

Системное программное обеспечение

Саранча Сергей Николаевич, к.т.
н., доцент каф ЭВМ ХНУРЭ
softpro@kture.kharkov.ua
702-13-54, ауд 37-з

Структура курса

2 семестра

- Первый семестр: современные технологии проектирования программного обеспечения
 - Технология .Net
 - Язык C#
 - Windows Forms и Windows Presentation Foundation
- Второй семестр: принципы построения и функционирования современных операционных систем
 - Процессы и потоки, алгоритмы планирования и синхронизации процессов
 - Управление памятью
 - Файловые системы и управление файлами
 - Системные ресурсы и внешние устройства
 - **Курсовой проект !!!**



Лекция 1 - Платформа .NET и ее особенности

Содержание лекции

1. .NET как концепция
2. .NET как вычислительная модель
3. .NET как технологическая платформа
4. .NET как инструментальное средство
5. Common Language Runtime и .NET Framework
6. Система типов Common Type System в .NET
7. Преимущества и недостатки .NET
8. Библиография

Что такое .NET ?

.NET включает следующие основные аспекты :

1. Идеология проектирования и реализации программного обеспечения
2. Модель эффективной поддержки жизненного цикла прикладных систем
3. Унифицированная, интегрированная технологическая платформа
4. Современный, удобный в использовании, безопасный инструментарий для создания, размещения и поддержки программного обеспечения

.NET как идеология (vision)

- Легкость развертывания приложений в глобальной среде Интернет
- Экономичная разработка программного обеспечения
- «Бесшовная», гибкая интеграция программных продуктов и аппаратных ресурсов
- Предоставление программного обеспечения как сервиса
- Новый уровень безопасности и удобства использования

.NET как вычислительная модель

- Компонентный подход как развитие объектно-ориентированной модели
- Универсальная система типизации: «всякая сущность есть объект»; унификация данных и метаданных
- Строго иерархическая организация кода, пространств имен и классов
- Универсальный интерфейс .NET Framework (включая поддержку различных подходов к программированию)
- Высокая вариативность экземпляров реализации (в частности, на основе веб-сервисов)

