Programación con **Visual Basic .NET**

1 – Plataforma .NET

Francisco Ruiz
Manuel Ángel Serrano

Escuela Superior de Informática
Universidad de Castilla-La Mancha

Programación con **Visual Basic .NET**

Contenidos sesión 1

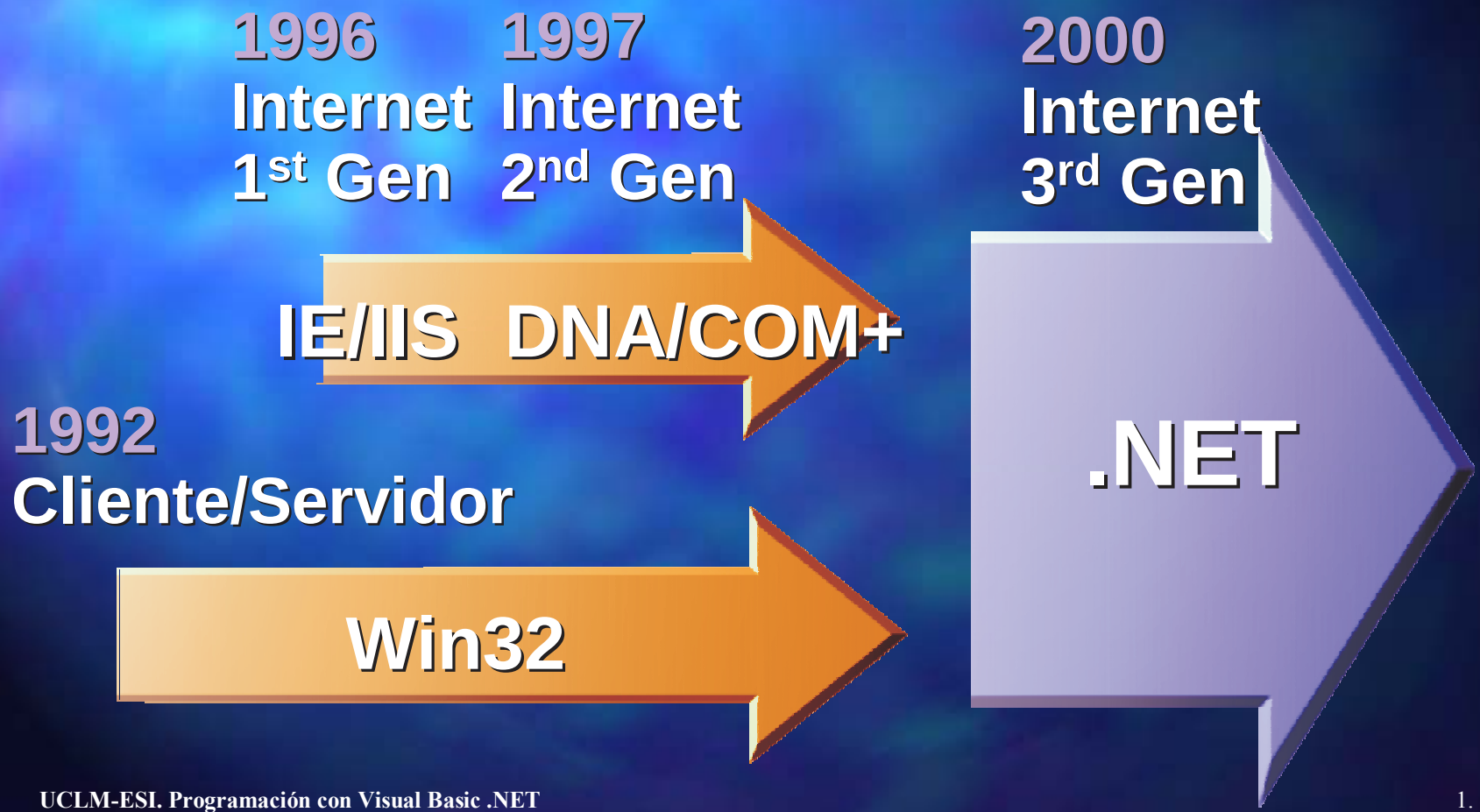
- Plataforma .NET
  - Objetivos
  - Estrategias de desarrollo e implantación
  - .NET Framework
  - Entorno de Ejecución común
  - Biblioteca de clases común
  - Tipos de aplicaciones
- Introducción al Visual Studio .NET (VS.NET)
  - Demo de uso
- Primer programa: "Hola Mundo"
  - Creación y ejecución
  - Código generado
  - Archivos en disco
- VS.NET Aspectos avanzados
  - Demo
- Segundo programa: "Preguntar usuario"
  - Escribiendo código

Plataforma .NET

- ¿Qué es .NET?
  - Una arquitectura tecnológica para la creación y distribución de software como servicio.
    - Servicio en cualquier plataforma, cliente en cualquier dispositivo, programación en cualquier lenguaje, integración basada en estándares.
- ¿Qué incluye?
  - .NET Framework, infraestructura para la creación y ejecución de las aplicaciones.
  - Visual Studio .NET, entorno de desarrollo integrado (IDE).
  - Otros servicios.

Objetivos fundamentales (i)

- Nueva manera de desarrollar software



Objetivos fundamentales (ii)

- Soporte multi-lenguaje
  - La plataforma .NET es independiente del lenguaje
    - Todos los lenguajes .NET son considerados por igual
    - Un Runtime único

      => mejor aprovechamiento de los conocimientos existentes
  - Es posible integrar otros lenguajes y crear otros compiladores
    - Common Language Specification (CLS)
  - Lenguajes disponibles:
    - Por Microsoft: Visual Basic, C++, C#, J#
    - Por terceros: APL, COBOL, Pascal, Eiffel, Haskell, ML, Oberon, Perl, Python, Scheme, Smalltalk, Fortran, ...

Objetivos fundamentales (iii)

- Soporte multi-lenguaje

VB.NET

```
Dim s as String
s = "authors"
Dim cmd As New SqlCommand("select * from " & s, sqlconn)
cmd.ExecuteReader()
```

C#

```
string s = "authors";
SqlCommand cmd = new SqlCommand("select * from "+s, sqlconn);
cmd.ExecuteReader();
```

C++

```
String *s = S"authors";
SqlCommand cmd = new
SqlCommand(String::Concat(S"select * from ", s),
                          sqlconn);
cmd.ExecuteReader();
```

J#

```
String s = "authors";
SqlCommand cmd = new SqlCommand("select * from "+s, sqlconn);
cmd.ExecuteReader();
```

Smalltalk

```
|s| := 'authors'.
|cmd| := SqlCommand('select * from '+s, sqlconn).
cmd.ExecuteReader().
```

Python

```
s = "authors"
cmd =SqlCommand("select * from " + s, sqlconn)
cmd.ExecuteReader()
```

Objetivos fundamentales (iv)

