Panorámica general de .NET

Juan Manuel Cueva Lovelle
cueva@lsi.uniovi.es
www.di.uniovi.es/~cueva

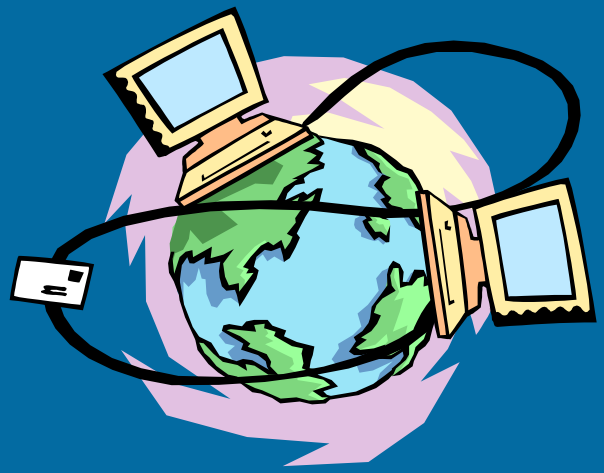
Departamento de Informática
OOTLab www.ootlab.uniovi.es

¿Qué es .NET?

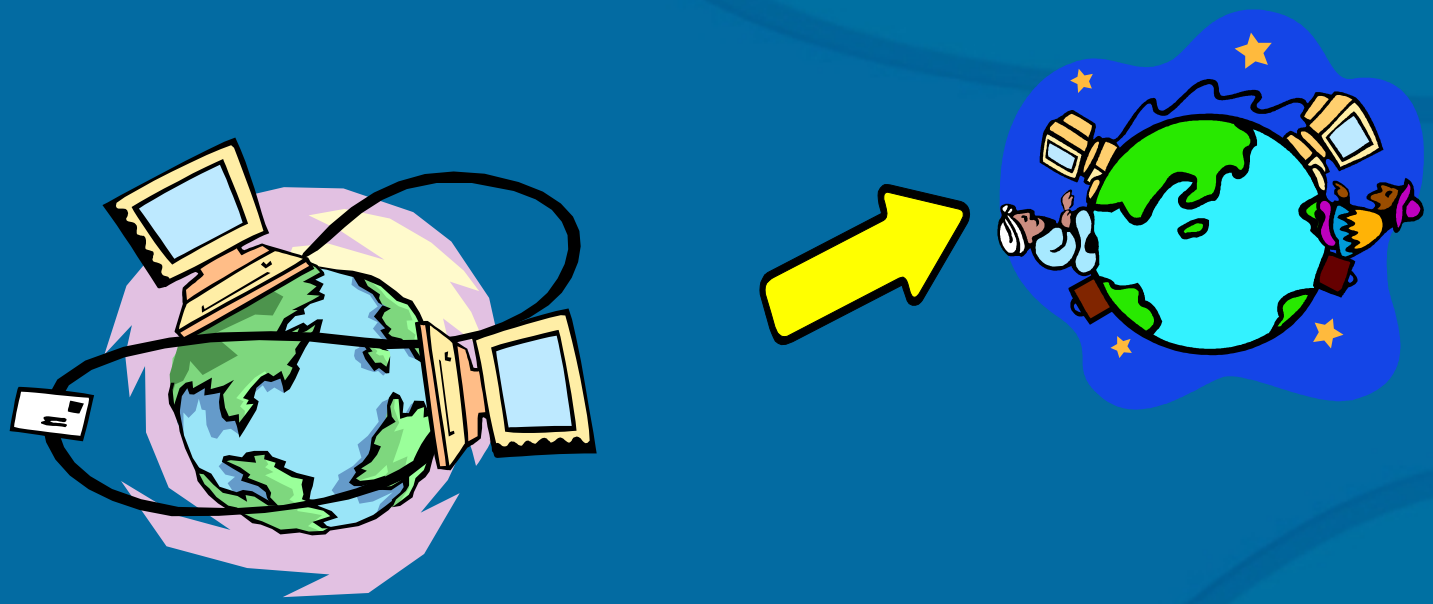
- Una plataforma de desarrollo de software
  - incluye interfaces, componentes y herramientas
  - El mayor cambio en Microsoft desde que Windows NT reemplazó a MS-DOS
  - El cambio incluye
    - Cambio de formato de los ejecutables
    - Cambio de compiladores y de su filosofía de trabajo
    - Cambio de la biblioteca de clases básicas
    - ...

Y a que se debe este cambio ...

