ГБПОУ г. Москвы
«Первый московский образовательный комплекс»

Объектно-ориентированное программирование

Лекция 1. Платформа разработки .Net

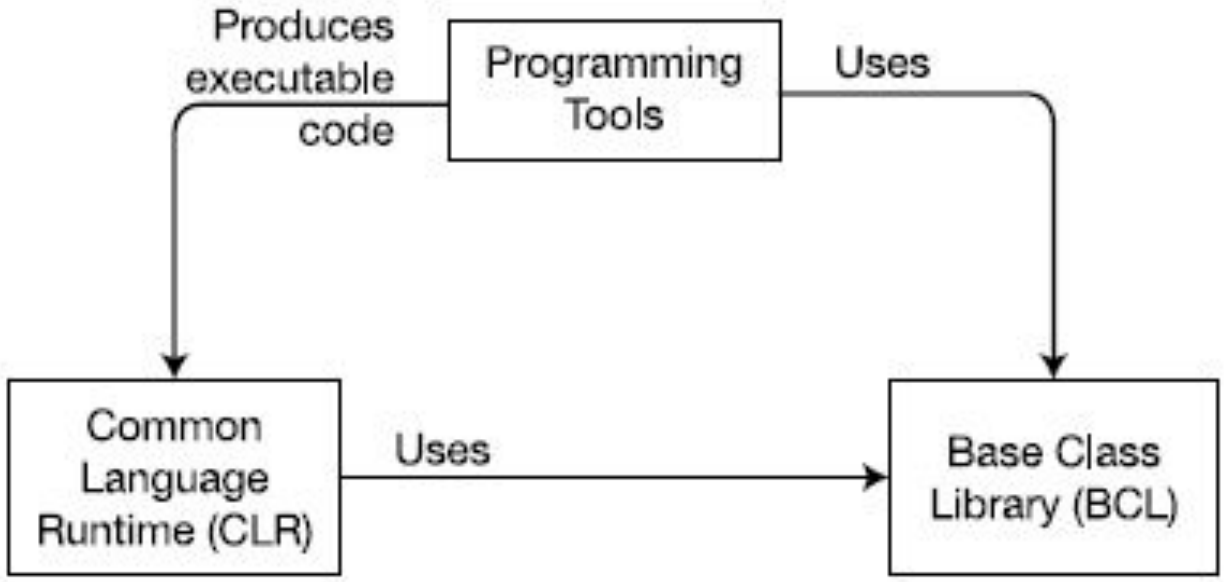
Платформа Microsoft .Net

- В 2002 году компания Microsoft выпустила платформу разработки и выполнения программ под управлением ОС Windows -.NET Framework.
  - новая интегрированная, объектно-ориентированная среда разработки и выполнения программ.
  - предлагает новый подход к решению проблем разработки ПО и соответствие целям информационных систем следующего поколения.
- С программной точки зрения .Net Platform это
  - набор библиотек классов;
  - среда выполнения программ CLR;
  - набор программных инструментов (Visual Studio, компиляторы, отладчики и пр.).

Платформа программирования

Любая платформа разработки и выполнения программ включает:

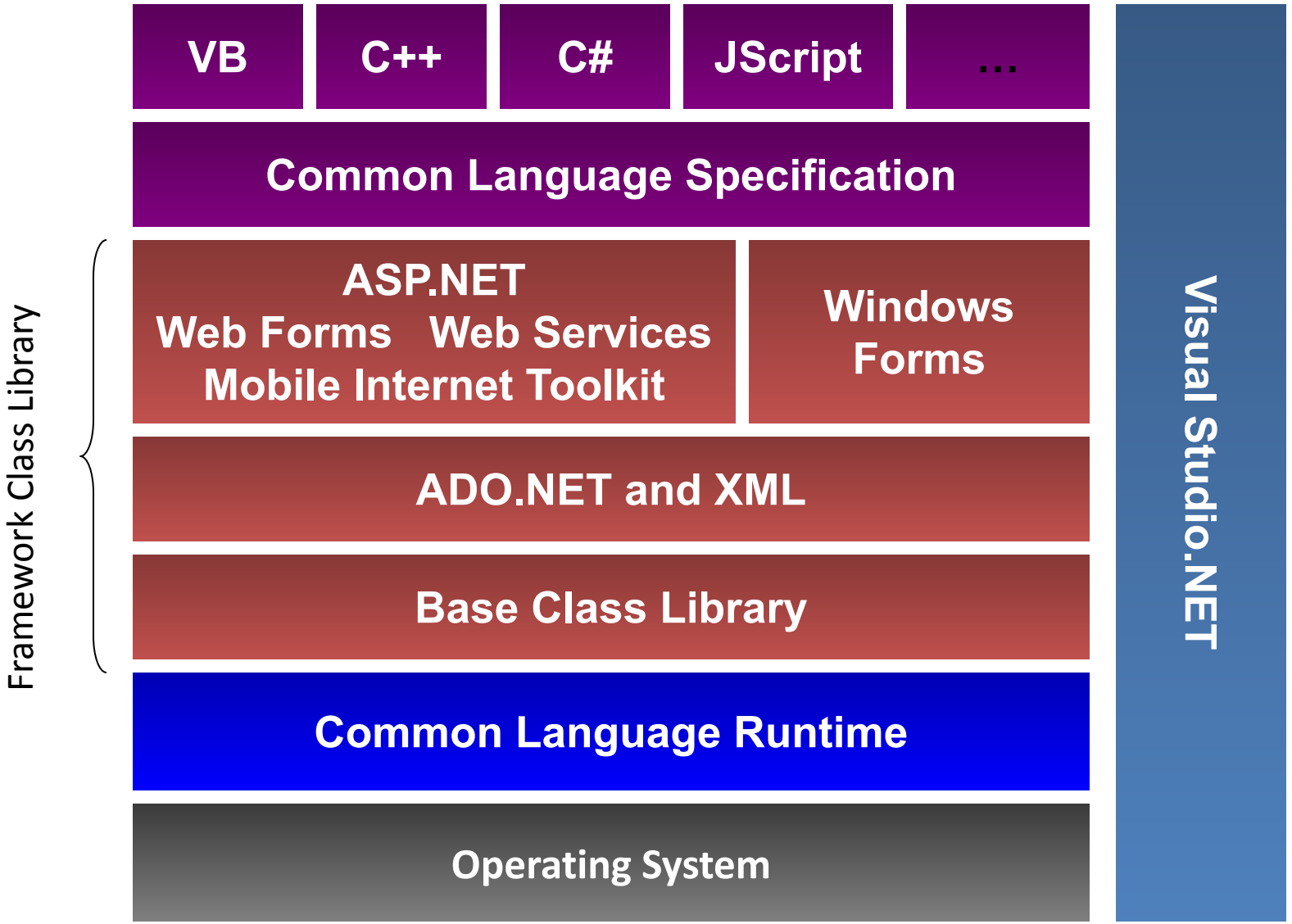
- Среду выполнения кода (программы)
- Среду разработки программы
- Библиотеку классов



Состав платформы .NET

- Общая среда выполнения (Common Language Runtime)
  - Runtime engine (виртуальная машина) для управляемого кода
  - Управление потоками и памятью
  - Хорошо гранулированная, ясная защищенность (security)
  - Межъязыковое управление исключениями, диагностика, отладка
- Библиотека классов (.NET Framework Class Libraries)
  - Набор иерархически организованных библиотек классов
  - Используется всеми языками .NET
  - Встроенная общая система типов данных (common type system)
  - Объектно-ориентированная, расширяемая
- Набор инструментов разработки и отладки программ
  - Компилятор (VB .NET, C# и C++)
  - Инструменты (AL.exe, Disasm.exe)

Платформа .NET Framework



Платформа .NET Framework

Список версий .Net Framework

Версия	Дата выхода	Visual Studio	По умолчанию в Windows
1.0	2002-01-05	Visual Studio .NET	
1.1	2003-04-01	Visual Studio .NET 2003	Windows Server 2003
2.0	2005-11-07	Visual Studio 2005	
3.0	2006-11-06		Windows Vista, Windows Server 2008
3.5	2007-11-09	Visual Studio 2008	Windows 7, Windows Server 2008 R2
4.0	2010	Visual Studio 2010	

Особенности платформы Microsoft.Net

- **Многоплатформенность**: она может работать на разных компьютерах, начиная от серверов и настольных компьютеров и заканчивая наладонными компьютерами (PDA) и сотовыми телефонами.
- **Единая модель и инструментарий** разработки всех типов приложений (локальных и сетевых)
- **Активная поддержка международных стандартов**: она использует такие стандартные протоколы коммуникации, как XML, HTTP, SOAP и WSDL.
- **Безопасность**: данная платформа предоставляет намного более безопасную среду выполнения, даже в случае получения программного кода из не надежных источников.

