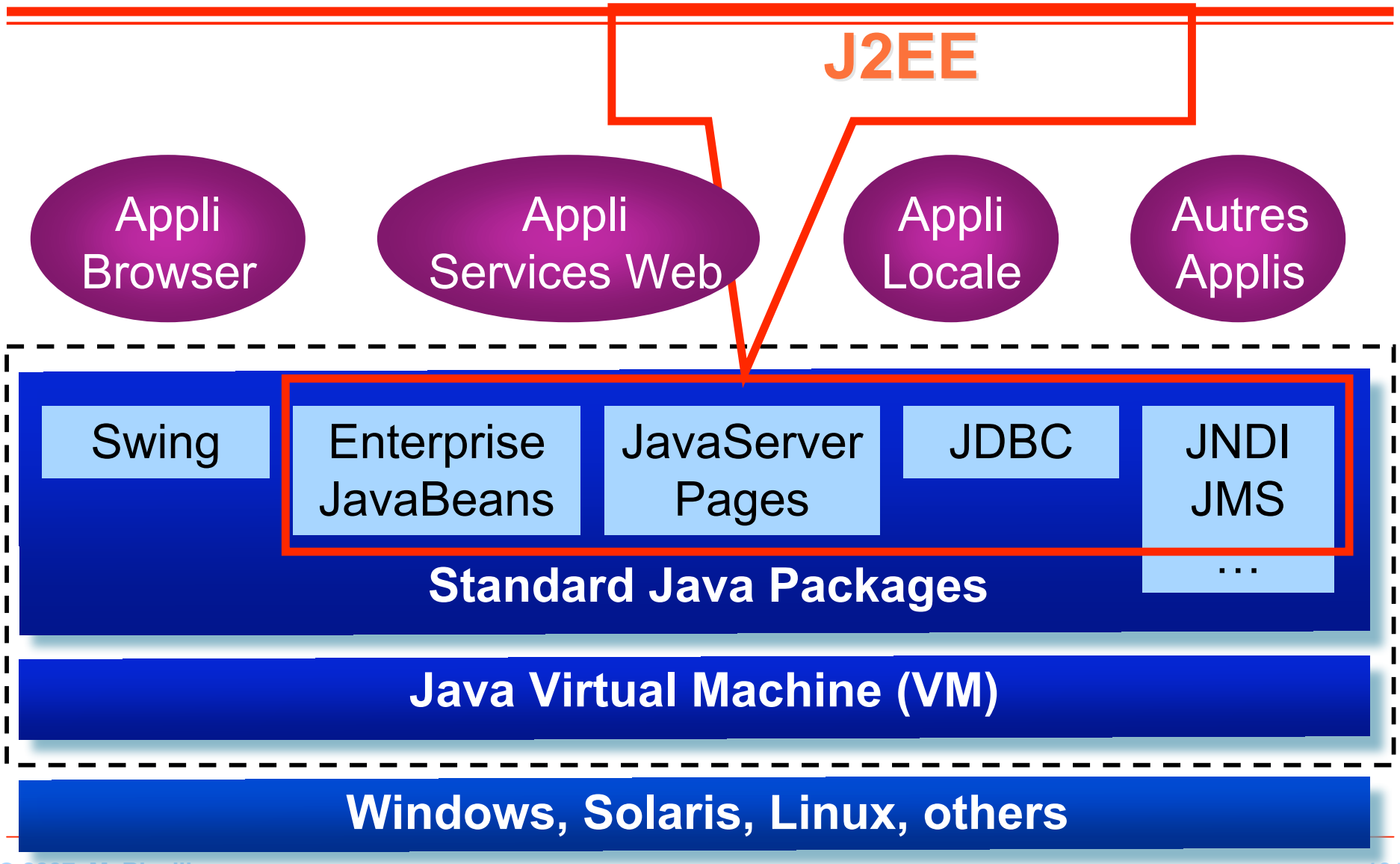
© 2007, M. Riveill

Serveur d' Applications



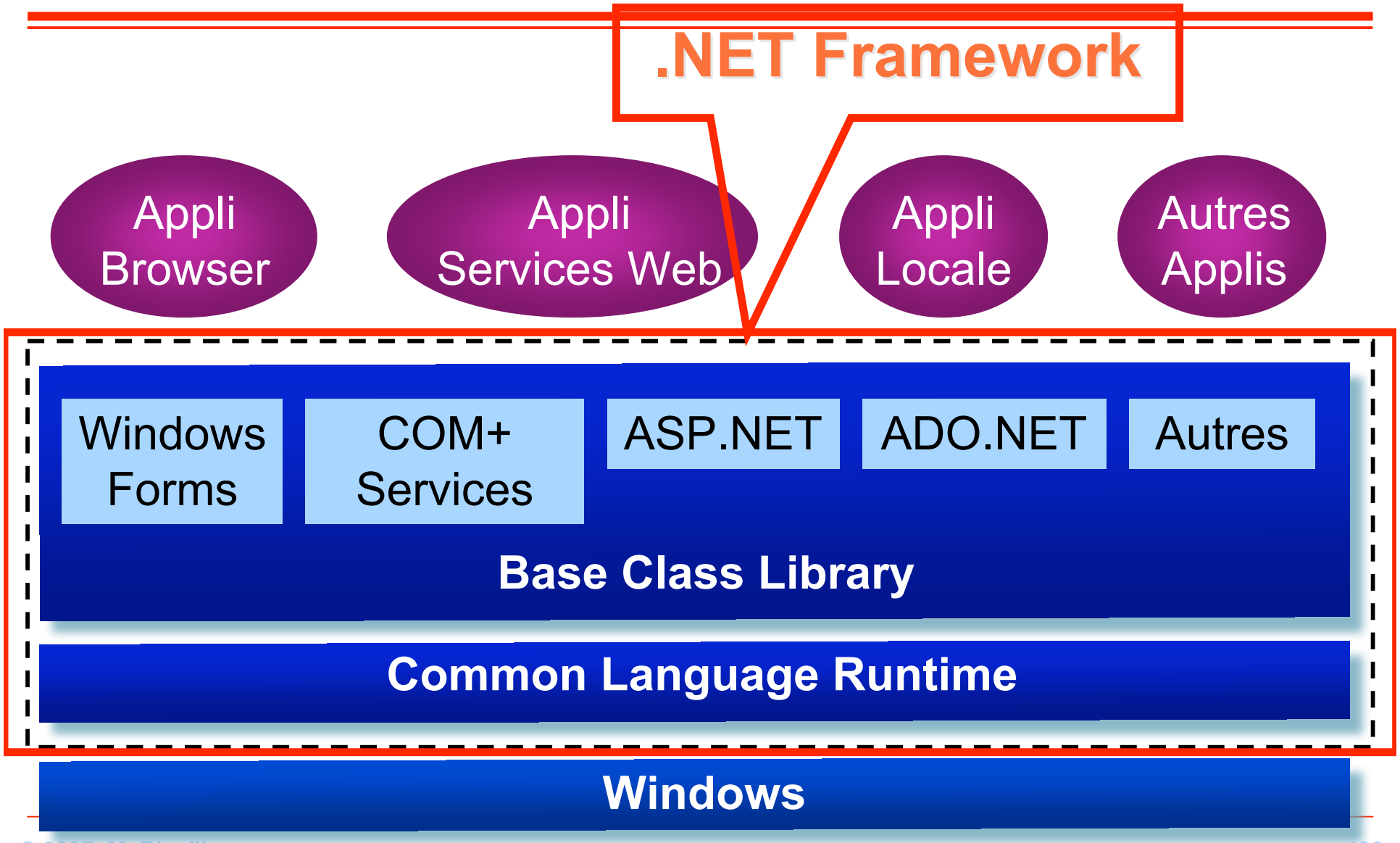