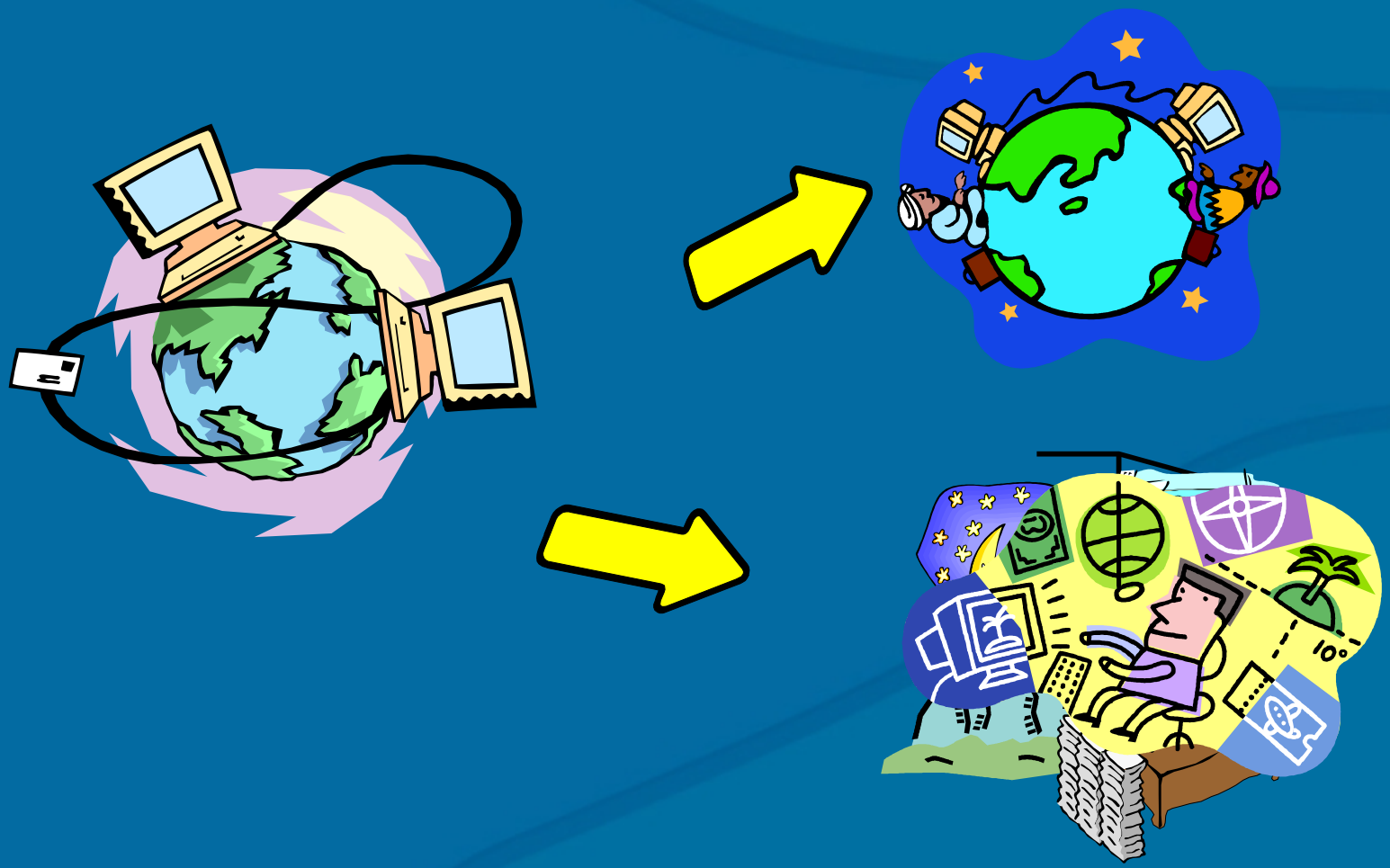
Internet como substrato esencial



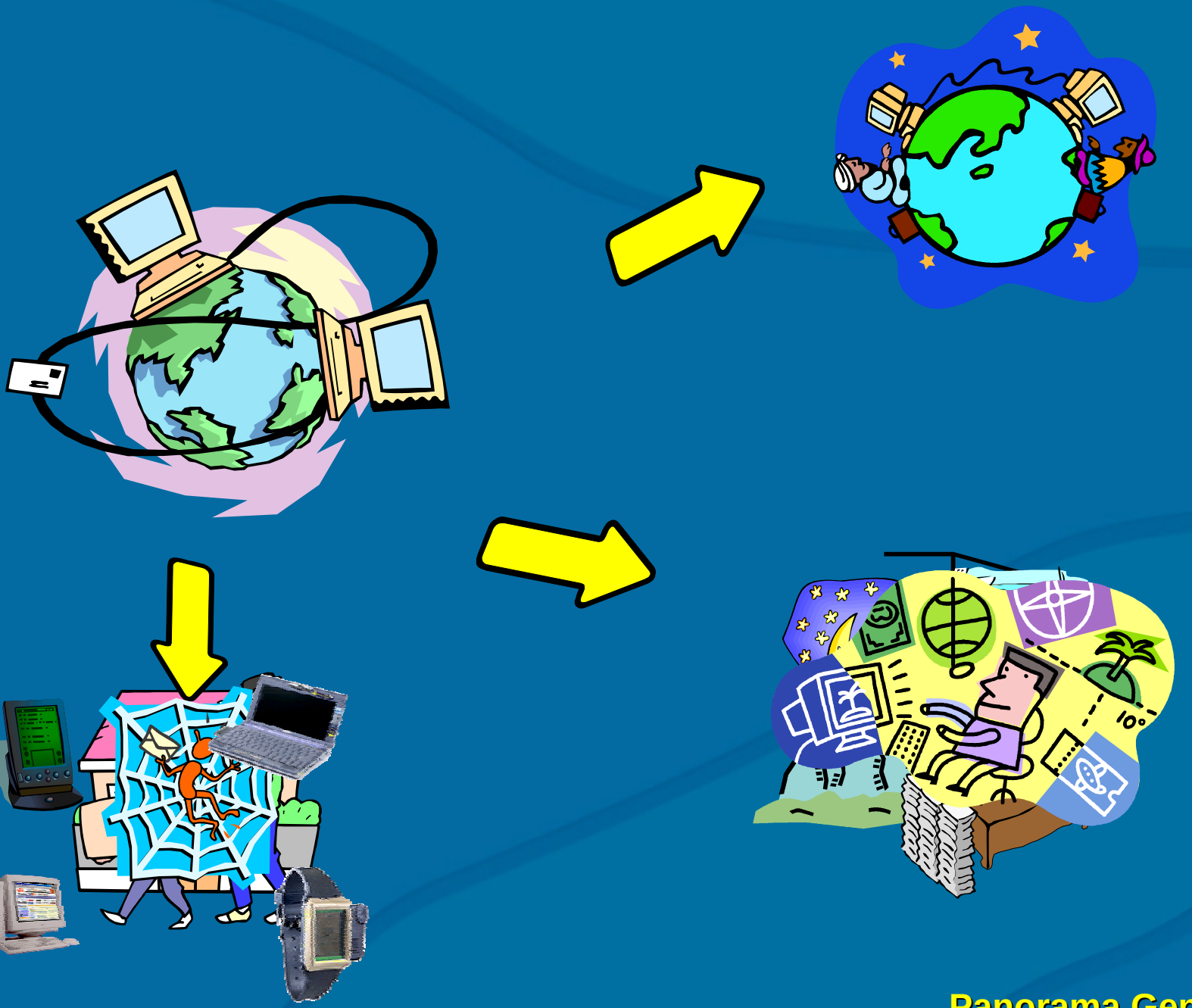
Un mundo de servicios



Acceso a la información desde cualquier sitio

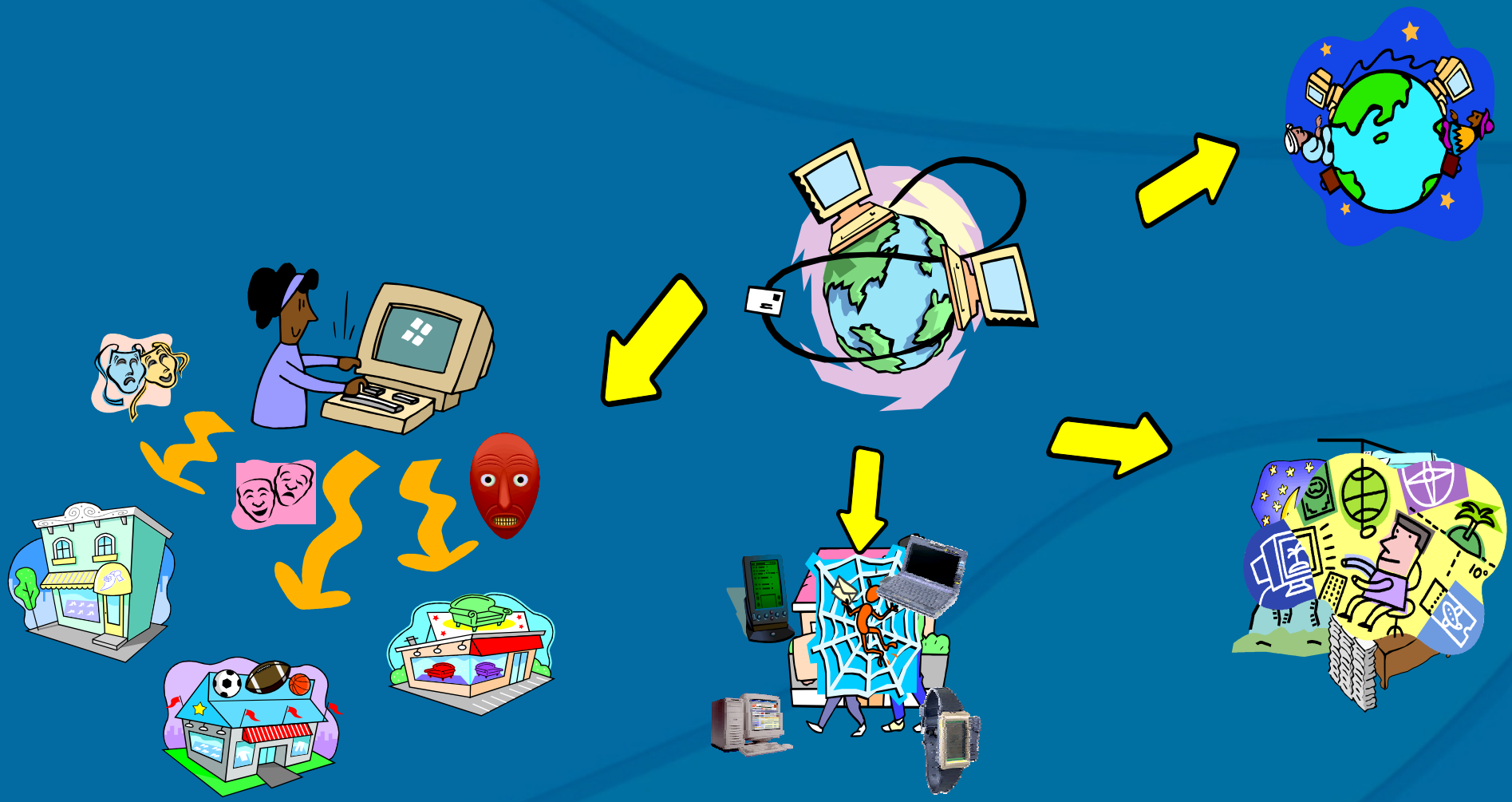


Gran variedad de dispositivos

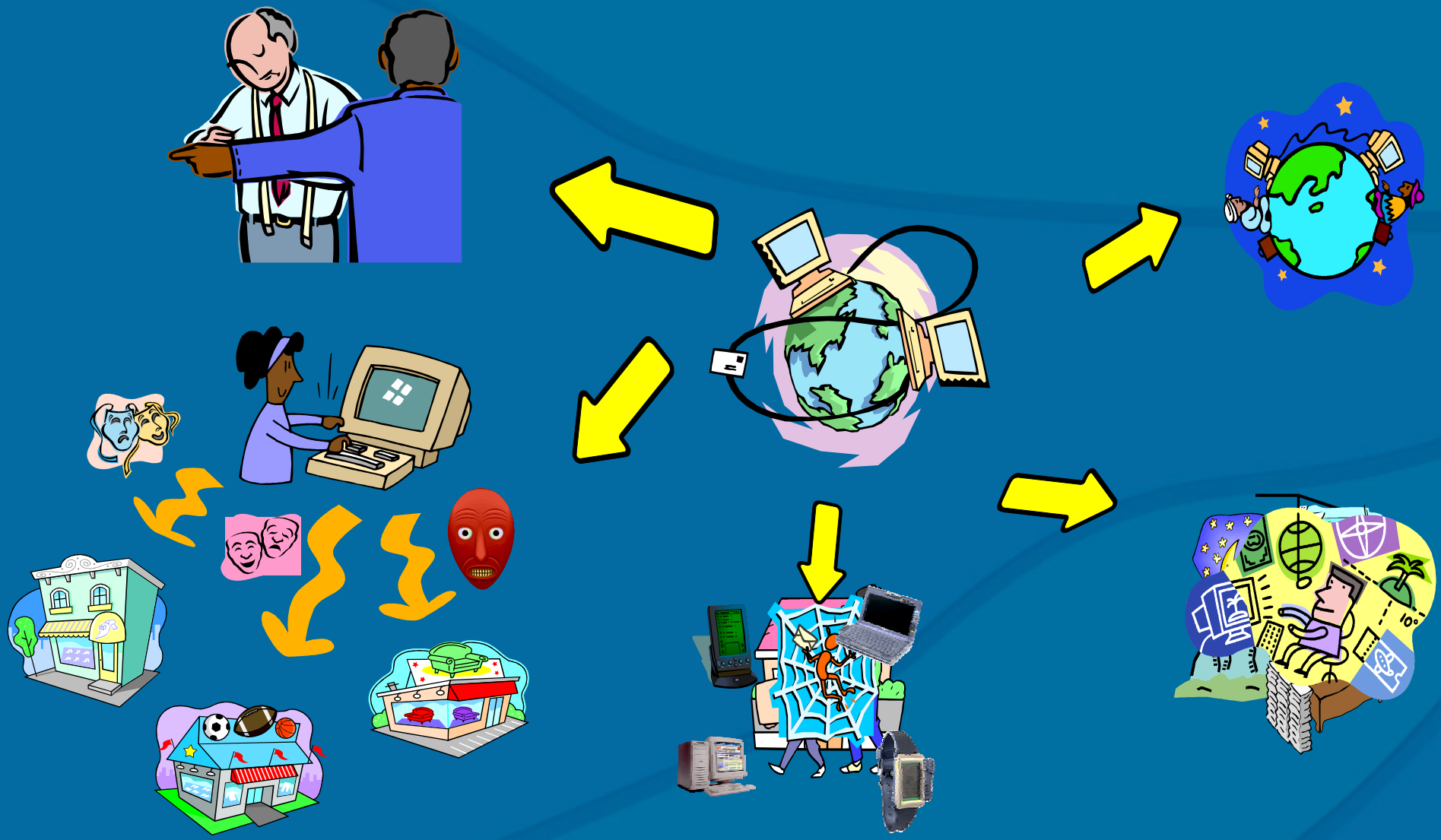


Passport

Identidad única



Servicios personalizados



¿Y cómo lo hace .NET?

- Una visión
  - Ejecutar software en cualquier lenguaje sobre cualquier dispositivo
  - Cómo la Internet puede hacer los negocios más eficientes y proporcionar servicios a los consumidores
    - Concepto y modelo de programación: Servicio Web XML
  - Nuevas formas de interactuar con PCs (uso de la voz, reconocimiento de escritura…)
  - Nuevos dispositivos: teléfonos, PDAs, Tablet PCs
- Una plataforma software
  - Nuevo nivel: .NET Framework
  - Nueva herramienta: Visual Studio.NET
- Un entorno de hospedaje de Servicios Web personales
  - Servicios Web “básicos”: autenticación, almacenar datos
  - Subscripción a software como Servicio: .NET myServices

Situación actual ...

