Платформа .NET
и ее применение для ООП

Комментарий к слайду

В данной лекции будут рассмотрены вопросы, относящиеся к идеологии, технологии и обзору практических возможностей создания объектно-ориентированных программных систем на основе наиболее современного подхода к проектированию и реализации программного обеспечения, известного под названием Microsoft .NET.

Современные языки программирования и .NET: II семестр
Лекция 2: Платформа .NET и ее применение для ООП

Содержание лекции

Комментарий к слайду

Коротко о содержании лекции.

В отличие от всех предшествующих подходов, компания Microsoft предлагает наиболее развитое и комплексное решение для проектирования и реализации программного обеспечения.

В частности, в данной лекции будут рассмотрены такие аспекты .NET, как:

- идеология;
- вычислительная модель;
- технологическая платформа;
- инструментальное решение;
- безопасность;
- интеграция приложений;
- организация вычислительных сред CLR и . NET Framework;
- универсальная система типов в .NET, известная как CTS;
- поддержка веб-сервисов;
- компонентный подход к программированию и его связь с ООП.

В заключение будут проанализированы достоинства и недостатки .NET и сделаны необходимые выводы.

Наконец, для желающих глубже исследовать предмет будут представлены ссылки на важнейшие работы теоретического и практического плана по теме лекции.

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

Комментарий к слайду

Прежде всего, необходимо ответить на важный вопрос: что такое .NET? Несмотря на широкое освещение в прессе, ответить однозначно непросто, прежде всего по той причине, что ответ представляется многоаспектным.

Итак, можно сказать, что .NET – это подход к проектированию и реализации программного обеспечения, включающий по меньшей мере четыре следующих компонента:

1) идеология проектирования и реализации программного обеспечения;
2) модель эффективной поддержки жизненного цикла прикладных систем;
3) унифицированная, интегрированная технологическая платформа для программирования;
4) современный, удобный в использовании, безопасный инструментарий для создания, размещения и поддержки программного обеспечения.

Остановимся подробнее на каждом из этих существенных аспектов.

.NET как идеология (vision)

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

Комментарий к слайду

Прежде всего, постараемся сформировать понимание идеологии подхода Microsoft .NET.

Самой корпорацией-разработчиком сформулированы приблизительно следующие важнейшие аспекты видения (vision) идеологии .NET:

1) легкость развертывания приложений в глобальной среде Интернет;
2) экономичная разработка программного обеспечения;
3) «бесшовная», гибкая интеграция программных продуктов и аппаратных ресурсов;
4) предоставление программного обеспечения как сервиса;
5) новый уровень безопасности и удобства использования.

Действительно, как мы увидим в ходе лекции, все аспекты видения .NET удалось реализовать на качественно новом уровне, обеспечив существенное продвижение вперед в направлении гибкости интеграции с программно-аппаратными ресурсами, безопасности и удобстве использования кода, а также снижении затрат на производство программного обеспечения.

.NET как вычислительная модель

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

Комментарий к слайду

Рассмотрим подробнее, как идеология .NET претворяется в практические вопросы проектирования программного обеспечения.

Корпорацией Microsoft предложен новаторский компонентно-ориентированный подход к проектированию, который является развитием объектно-ориентированного направления. Согласно этому подходу, интеграция объектов (возможно, гетерогенной природы), производится на основе интерфейсов, представляющих эти объекты (или фрагменты программ) как независимые компоненты. Такой подход существенно облегчает написание и взаимодействие программных «молекул»-компонент в гетерогенной среде проектирования и реализации. Стандартизируется хранение и повторное использование компонент программного проекта в условиях распределенной сетевой среды вычислений, где различные компьютеры и пользователи обмениваются информацией, например, взаимодействуя в рамках исследовательского или бизнес-проекта.

Существенным преимуществом является и возможность практической реализации принципа «всякая сущность является объектом гетерогенной программной среды». Во многом это стало возможным благодаря усовершенствованной, обобщенной системе типизации Common Type System, или CTS, которая будет подробнее рассмотрена в одной из следующих лекций.

Строгая иерархичность организации пространств для типов, классов и имен сущностей программы позволяет стандартизировать и унифицировать реализацию.