Основные идеи .Net технологии

1. Общий промежуточный язык
(Common Intermediate Language - CIL)
   Все компиляторы .Net создают программу на специальном языке CIL

2. Общая среда выполнения
(Common Language Runtime - CLR)
   Все программы выполняются под управлением специальной программы (CLR)

3. Framework Class Library (FCL)
   При выполнении программы, написанные на любом языке, используют общую библиотеку

Упрощенная разработка

- Высокий уровень абстракции
  - Нет низкоуровневой инфраструктуры COM
  - Полностью объектно-ориентированная
- Единая система типов
  - Все является объектами некоторых классов, нет variants (без типовых переменных),
  - Один тип string,
  - Все символы кодируются в системе Unicode
- Программные компоненты
  - Свойства, методы, события и атрибуты являются базовыми элементами классов.
- Бесшовное взаимодействие между языками

Установка .Net Framework

- В ОС Windows XP .Net Framework необходимо устанавливать. В последующих ОС (Vista, 7) данная платформа уже установлена.

- C:\WINDOWS\Microsoft.NET\Framework
  - Версии платформы
    - v2.0.50727

- Варианты установки
  - Software Development Kit (SDK) (354 Mb) – для выполнения и создания управляемых приложений
  - Redistributable Package (22.4 Mb) – выполнения управляемых приложений

Факты об .NET Framework

- .NET Framework SDK свободно распространяется
  (SDK – Software Development Kit)
- .NET Framework SDK включает компиляторы для языков: C#, VB.NET и C++.
- Программирование на .NET Framework SDK не требует наличия среду разработки Visual Studio .NET
- Имеются бесплатные версии среды разработки Visual.Studio (Express Edition)
- .NET Framework SDK включает набор инструментов, запускаемых из командной строки, такие как компиляторы, отладчики, и разные утилиты
- Rotor это открытый код реализации .NET Common Language Runtime (CLR) и C# языка

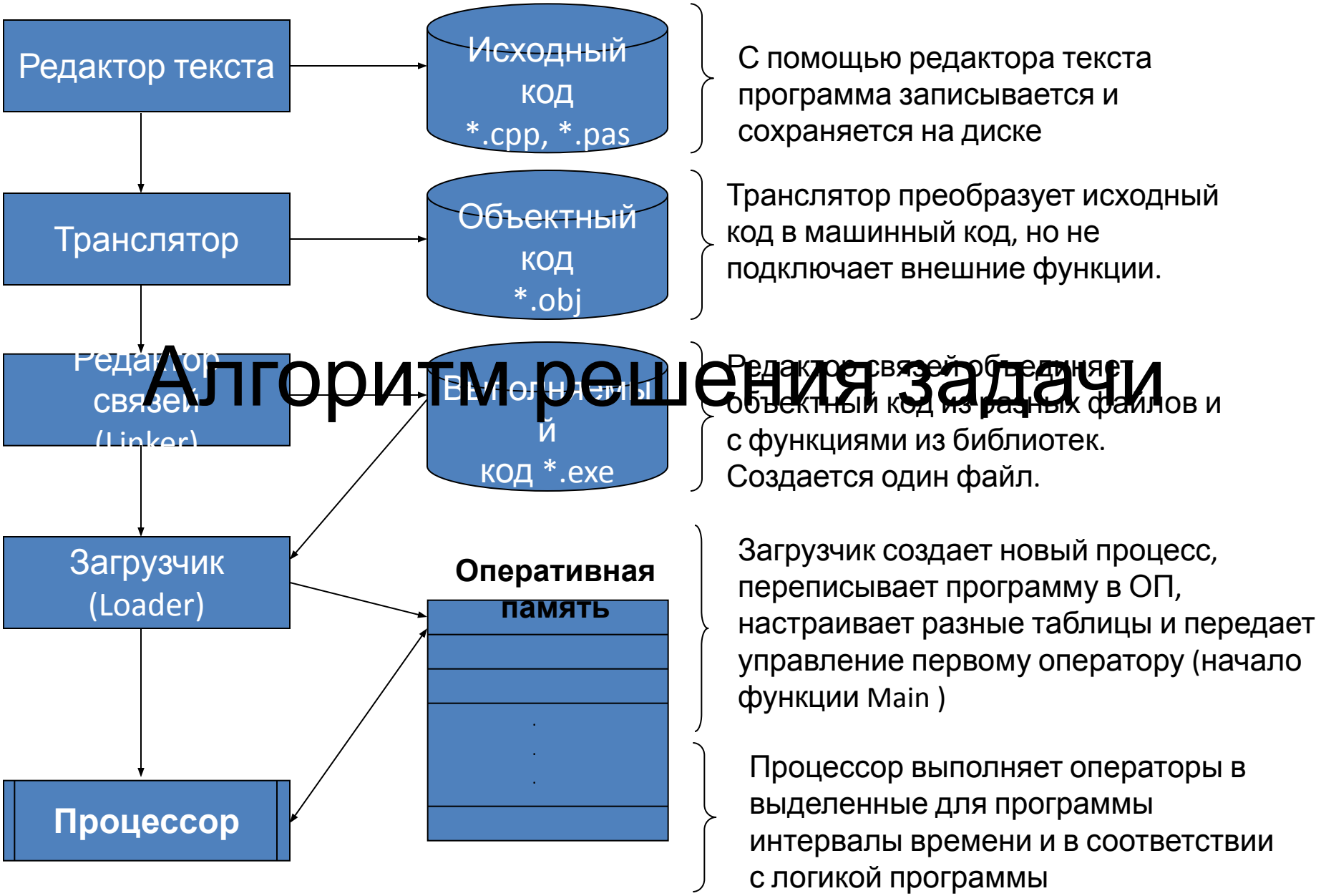
Проект Mono

- Проект по созданию полноценной реализации платформы системы .NET на базе свободного программного обеспечения. (выполняется компанией Novell, руководит Мигель де Иказа, известный разработчик, участник проекта GNOME и др.)
- Включает следующие компоненты:
  - компилятор языка C# — mcs,
  - среду исполнения (CLR) — mono (с поддержкой JIT) и mint (без поддержки JIT),
  - отладчик, а также
  - ряд библиотек, включая реализацию ADO.NET и ASP.NET.
  - В рамках проекта также разрабатываются привязки для графической библиотеки GTK+ на платформу .NET.
- Среда исполнения mono может исполнять модули, написанные на языках C#, Visual Basic .NET, Java, Boo, Nemerle, Python, JavaScript, PHP и Object Pascal (при наличии компилятора в среду .Net/Mono). Ожидается также поддержка языков C, Ada 2005 и Eiffel.
- Реализации Mono существуют для таких операционных систем, как:
  - GNU/Linux,
  - FreeBSD,
  - Solaris,
  - Mac OS X,
  - Microsoft Windows и
  - Unix.

Два типа программ в ОС Windows

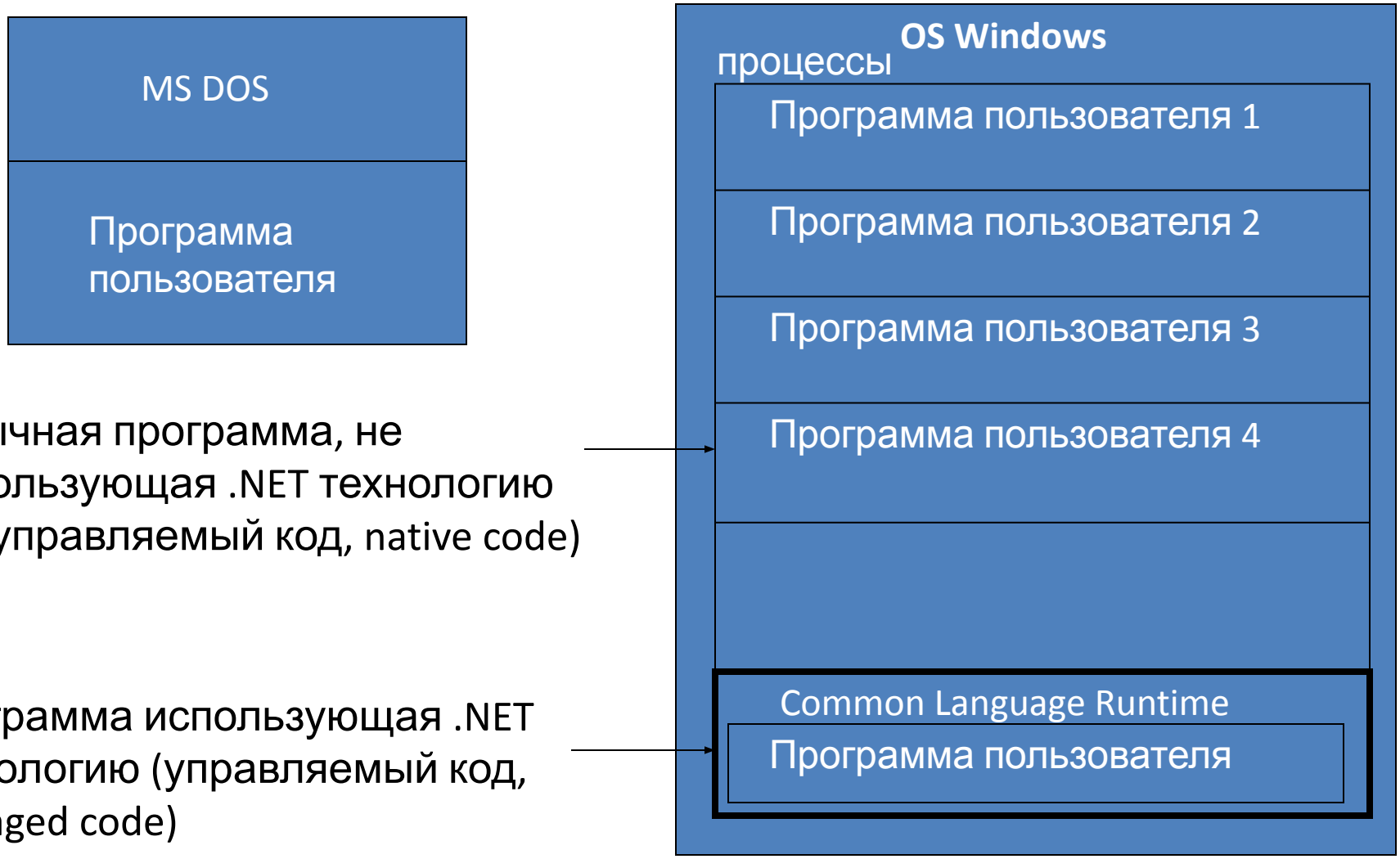
- Программы (exe модули) в виде набора инструкций процессора (native code)
  - выполняются процессором непосредственно
  - все ранее созданное программное обеспечение
- Программы имеющие специальную структуру на промежуточном языке - управляемый код (managed code)
  - создаются на платформе .Net
  - выполняются в среде CLR

