INTRODUCTION TO .NET

- IDE:
  - Visual Studio;
  - Visual Studio Code.
- Database:
  - Microsoft SQL Server;
  - SQL Server Management Studio (SSMS).

- Life As a C/Win32 API Programmer:
  - Manual memory management, ugly pointer arithmetic, and ugly syntactical constructs.
- Life As a C++/MFC Programmer:
  - Difficult to learn and error-prone experience.
- Life As a Visual Basic 6.0 Programmer:
  - Not a fully object-oriented, no multithread supports (unless working with low level of Win32 API calls).
- Life As a Java/J2EE Programmer
  - No language integration, not appropriate for many graphically or numerically intensive applications.
- Life As a COM Programmer
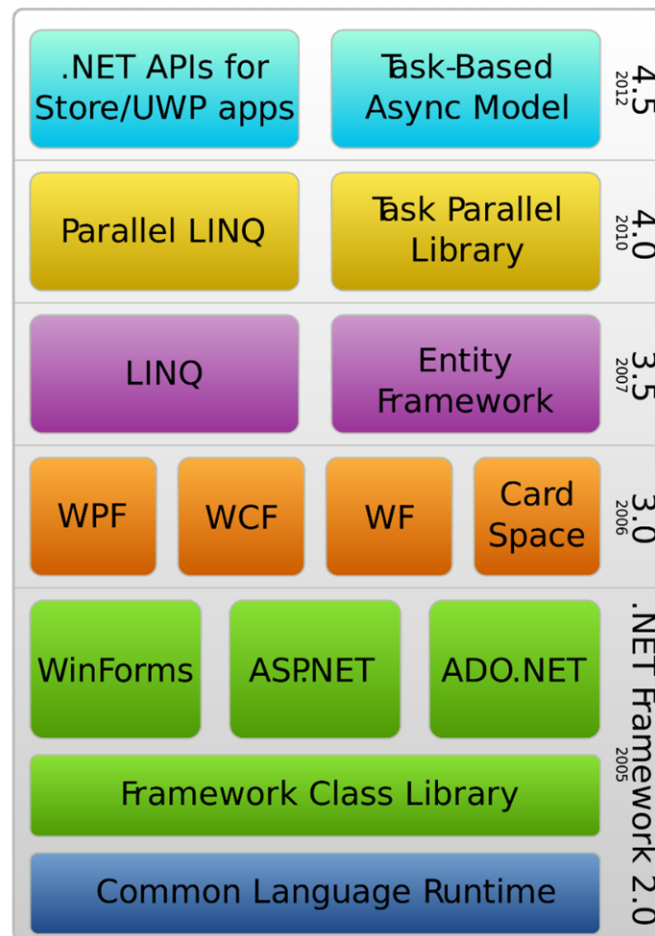 - No support for classical inheritance, very complex to build and use.