Новый подход к интеграции компонент приложений в среде вычислений Интернет (или так называемые веб-сервисы), дает возможность ускоренного создания приложений для глобальной аудитории пользователей.

Универсальный интерфейс .NET Framework обеспечивает интегрированное проектирование и реализацию компонент приложений, разработанных согласно различным подходам к программированию.

.NET как технологическая платформа

1. Многоязыковая поддержка (десятки языков программирования)
2. Использование технологии веб-сервисов для обеспечения интероперабельности и масштабируемости в глобальной сетевой среде
3. Унификация доступа к библиотекам API-интерфейса независимо от языка и программной модели
4. Соответствие современным технологическим стандартам

Комментарий к слайду

Говоря о .NET как о технологической платформе, нельзя не отметить тот факт, что она обеспечивает одновременную поддержку проектирования и реализации программного обеспечения с использованием различных языков программирования. При этом поддерживаются десятки языков программирования, начиная от самых первых (в частности, COBOL и FORTRAN) и заканчивая самыми современными (например, C# и Visual Basic). Ранние языки программирования до сих пор активно используются, в частности, для обеспечения совместимости с ранее созданными приложениями, критичными для бизнеса (скажем, COBOL весьма широко использовался для создания прикладных программ, поддерживающих финансовую деятельность).

Применение технологии веб-сервисов – это не просто дань моде на Интернет, а реальная (и, пожалуй, наиболее приемлемая практически возможность) обеспечения масштабируемости и интероперабельности приложений. Под масштабируемостью понимают возможность плавного роста времени ответа программной системы на запрос с ростом числа одновременно работающих пользователей; в случае веб-сервисов масштабируемость реализуется посредством распределения вычислительных ресурсов между сервером, на котором выполняется прикладная программа (или хранятся данные) и компьютером пользователя.

Под интероперабельностью понимается возможность интегрированной обработки гетерогенных данных, поступающих от разнородных прикладных программ. Именно благодаря интероперабельности возможна унификация взаимодействия пользователей через приложение с операционной системой на основе специализированного интерфейса прикладных программ, или API-интерфейса (Application Programming Interface).

Немаловажно отметить и то обстоятельство, что новая технология .NET не только востребована мировой общественностью, но и официально признана, что отражено в соответствующих стандартах ECMA (European Computer Manufacturers Association).

.NET - универсальное инструментальное средство

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

Комментарий к слайду

Теперь рассмотрим инструментальные возможности .NET как средства проектирования и реализации программного обеспечения, т.е., собственно, программирования в широком смысле этого слова.

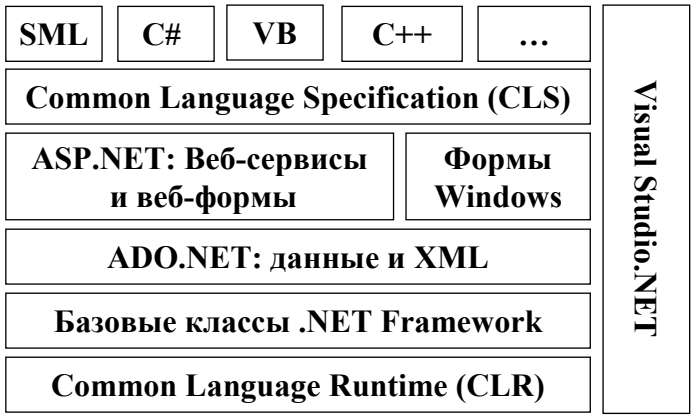
Прежде всего, необходимо отметить поддержку многоязыковой среды разработки приложений CLR (Common Language Runtime). Эта возможность появилась благодаря универсальному межъязыковому интерфейсу Common Language Infrastructure, или CLI, который поддерживает разработку программных компонент на различных языках программирования.

При этом несомненным преимуществом для программистов является то обстоятельство, что они могут разрабатывать (или дорабатывать) программное обеспечение на наиболее подходящем языке программирования. Здесь следует учитывать характер задачи (скажем, рекурсия или символьная обработка прозрачнее и с меньшими трудозатратами реализуема на языке функционального программирования, а формализация структуры предметной области – на объектно-ориентированном языке). Кроме того, необходимо принимать во внимание опыт работы программистов в команде разработчиков и тот язык программирования, на котором изначально создавалась система.