- Soporte multi-plataforma
  - Proyecto Rotor
    - Common Language Infraestructure (CLI)
    - Código fuente abierto, no comercial, modificable
    - Compilar/ejecutar en FreeBSD, Windows, Mac OS X

      http://msdn.microsoft.com/net/sscli
  - Proyecto Mono
    - Implementación open source del .NET Framework
    - Compilador de C#, VB.NET, CLR, librería de clases, etc.
    - Linux, S390, SPARC, HPPA
    - Implementa ADO.NET, ASP.NET, etc.

      www.go-mono.com

Objetivos fundamentales (v)

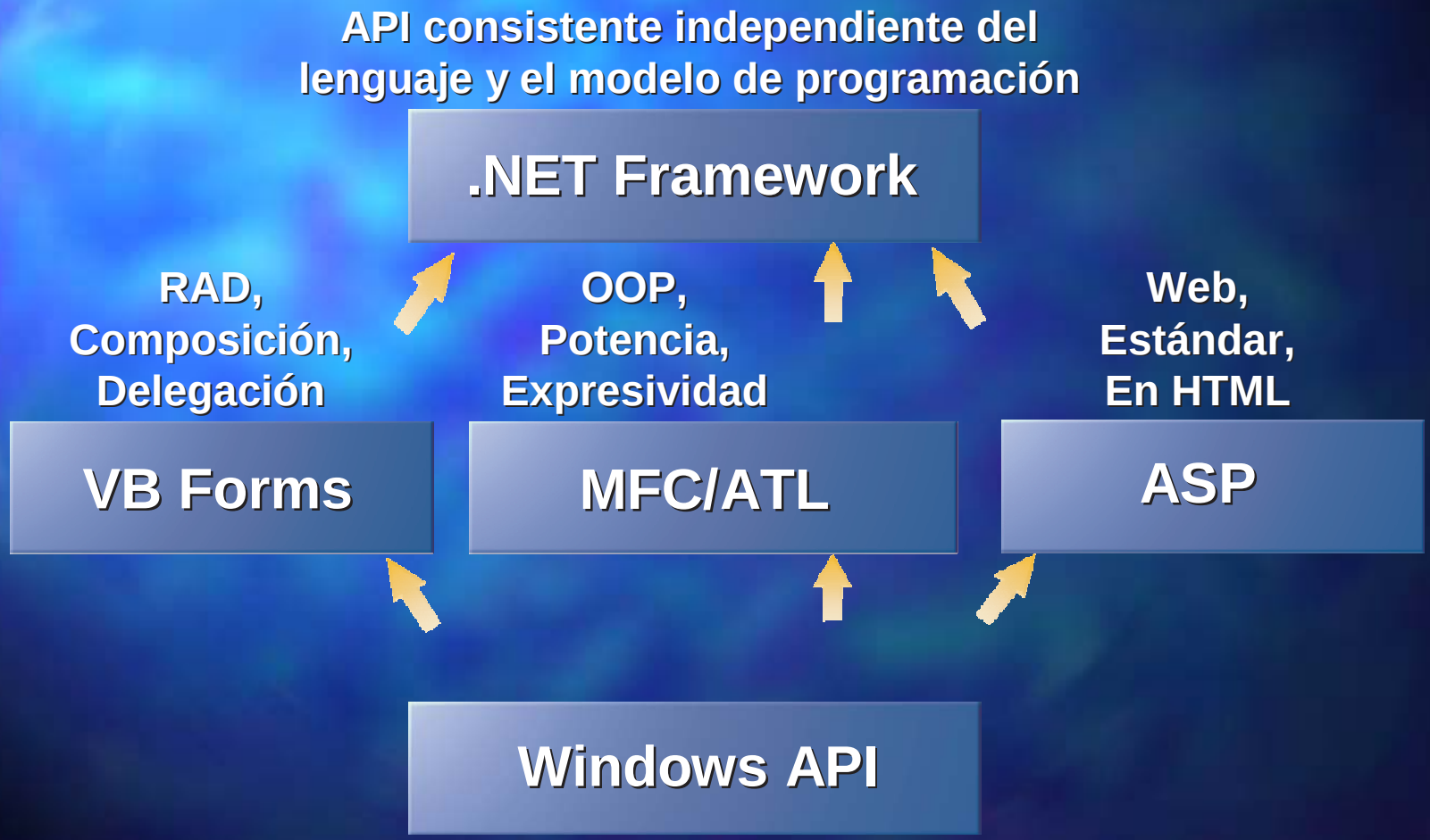
- Basado en estándares
  - XML (XML Schemas, Xpath, XSLT)
  - Servicios Web
    - SOAP (Simple Object Access Protocol)
    - UDDI (Universal Description, Discovery & Integration)
    - WSDL (Web Service Description Language)
  - Lenguaje intermedio común CIL
  - Lenguaje de programación C#

Estrategias de Desarrollo (i)

- Facilitar la integración del software permitiendo
  - Diversas plataformas (Windows, UNIX, Mainframe)
  - Diferentes middleware de componentes (DDE, COM, CORBA)
  - Diferentes sistemas operativos
- Homogeneizar lenguajes mediante
  - Sistema de Tipos Común
  - Biblioteca de Clases base comunes
- Redefinir soluciones preexistentes como servicios
- Simplificar la forma de programar mediante
  - Orientación a objetos plena: clases, interfaces, constructores, atributos, métodos, herencia inter-lenguajes, …
  - Herramientas y entorno comunes para todos los desarrollos.
  - Interoperabilidad pre-construida (con COM, DLL's, etc.)
  - Un Framework que libera al programador de muchas tareas: memoria, seguridad, etc.

Estrategias de Desarrollo (ii)

- Paradigma de desarrollo unificado



Estrategias de Implantación (i)

- Instalación de impacto cero
  - Aplicaciones y componentes compartidos o privados
- Ejecución "side-by-side"
  - Coexistencia de varias versiones del mismo componente
- Ensamblados (*Assemblies*)
  - Colección de funcionalidad creada, versionada e implantada como una unidad de implementación única (incluyendo uno o varios archivos)
  - Son autodescriptivos
  - En tiempo de ejecución se resuelven las referencias entre componentes, se garantiza la política de enlace de versiones y se valida la integridad de los ensamblados cargados.
  - El Manifiesto (*Manifest*) contiene los metadatos del ensamblado incluyendo
    - la identidad del ensamblado
    - los archivos que implementan el ensamblado
    - las dependencias en tiempo de compilación con otros ensamblados
    - los tipos y recursos que forman el ensamblado
    - el conjunto de permisos para ejecutarse apropiadamente