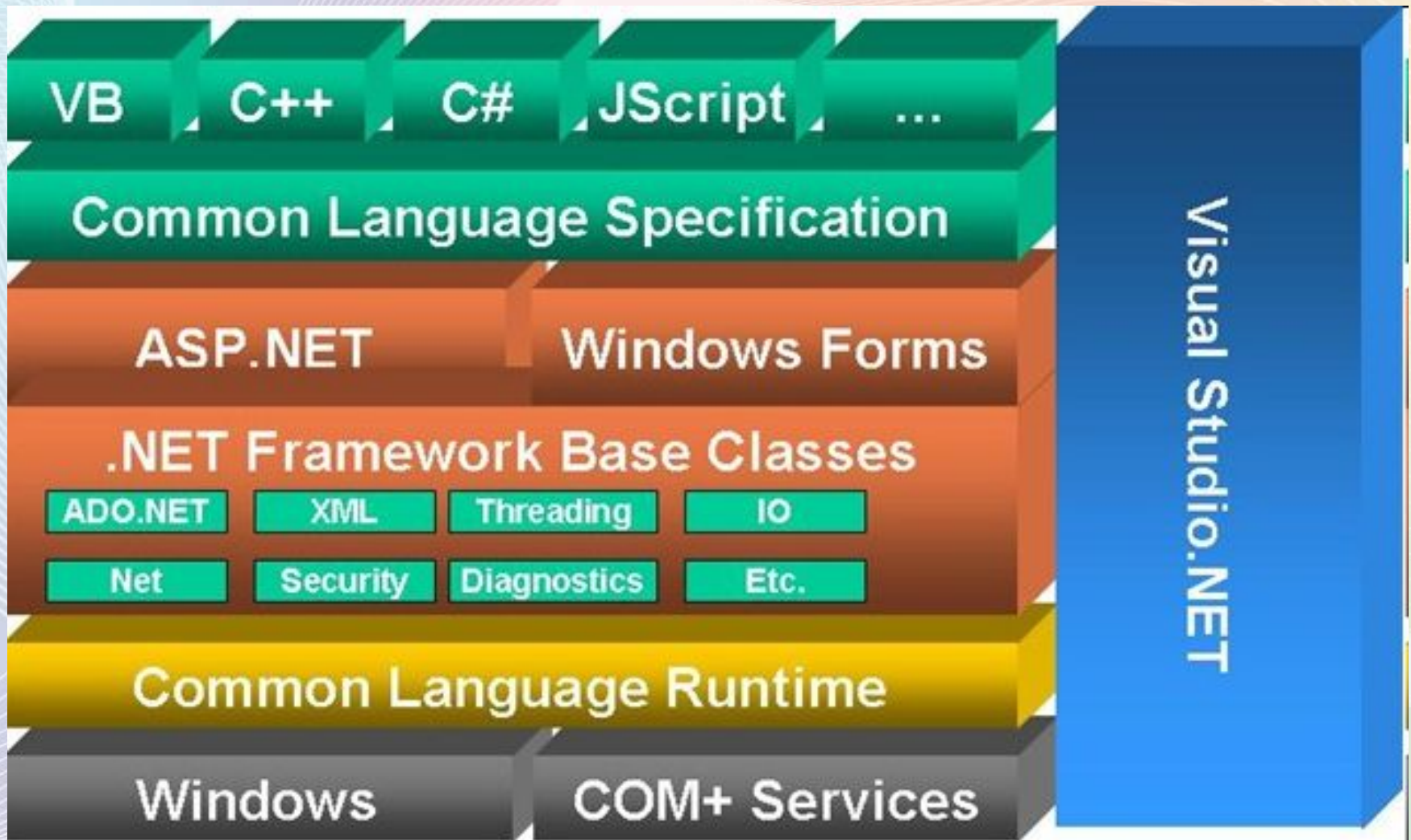
.NET как технологическая платформа

- Многоязыковая поддержка
- Использование технологии веб-сервисов для обеспечения интероперабельности и масштабируемости в глобальной сетевой среде
- Унификация доступа к библиотекам API-интерфейса независимо от языка и программной модели
- Соответствие современным технологическим стандартам

.NET - универсальное инструментальное средство

- Поддержка многоязыковой среды CLR (Common Language Runtime)
- Возможность создавать компоненты проекта в единой среде на наиболее подходящем языке программирования
- Доступность всех средств .NET для каждого из широкого спектра языков программирования
- Сервисные возможности для разработчиков, (отладка, анализ кода, ...) одинаковы для всех языков
- Возможность облегченной самостоятельной разработки транслятора для любого языка программирования (Microsoft – VB, C#, ... другие – APL, COBOL, Eiffel, Fortran, Haskell, SML, Perl, Python, Scheme, Smalltalk, ...)

Архитектурная схема .NET Framework и Visual Studio.NET



Что такое CLI?

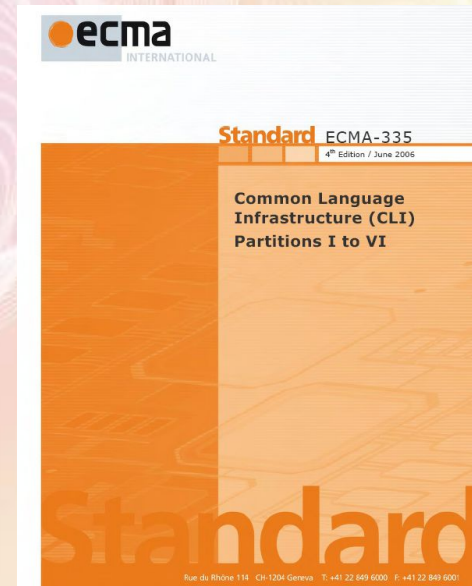
- CLI это открытая спецификация разработанная фирмой Microsoft, которая описывает код исполняемой программы и среду выполнения. Спецификация подразумевает среду разрешающую нескольким языкам высокого уровня быть использованными на разных компьютерных платформах без переписи под специфику архитектур.

Основная идея

- Основная идея состоит в том, чтобы был многоязыковой стандарт позволяющий разработчикам использовать для решения задачи использовать тот язык, который лучше всего для этого подходит.
- Более того, модули, написанные таким образом, должны не только корректно работать вместе, но и могли выполняться в любой операционной системе без переписывания или перекомпилирования.