Отметим еще два существенных обстоятельства. Во-первых, основные сервисные возможности для разработчиков, которые предоставляет среда .NET (отладка, анализ кода и т. д.) не зависят от конкретного языка программирования, и, следовательно, программистам нет необходимости заново постигать особенности среды разработки, если необходимо перейти с одного языка на другой. Во-вторых, несмотря на то, что еще не все языки программирования поддерживаются .NET, существует возможность самостоятельной разработки транслятора для любого языка программирования, причем его реализация не вызывает трудностей даже у программистов, практически не имеющих профессиональной подготовки в области разработки компиляторов.

Современные языки программирования и .NET: II семестр
Лекция 2: Платформа .NET и ее применение для ООП

Архитектурная схема .NET Framework и Visual Studio.NET



Комментарий к слайду

Кратко обсудим основные аспекты архитектурного решения Microsoft .NET Framework, отметив прежде всего то обстоятельство, что важную роль играет среда разработки Microsoft Visual Studio.NET, а первостепенное значение отводится среде выполнения программ – Common Language Runtime (CLR).Среда выполнения программ CLR реализует управление памятью, типами данных, межъязыковым взаимодействием, разворачиванием (deployment) приложений.

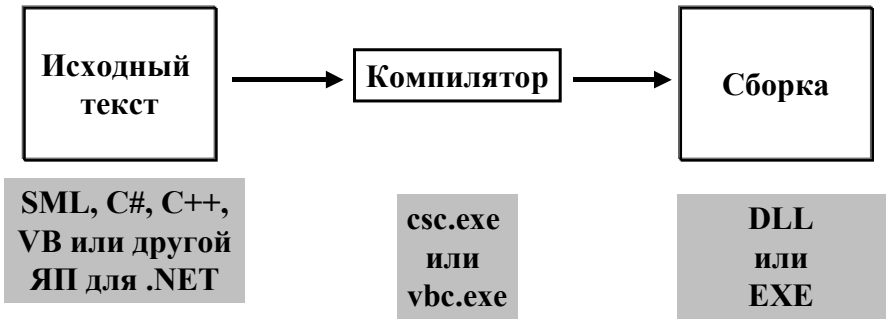
Существенным преимуществом конструктивного решения .NET является компонентно-ориентированный подход к проектированию и реализации программного обеспечения, который будет подробнее рассмотрен в ходе настоящей лекции. Суть подхода состоит в принципиальной возможности создания независимых составляющих программного обеспечения с унифицированной интерфейсной частью для многократного повторного и распределенного использования. При этом продуктивность решения обусловлена многоязычностью интегрируемых программных проектов (концепция .NET потенциально поддерживает произвольный язык программирования, в числе наиболее известных языков – C#, Visual Basic, C++ и др.)

В ходе компиляции программа на .NET-совместимом языке программирования трансформируется в соответствии с заранее заданной обобщенной спецификацией языка Common Type System (CTS). Система типов CTS полностью описывает все типы данных, поддерживаемые средой выполнения, определяет их взаимосвязи и хранит их отображения в систему типов .NET.

Под Common Language Specification (или CLS) понимается набор правил, определяющих подмножество обобщенных типов данных, в отношении которых гарантируется, что они безопасны при использовании во всех языках .NET.

Интерфейсы реализуются посредством форм Windows и ASP.NET для веб-приложений.