© 2007, M. Riveill

Les Serveurs d'Applications Java



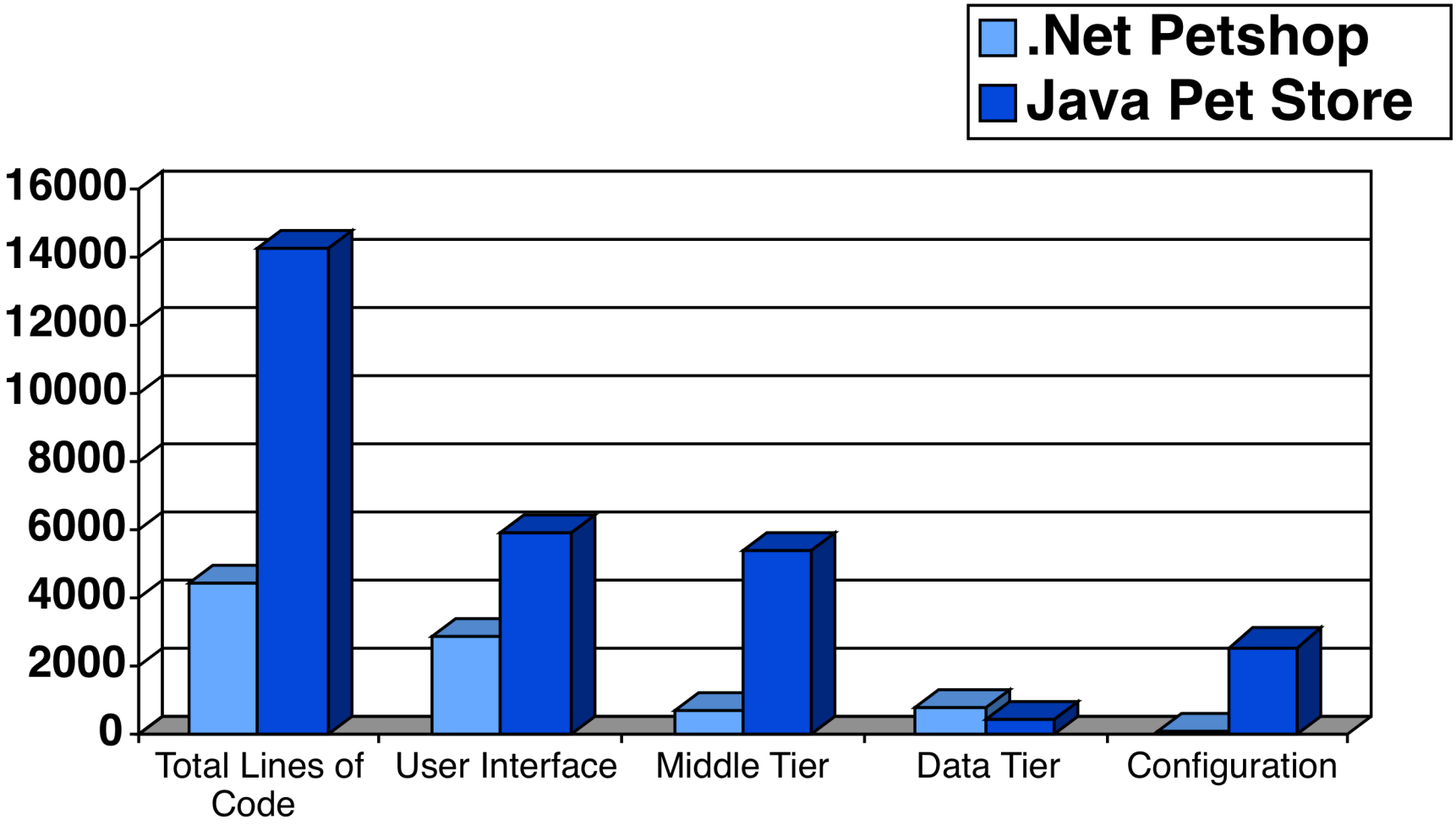
© 2007, M. Riveill

Le .NET Framework