Классическая последовательность создания программы



Алгоритм решения задачи

Работа программ в MS DOS и OS Windows



Обычная программа, не использующая .NET технологию (не управляемый код, native code)

Программа использующая .NET технологию (управляемый код, managed code)

Последовательность создания и выполнения программы на платформе .Net

С помощью редактора текста программа записывается в файл и сохраняется на диске.

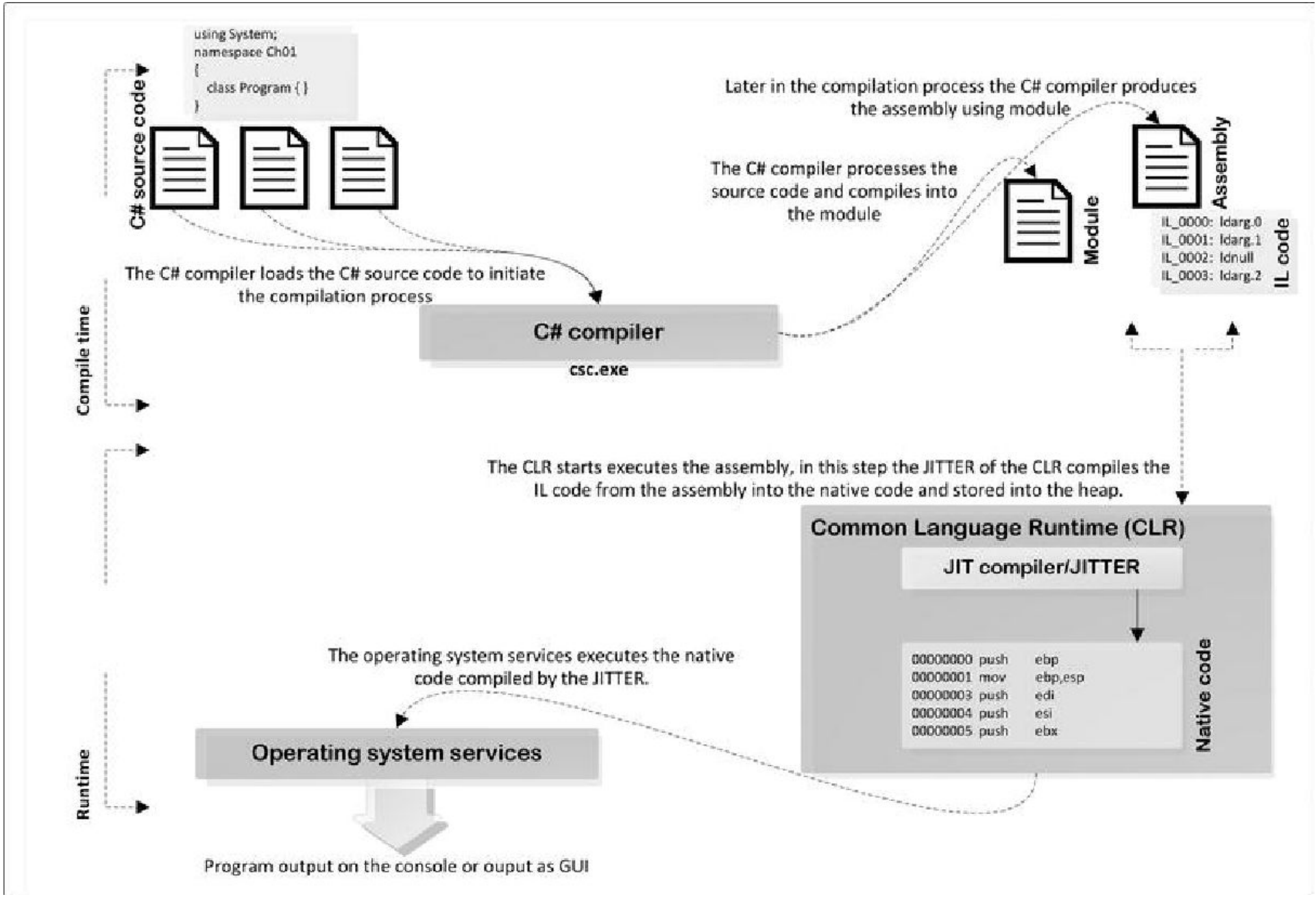
Компилятор с любого языка преобразует исходный код в промежуточный код, и задает метаданные модуля с описаниями всех типов (классов) в модуле.

Может использоваться AL.exe утилита для создания сборки (assembly). Для простых сборок компилятор автоматически создает сборку. Сборка не объединяет модули в единый модуль, а соединяет их логически.

Загрузчик создает новый процесс, в который загружает среду выполнения CLR (Common Language Runtime).

CLR вызывает JIT компилятор, который по мере использования классов сборки выполняет компиляцию на язык машинных инструкций. Вначале управление передается статической функции Main.

Процессор выполняет операторы в соответствии с логикой программы. Если класс не на машинном языке, то формируется exception, которое вызывает JIT


using System;
namespace Ch01
{
class Program { }
}
C# source code
The C# compiler loads the C# source code to initiate the compilation process
C# compiler
csc.exe
Compile time
Later in the compilation process the C# compiler produces the assembly using module
The C# compiler processes the source code and compiles into the module
Module
Assembly
IL_0000: ldarg.0
IL_0001: ldarg.1
IL_0002: ldnull
IL_0003: ldarg.2
IL code
The CLR starts executes the assembly, in this step the JITTER of the CLR compiles the IL code from the assembly into the native code and stored into the heap.
Common Language Runtime (CLR)
JIT compiler/JITTER
00000000 push ebp
00000001 mov ebp,esp
00000003 push edi
00000004 push esi
00000005 push ebx
Native code
The operating system services executes the native code compiled by the JITTER.
Operating system services
Runtime
Program output on the console or ouput as GUI

Компиляция программ в .Net

Основные типы модулей с инструкциями компьютера в ОС Windows

- Выполняемые программы (*.exe)
- Статические библиотеки (*.lib)
- Динамические библиотеки (*.dll)

Типы программных модулей в .Net платформе

- сборки (assembly)
  - exe (может быть запущен на выполнение)
  - dll (библиотека классов, может использоваться в других программах, которые на нее ссылаются - reference)
- специальные модули (не включает метаданные о сборке, а только метаданные с описанием типов)
  - netmodule (может быть включен в сборку).