Схема компиляции в Common Language Runtime



Комментарий к слайду

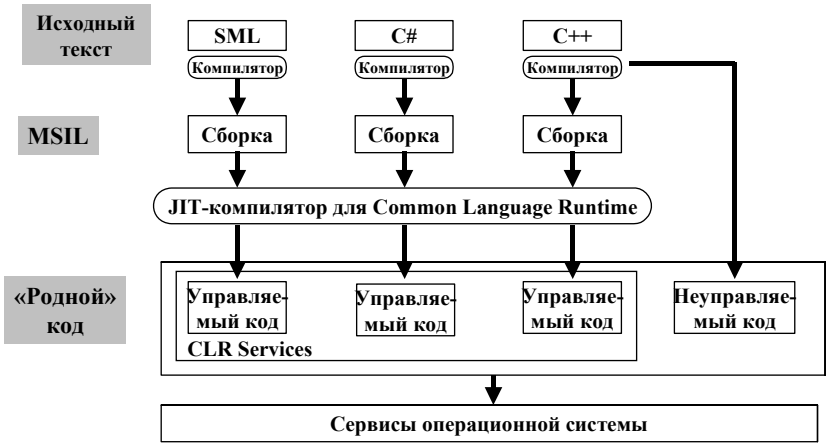
Как уже упоминалось, среда выполнения программ CLR реализует управление памятью, типами данных, межъязыковым взаимодействием, разворачиванием (deployment) приложений.

В ходе выполнения процедуры трансляции исходный текст программы (написанный на SML, C#, Visual Basic, C++ или любом из множества других языков программирования, который поддерживается .NET) преобразуется компилятором в так называемую сборку (assembly) и сохраняется в виде файла динамически присоединяемой библиотеки (Dynamically Linked Library, DLL) или исполняемого файла (Executable, EXE).

Естественно, что для каждого компилятора (будь то компилятор языка C#, csc.exe или Visual Basic, vbc.exe) средой времени выполнения производится необходимое отображение используемых типов в типы CTS, а программного кода – в код «абстрактной машины» .NET – MSIL (Microsoft Intermediate Language).

В итоге программный проект формируется в виде сборки – самодостаточного компонента для разворачивания, тиражирования и повторного использования. Сборка идентифицируется цифровой подписью автора и уникальным номером версии.

Схема выполнения CLR



Комментарий к слайду

Рассмотрим достаточно обобщенный пример трансляции многокомпонентного гетерогенного программного проекта под управлением Microsoft .NET.

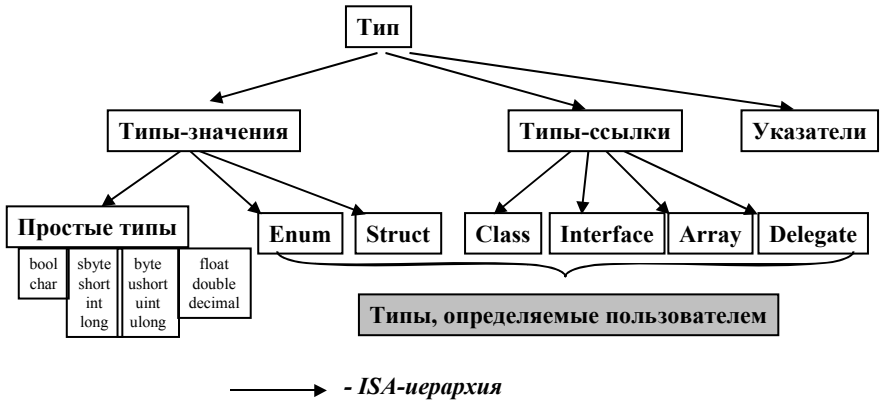
Предположим, что компоненты проекта написаны на трех языках программирования: уже знакомого нам языка SML, изучаемого языка C#, а также языка C++, который характеризуется возможностью написания потенциально небезопасного кода (в частности, динамического распределения памяти).

Исходные тексты компонент проекта транслируются соответственно компиляторами с языков SML, C# и C++ в унифицированный MSIL-код и сохраняются в файлах в виде сборок.

В ходе компоновки и выполнения программного проекта Just-In-Time (JIT) компилятор среды CLR производит выполнение проекта с ленивым (по мере необходимости) означиванием оттранслированного промежуточного кода сборок.

Существенно, что потенциально небезопасный код на языке C++ принципиально невыполним собственно JIT-компилятором, но исполняется посредством сервисов операционной системы. Ответственность за работоспособность программы и безопасность кода в этом случае лежит уже не на среде проектирования и разработки программного обеспечения .NET, а на программисте-разработчике.