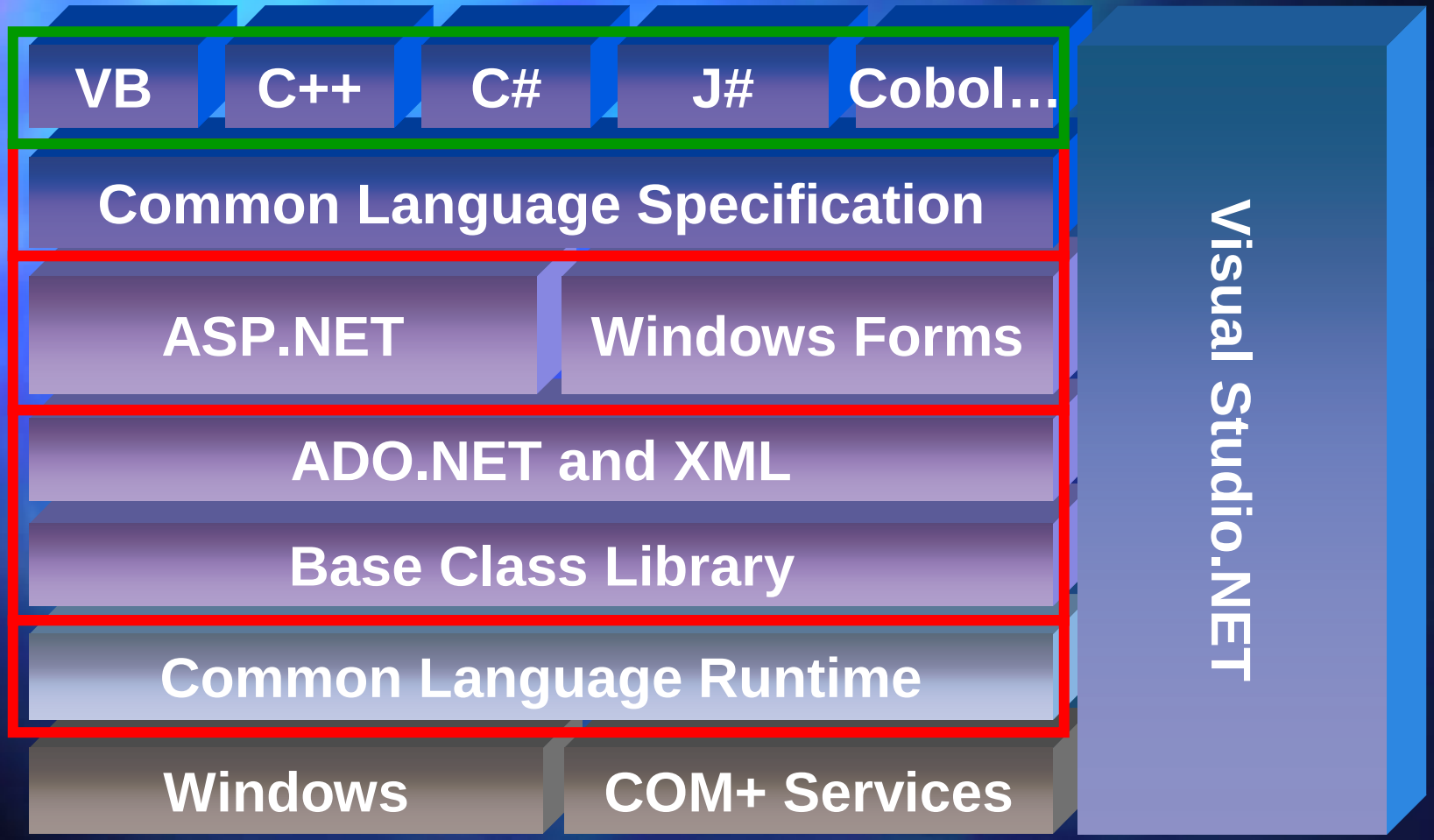
Estrategias de Implantación (ii)

- Ventajas de los Ensamblados:
  - No son necesarios más archivos para usar el componente
  - No más archivos de cabeceras, IDL, librerías de tipos, ...
  - Se reduce la dependencia del Registro
  - No más "infierno de las DLLs": se evitan las faltas de sincronismo entre una librería compartida respecto de la aplicación que la invoca
  - Sistema de nombres robusto
  - Auto-reparación de aplicaciones

.NET Framework (i)

- De cara al programador, es la pieza base de .NET, ya que proporciona las herramientas y servicios para desarrollar el software:
  - Entorno de Ejecución Común (CLR)
  - Biblioteca de clases básicas
  - Motor de generación de interfaces de usuario
    - Web (ASP .NET)
    - Tradicionales Windows (Windows Forms)
  - Especificación de Lenguaje Común (CLS)

.NET Framework (ii)



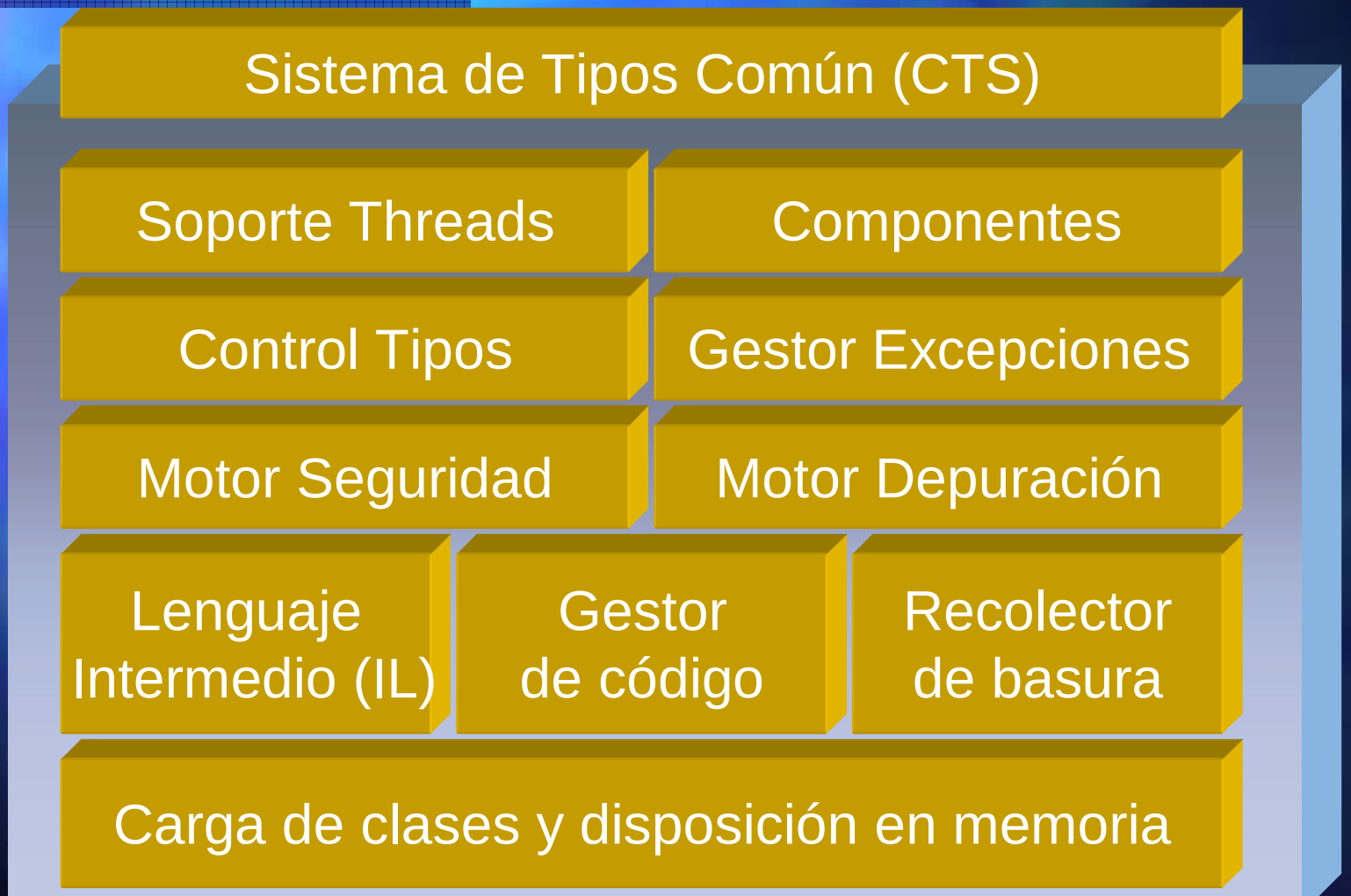
Entorno de Ejecución Común

Common Language Runtime (CLR) (i)