Новый тип программы – Сборка (assembly)

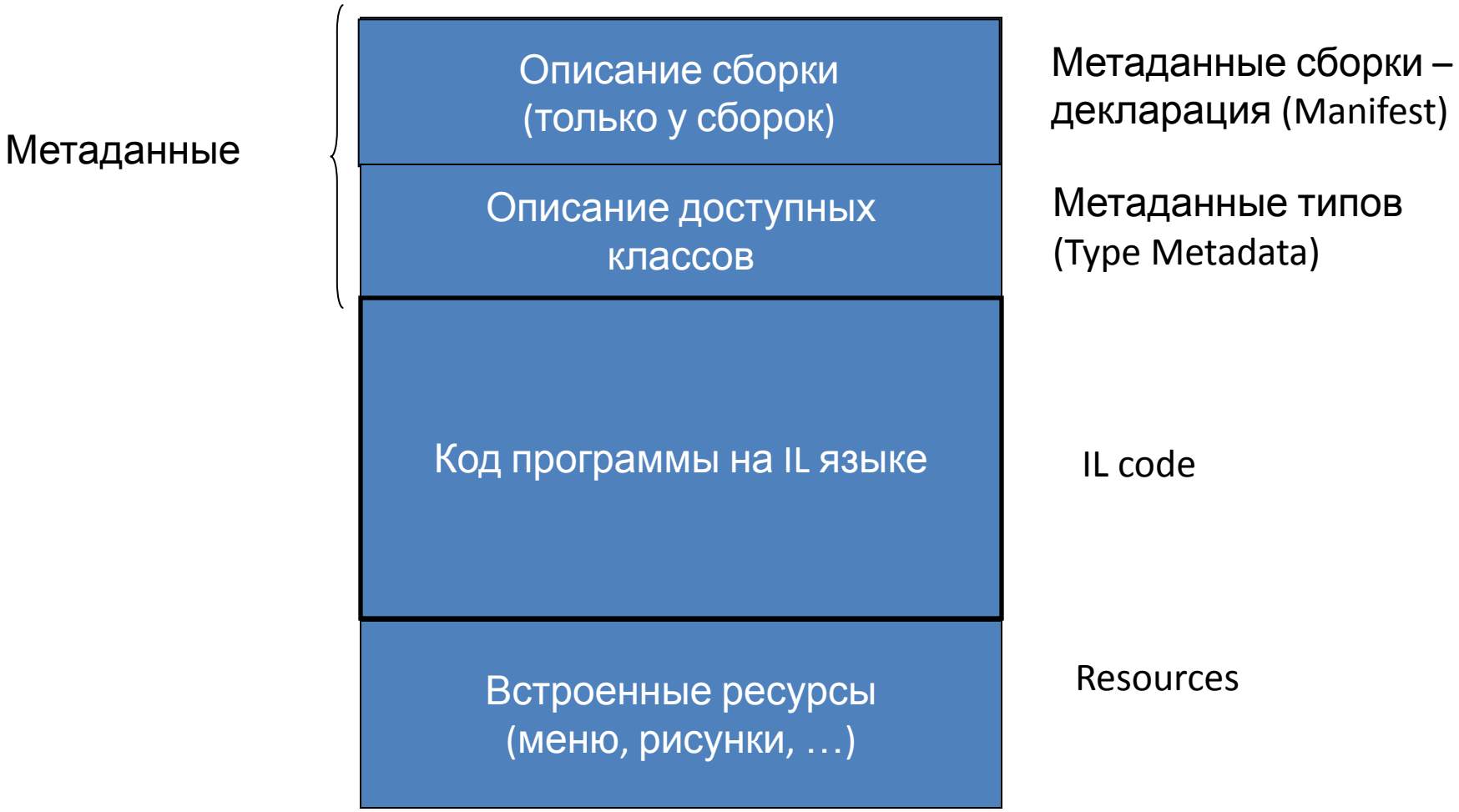
- Сборка (assembly) – включает 1 или более управляемых модулей (УМ)
- Управляемый модуль (managed module) – содержит 1 или более классов
- Один класс должен включать 1 статический метод (static method) Main()
- В методе Main должно быть решение задачи, или создание экземпляров класса, которые решают задачу

Сборка (продолжение)

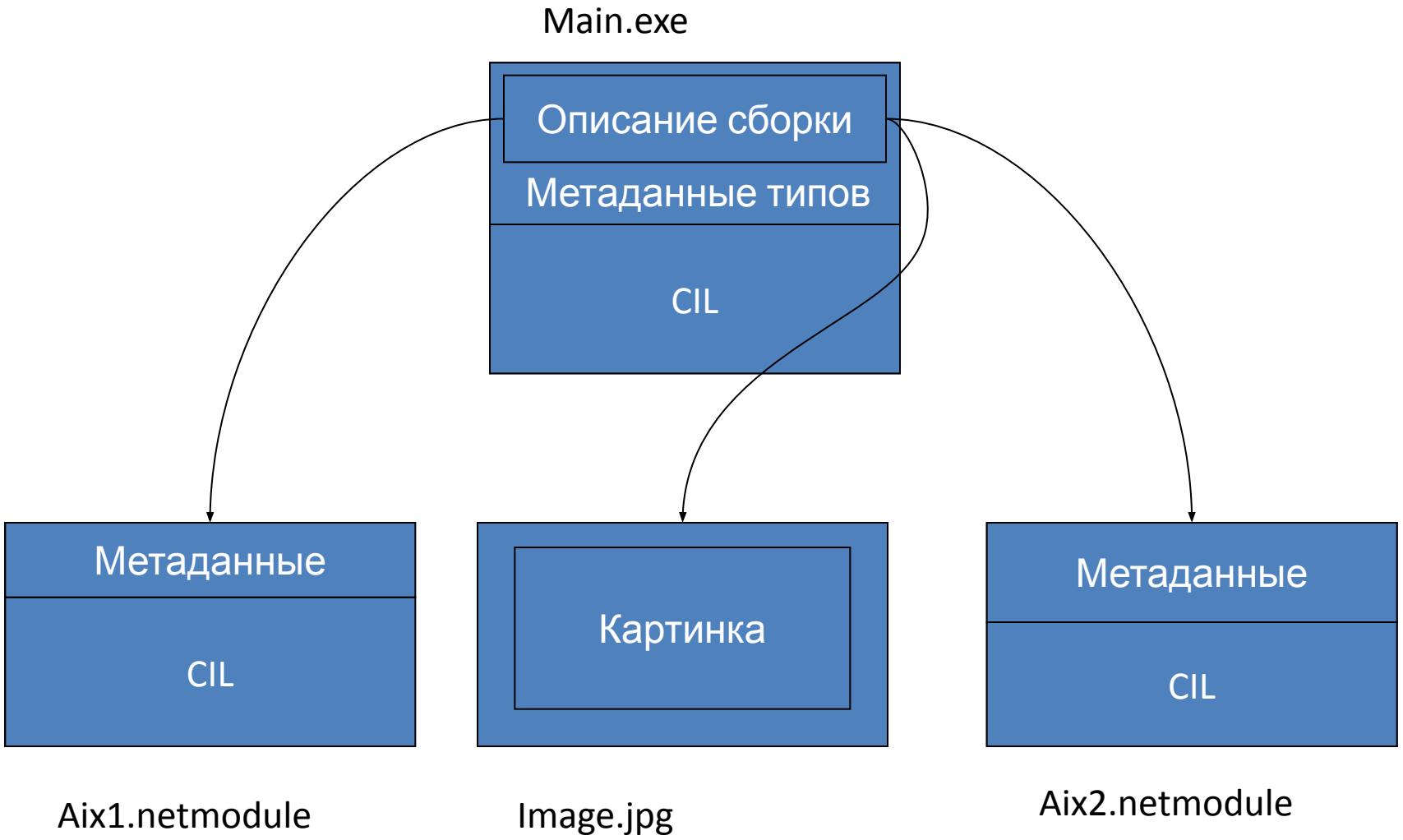
- Компилятор сразу создает управляемый модуль и сборку
- Все модули сборки хранятся (один или несколько файлов) хранятся в одном каталоге
- Утилита AL.exe – для создания многофайловых сборок (может быть на разных языках)
- В сборке есть *декларация* – дополнительные метаданные, которые описывают состав сборки

Формат исполняемых файлов, объектного кода и динамических библиотек

Формат управляемых программных единиц



Многофайловая сборка



Метаданные сборки - декларация (Assembly Manifest)

- Декларация (Манифест) это часть метаданных модуля
- Декларация – метаданные, описывающие содержимое сборки, в частности, PE файлы, представляющий данную сборку.
- Состав декларации
  - Идентификация – имя, номер версии, открытый ключ
  - Список файлов сборки
  - Список сборок на которые есть ссылки (references)
  - Экспортируемые типы (классы)

- Хранится в одном выделенном файле сборки
- Первым делом CLR читает декларацию

Метаданные типов

- Единое (табличное) представление информации о типах и других именованных сущностях, определенных и используемых в .NET-приложении. По структуре очень близки к реляционной СУБД. Создаются компилятором
- ***Полное*** описание информации о типах (классах) (определяемых и используемых)
- Хранятся в виде набора таблиц
- Расширяют возможности старых технологий, таких как IDL
- Всегда связаны с кодом на IL (генерируются одновременно) => синхронизированы

Использование метаданных

- Устранение необходимости в заголовочных и библиотечных файлах при компиляции
- Интеллектуальные функции наподобие IntelliSense в VS.NET (браузер типов) – раньше такое тоже было, но использовались TLB
- Верификация кода (проверка на безопасность)
- Основа для сборки мусора