Универсальная система типизации (UTS)



Комментарий к слайду

Существенным позитивным отличием Microsoft .NET от существующих аналогов на современном рынке программного обеспечения является универсальная система типизации.

В ходе компиляции программа на .NET-совместимом языке программирования трансформируется в соответствии с заранее заданной обобщенной спецификацией языка Common Type System (CTS). Система типов CTS полностью описывает все типы данных, поддерживаемые средой выполнения, определяет их взаимосвязи и хранит их отображения в систему типов .NET.

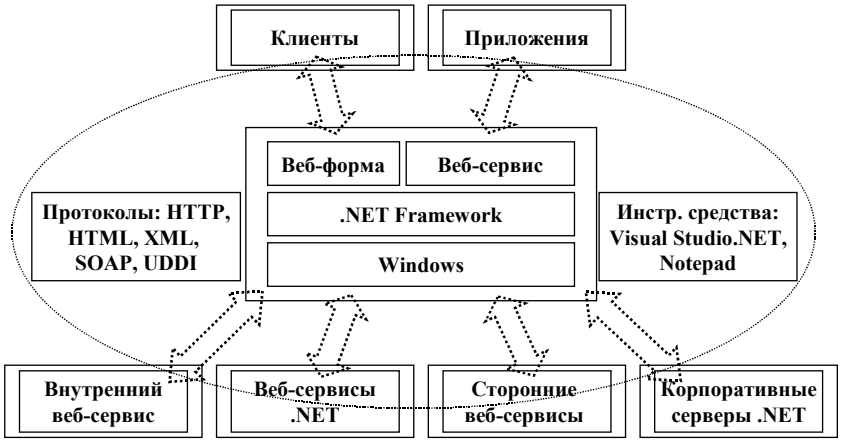
Система типизации Microsoft .NET представляет собой частично упорядоченное множество, которое на качественном уровне может пониматься как ISA-иерархия (ISA происходит от английских слов “is a”, которые означают «является одним из»).

Так, например, высказывание STUDENT ISA PERSON означает, что тип STUDENT является подтипом типа PERSON (здесь вполне уместна аналогия с множествами и вполне точна аналогия с доменами).

Таким образом, система типов Microsoft .NET образует иерархию с возрастанием общности снизу вверх (см. слайд), в которой явно выделяются две большие группы типов, а именно, типы-ссылки и типы-значения. Различие между последними определяется особенностями вызова в процедурах: по имени или по значению (call-by-name, CBN) и по ссылке (call-by-reference, CBR).

Заметим также, что система типизации Microsoft .NET помимо развитой иерархии предопределенных типов позволяет пользователю создавать собственные типы (как типы-ссылки, так и типы-значения) на основе уже существующих.

Веб-сервисы в .NET (1)



Комментарий к слайду

Изложение важнейших аспектов Microsoft .NET было бы не полным, если бы мы не упомянули о таком существенном архитектурном принципе как веб-сервисы.

Значение веб-сервисов заключается в распределении возможностей разработанных прикладных систем по каналам глобальной сети Интернет.

Заметим, что центральным блоком в схеме является .NET Framework, который можно рассматривать как библиотеку базовых объектов и операций над ними.

В качестве среды разработки прикладных систем целесообразно использовать Microsoft Visual Studio .NET, предоставляющей целый комплекс развитых средств создания, редактирования и отладки программного кода на различных языках программирования. В случае несложных задач можно ограничиться примитивными редакторами текста программ, подобных Notepad.

Интерфейсная часть прикладной программной системы в Интернет-архитектуре представлена так называемыми веб-формами, предназначенными для ввода и вывода данных в унифицированном формате.

В качестве языка реализации может использоваться язык гипертекстовой разметки HTML (HyperText Markup Language). Взаимодействие между клиентом и приложением в простейшем случае осуществляется с использованием традиционного Интернет-протокола передачи данных HTTP (HyperText Transfer Protocol).

Структурированные данные хранятся в формате XML (вариант HTML с более строгим синтаксисом).

Заметим, что технология веб-сервисов, реализованная Microsoft, допускает интеграцию с компонентами сторонних производителей.