- Es un motor encargado de la gestión del código de las aplicaciones en cuanto a su
  - Carga
  - Ejecución
  - Manipulación de memoria
  - Seguridad
  - Etc.
- Servicios que proporciona:
  - En tiempo de ejecución
    - Gestión de memoria (incluida recolección de basura), gestión de procesos, *threads* (hilos), garantizar seguridad, satisfacer dependencias sobre otros componentes.
  - En tiempo de desarrollo
    - Gestión del tiempo de vida, nominación de tipos robusta, tratamiento de excepciones multilenguaje, gestión de eventos basada en delegados, enlace dinámico, ..

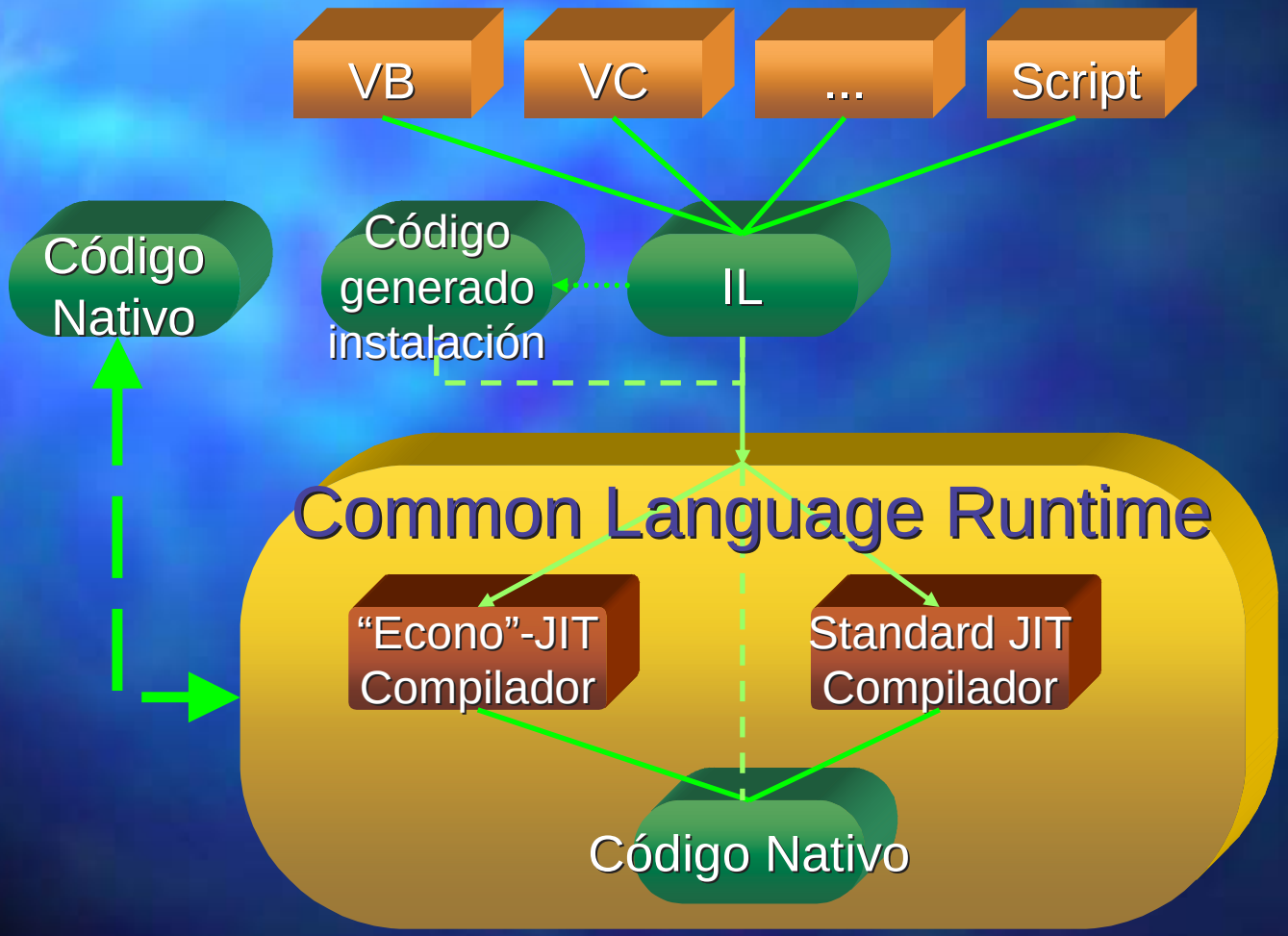
Entorno de Ejecución Común

Common Language Runtime (CLR) (ii)