- Why .NET ???
- .NET Framework.
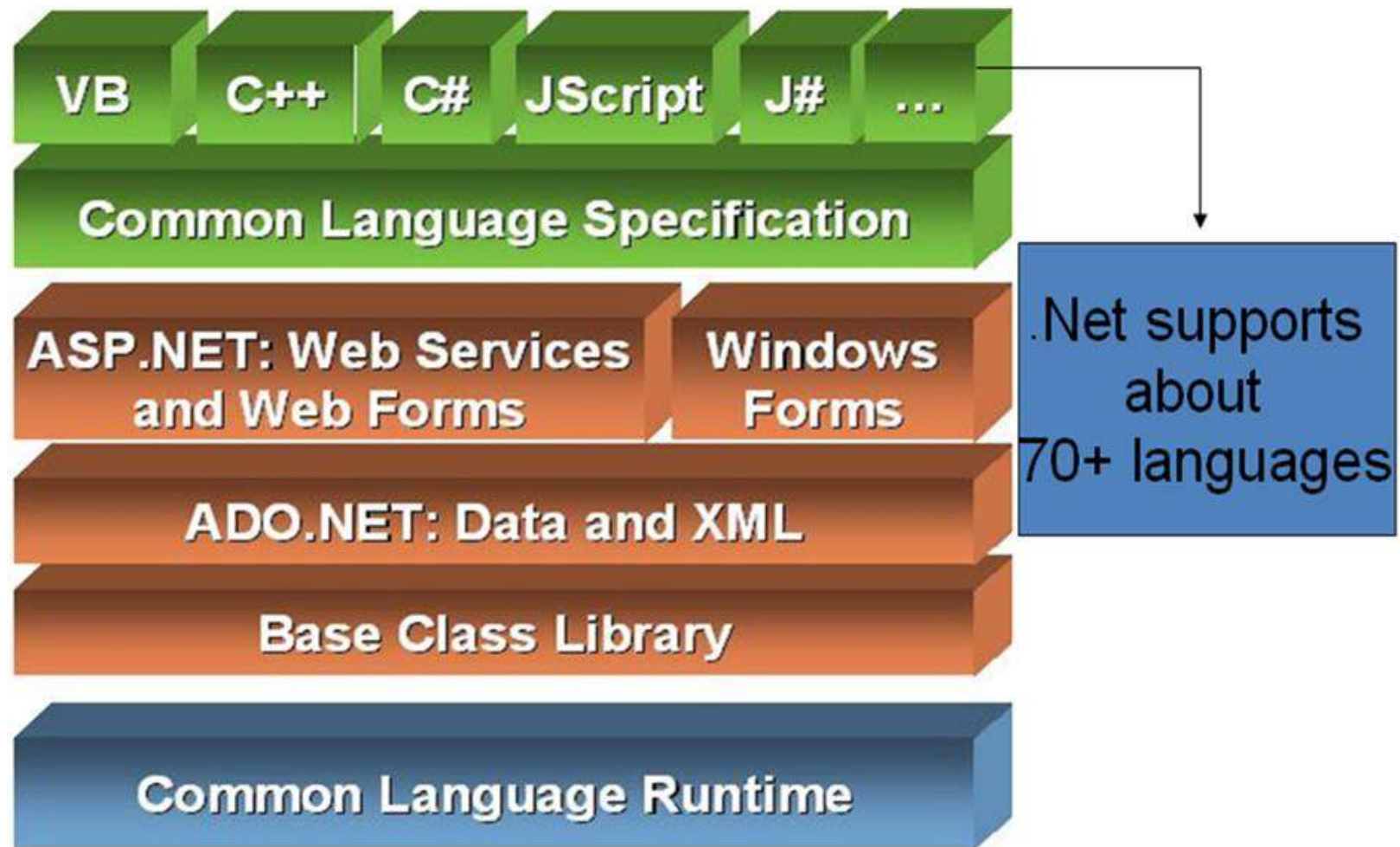- .NET Core (cross-platform).
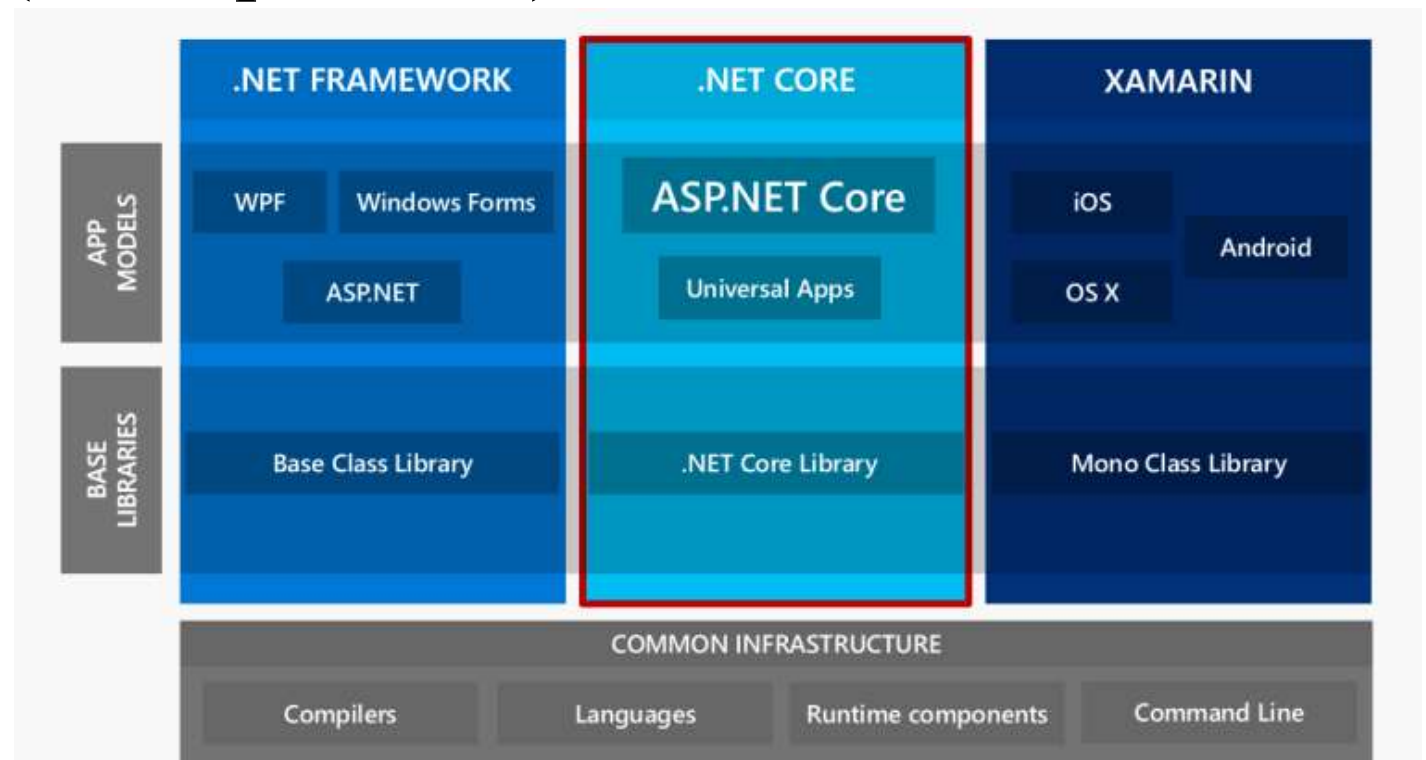
- .NET Framework Version History:

- Components of .NET Framework:

- CLR is physically represented by **mscoree.dll** library (Common Object Runtime Execution Engine):
  - This library is loaded automatically when an assembly is referenced for use.
- CLR responsibilities:
  - Resolving the location of an assembly and finding the requested type within the binary by reading the contained metadata;
  - Loading the type into memory;
  - Compiling CIL into platform-specific instruction;
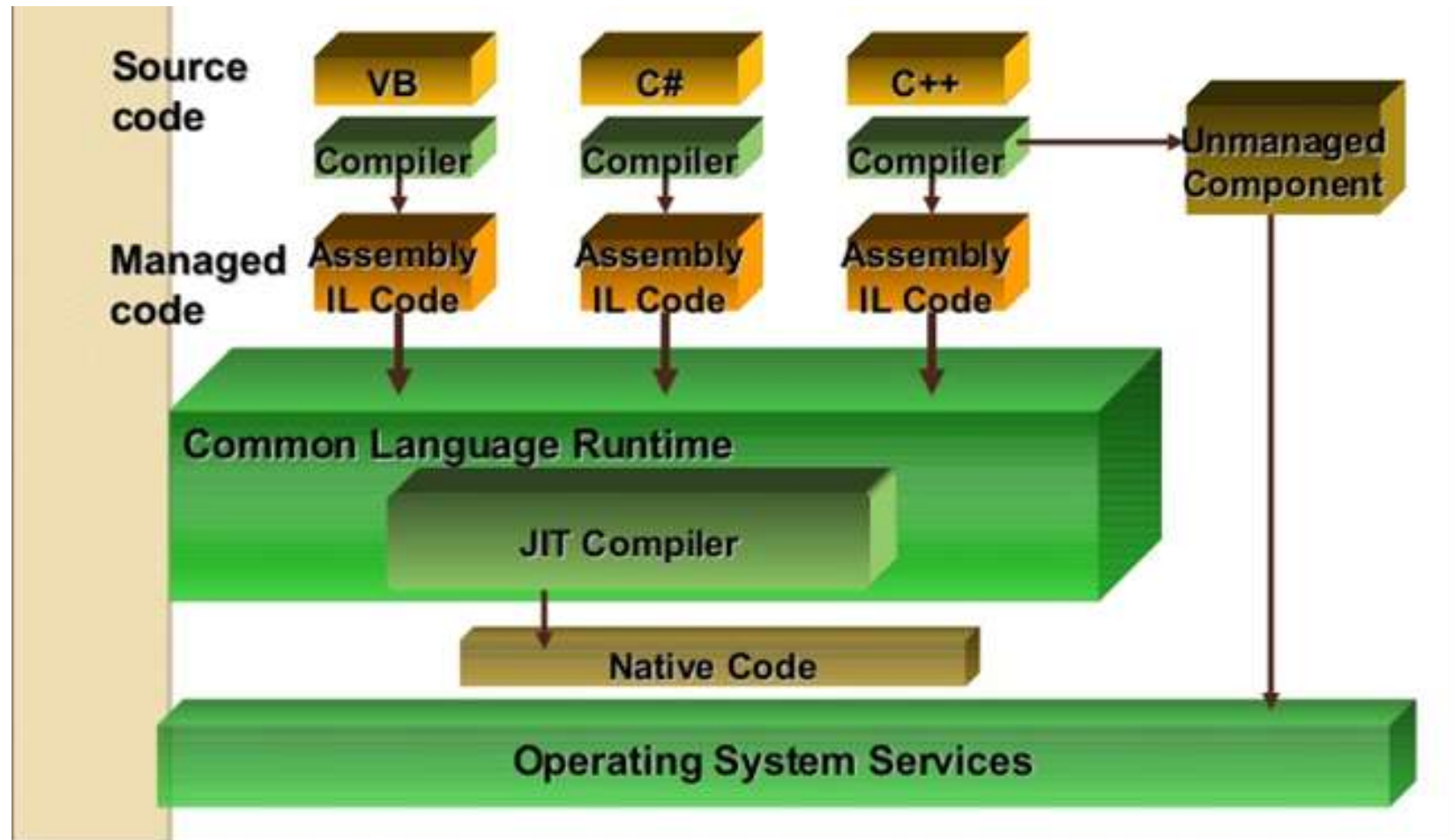 - Performing security checks, executing the code.

