XXX Юбилейная Неделя науки СПбГТУ. Материалы межвузовской научной конференции. Ч. VII: С.39-42, 2002.

УДК 681.324

А.М. Павлов (5 курс, каф. АиВТ), Е.В. Пышкин, к.т.н. доц.

ОБЗОР АРХИТЕКТУРЫ ПЛАТФОРМЫ MICROSOFT.NET ДЛЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Цели разработки платформы NET.

В числе основных целей, преследовавшихся разработчиками новой платформы, отмечаются следующие:

– реализация возможностей разработки Web-приложений, обеспечивающих комфортное взаимодействие с пользователями различных устройств;
– обеспечение независимости от операционной системы, установленной на компьютере, на котором разрабатываются и исполняются приложения, построенные на основе .NET;
– интеграция возможностей наиболее распространённых языков программирования;
– эффективное использование ранее спроектированных компонентов;
– упрощение и ускорение разработки приложений и Web-сервисов;
– повышение надежности разрабатываемого ПО.

Web-сервисы.

В основе архитектуры .NET – новая модель программирования распределённых приложений - Web-сервисы. В общем, под Web-сервисом понимается небольшой неоднократно используемый компонент распределённого приложения, доступный как разделяемый ресурс через компьютерную сеть.

Важно отметить, что для интеграции Web-сервиса в Web-приложение разработчику не обязательно знать, как именно конкретный Web-сервис реализован и где именно он находится. В отличие от существующих компонентных технологий, Web-сервис не использует DCOM, RMI или IIOP, которые требуют специальной, однородной инфраструктуры как на клиентской, так и на серверной частях. В качестве средства описания интерфейса, представления команд и типов данных используется XML – язык, специально созданный в качестве стандартного метаязыка для описания данных и типов данных. Обмен информацией и вызов Web-сервиса обеспечивает протокол SOAP.

Среда Common Language Runtime.

CLR – основа архитектуры .NET. Для платформы .NET свойственно построение и функционирование компонентов в общей среде. Нет необходимости в специальных средствах организации взаимодействия компонентов. Это позволяет использовать предоставляемые CLR сервисы, разделяемые разными компонентами – автоматическое управление памятью («сборка мусора» - Garbage Collection), отладка, сервисы безопасности, обработка исключений. В отличие от разнообразных механизмов обработки ошибок в существующих библиотеках, в .NET Runtime все ошибки обрабатываются при помощи исключений, и не нужно постоянно переключаться между кодами ошибок, значениями HRESULT и исключениями.

Контролируемый код и MSIL

Код, генерируемый трансляторами языков программирования, называется управляемым кодом (managed code). Отличия контролируемого кода от обычного исполняемого кода заключаются в том, что, во-первых, контролируемый код является кодом на специальном промежуточном языке – Microsoft Intermediate Language (MSIL), во-вторых, вместе с кодом генерируются метаданные. Для каждого объекта среды метаданные содержат информацию об именах объектов, определениях всех типов, сигнатурах всех типов, обо всех функциях объектов. Метаданные в CLR предоставляют подобную информацию, что и библиотеки типов, записи в системном реестре и прочая информация для приложений COM, но при этом

упакованы непосредственно в код, а не распределены в разных местах. С одной стороны это упрощает работу программиста, с другой стороны – предоставляет дополнительные удобства пользователю. В частности, метаданные используются CLR для проверки наличия необходимых приложению версий файлов.

Контролируемый код исполняется медленнее, чем обычный неконтролируемый код, - это во многом объясняется особенностями компиляции. Неуправляемый код компилируется под конкретный процессор, и потом просто исполняется. Компилирование управляемого кода разбито на две фазы – генерация MSIL и трансляция его в родной код процессора, что, конечно, требует больше памяти и процессорного времени. Однако разработчики в MS уверены, что со временем управляемый код будет работать быстрее. Они аргументируют это тем, что MSIL будет лучше приспосабливаться к особенностям различных процессоров и преимуществам одного процессора над другим, а также при управляемом коде будут лучше использоваться регистры процессора и лучше будет распределение памяти.

Сборки (assemblies)

Единицей распространения в .NET является сборка (assembly). Сборка содержит IL-код, метаданные и остальные необходимые файлы, объединённые в одно целое. Каждая сборка содержит манифест, содержащий перечень файлов, входящих в сборку, а также информацию о типах и ресурсах сборки, доступных за её пределами. Сборки могут принадлежать одному приложению или разделяться между несколькими приложениями. Механизм сборок позволяет также обеспечить корректное согласование различных версий программных продуктов. CLR может загружать различные версии одной сборки для двух различных приложений, которые могут выполняться одновременно. Поскольку вся необходимая информация содержится в манифесте сборки, никакая регистрация сборок в системном реестре не требуется, и инсталляция приложения может представлять собой простое копирование файлов.

Надёжность и безопасность приложений, управляемых CLR

Контролируемость кода проявляется, прежде всего, в построении надежных и безопасных приложений. Для обеспечения этого CLR предоставляет следующие возможности:

- автоматическое управление резервированием и освобождением ресурсов памяти – сборщик мусора уничтожает объекты или переменные, на которые больше нет ссылок; от программиста больше не требуется заботиться об освобождении памяти;
- во время выполнения среда следит за безопасностью по отношению к типам (несоответствие преобразований типов, выход за пределы выделенных областей памяти, неинициализированные переменные);
- механизм обработки ошибочных ситуаций посредством обработки исключений;
- обеспечение двух уровней безопасности: на уровне кода и на уровне ролей пользователя.

Visual Studio .NET

Microsoft предоставляет четыре языка в составе visual Studio .NET: C++, C#, VB и JScript, но при этом реализована возможность разрабатывать программы на любом языке, перенесенным на эту платформу. Причем вся работа над кодом ведется в рамках одной и той же визуальной среды разработки. На данный момент, по сообщениям Microsoft, поддерживаются, в частности, Cobol, Pascal, Eiffel, Perl, Python, Oberon, Smalltalk, Java. Предусмотрена возможность автоматического преобразования программного кода с языка Java на C#.

Одним из средств обеспечения полноценной межъязыковой интеграции является базовая библиотека классов (Base Class Library), которая является общей для всех языков .NET. Для корректного взаимодействия компонентов на различных языках разработана обобщенная спецификация языка CLS (Common Language Specification). CLS описывает требования к открытым интерфейсам классов, то есть устанавливает соглашения, которым нужно следовать при написании тех компонентов, к которым предполагается доступ из приложений, разработанных на другом языке .NET.

Базовая библиотека классов содержит функции, традиционно находящиеся в runtime-

библиотеках, а также ряд новых, в частности:

- классы-коллекции (очереди, массивы, стеки и хэш-таблицы);
- классы для работы с базами данных;
- классы ввода-вывода;
- классы WinForms для создания пользовательских интерфейсов локальных приложений;
- классы для организации сетевых взаимодействий;
- классы WebControls для создания пользовательских интерфейсов, ориентированных на Web.

Становится возможным и перекрестное наследование, когда класс на C++ формируется как производный от класса на Cobol или VB. Также ваше приложение может состоять из нескольких частей, написанных на разных языках. Это означает, что в команде разработчиков программных продуктов могут совершенно равноценно работать профессионалы в разных языках программирования.

Введение иерархического пространства имен позволяет избавиться от GUID а множественное наследование от генерации IDL файлов.