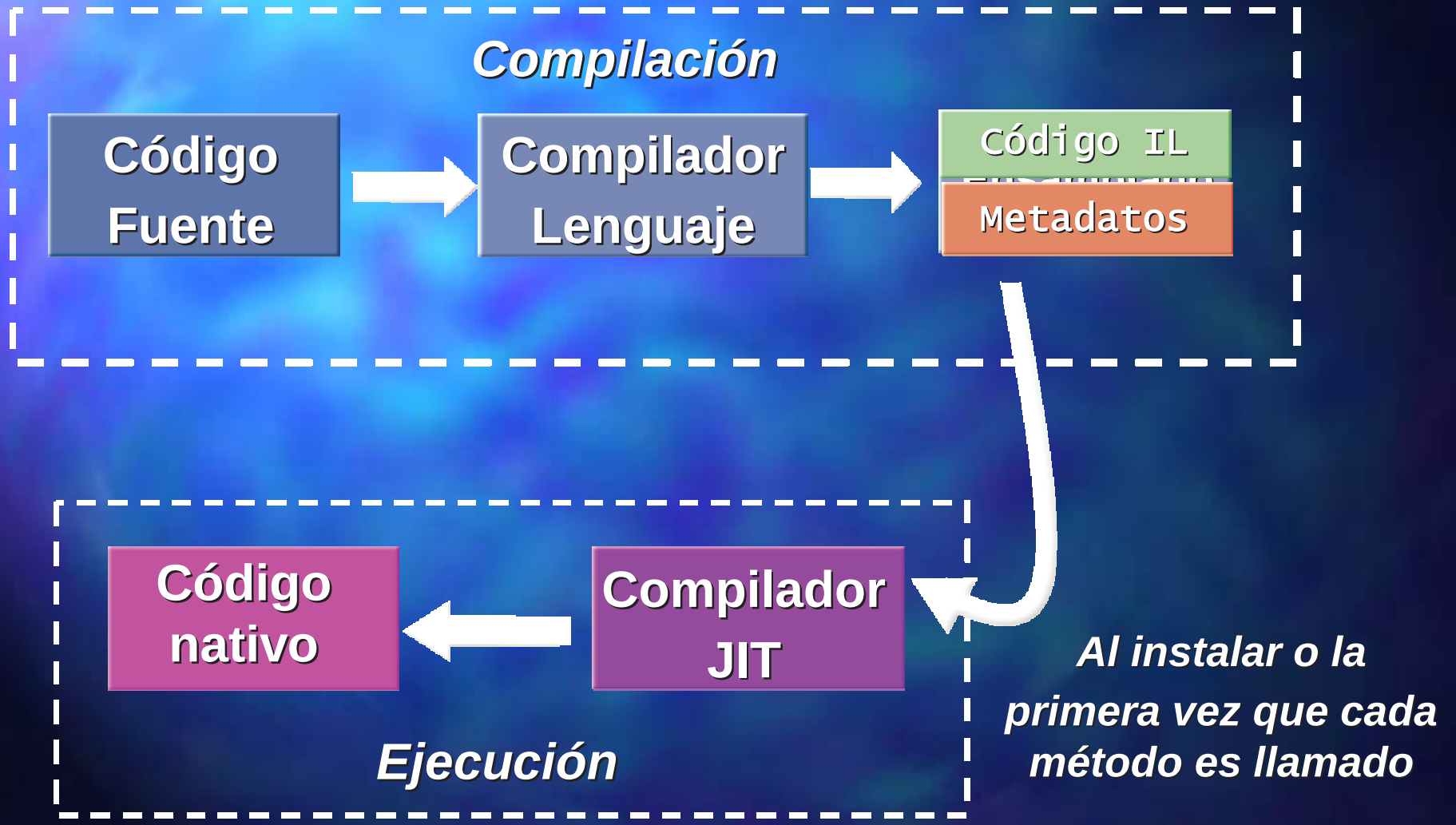
CLR

Lenguaje intermedio IL



CLR

Compilación y ejecución



CLR

Sistema de Tipos Común (CTS) (i)

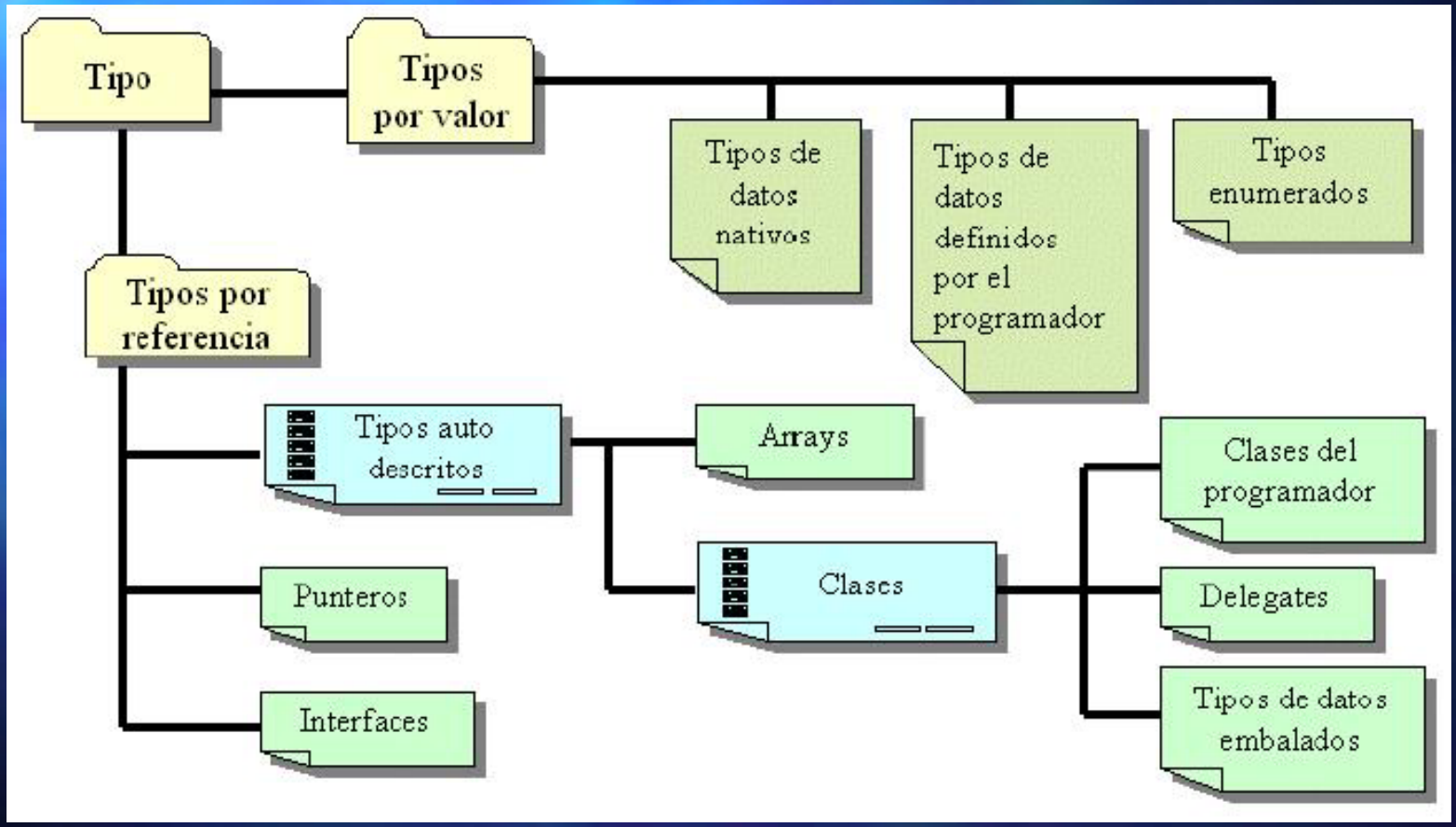
- *Common Type System*
  - Conjunto estándar de tipos y reglas para crear nuevos tipos
  - Integración multi-lenguaje:
    - Heredar implementaciones de clases escritas en otros lenguajes
    - Invocar excepciones de código entre lenguajes
    - Depurar transparentemente
  - ¡No más versiones de librerías para cada lenguaje o compilador!
  - ¡No más librerías de clases limitadas a un lenguaje concreto!
  - Todas las implementaciones de clases, interfaces, estructuras, etc., nativas o creadas por el programador, son tipos .NET.
  - Todos los tipos .NET son objetos.

```
Dim sNombre As String
sNombre = "coche"
MessageBox.Show(sNombre.Length) ' devuelve 5
```