Стандарты

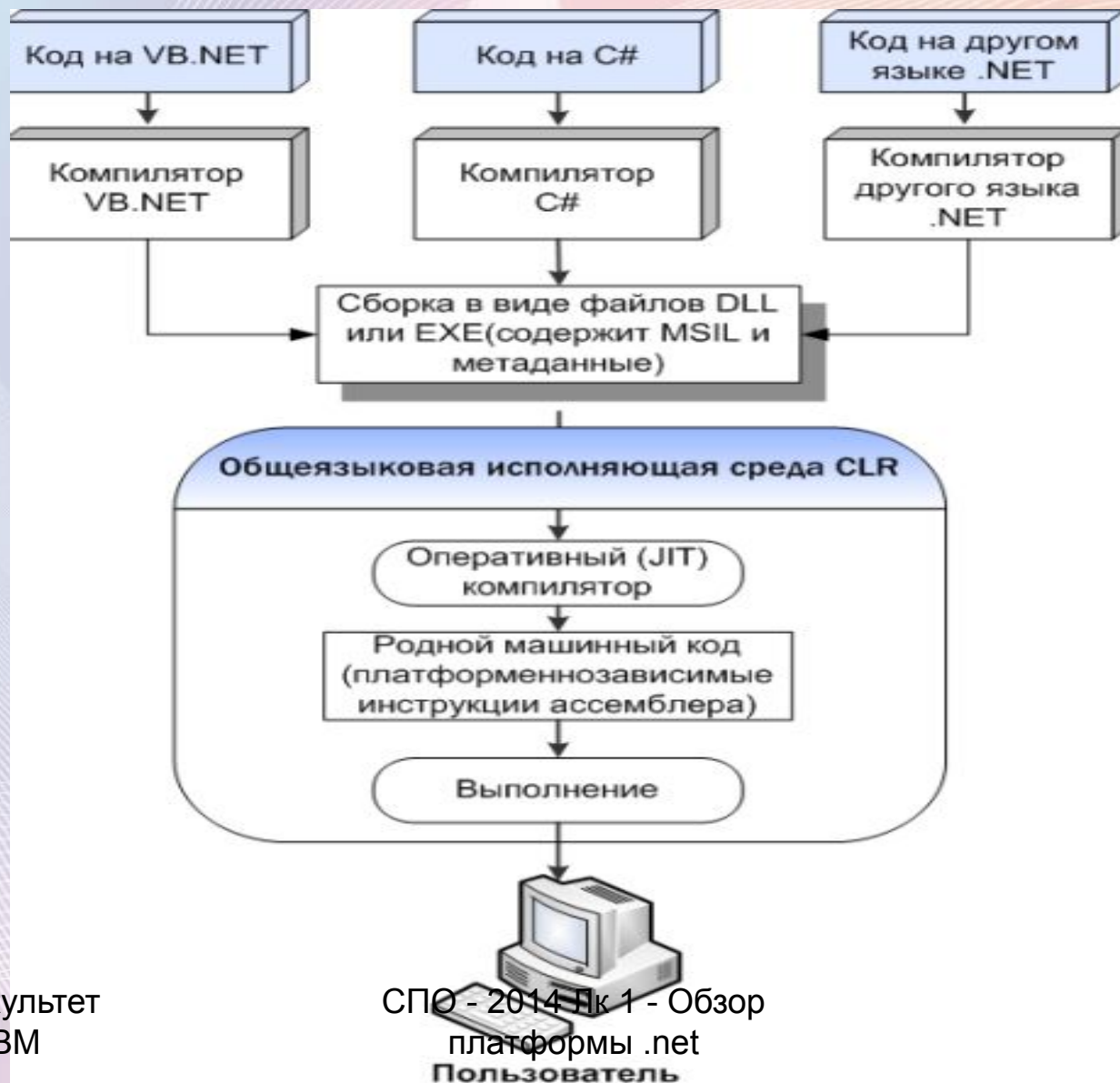
- ISO: ISO/IEC 23271:2006, Information Technology— Common Language Infrastructure (CLI).
- ECMA: Standard ECMA-335, 4th Edition, June 2006.