© 2007, M. Riveill

Pet Store
Lignes de codee



© 2007, M. Riveill

Pet Store

Influence du serveur application

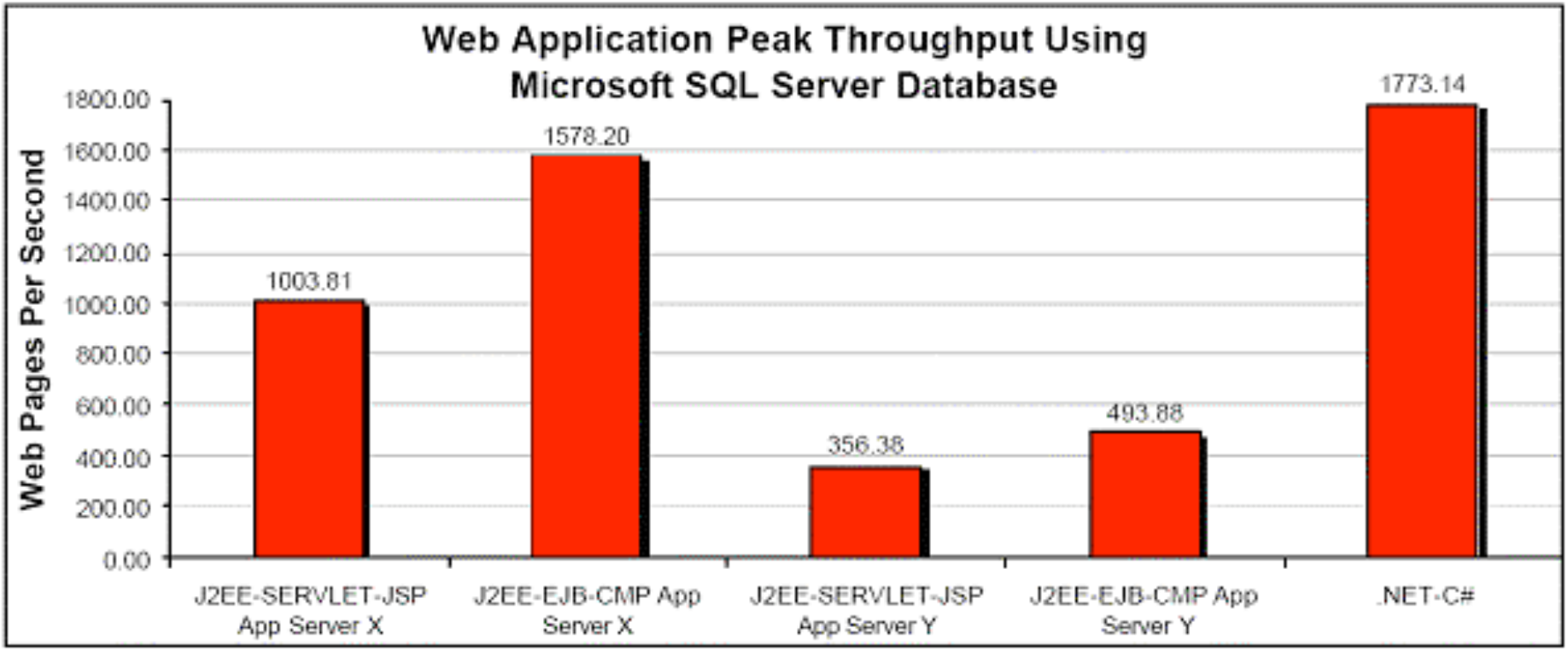


Figure 4, the maximum throughput achieved during the web application tests using Microsoft SQL Server 2000 database.