Веб-сервисы в .NET (2)

1. Программируемые компоненты приложений, доступные посредством стандартных Интернет-протоколов
2. Центральная часть архитектуры .NET
3. Распределяют функциональность по глобальной сети
4. Строятся на существующих и развивающихся стандартах: HTTP, XML, SOAP, UDDI, WSDL и др.

Комментарий к слайду

Попытаемся сформулировать определения понятия «веб-сервис» (или, иначе, «веб-служба»).

Под веб-сервисами обычно понимают программируемые компоненты прикладных программных систем, которые доступны для клиента (пользователя) посредством стандартных протоколов, применяемых для работы в Интернет-среде.

Как уже упоминалось ранее, именно веб-сервисы являются одной из важнейших составляющих идеологии .NET и центральной частью данной архитектуры, поскольку предназначены для реализации декларируемого Microsoft основополагающего принципа «программное обеспечение как сервис».

Смысл использования веб-сервисов состоит в возможности распределения функциональных возможностей разработанных прикладных систем по глобальной сети.

Для реализации этой задачи веб-сервисы надстраиваются на прошедших долговременную апробацию, традиционных, а также на развивающихся стандартах взаимодействия приложений в Интернет, а именно:

1) HTTP – стандартный протокол обмена гипертекстовыми документами в Интернет с возможностью передачи данных посредством веб-форм;
2) XML – формат хранения структурированных данных с возможностью обмена ими по Интернет-каналам;
3) SOAP – стандартный протокол взаимодействия компонент (глобально) распределенного приложения (Simple Object Access Protocol);
4) UDDI – стандарт интеграции приложений (Universal Description, Discovery and Integration);
5) WSDL – универсальный язык описания веб-сервисов (Web Service Description Language);

а также целом ряде других менее употребительных протоколов.

Компонентное программирование в .NET (1)

- Компоненты – это:
  - независимые повторно используемые и тиражируемые модули;
  - в целом более крупные, чем объект (объекты – конструкции уровня ЯП);
  - могут содержать множественные классы;
  - независимы от языка реализации.
- В общем случае, разработчик и пользователь компонента территориально разделены и пользуются разными языками в единой среде.

Комментарий к слайду

Одним из принципиальных технологических преимуществ проектирования и реализации программного обеспечения, декларируемых Microsoft, является так называемый компонентный подход к программированию.

В своей основе указанный подход совпадает с традиционным объектно-ориентированным, однако имеет ряд важных особенностей. Поскольку конечной целью данной части учебного курса является гетерогенное компонентное программирование, необходимо с самого начала уяснить смысл основных понятий, на которых зиждется компонентный подход.

Центральной концепцией подхода (и это очевидно уже из названия) является понятие компонента.

Под компонентом в дальнейшем будем понимать независимый модуль программного обеспечения, который возможно повторно использовать, а также тиражировать.

В отличие от «традиционных» объектов ООП компоненты обладают следующими характеристическими свойствами:

- в целом компонент обладает более высоким уровнем абстракции по сравнению с объектом (если под последним понимается конструкция уровня языка программирования);
- компоненты могут содержать в своем составе множественные классы;
- компоненты с точки зрения пользователя являются инвариантами по отношению к тому языку программирования, на котором они реализованы.

Таким образом, оказывается, что в общем случае разработчик и пользователь компонента могут быть территориально разделены и могут использовать различные языки программирования в рамках единой среды разработки приложений Microsoft .NET.

Компонентное программирование в .NET (2)

- Компонентная объектная модель (COM):
  - основной стандарт Microsoft для компонент;
  - содержит протокол для инициализации и использования компонентов внутри одного процесса, между процессами или между компьютерами;
  - основа для ActiveX, OLE и многих других технологий;
  - поддерживается в Visual Basic, C++, .NET и др.
- Модель Java Beans:
  - основной стандарт Sun Microsystems для компонент;
  - зависима от языка реализации.

Комментарий к слайду

Заметим, что попытки построения компонентных программных систем предпринимались и рядом других компаний – разработчиков программного обеспечения (в частности, технология JavaBeans производства Sun Microsystems), а также международных ассоциаций, объединяющих усилия исследователей и практиков в области объектного программирования (например, стандарт брокеров объектных запросов CORBA организации Object Management Group, или OMG).