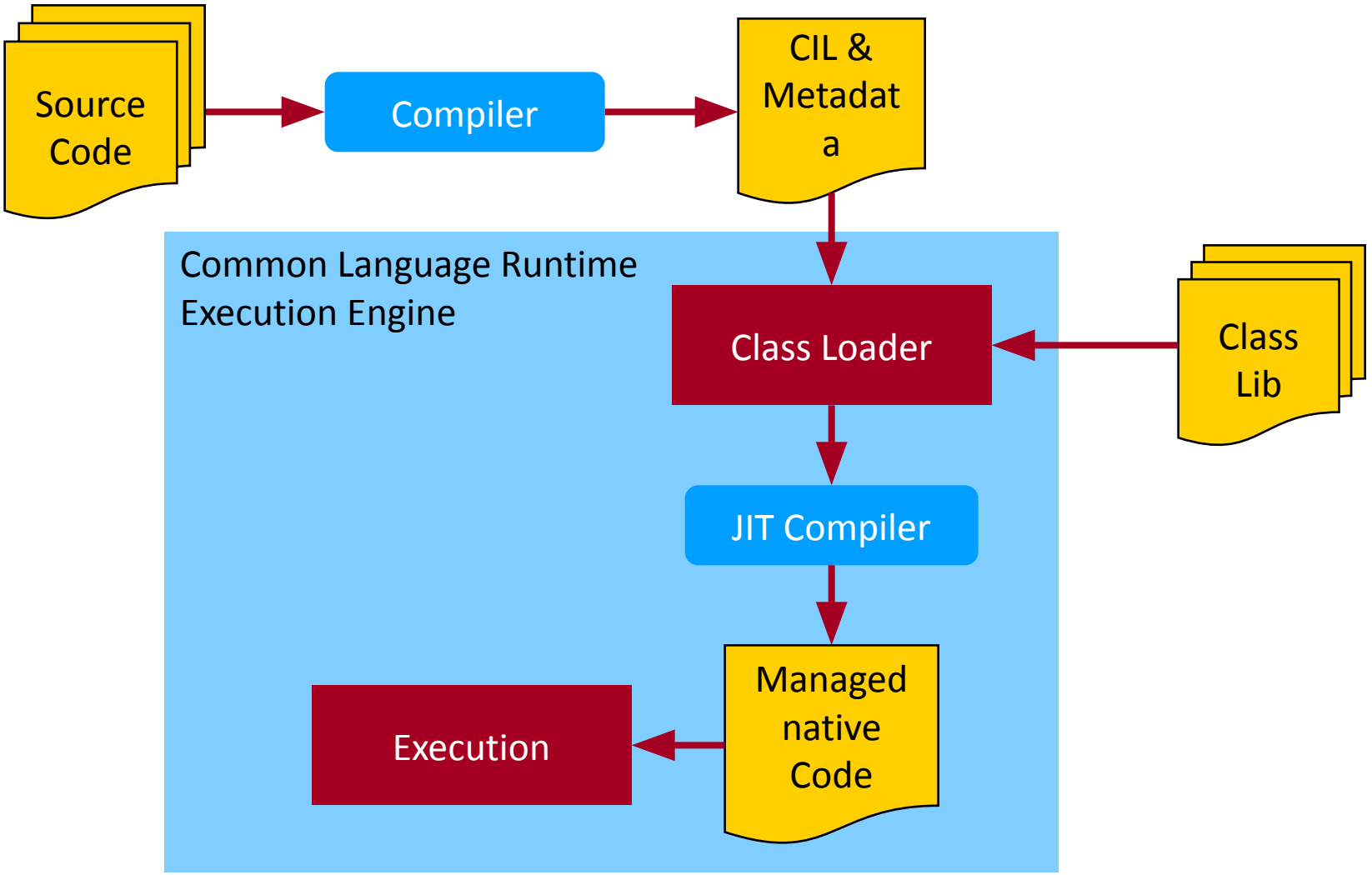
Типы сборок

- Нестрого именованные сборки (weakly named) – сборки без криптографической подписи
- Строго именованные сборки (strongly named) – имеют
  - Открытый ключ создателя
  - цифровую подпись, вычисляемую по содержанию сборки и закрытого ключа создателя.
  - Имя строго именованной сборки включает открытый ключ создателя и номер версии.

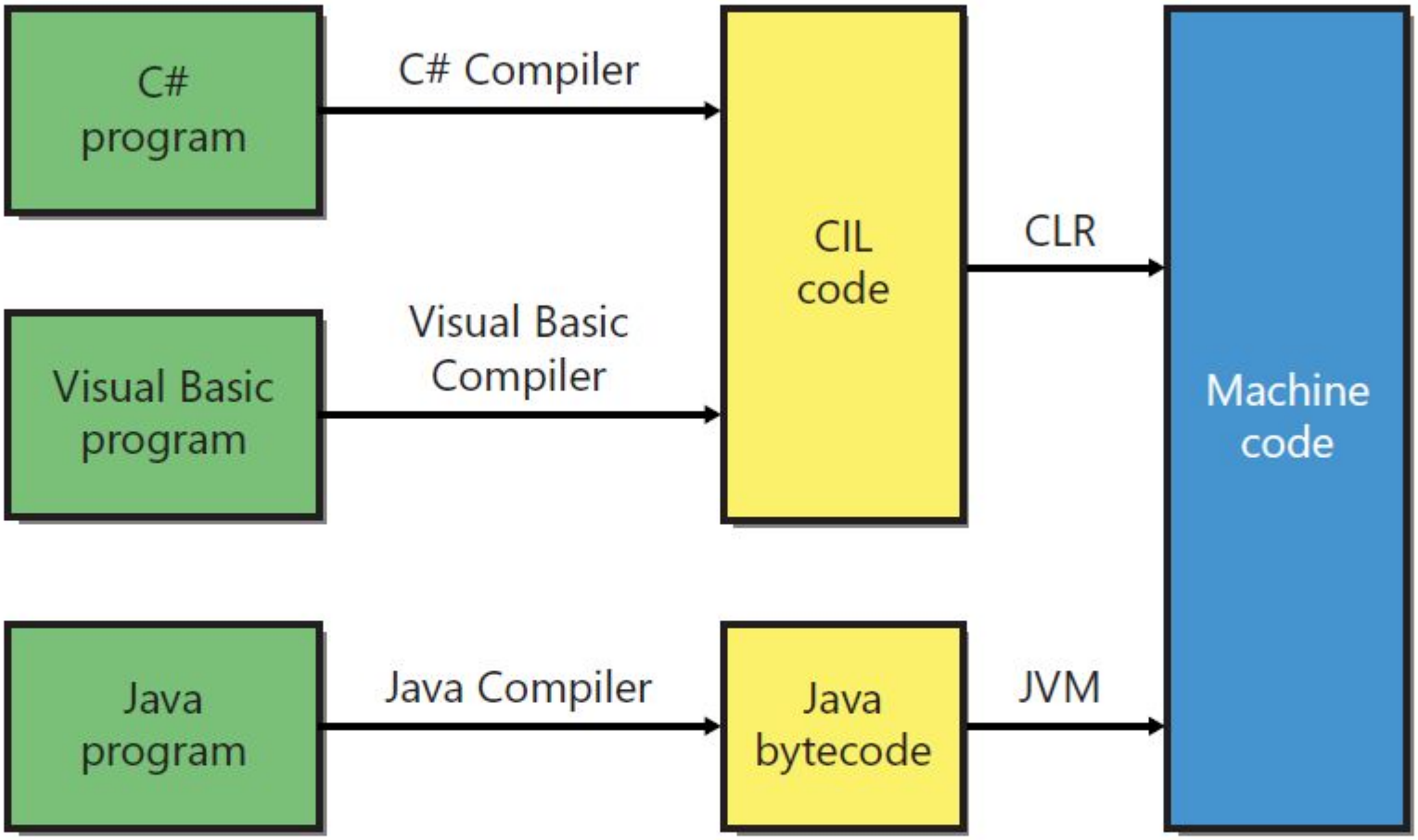
Включение ресурсов в сборку

- AL.exe, CSC.exe, VBC.exe
- Может быть любой ресурс (не только стандартный ресурс Windows)
- Ресурс может быть внедрённым (embedded) или отдельным
- Таблица ManifestResourceDef