- Lenguajes de programación y compiladores
  - Archivos fuentes vs. binarios
- Ejecutables y enlace dínámico en DLLs
  - Tiempo de ejecución, símbolos
- Modelos de componentes y encapsulamiento
  - Tipos, interfaces, clases, objetos
- Aplicaciones distribuidas
  - Arquitecturas cliente/servidor, en 3 niveles
- Internet
  - Páginas activas
  - Middleware, seguridad, transacciones, atributos
  - Máquinas virtuales, interpretación y librerías de abstracción

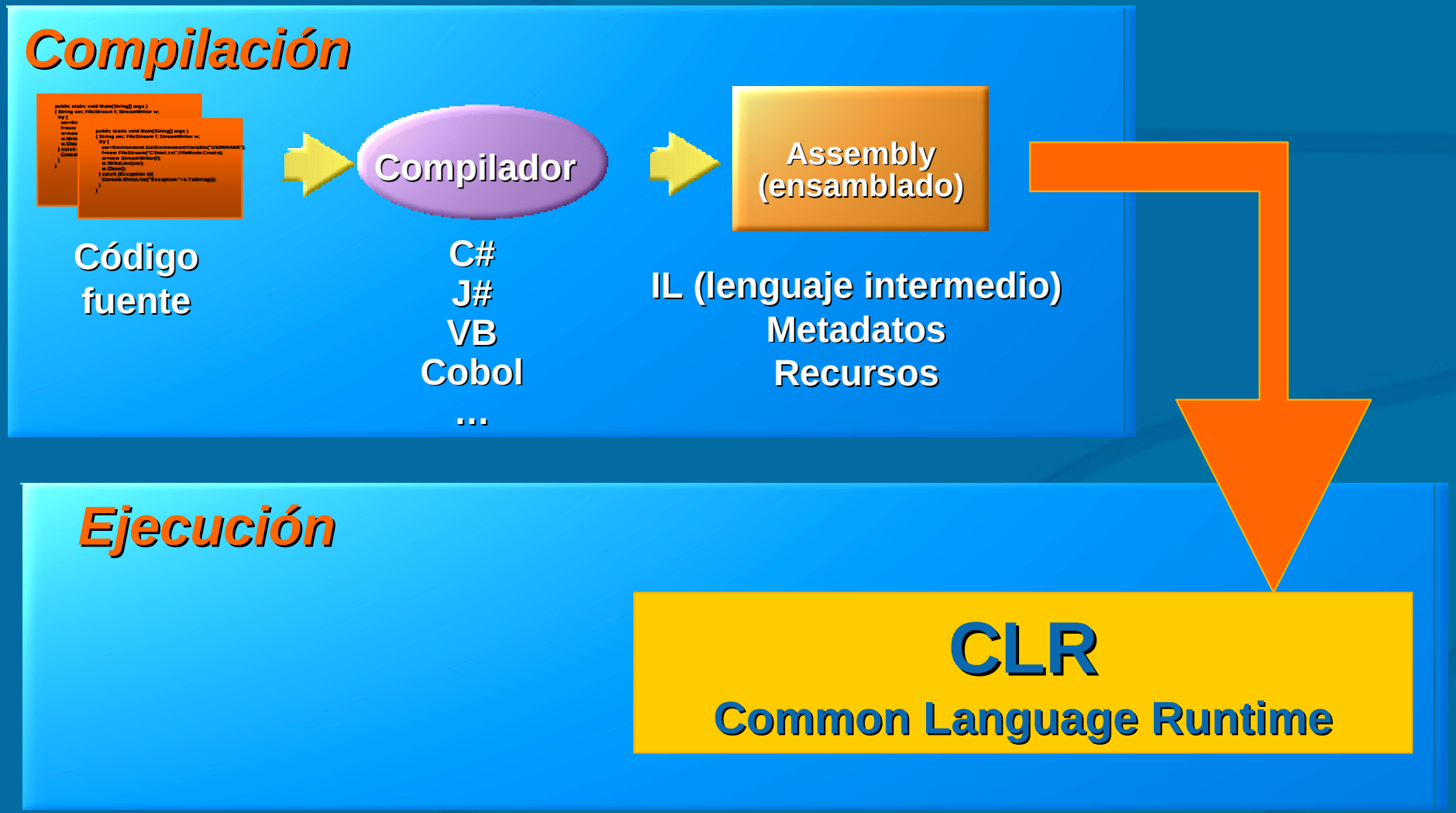
Algunos problemillas…

- IDLs y Librerías de Tipos son complejos
  - Se separa el interfaz de la implementación (ejecutable)
  - El compilador tira metadatos útiles
- Cada entorno de desarrollo debe implementar costosos mecanismos de infraestructura
  - Factorías de clases, de interfaces
- Control explícito del flujo de ejecución
  - Concurrencia: STA, MTA, FTA, etc…
- Windows
  - Visual C++, punteros y API Win32
  - Incompatibilidades de tipos (Visual Basic String vs. VC char array) y la reutilización de implementación: herencia
  - Sin certeza al 100% sobre cómo se usan las DLLs
  - Versiones: Infierno de las DLLs
  - Complejidades de Instalación: Inicio, Registry, archivos…
  - Construir seguridad implícitamente sobre el sistema

Posibles arreglos…

- Compartir características y niveles de abstracción: un framework
- Múltiples lenguajes compilados
- Servicios en el desarrollo y ejecución de código
  - Nueva máquina virtual multilenguaje (CLR)
- Lenguaje Intermedio (MS-IL)
- Espacios de nombres (librerías de clases y tipos unificados)
- Metadatos y ensamblados (assemblies)
- Simplificar y unificar

Desarrollo de software con .NET



Facilidades al desarrollo ...