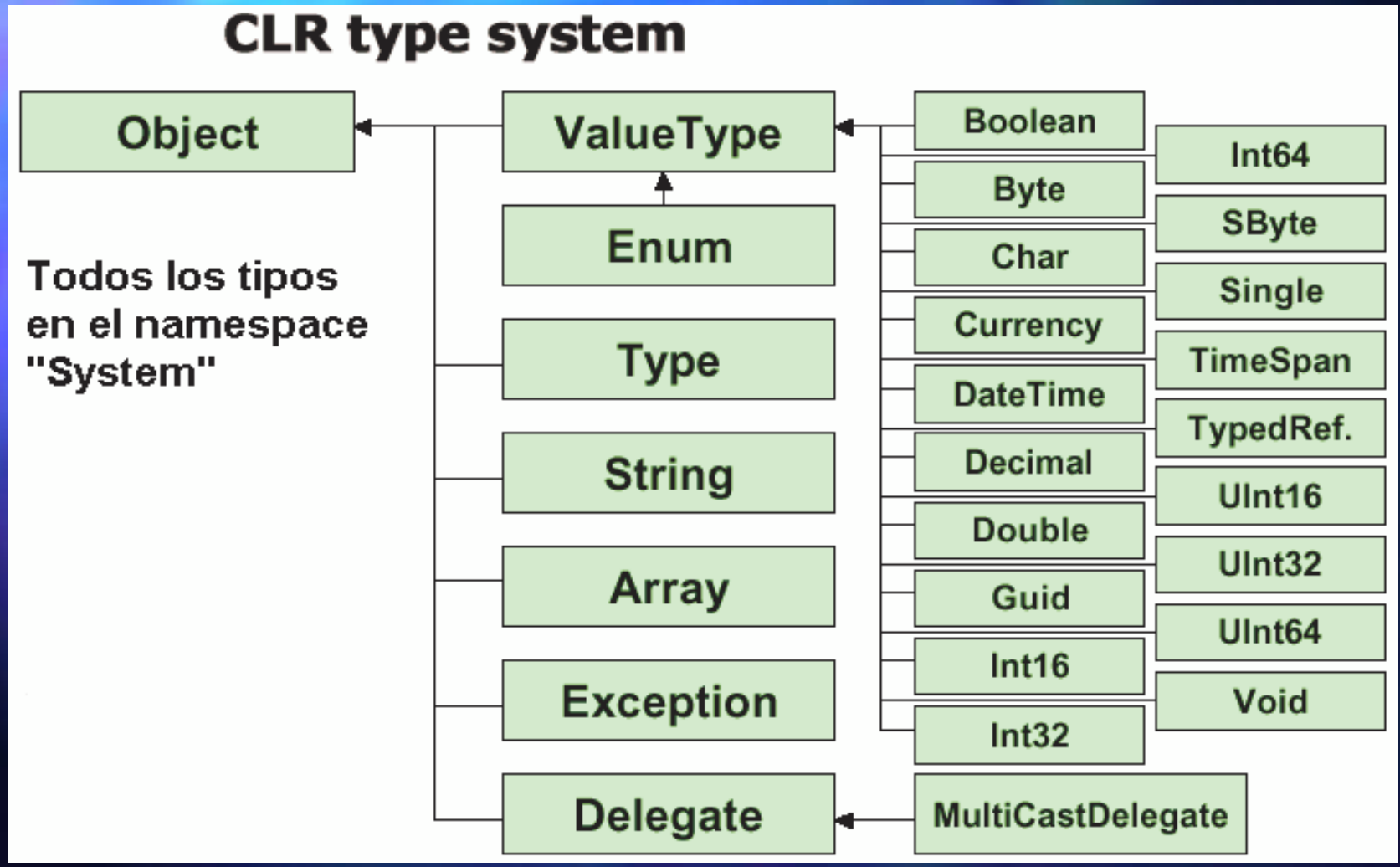
CLR

Sistema de Tipos Común (CTS) (ii)

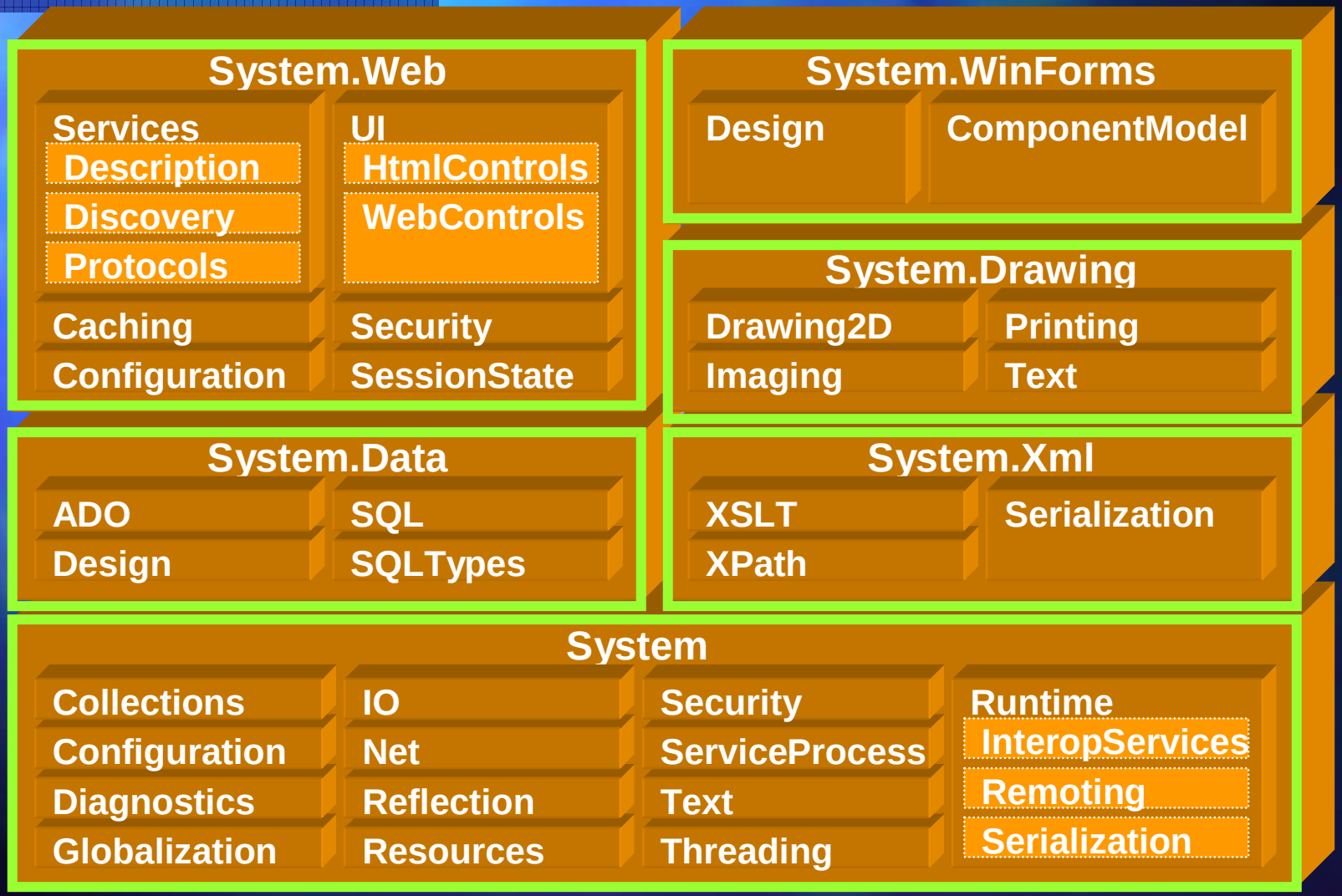
- Categorías de tipos



Sistema de Tipos Común (CTS) (iii)