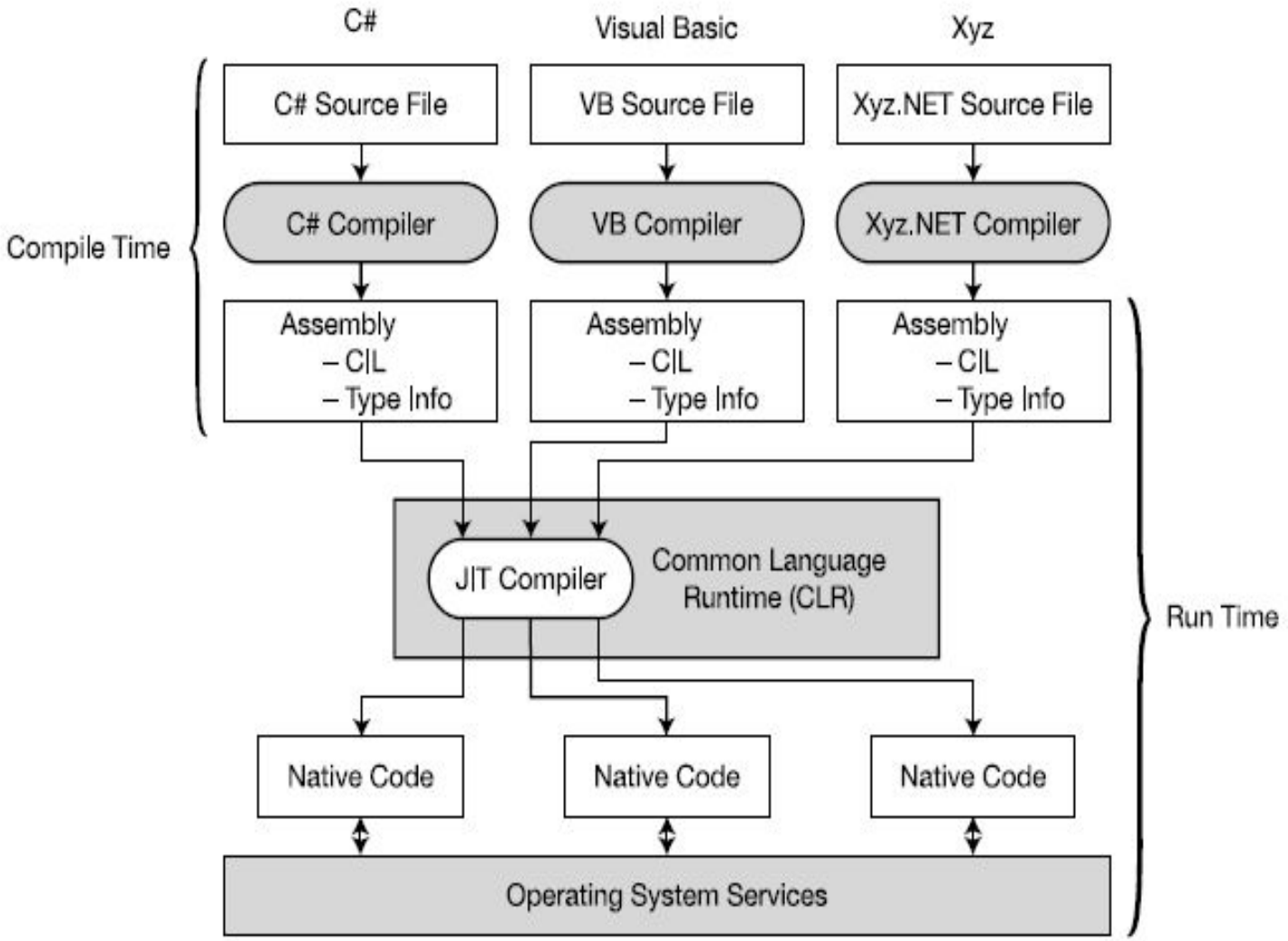
Модель разработки и выполнения программы



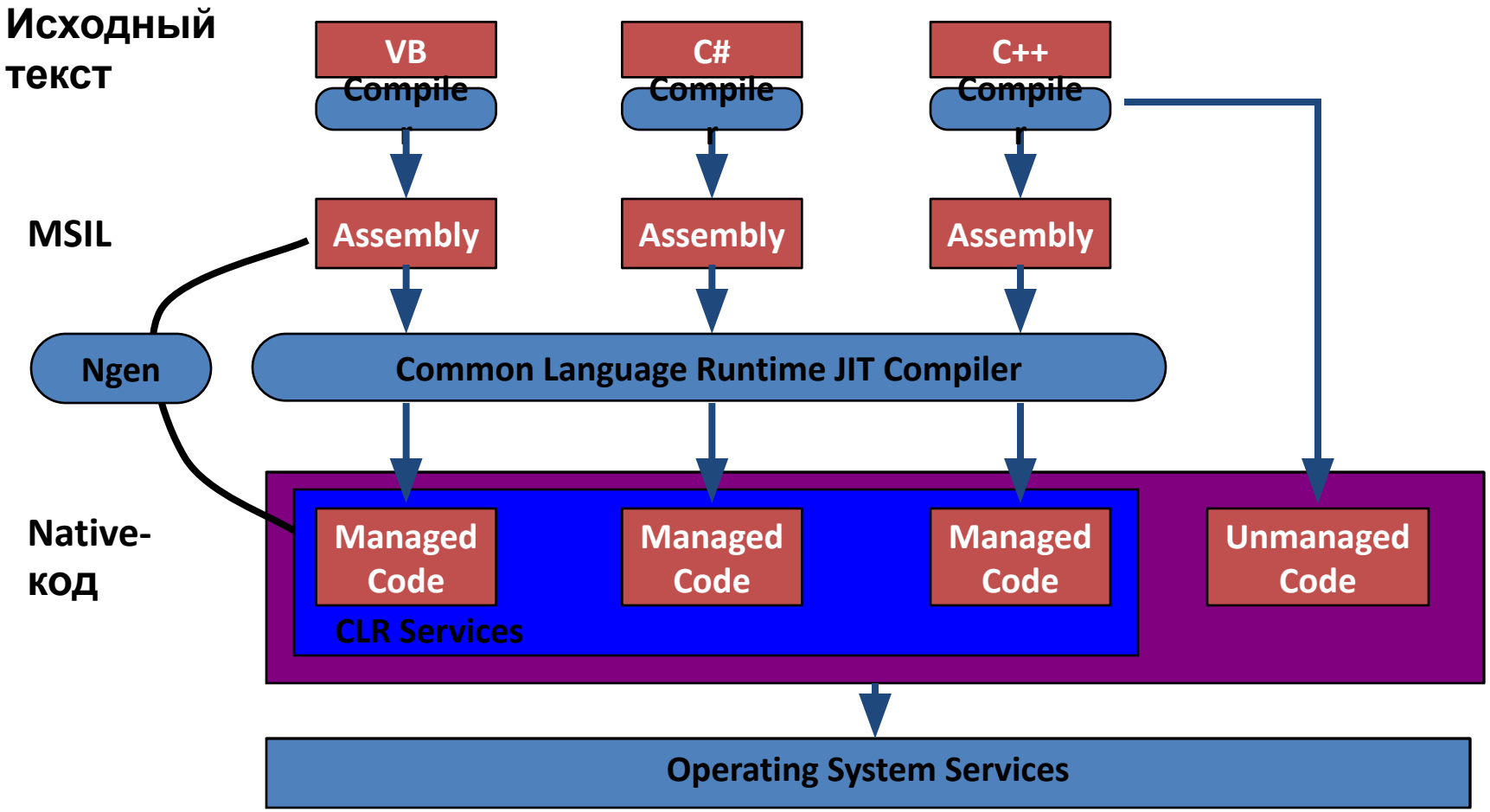
Languages such as Visual Basic, C#, and Java compile programs into intermediate languages before eventually converting them into machine code.



Создание и выполнение управляемых программ



Исполнение в .NET



Вызов компилятора

- csc.exe progr.cs
  /**r**eference:System.Drawing.dll,System.Windows.Forms.dll
  /**t**arget:exe /out:myprg.exe

- По умолчанию подключается модуль
  - *mscorlib.dll*

- /**r**eference:<подключаемые библиотеки>
- /**t**arget:<тип результата>
  - exe – консольное приложение
  - winexe – GUI приложение
  - library – библиотека классов (dll)
  - module – управляемый модуль
- /**out**: <имя полученного файла>

Создание сборки из управляемых модулей с помощью Assembly Linker

csc /t:module a.cs

csc /t:module b.cs

- создание управляемого модуля

AL.exe /target:library /out:lib.dll a.netmodule b.netmodule

- создание сборки

csc /t:exe /r:lib.dll demo.cs

Глобальный кэш сборок
Global assembly cache (GAC)

- Обычные сборки хранятся в том файле, где они используются
- В глобальном кэше (хранилище) сборок хранятся сборки, которые могут использоваться разными приложениями

Общий промежуточный язык

Microsoft Intermediate Language (MSIL)

- MCIL это ассемблерный язык виртуальной машины. Однако реально система команд этой машины переводится в исполняемый код конкретного процессора перед исполнением (так называемая компиляция времени исполнения)
- При этом выполняется довольно сложный анализ типов программы и проверки условий корректности кода