• **mscoree.dll**
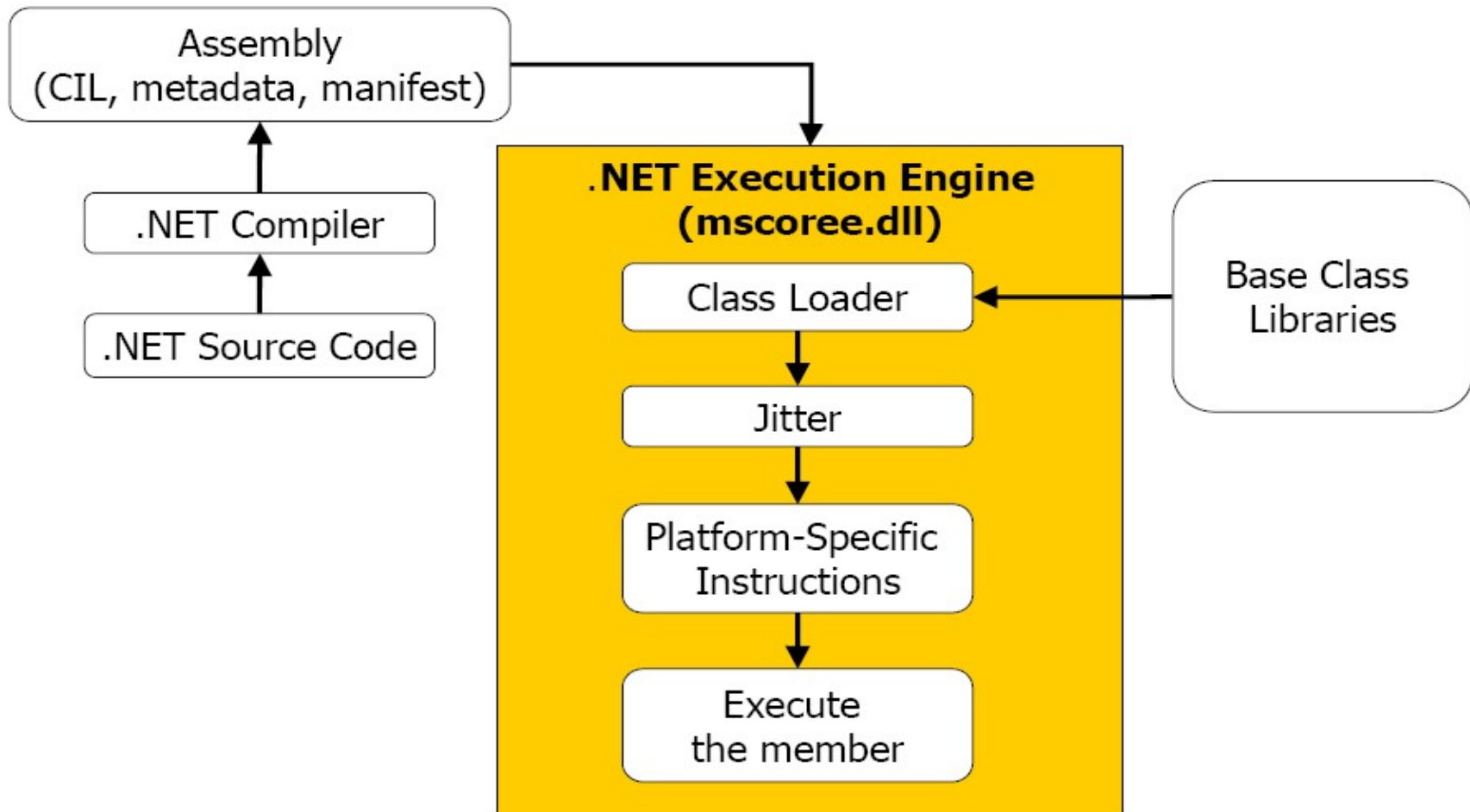
- CLR Architecture:
  - Managed Code;
  - Microsoft Intermediate Language (MSIL);
 - Native Code.
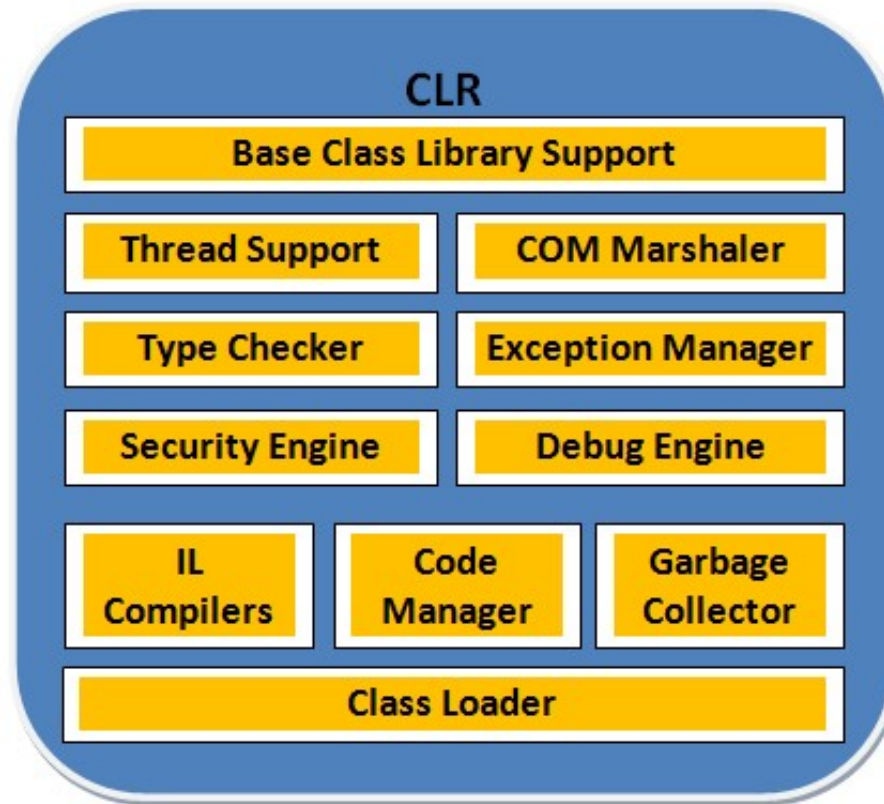
- ## CLR Execution Model:

- Also known as Microsoft Intermediate Language (MSIL);
- CIL is a language that sits above any particular platform-specific instruction set:
  - The same idea as Java's virtual machine.
- Compilers of all .NET-aware languages emit CIL instructions:
  - Binaries are platform-independent.
- When the CIL code is about to run, the Jitter (just-in-time compiler) compiles it into native (machine) code:
  - Jitter will cache resulting machine code in memory.

- Defines classes available to all .NET Framework languages.
- Defines various primitives:
  - Threads, file input/output, graphical rendering, interaction with external hardware devices;
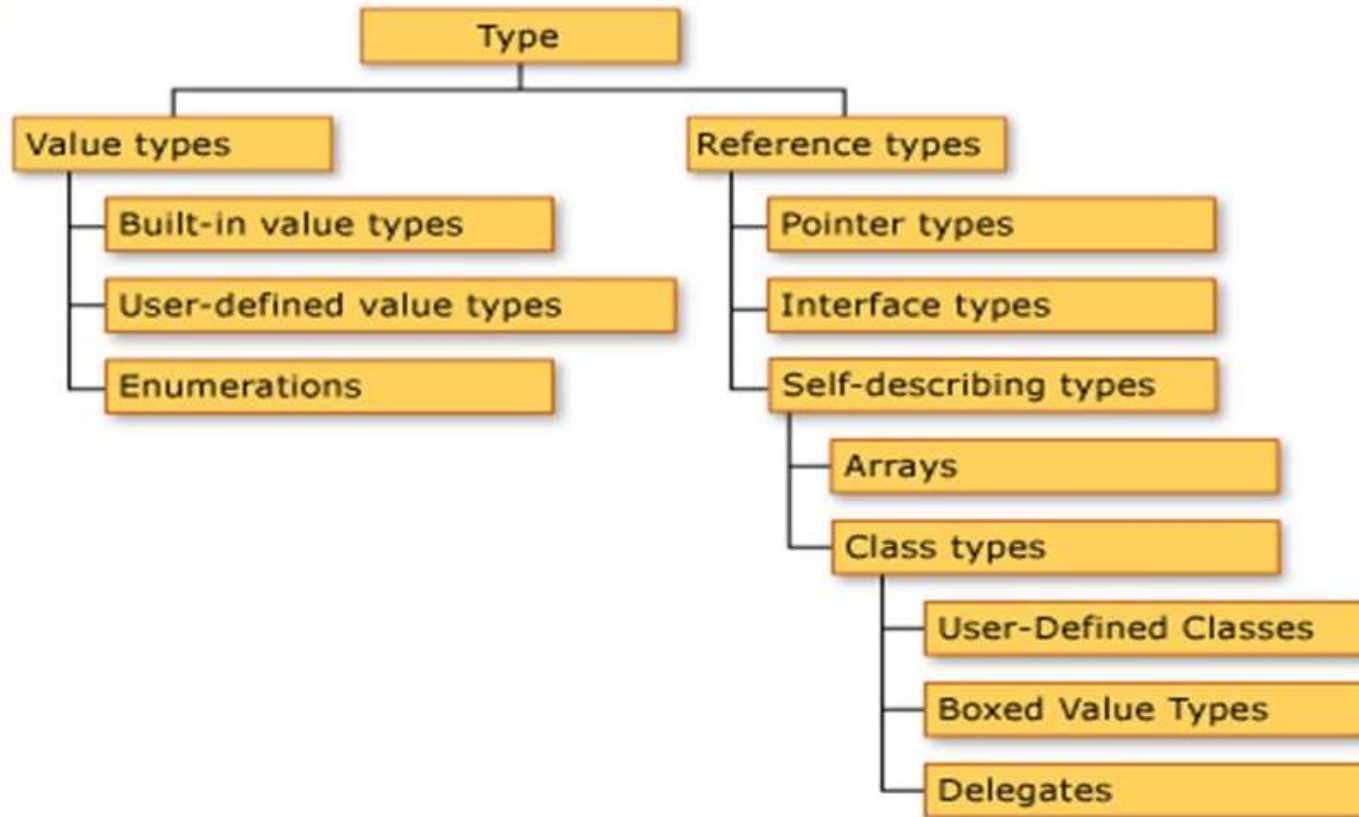 - Database access, XML manipulation, security.
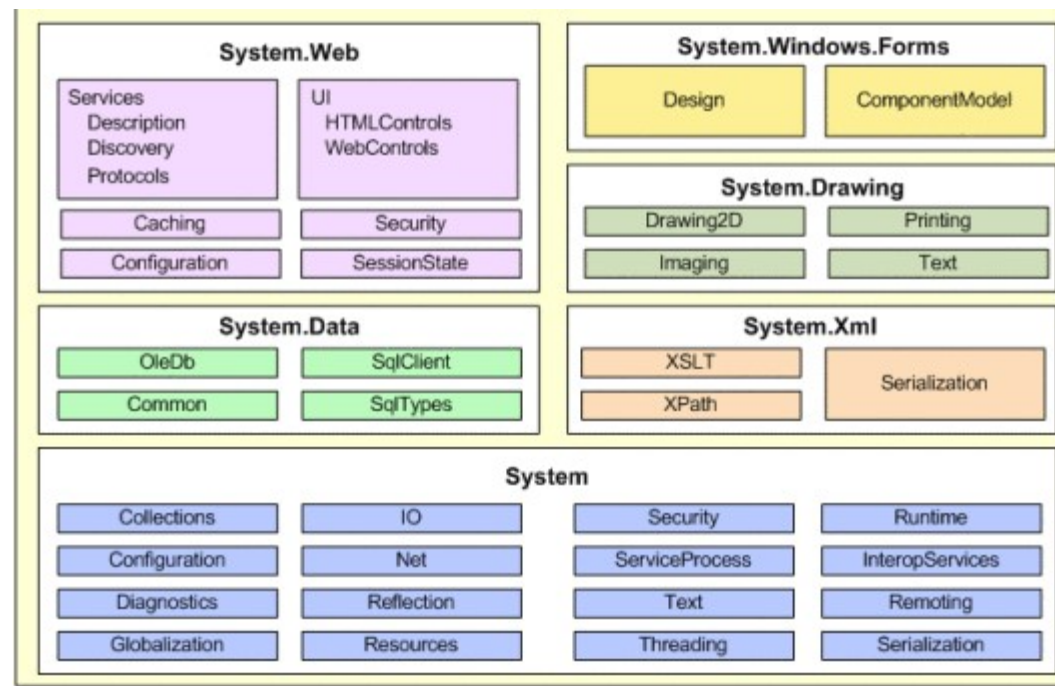
- Types in .NET:
  - Value types;
  - Reference types.