Biblioteca de clases común (i)

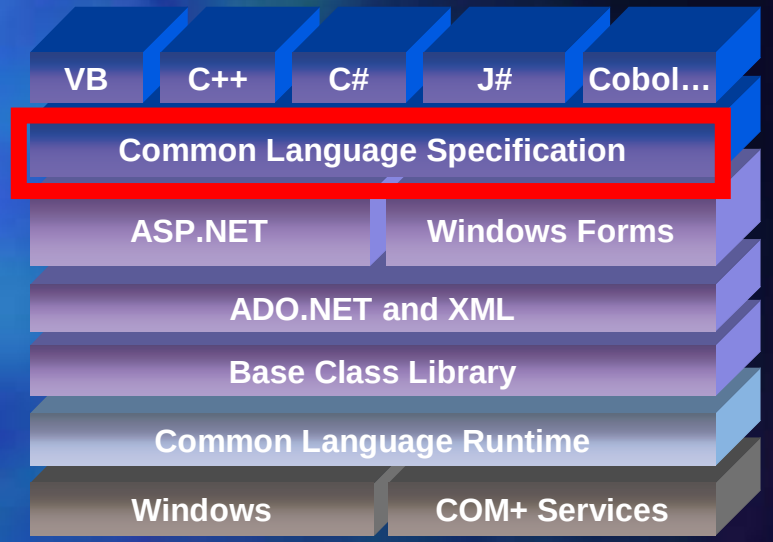


Biblioteca de clases común (ii)

- Lo que vé el desarrollador
  - Un API formada por un conjunto de bibliotecas de clases comunes, orientadas a objetos, jerárquicas y extensibles que
    - proveen soporte completo para la programación de diversos tipos de aplicaciones
    - permiten herencia entre lenguajes, gestión de errores y depuración
  - Las clases están organizadas de forma lógica y jerárquica en "namespaces"
    - System, XML, Data (ADO .NET), Drawing, WinForms, Web (ASP.NET)
    - Facilita reutilización
    - Evita colisiones
  - No es necesario aprender múltiples modelos de objetos:
    - Visual C: MFC, Visual J++: WFC, Visual Basic: Fw

Especificación de Lenguaje Común

Common Language Specification (CLS)



- Conjunto de características comunes que deben cumplir todos los lenguajes.
- Su finalidad es
  - Independencia del lenguaje
  - Integración entre lenguajes
  - Apertura a nuevos lenguajes

Tipos de Aplicaciones

- Aplicaciones normales:
  - De consola
  - Windows Service (no interactiva)
  - Windows Form (interactiva)
  - Windows Control Library (biblioteca de controles)
- Aplicaciones para la web:
  - Web Application (aplicación web cliente-servidor)
  - XML Web Service (servicio web)

Tipos de Aplicaciones

De Consola

- Ejecutables de línea de comandos
- Escritos con cualquier lenguaje .NET
- Ejecución desatendida
- Muy ligeros
  - Hola Mundo ~ 5k

Tipos de Aplicaciones

Windows Service

- Aplicaciones windows no interactivas.
- Escritas con cualquier lenguaje .NET
- Incluyen el código para el comienzo, el fin y mientras se ejecuta.
- Tipos de comienzo:
  - Manual
  - Automático
  - Deshabilitado

Tipos de Aplicaciones
Windows Form

- Aplicaciones Windows interactivas
- Basadas en formularios
  - p.e., interfaz a bases de datos
- SDI/MDI (multidocumento)
- Estilo Explorer
  - Con conectores y divisores entre “frames”
- Instalación
  - XCopy
  - No-tocar (!todavía más fácil que con XCopy!)
    - Ejecutar desde una URL
- Impresión
  - Configuración de páginas
  - Previsualización

Tipos de Aplicaciones

Windows Control Library

- Usables en cualquier formulario con cualquier lenguaje.
- Derivados desde cualquier otro control
  - Extender controles .NET pre-construidos.
  - Construir nuevos controles.
- Ejemplo:
  - Acceso a datos particularizado y ampliado.

Tipos de Aplicaciones

Web Application

- Mucho menos código
  - Con Web Forms que con ASP clásico
- Estilo de programación a lo VB
- Controles para validar entradas
- Controles complejos mejorados
  - DataGrid
  - DataList
  - Calendar
- Sesiones seguras
  - Cookies no requeridos
- Traza y depuración mejoradas
- Caching más potente
- Más eventos detectables

Tipos de Aplicaciones

XML Web Service

- Evolución de aplicaciones y sitios web
  - Interacciones seguras
- Nueva metodología de integración
  - Entre cualquier sistema operativo y/o lenguaje
  - Olvidando procolos propietarios (Java RMI, CORBA IIOP, DCOM)
- Componentes de aplicación programables
  - Accesibles via protocolos Internet estándares
  - HTTP, XML, SOAP, WSDL, UDDI
- Soporte provisto por el Framework
  - Generación automática de XML, SOAP, WSDL
  - Conversión automática de un componente en Web Service (WebMethod)

VS.NET

Introducción al Visual Studio .NET

- Entorno Integrado de Desarrollo (IDE) para .NET
- Multi-lenguaje integrado
  - Lo mismo se hace siempre igual, independientemente del lenguaje.
- Permite elegir los tipos de aplicación
- Abierto
  - se pueden añadir nuevos lenguajes o nuevas herramientas.
- Integración con arquitectura COM

Introducción al VS.NET

Demo de uso (i)

- Página de inicio
  - Editar "Mi perfil"
  - Ver "recursos en línea"
  - Proyectos (nuevo / abrir)
- Crear nuevo proyecto
  - Aplicación windows, biblioteca de clases, servicio web, ...
- Elementos principales del entorno
  - Menú
  - Barra de herramientas
  - Ventana principal de trabajo
    - Fichas
  - Ventanas desplegables (pestañas)
  - Ventanas combinadas

Introducción al VS.NET

Demo de uso (ii)