- .NET Framework
  - Nivel software para el desarrollo y ejecución de aplicaciones sobre un sistema operativo
    - Common Language Runtime (CLR)
    - Base Class Libraries
    - Distintos lenguajes de Programación
- Modelos de programación ASP.NET
    - Formularios Web
    - Servicios Web XML

.NET Framework gráficamente

Aplicaciones .NET

- 3 entidades primarias
  - Assembly: unidad primaria de implantación de una aplicación .NET
  - Módulos: archivos individuales que forman assembly
  - Tipos: unidad básica de encapsulación de datos con un conjunto de comportamientos
- Problemática
  - Cómo desarrollar, empaquetar e implantar aplicaciones y tipos
    - Cómo crear tipos y empaquetarlos en archivos
  - Cómo crear componentes
  - Como crear y usar componentes compartidos

Assembly

- Unidad primaria de implantación
- Autodescriptivo e independiente de la plataforma
- Compuesto de un manifiesto y uno o más módulos
- Manifiesto
  - Contiene la identidad del assembly
    - Nombre textual y número de versión
    - Si es público, también contiene la clave pública: garantiza unicidad e identifica la fuente
  - Responsable de declarar la seguridad que requiere
  - Lista de de todos los tipos expuestos y recursos (GIFs, JPGs…) en otros assemblies dependientes
    - El CLR lo usará para localizarlos

Módulo

- Un archivo DLL o EXE Windows PE (Portable Executable)
- Contiene código en lenguaje intermedio (IL), metadatos y opcionalmente el manifiesto del assembly
- IL: modo independiente de la plataforma de representar código gestionado
- Metadatos: datos sobre los datos
  - Los metadatos proporcionan información adicional a los tipos declarados en el IL y son usados por el CLR
    - IDLs no son necesarios
- El CLR optimiza IL a código nativo
  - En tiempo de instalación o JIT

Tipo

- Describe la encapsulación de datos y un conjunto de comportamientos
- 2 tipos
  - Tipo referencia: como clases
    - Creados en el Managed Heap
  - Tipo valor: como estructuras
    - Creados en la pila
  - Afecta a la gestión de memoria y al funcionamiento del GC
- Propiedades, métodos y campos
  - Campos: datos miembro dentro de un tipo
  - Los miembros definen conductas particulares del tipo
  - Propiedades: como campos pero pueden tener código que realice algún tipo de validación de datos (sexo: varón, hembra)

Sistema de Tipos Comunes

- Common Type System (CTS)
  - Todos los tipos de datos más frecuentemente utilizados (enteros, reales, texto) son implementados como objetos
  - Derivan de System.Object
  - Elimina la necesidad de que cada lenguaje implemente sus propios tipos de datos de modo incompatible
    - Un Integer en VB6 es ahora un Short en VB.NET
  - Todos los lenguajes utilizan la misma librería de tipos
  - Proporciona mecanismos para extender estos tipos