В основе таких попыток лежали варианты объектных моделей. Один из подобных вариантов, детально проработанный с математической точки зрения, а именно, модель двухуровневой концептуализации, нам предстоит рассмотреть в ходе данного курса. Сейчас же лишь кратко охарактеризуем основные отличительные особенности наиболее известных из представленных на рынке современного программного обеспечения объектных моделей.

Прежде всего, охарактеризуем компонентную модель Microsoft, которая обычно именуется в литературе аббревиатурой COM (что происходит от слов Component Object Model).

Компонентная объектная модель COM является основным стандартом Microsoft для компонентного проектирования и реализации программного обеспечения. На сегодня это самая развитая, и, пожалуй, самая удачная в практическом плане модель, которая практически обеспечивает возможность инициализации и использования компонентов как внутри одного процесса, так и между процессами или между компьютерами независимо от языка реализации. COM-модель поддерживается в идеологии .NET для целого ряда языков программирования (C#, SML, Visual Basic, C++ и др.), является основой для ActiveX, OLE, а также для многих других технологий Microsoft.

В отличие от COM, модель Java Beans, базовый стандарт Sun Microsystems для компонент, оказывается зависимой от языка реализации.

Сравнение компонентно- и объектно-ориентированного программирования

1. Основные понятия объектно-ориентированного программирования:
- класс (class);
- интерфейс (interface)

2. Основные понятия компонентно-ориентированного программирования:
- свойство (property);
- событие (event);
- сборка (assembly)

Комментарий к слайду

Исследовав основные особенности объектно-ориентированного и компонентного подходов к проектированию и реализации программного обеспечения, произведем краткий сравнительный анализ этих особенностей и подходов в целом.

Прежде всего, перечислим основополагающие понятия, характеризующие каждый из подходов. Затем сопоставим эти подходы друг с другом с целью поиска аналогий между ними.

В объектно-ориентированном подходе ключевыми являются, в частности, понятия класса и интерфейса. Заметим, что в компонентно-ориентированном подходе эти понятия также являются системообразующими.

При этом под ***классом*** понимается базовая сущность, определяемая как совокупность своих элементов.

Под ***интерфейсом*** понимается набор семантически связанных абстрактных элементов. Для компонентно-ориентированного подхода понятие интерфейса имеет первостепенное значение, поскольку исключительно посредством этого механизма клиент в архитектуре с моделью COM может непосредственно осуществлять взаимодействие с COM-классом. Заметим, что интерфейсы повышают безопасность кода, т.к. взаимодействие с объектом происходит не непосредственно, а через указатель (ссылку). Понятия свойства (как атрибута объекта) и метода (как операции над объектом), также как и механизма событий (соотнесений над объектами предметной области) свойственны обоим подходам.

Принципиально новым является наличие в COM-модели сборок – самодостаточных единиц информации для инсталляции и распространения программных продуктов.

В целом COM-подход является более удобным с практической точки зрения, хотя механизмы, реализованные в нем, принципиально сравнимы с возможностями ООП.

.NET – наиболее существенные недостатки

1. Высокие требования к аппаратному обеспечению (минимум 256M RAM, 10G HDD для работы с Microsoft Visual Studio .NET)
2. Сложности работы с некоммерческими релизами программного обеспечения (некоторая неустойчивость, отсутствие полномасштабной документации);
3. Поддержка ряда теоретически интересных и практически полезных языков программирвоания не в полном объеме (SML для Visual Studio .NET – в процессе реализации);
4. Инструментарий .NET (и компиляторы для языков программирования) не ратифицированы по международным стандартам.

Комментарий к слайду

Несмотря на перечисленные выше инновации в области теории, технологии и практической реализации, в силу масштабности идеологии и новизны исследуемой проблематики, подход .NET не лишен отдельных недостатков, большинство из которых, по-видимому, носит временный характер. Отметим, по нашему мнению, наиболее существенные из них.