- Agregar nuevos elementos a un proyecto
  - Windows Form, Clase, Módulo, Clase de componentes, Control de usuario, Dataset, Archivo XML, ... hasta 33 diferentes
- Configuración del entorno
- Estados de las ventanas
  - Acoplable / Ocultar / Flotante / Ocultar automáticamente
- Explorador de soluciones
  - Solución: colección de proyectos abiertos en una misma sesión de trabajo.
  - Para cada proyecto muestra sus elementos: Formularios, módulos, clases, recursos, referencias, etc.
    - References => referencias a los espacios de nombres usados
    - AssemblyInfo.vb => información del Ensamblado

Introducción al VS.NET

Demo de uso (iii)

- Propiedades de un proyecto
- Propiedades de la solución
- Agregar un control a un formulario
- Menú contextual
- Ventana de propiedades
  - Formulario / control

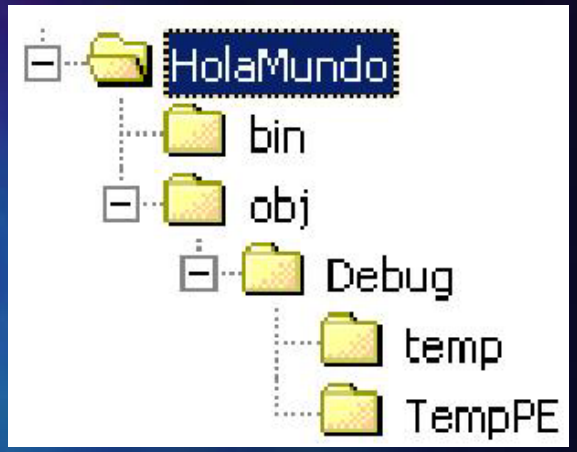
Primer programa "Hola Mundo"
Creación y ejecución

1. Crear un nuevo proyecto de Visual Basic, tipo "Aplicación para Windows"
2. Añadir un formulario *(clase de objeto visual)*
3. Editar sus propiedades
   text <- Programa de prueba 1"
4. Añadir un control de tipo "label" *(clase de objeto visual incrustado dentro de un formulario)*
5. Editar sus propiedades
   text <- "Hola Mundo"
   font <- MS Sans Serif, 20 pt
6. Ejecutar la aplicación

Primer programa "Hola Mundo"
Código generado

1. Ver código
   - Expandir/contraer regiones
2. Clase del formulario
   - Un formulario es un objeto => una clase
3. El método constructor New()
   - Se hereda de la clase base (padre)
4. El método InitializeComponent()
   - Crea un nuevo objeto Label
   - Asigna valores a las propiedades del Label
   - Asigna valores a las propiedades del Form
   - Añade el objeto Label al Form
5. El método Dispose()
   - Libera recursos de memoria

Primer programa "Hola Mundo"
Archivos en disco



1. Ver lo que se ha creado en el disco.
   - Directorios de la figura
2. Tipos de Archivos:
   - VB: código fuente en Visual Basic (clases, módulos, etc.)
   - VBPROJ: información sobre los elementos del proyecto
   - SLN: solución
   - VBPROJ.USER: opciones de usuario del proyecto
   - RESX: plantilla de recursos en XML
   - EXE: aplicación ejecutable
     - Guardado en el directorio bin
     - Es lo único necesario para instalar en otro equipo que tiene .NET Framework
   - PDB: información para depuración

VS.NET

Aspectos avanzados – demo (i)

- Cuadro de herramientas
  - Agregar/eliminar controles
  - Contenedor de código fuente (anillo del portapapeles)
- Barras de herramientas
  - Menú contextual
  - Personalizar
- Barras personalizadas
  - Crear nueva
  - Añadir/quitar comandos
- Editor de código
  - Fuentes y colores
  - Otras opciones

VS.NET

Aspectos avanzados – demo (ii)

- Dividir ventana edición
- Menú Edición
  - Buscar/Reemplazar
  - Ajuste de línea
  - Marcadores
  - Mostrar espacios en blanco
  - Regiones y esquematización
  - Comentarios en bloques de líneas
  - Intellisense (ayuda "inteligente" al escribir código)
- Ir a definición de procedimiento
- Mostrar pantalla completa

VS.NET

Aspectos avanzados – demo (iii)

- Ventana vista de clases
- Ventana explorador de objetos
  - Buscar símbolo
- Tareas

  {Agenda para registrar trabajos pendientes}
  - Crear/Eliminar tareas
    - Comentarios / Marcadores
  - Ventana Lista de Tareas
- Macros
  - Explorador
  - Ejecución
  - Grabación (estilo Office)
  - IDE de macros

VS.NET
Aspectos avanzados – demo (iv)

- Sistema de Ayuda
  - *MSDN (Microsoft Development Network Library)*
  - *Son CD's separados de los de Visual Studio .NET*
  - *Formato HTML*
  - Ayuda dinámica
  - Contenido
  - Índice
  - Buscar
  - Ayuda externa al IDE
  - Mantener temas visibles
  - Documentación completa

    .NET Framework SDK / MSDN for Visual Studio 2003

Segundo programa "Preguntar usuario"
Escribiendo código (i)

1. Crear nuevo proyecto EscribirCodigo de tipo "Aplicación para Windows"
2. Añadir módulo MiCodigo *[contenedor de código]*

```
Module MiCodigo
....
End Module
```

3. Añadir comentario

```
'Aplicación EscribirCodigo
```

4. Añadir procedimiento Main dentro de MiCodigo

```
Sub Main() 'punto de entrada a la aplicación
....
End Module
```

5. Mostrar mensaje de inicio usando la clase del sistema MessageBox

```
MessageBox.Show("Empieza el programa")
```

Segundo programa "Preguntar usuario"
Escribiendo código (ii)

6. Configurar propiedades del proyecto
   - Nombre ensamblado: EscribirCodigo
   - Espacio de nombres: EscribirCodigo
   - Objeto inicial: Sub Main
7. Declarar la variable Nombre de tipo string

```
Dim Nombre As String
```

8. Usar función InputBox() para preguntar nombre del usuario y almacenarlo en la variable Nombre

```
Nombre=InputBox("Nombre del usuario:")
```

9. Utilizar el operador de concatenar textos (&) para mostrar mensaje de control

```
MessageBox.Show("El usuario es " & Nombre, "Programa de
        prueba")
```

10. Ejecutar el programa y grabarlo en disco
11. Probar depuración errores *(poner la declaración Dim como comentario)*

Ir pensando en la Práctica 1

Resolver ecuación 2do grado (i)

- Mostrar un formulario para preguntar los tres coeficientes de una ecuación de segundo grado:

  A*x^2 + B*x + C = 0

  - Preguntar cada coeficiente en un control de tipo TextBox
- Añadir un botón "Calcular" para obtener las soluciones invocando al procedimiento de igual nombre.
  - Indicar si las soluciones son reales o imaginarias con un control CheckBox.
  - Mostrar las 2 soluciones en un control etiqueta "Solución" con texto azul si son reales y rojo si son imaginarias.
  - Ejemplo formato 2 soluciones reales: "2'45 y 78'23"
  - Ejemplo formato 1 solución real: "-9'06"
  - Ejemplo formato 2 soluciones imaginarias: "1'48+2'1i y 0'63-1'86i"
- Añadir un botón "Salir" para acabar.