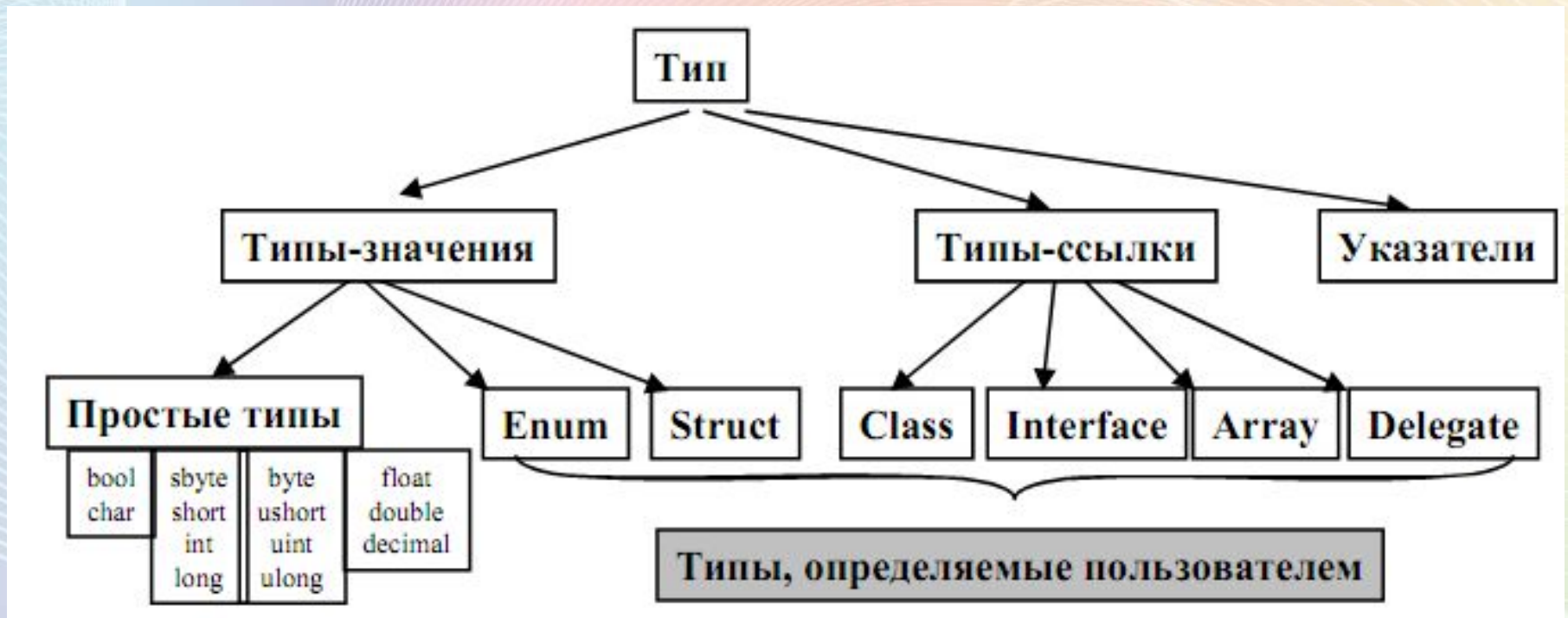
Аспекты CLI

- CLI, среди прочих вещей, описывает следующие 4 аспекта:
 - Common Type System (CTS)
 - Metadata
 - Common Language Specification (CLS)
 - Common Language Runtime (CLR)
- Все совместимые языки компилируются в Microsoft Intermediate Language (MSIL). Когда код будет запущен, платформенно-зависимая среда исполнения докомпилирует промежуточную сборку в машинный код

Схема компиляции в .net framework



Универсальная система типизации (UTS)



Common Language Infrastructure

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static int Add(int a, int b)
        {
            return a + b;
        }
        static void Main(string[] args)
        {
            Console.WriteLine(Add(2, 3));
            Console.ReadKey();
        }
    }
}
```

```
Module Module1
Public Function Add(ByVal x As Integer, ByVal y As Integer)
    Return x + y
End Function