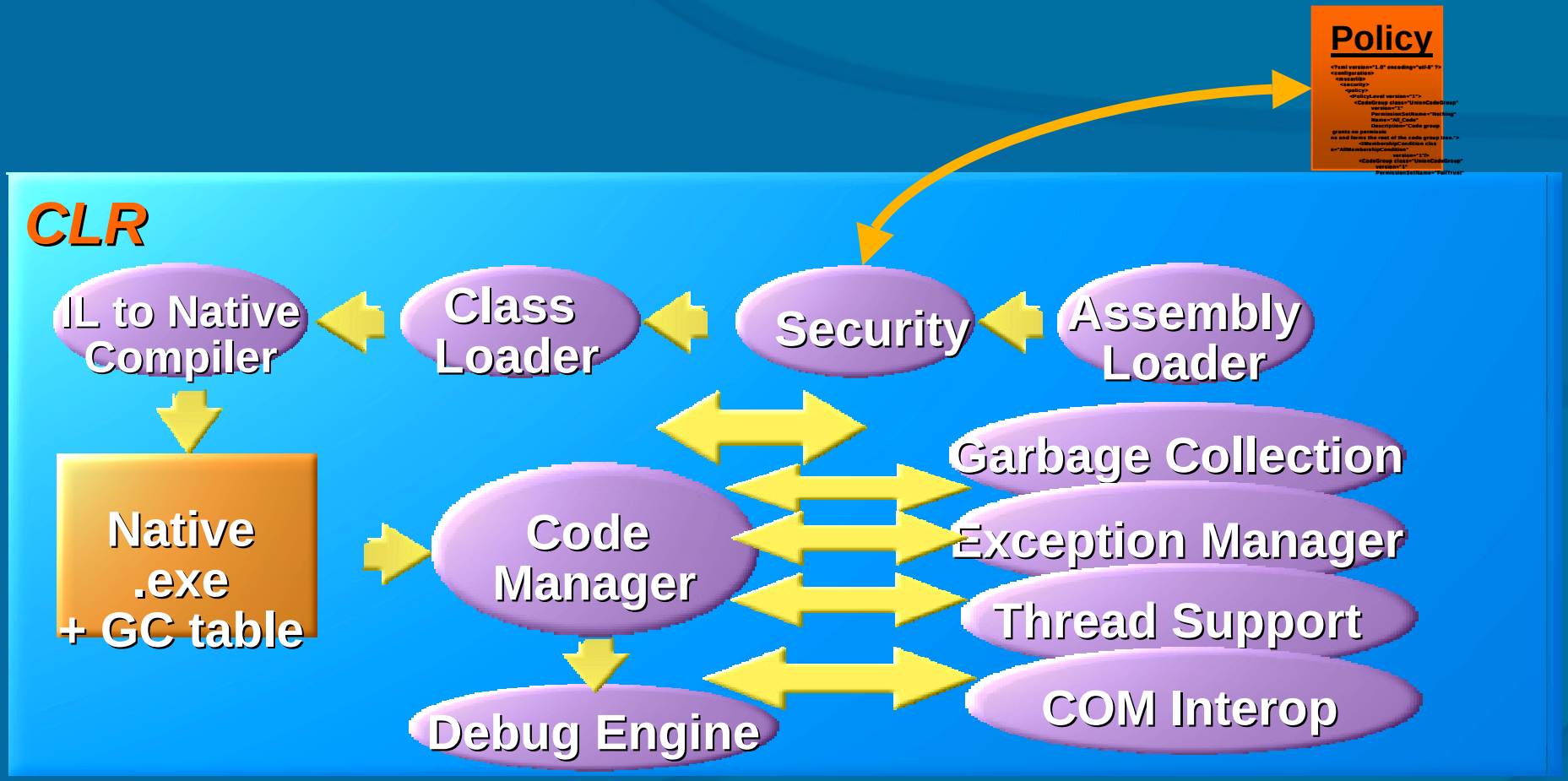
Metadatos

- Información que permite a los componentes ser autodescriptivos
- Describir aspectos: clases, métodos, campos o el propio assembly
- Usados por el CLR
  - Validar un assembly antes de ser ejecutado
  - Realizar GC en la ejecución
  - Encontrar y cargar tipos
- Modelos anteriores
  - Metadatos COM: la Librería de Tipos describe clases expuestas por el componente para facilitar automatización
  - Metadatos COM+: permite declarar atributos: transacciones, serialización, pooling…
  - Problemas: falta de sincronismo, actualización, registro, versiones…
- Accesible mediante reflexión: examinar metadatos asociados con el assembly actualmente ejecutándose

Atributos en .NET

- Una forma de expresar de forma declarativa cualidades de componentes
  - Definidas en el mismo código: atributo []
  - Ej: persistencia, transacciones, seguridad…
- Diferencia con atributos en COM+
- Asocian y decoran decorar assemblies, clases o miembros como métodos y propiedades con información adicional
- Varios propósitos
  - Informativo: obtener comportamiento en tiempo de ejecución
  - Invocar una cierta conducta
- En tiempo de compilación, los atributos se convierten en metadatos y se almacenan con el código IL
  - Como tener una librería de tipos en la misma DLL que el código
- En tiempo de ejecución, se acceden a través de clases estándares .NET y disponibles por parte del CLR

CLR


Policy
CLR
IL to Native Compiler
Class Loader
Security
Assembly Loader
Native .exe + GC table
Code Manager
Garbage Collection
Exception Manager
Thread Support
COM Interop
Debug Engine

Beneficios de la CLR

- Verifica que el código es “type safe”: sólo realizará operaciones apropiadas sobre ciertos tipos de datos
  - No usar un entero como puntero a función
  - Acceso sólo a la memoria autorizada
- Reduce conflictos entre diferentes versiones de componentes
  - Usa el manifiesto y metadatos para cargar la versión correcta
    - Ejemplo: acceso a datos
  - Tanto assemblies públicos (GAC) como privados
  - Algoritmo bien definido para enlazar los diversos assemblies y garantizar compatibilidad
  - Ejecución “side by side”