COMMON TYPE SYSTEM (CTS)

- Intrinsic CTS Data Types

| CTS Data Type	VB.NET	C#	Managed C++
System.Byte	Byte	byte	unsigned char
System.SByte	SByte	sbyte	signed char
System.Int16	Short	short	short
System.Int32	Integer	int	int or long
System.Int64	Long	long	__int64
System.UInt16	UShort	ushort	unsigned short
System.UInt32	UInteger	uint	unsigned int or unsigned long
System.UInt64	ULong	ulong	unsigned __int64
System.Single	Single	float	Float
System.Double	Double	double	Double
System.Object	Object	object	Object^
System.Char	Char	char	wchar_t
System.String	String	string	String^
System.Decimal	Decimal	decimal	Decimal
System.Boolean	Boolean	bool	Bool

- CLS defines a subset of common types and programming constructs that all .NET programming languages can agree on in order to interoperate successfully.

  - CLS can be viewed as a subset of the full functionality defined by the CTS.

- Also, CLS defines a set of rules/features a given .NET-aware compiler must support to produce code that can be hosted by the CLR.

  - Ex: overloading: methods, and constructors are allowed to be overloaded; fields and events must not be overloaded.

CTS
(Common Type System)   C#   CLS   C++   VB .NET
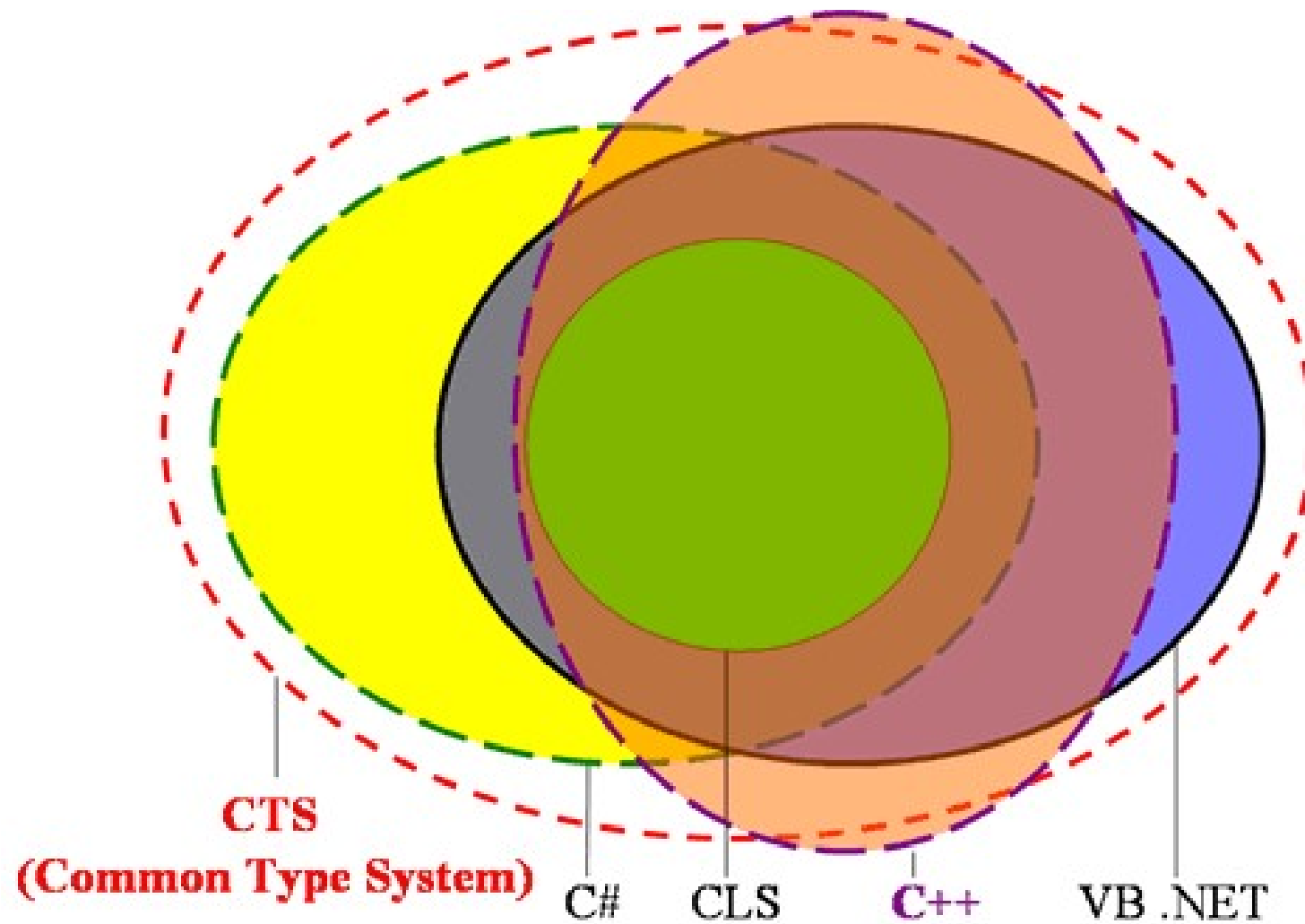
- ## .NET Assemblies:
  - Binaries containing Common Intermediate Language (CIL) instructions and type metadata
    - .dll or .exe files, which cannot be run without the .NET runtime.
  - The most important features:
    - versioning;
    - self-describing;
    - configurable (using private/global assembly via App.config).
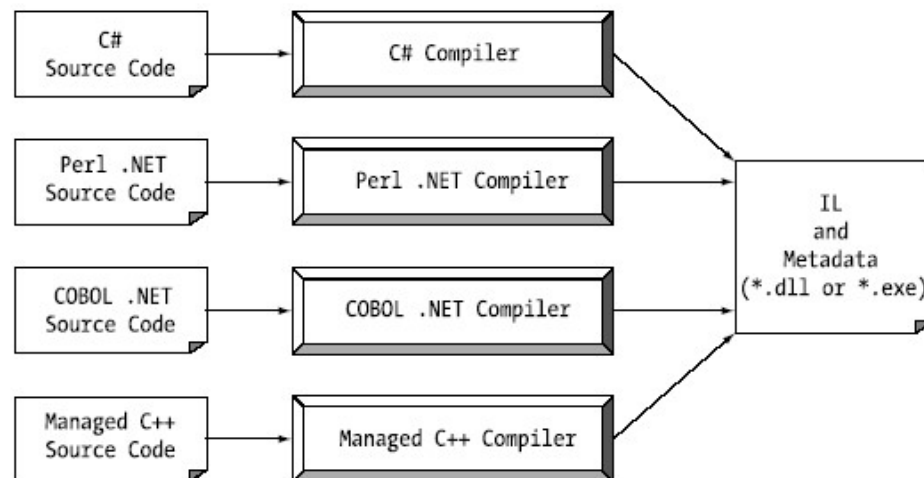
- .NET Assembly's Format:
  - Win32 File Header;
  - CLR File Header;
  - CIL code;
  - Type metadata;
  - Assembly manifest;
  - Optional embedded resource.

Thank You !

IT FACULTY