Характеристики MSIL

- Псевдоассемблер – определяет набор команд виртуального процессора (примерно 100 команд)
- Использует стековую модель выполнения (сперва значения загружаются в стек, вызывается команда операции, а затем результаты сохраняются в памяти)
- При запуске программы CLR компилирует с CIL в машинные коды
- Утилита ildasm.exe - дизассеблер

Трансляция в MSIL

Исходный текст на C#

```
using System;

class Fib // числа Фибоначчи
{
  public static void Main (String [] args)
    {
      int a = 1, b = 1;
      for (int i = 1; i != 10; ++ i)
        {
          Console.WriteLine (a);
          int c = a + b;
          a = b; b = c;
        }
    }
}
```

Трансляция в MSIL

Сгенерированный код (начало)

```
// объявление имени assembly
.assembly fib as "fib" {
// здесь могут быть параметры assembly
}
.class public Fib
{
  .method public static void Main ()
  {
    .entrypoint // означает начало assembly

    // декларация локальных переменных:
    .locals (int32 a, int32 b)
       ldc.i4.1  // загрузка константы 1
       stloc   a // сохранение 1 в a (a = 1)
       ldc.i4.1
       stloc   b // аналогично: b = 1
       ldc.i4.1  // загрузка 1 на стек
                 // (счетчик цикла)
```

Трансляция в MSIL (2)

Сгенерированный код (окончание)

```
Loop:
        ldloc   a
        call    void System.Console::WriteLine(int32)
                                        // печать a
        ldloc   a // stack: 1 a
        ldloc   b // stack: 1 a b
        add       // stack: 1 (a+b)
        ldloc   b
        stloc   a // a = b
        stloc   b // b = (a+b)
        ldc.i4.1
        add     // инкремент счетчика
        dup
        ldc.i4.s   10
        bne.un.s   Loop // сравнение и переход
                        // на следующую итерацию
        pop  // удаление счетчика цикла со стека
        ret
   }
}
```

Достоинство MSIL

- Многоплатформенность
- Интеграция языков программирования
- Возможность отладки многоязыковых приложений
- Единая модель обработки ошибок

Ассемблер и дизассемблер MSIL

- Ассемблер ILAsm.exe (входит в .NET Framework)
- Дизассемблер ILDasm.exe (не входит в .NET Framework, но входит в VS.NET)

MSIL и безопасность

- При компиляции IL в команды процессора выполняется верификация (проверка кода на безопасность)
- Верификация основывается на метаданных
- При обнаружении небезопасного кода возбуждается исключение (System.Security.VerificationException)
- Не исполняется для небезопасного кода (например, помеченного с помощью ключевого слова unsafe в C#)

Инструменты программирования

включают все, что необходимо для кодирования и отладки:

- Согласованные с .Net компиляторы (например, C#, VB, JScript, и управляемый (managed) C++, а также компиляторы, разработанные другими компаниями).
- Отладчики (debugger).
- Серверные компоненты (дополнения), такие как обработчики ASP.NET страниц.
- Интегрированную среду разработки Visual Studio .Net (или другие версии среды разработки).

Выполнение сборки в процессе ОС домены приложений

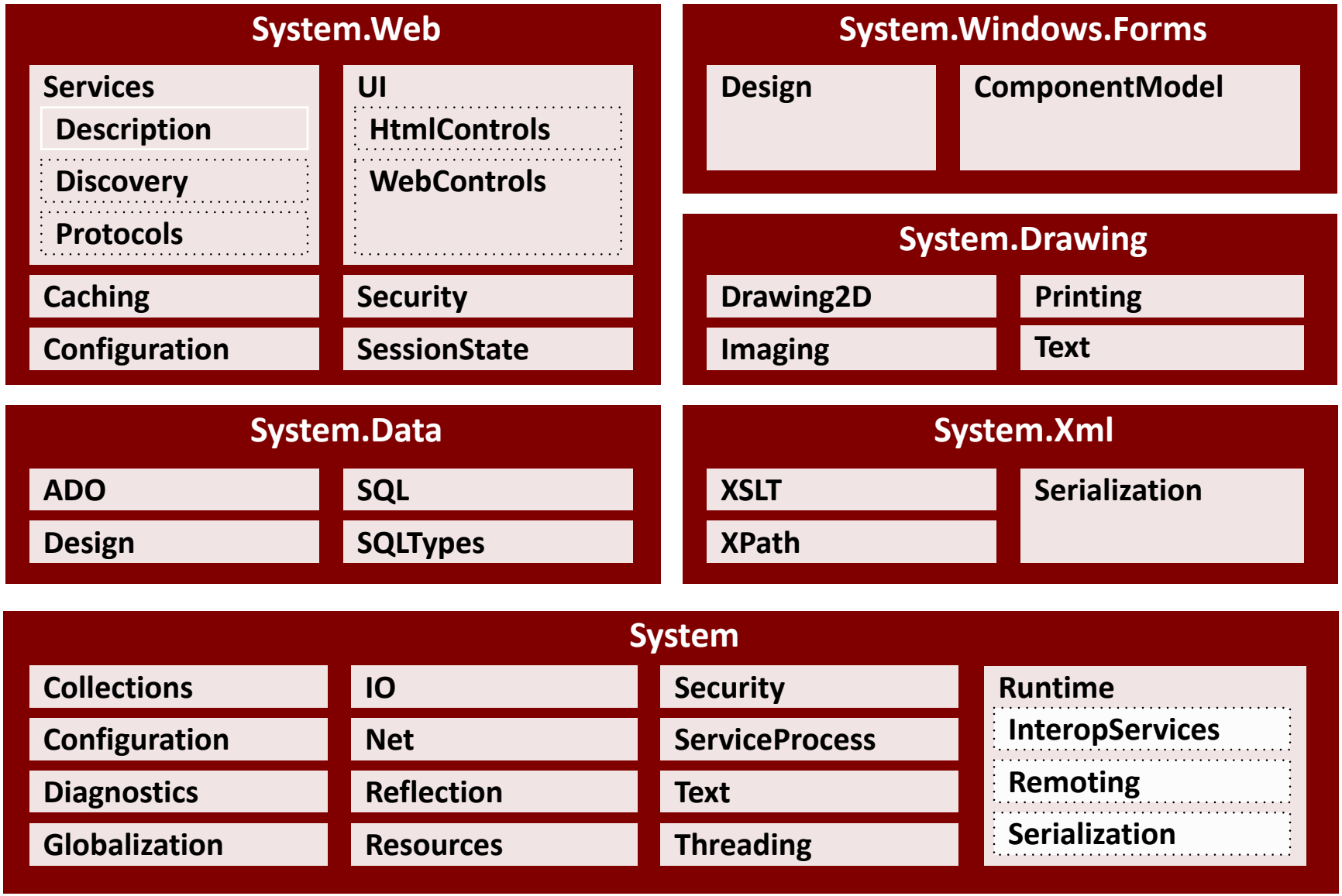
- Application domain - концепция для совместного использования и изоляции приложений
- Накладные расходы меньше, чем при создании нового процесса
- AppDomains создаются в рамках “CLR-хоста” (одного запуска CLR), исполняющего .NET-приложения
- Иерархия: Процесс ОС -> CLR-хост -> AppDomains
- AppDomain создается для каждого .NET-приложения (по умолчанию – DefaultDomain)
- Явные ссылки между AppDomains запрещены
- Связь между AppDomains требует прокси-классов и сериализации (обеспечиваемых .NET Remoting API)

Библиотека классов .NET Framework (FCL)