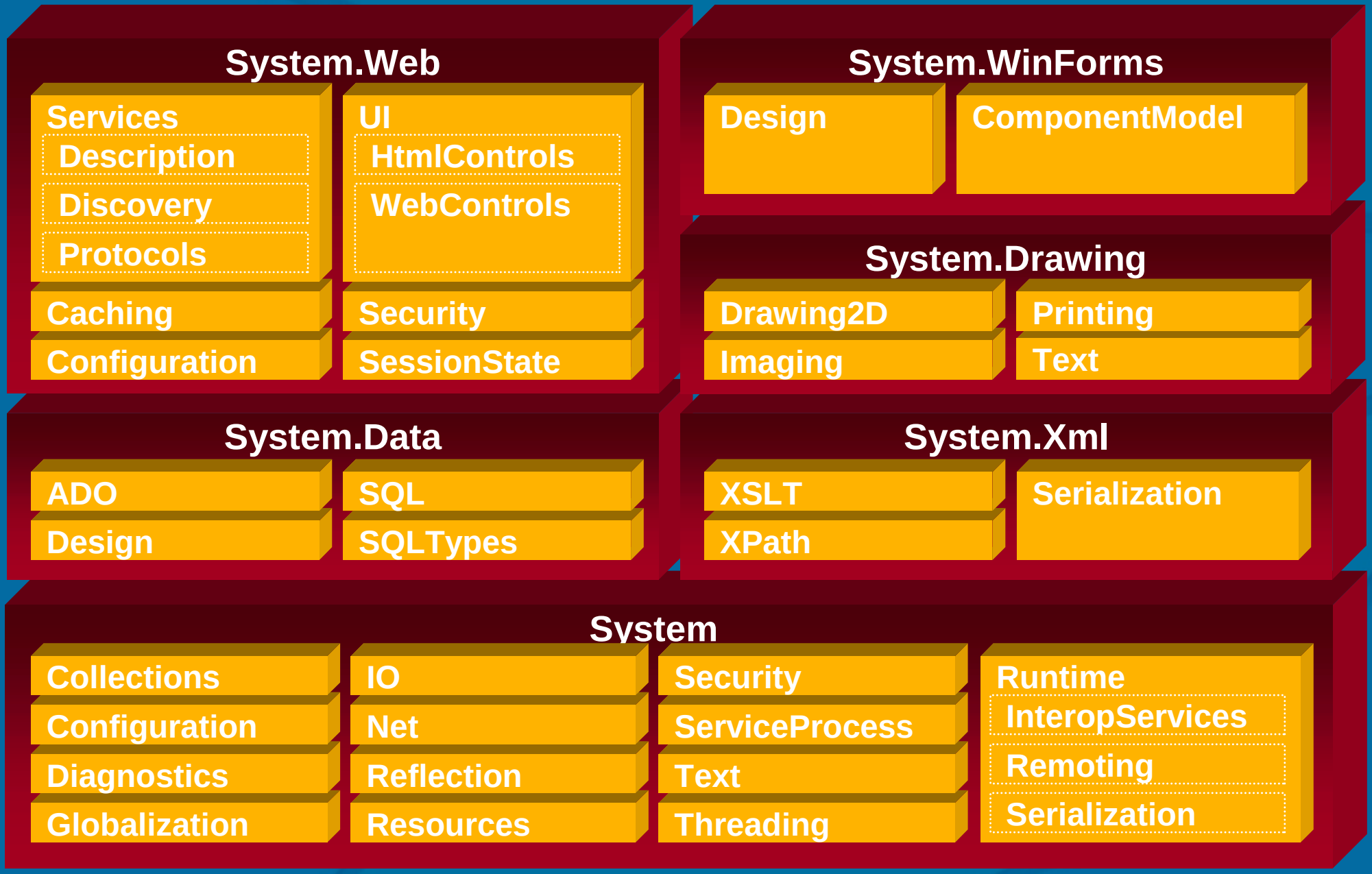
Beneficios de la CLR (II)

- Seguridad: identifica la identidad y origen del código y determina qué puede hacer
  - De dónde ha sido descargado o instalado
  - Qué métodos intenta invocar
  - Qué usuario lo ejecuta
  - Qué firma digital (si la hay) está almacenada en el manifiesto del assembly
- Monitoriza el código durante la ejecución y libera memoria (recolección de basura)
- Menos memoria perdida por “leaks”
- Gestión a bajo nivel de objetos y memoria, marshalling de datos y threads
- El CLR soporta políticas de seguridad configurables aplicadas a aplicaciones individuales: PKI

.NET class library

- Librería de clases nueva, orientada a objetos, jerárquica y unificada
  - Similar en funcionamiento a las funciones de APIs y SDKs
- Organiza tipos en base a un espacio de nombres (System)
  - Cientos de clases, interfaces y estructuras
- Definiciones de los tipos de datos básicos (Object, Integer, String), eventos y descriptores, interfaces y atributos
- Beneficios
  - Ineroperabilidad consistente entre lenguajes y plataformas
  - Modelo de programación unificado y consistente
  - Orientada a objetos y extensible
    - Heredar clases, sobrecargar métodos
  - Desarrollo de aplicaciones de consola y Servicios Windows

.NET Framework Namespace



.NET Framework es más simple que el API Win32

Windows API

```
HWND hwndMain = CreateWindowEx(
    0, "MainWClass", "Main Window",
    WS_OVERLAPPEDWINDOW | WS_HSCROLL | WS_VSCROLL,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    (HWND)NULL, (HMENU)NULL, hInstance, NULL);
ShowWindow(hwndMain, SW_SHOWDEFAULT);
UpdateWindow(hwndMain);
```

.NET Framework

```
Form form = new Form();
form.Text = "Main Window";
form.Show();
```

Pasos hacia la estándarización

- Apoyo y adopción de XML
  - Clases en el Framework
  - Explotar Servicios Web: nuevos estándares: WSDL, SOAP…
  - Acceso a datos y servidores (bases de datos, mensajería…)
- Remisión al ECMA (European Computer Manufacturers Association)
  - CLI (Common Language Infrastructure)
    - Porción del .NET Framework: CTS y CLR
    - Permite tratar adecuadamente excepciones, threading, depuración, páginas y Servicios Web entre lenguajes
    - Proyecto Mono (Ximian,Linux)
    - ROTOR (Microsoft, BSD)
  - CLS (Common Language Specification)
    - Integración de lenguajes, desarrollo de compiladores,
  - C# y otros lenguajes
    - Compilador del lenguaje en otros entornos
  - Cualquiera puede desarrollar un compilador o evolucionar el lenguaje

Referencias

- Así es Microsoft .NET
  - David S. Platt. McGraw-Hill, 2001
- http://msdn.microsoft.com/downloads/
  - CLR y herramientas en línea de comandos (SDK)
- http://www.go-mono.com/
  - Proyecto .NET para Linux
- http://www.gnu.org/projects/dotgnu
  - Proyecto .NET de GNU