© 2007, M. Riveill

Pet Store
services Web

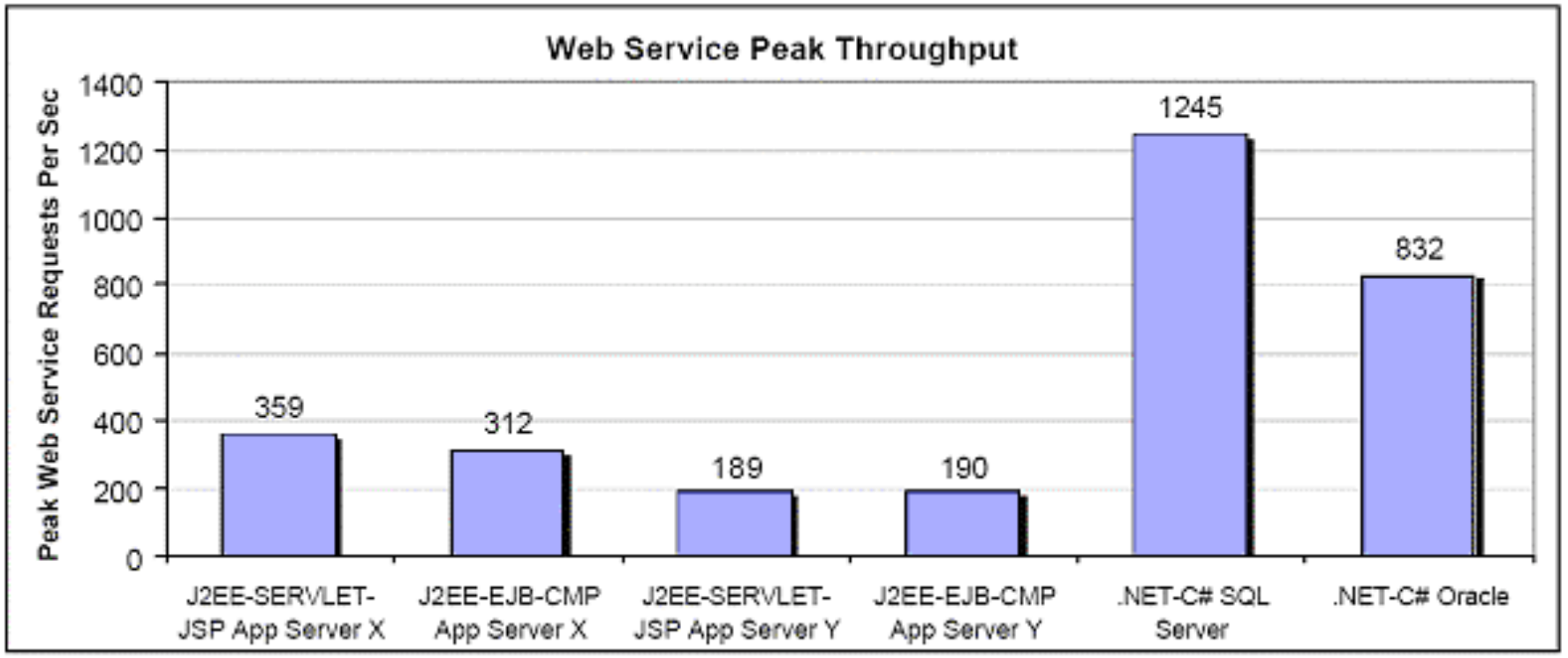


Figure 2, the maximum throughput achieved by each configuration during the web service tests.

© 2007, M. Riveill

Quatrième Partie

Pour aller plus loin

Inside ROTOR

- Pour ceux qui veulent regarder le cœur du CLR
  - Gestion de composants, chargement dynamique de code
- Exemple d'implémentation des standards ECMA
  - 334: C# language
  - 335: Common Language Infrastructure (CLI)
- Non-commercial, distribution du source code
  - Modifiable (et documenté pour modification)
  - Facile à re-distribuer et partager des ajouts
- Implémentation sur plusieurs plates-formes
  - Code qui s'exécute sur FreeBSD, Windows™ XP et MacOS X
  - Architecture du code permettant d'être aisément déployé sur d'autres plates-formes

© 2007, M. Riveill

Programmer avec .Net

- Travaux dirigés permettant de démarrer avec la plate-forme
- Peut-être fait de manière autonome
  - http://rangiroa.polytech.unice.fr/riveill/enseignement/tp/carteVisite.html

1. Mise en place d'un environnement 'minimal' de développement
   - .Net SDK et .Net framework (C#, CLR)
   - WebMatrix (un IDE simple d'utilisation)
     - serveur Web inclue dans WebMatrix – nécessaire pour pages dynamiques et web services - ASP.Net/ASM.Net)
     - Pour les vielles versions de Windows, penser à installer MDAC (Microsoft Data Access Component v1.6 ou ultérieur) – nécessaire pour accéder aux données
2. Etude d'une application .Net pour la gestion de carte de visite

© 2007, M. Riveill