Во-первых, разработчики отмечают достаточно высокие требования к аппаратному обеспечению (в частности, объем оперативной памяти должен быть не менее 256 мегабайт, свободный объем жесткого диска для работы с Microsoft Visual Studio .NET – не менее 10 гигабайт).

Кроме того, некоммерческие версии программных продуктов Microsoft, которые зачастую предоставляют новые существенные возможности, в недостаточной степени устойчивы в работе; документация по ряду новых функций программного обеспечения представлена не в полном объеме.

Поддержка ряда теоретически интересных и практически полезных языков программирования реализована в ограниченном объеме (скажем, компилятор для языка программирования SML для Visual Studio .NET находится в процессе реализации). Поскольку целый ряд компиляторов для языков программирования предоставляется сторонними по отношению к Microsoft компаниями-разработчиками или некоммерческими учреждениями, результаты их деятельности поддаются контролю и доработке с ограничениями.

Комплекс программно-инструментальных средств, реализующий подход .NET (включая и компиляторы для языков программирования) ратифицирован по международным стандартам не в полном объеме.

Платформа .NET – выводы

1. Стратегическая идеология и технологическая платформа Microsoft на ближайшее десятилетие
2. Несомненное качественное превосходство над аналогами (Inprise Delphi, Microsoft Visual Studio и др.) за счет:
   - интероперабельности и межъязыкового взаимодействия;
   - многоуровневой безопасности;
   - интеграции с веб-сервисами;
   - облегчения разворачивания и использования.
3. Некоторая незавершенность решения для широкого коммерческого использования в силу концептуальной новизны и грандиозности проекта.

Комментарий к слайду

Кратко резюмируем итоги лекции.

Безусловно, .NET является выдающимся достижением современной индустрии программирования. Достаточно сказать, что корпорация Microsoft считает именно .NET своей стратегической идеологией и технологической платформой на ближайшее десятилетие.

Несомненное качественное превосходство над существующими средствами автоматизированного проектирования и быстрой реализации прикладного программного обеспечения (в частности, Inprise Delphi и JBuilder, Oracle Developer, Microsoft Visual Studio и др.) достигается за счет следующих основных факторов:

- интероперабельность и межъязыковое взаимодействие;
- многоуровневая, гибкая и надежная политика безопасности;
- интеграция с технологией веб-сервисов;
- упрощение процедуры разворачивания и использования создаваемого программного обеспечения.

Несмотря на некоторую незавершенность решения для широкого коммерческого использования в силу концептуальной новизны и грандиозности проекта, подход .NET, безусловно, оказывает значительное влияние на коммерческую индустрию программирования в целом и способствует радикальному совершенствованию отрасли в ходе рыночной конкуренции.

Библиография

1. http://msdn.microsoft.com/net
2. Nathan A. .NET and COM: The Complete Interoperability Guide. Sams, 2002, 1608 pp.
3. Box D. Essential .NET, Vol.1: The Common Language Runtime. Addison Wesley, 2002, 432 pp.
4. Grimes F. Microsoft .NET for Programmers. Manning Publications, 2002, 386 pp.
5. J. Richter. Applied Microsoft .NET Framework Programming. Microsoft Press, 2002, 556 pp.

Комментарий к слайду

К сожалению, в рамках одной лекции невозможно представить такой многоаспектный подход как .NET в полном объеме.

Для более детального ознакомления с последними достижениями и проблемами в ходе развития подхода рекомендуется следующий список литературы:

1.http://msdn.microsoft.com/net

2. Nathan A. .NET and COM: The Complete Interoperability Guide. Sams, 2002, 1608 pp.

3. Box D. Essential .NET, Vol.1: The Common Language Runtime. Addison Wesley, 2002, 432 pp.

4. Grimes F. Microsoft .NET for Programmers. Manning Publications, 2002, 386 pp.

5. Richter J. Applied Microsoft .NET Framework Programming. Microsoft Press, 2002, 556 pp.

Кратко остановимся на источниках. Последние сведения о .NET из первых рук доступны с Интернет-ресурса [1]. Работы [2,3] посвящены интероперабельности; в работах [4,5] рассмотрены проблемы практической реализации программного обеспечения согласно подходу .NET.