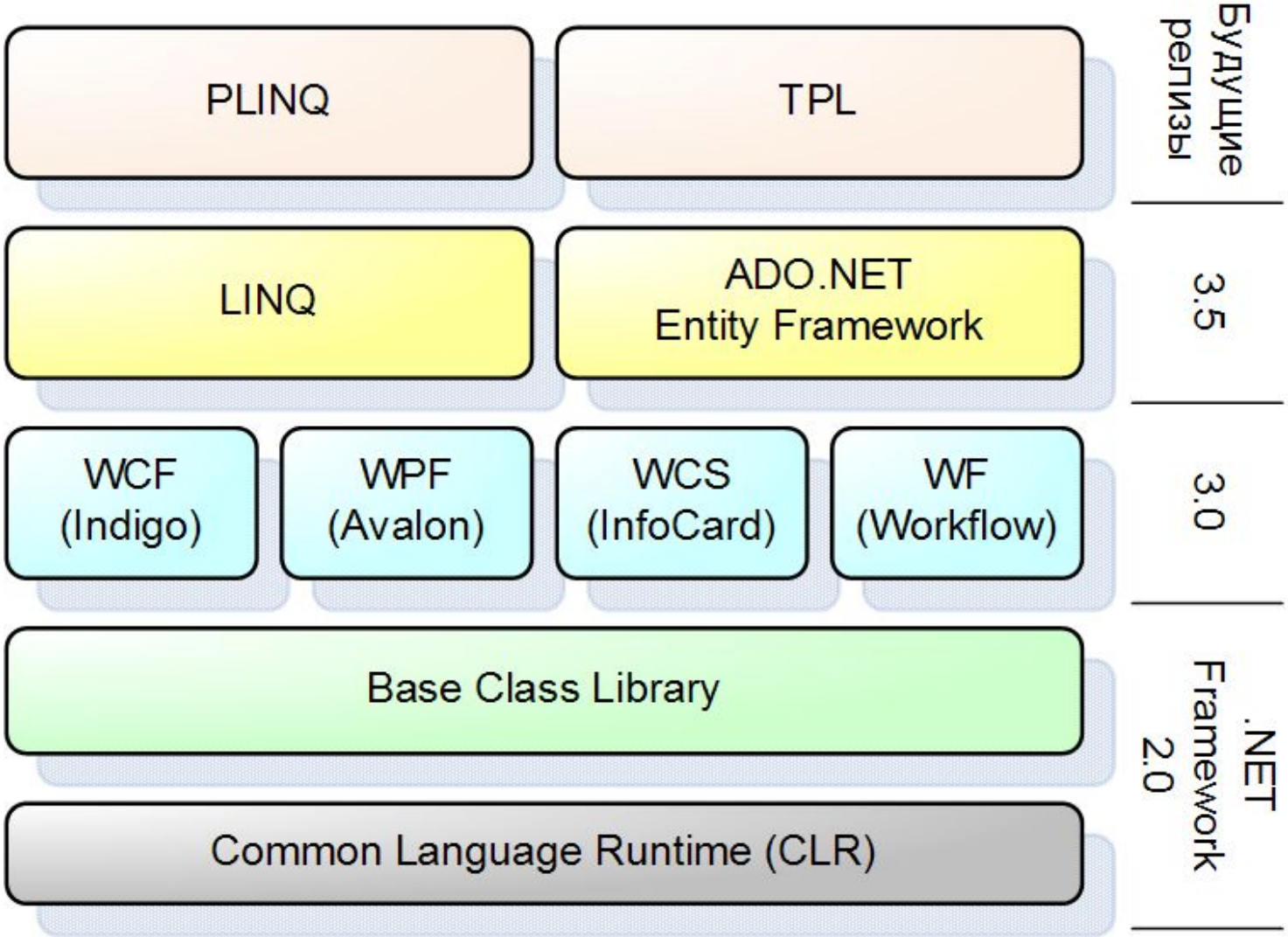
Основные сведения о FCL

- В начальной версии было более 13 000 типов (классы, интерфейсы, перечисления и делегаты)
- Некоторые классы включают описание до 100 методов.
- В библиотеке описаны 184000 методов.
- В библиотеке описаны 2800 методов вызова функций Microsoft Win32 API
- Все языки программирования используют одни и те же типы.
- Библиотека разделена на иерархическое пространство имен (около 100)
- Физически классы размещаются в DLL.
- Классы одного и того же пространства имен могут находиться в разных DLL

Пространства имен FCL
(FCL Namespaces)



Стек технологий .NET Framework



Организация библиотеки FCL

- Размещены в наборе библиотек – dll (Dynamic Link Library)
- В разных библиотеках включены разные пространства имен – namespaces
- Объекты одного пространства имен могут включаться в разные библиотеки
- В пространствах имен включены классы, структуры, интерфейсы, перечисления, делегаты.

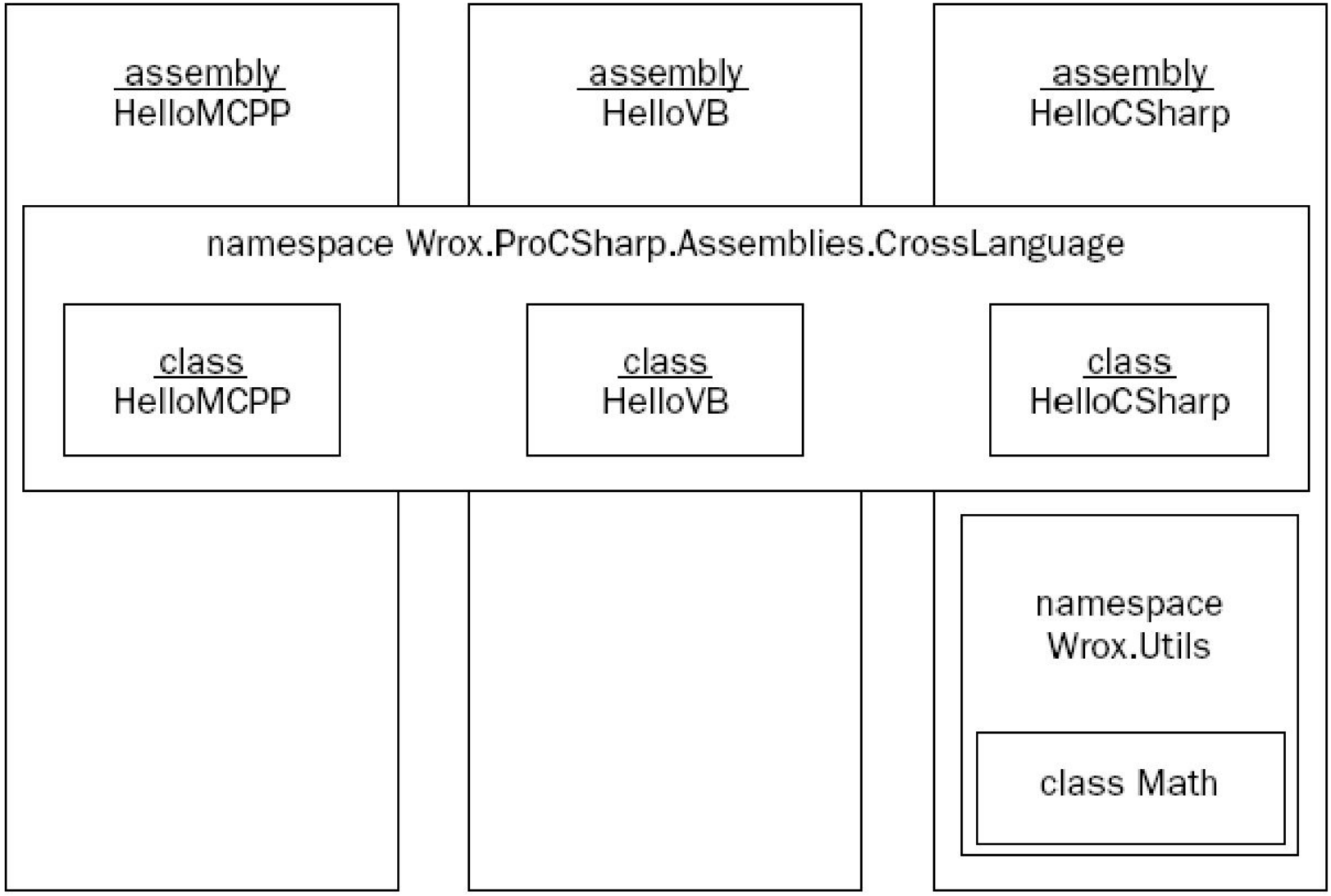
Пространство имен – name space

- Разделение объектов по иерархически организованным группам (категориям).
- *Например:*
  - *Все классы связанные с работой с файловой системой размещены в пространстве имен* System.IO
  - Все классы работающие с БД Microsoft SQL Server размещены в пространстве имен System.Data.SqlClient.
- Используется иерархическое пространство имен

<имяПространства>.<имяТипа>.<имяПодтипа>.<имяСобственное>

  - Вложенность нескольких имен (как почтовый адрес)
  - Значительно понижается вероятность совпадения имен классов разработанных разными компаниями
- Для описания связей между классами (классы близкие по функциональности включены в одно пространство)
- Пространство имен включает - классы (Class); интерфейсы (Interface); перечисления (Enum); делегаты (сигнатуры классов, Delegate); другие пространства имен.
- В одном DLL модуле могут содержаться элементы из разных пространств имен.

Связь сборок и пространств имен



Библиотек классов платформы .Net Framework Class Library (FCL)

- В состав FCL входит Base Class Library (BCL) - это часть FCL, которая поддерживает базовую функциональность программ.
- BCL включает классы пространства имен
  - System,
  - System.CodeDom,
  - System.Collections,
  - System.Diagnostics,
  - System.Globalization,
  - System.IO,
  - System.Resources,
  - System.Text,
  - System.Text.RegularExpressions.

Основные пространства имен FCL

- System – общие базовые типы
- System.VisualBasic – базовые типы для VBasic
- System.Drawing – классы для рисования
- System.Windows.Forms – классы для приложений с графическим интерфейсом
- System.Data – классы для работы с данными в БД
- System.Web – классы для ASP.NET и Web-форм
- System.Net – классы для работы с сетевыми протоколами
- System.Web.Services - классы для разработки Web сервисов
- System.Web.UI – основные классы используемые ASP.Net