Sub Main()
    Console.WriteLine(Add(2, 3))
    Console.ReadKey()
End Sub

End Module
```

Общая среда выполнения – идентичный результат компиляции

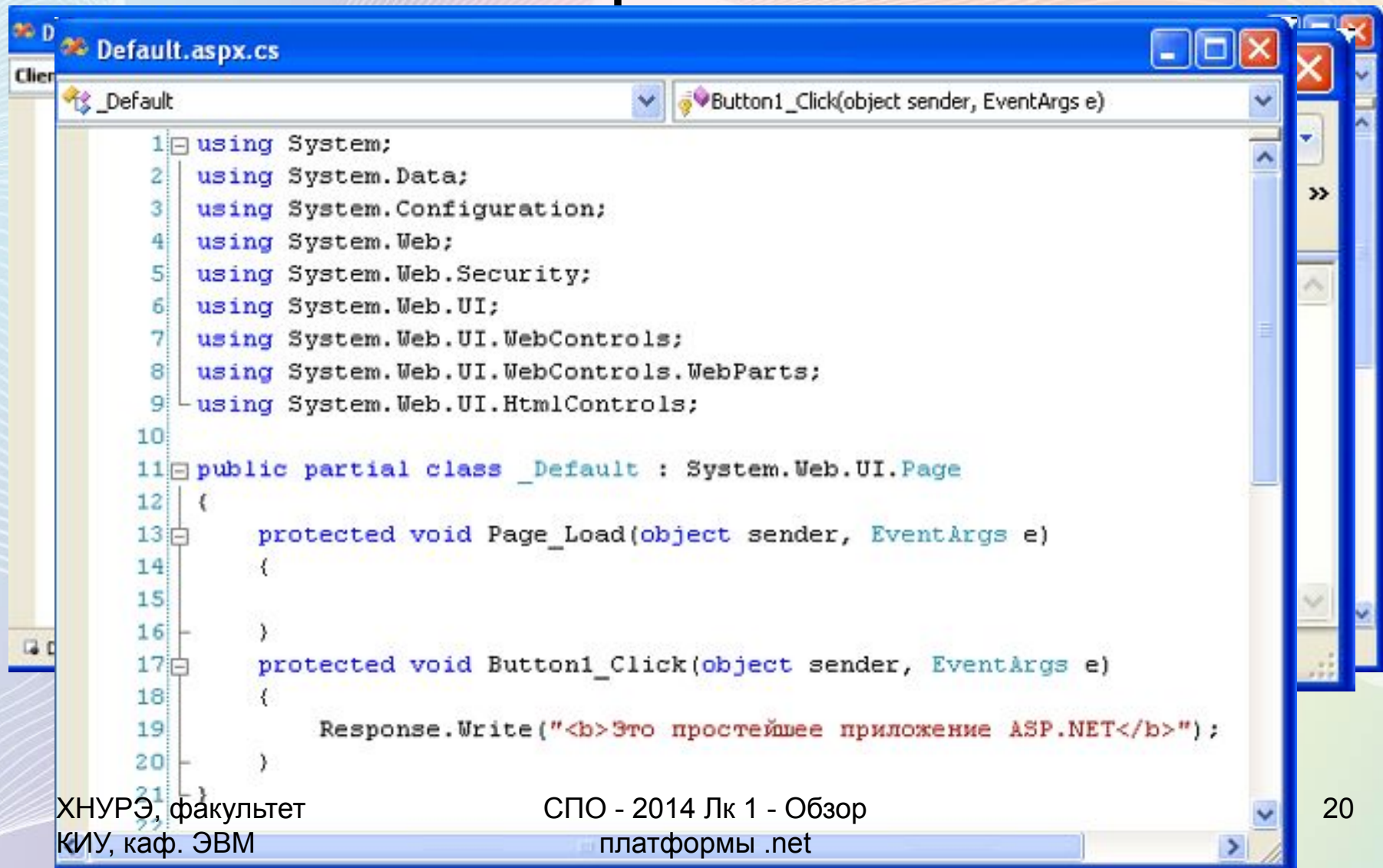
```
ConsoleApplication1.Program::Add : int32(int32,int32)
Найти Найти далее
.method private hidebysig static int32 Add(int32 a,
                                           int32 b) cil managed
{
    // Размер кода:          4 (0x4)
    .maxstack 8
    IL_0000: ldarg.0
    IL_0001: ldarg.1
    IL_0002: add
    IL_0003: ret
} // end of method Program::Add
```

```
VBSample.Module1::Add : int32(int32,int32)
Найти Найти далее
.method public static int32 Add(int32 x,
                                int32 y) cil managed
{
    // Размер кода:          4 (0x4)
    .maxstack 2
    .locals init ([0] int32 Add)
    IL_0000: ldarg.0
    IL_0001: ldarg.1
    IL_0002: add.ovf
    IL_0003: ret
} // end of method Module1::Add
```

Общая среда выполнения

- Единая программная модель
- Упрощенная модель программирования
- Отсутствие проблем с версиями
- Упрощенная инсталляция и удаление
- Работа на разных платформах
- Упрощенная интеграция языков программирования и повторное использование кода
- Автоматическое управление памятью (сборка мусора)
- Единый принцип обработки сбоев

.net как платформа построения сетевых приложений



```
1 using System;
2 using System.Data;
3 using System.Configuration;
4 using System.Web;
5 using System.Web.Security;
6 using System.Web.UI;
7 using System.Web.UI.WebControls;
8 using System.Web.UI.WebControls.WebParts;
9 using System.Web.UI.HtmlControls;
10
11 public partial class _Default : System.Web.UI.Page
12 {
13     protected void Page_Load(object sender, EventArgs e)
14     {
15     }
16 }
17 protected void Button1_Click(object sender, EventArgs e)
18 {
19     Response.Write("<b>Это простейшее приложение ASP.NET</b>");
20 }
21 }
```

.net framework

Преимущества

- Простота интеграции «разноязыких» проектов
- Возможности повторного использования кода
- Управляемая память
- Улучшенная безопасность кода

Недостатки

- Необходимость установки .net framework
- Низкая скорость запуска .net - приложений на XP
- Отсутствие CLR для других ОС (для Unix/Linux – проект Mono)

Литература

