ную тему, то с помощью блоков тестов можно создать итоговый тест, содержащий в себе вопросы на все темы, что очень удобно для подготовки контроля для студентов.

Создание групп и пользователей. Для упрощения оценки знаний студентов в университете создана возможность создания групп пользователей. В группы заносится имена студентов; необходимые атрибуты для нового студента (имя студента, пароль).

*Вывод результатов для каждой группы***.** Имеется возможность удаления плохих результатов (например <40 баллов) одной кнопкой и создание версии для печати.

Для полноценной работы системы необходимы: веб-сервер с поддержкой PHP, MySQL на стороне сервера и веб-браузер с поддержкой JavaScript на стороне клиента.

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ РЕФЛЕКСИИ В MICROSOFT.NET ДЛЯ ОПТИМИЗАЦИИ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ПАМЯТИ НА ОСНОВЕ ПРИМЕНЕНИЯ МЕТАДАННЫХ

В. В. Бухвальд

Белорусский национальный технический университет, г. Минск

Научный руководитель А. Б. Севрук

Метаданные являются основным фундаментом разработки платформы Net Framework. Метаданные описывают поля и методы типов, с помощью метаданных сборщик мусора определяет достижимые объекты, т. к. метаданные указывают объекты, на которые ссылается данный объект. Применение метаданных с целью простой сериализации и десериализации объекта, чтобы, например, передать его через сеть, позволяет легко создавать Web-сервисы в.Net Framework. Процесс изучения метаданных называется отражением, иначе говоря, пользователь видит метаданные модуля или сборки, отраженные тем или иным инструментом. Отражение позволяет создать динамически расширяемые приложения. Тем самым обеспечивается создание типа и упаковка его в сборку. Такая глубина интеграции и простота её достижения не имеет аналогов среди более ранних технологий вроде Win32 и COM. Отражение позволяет методу изучать другой код и корректировать свою работу в зависимости от полученных сведений. На самом деле все специализированные атрибуты основаны на отражении. Кроме того, отражение позволяет методу корректировать свою работу в зависимости от того, кем он вызван. Так, можно реализовать метод, который будет выполнять одно действие, если вызван код этой же сборки, и совсем другое, если вызывающий код находится в другой сборке.

Метаданные – это набор таблиц. При компоновке сборки или модуля компилятор создает таблицы определений типов, полей, методов и т. д. В пространстве имен System. Reflection FCL есть несколько типов, позволяющих писать код для отражения (т. е. для синтаксического разбора) этих таблиц. На самом деле типы из этого пространства имен предлагают модель объектов для отражения метаданных сборки или модуля.

Типы, составляющие эту модель объектов, позволяют легко перечислить все типы из таблицы определений типов, а также получить для каждого из них базовый тип, интерфейсы и ассоциированные с ним флаги. Остальные типы из пространства имен System. Reflection позволяют запрашивать поля, методы, свойства и события типа путем синтаксического разбора соответствующих таблиц метаданных. Reflection позволяют запрашивать таблицы определений в метаданных, однако типов, запрашивающих таблицы ссылок, нет. Типы отражения также не поддерживают чте-

ние IL-кода метода, для этого приходится выполнять синтаксический разбор байтов файла. К счастью, формат файлов управляемых модулей является открытым и соответствует стандарту, определенному техническим комитетом ECMA. Итак, если в приложении, запрашивающем таблицы ссылок в метаданных или извлекающем байты IL-кода управляемого метода, то придется написать код, «вручную» выполняющий синтаксический разбор файла модуля или сборки, так как отражение не предоставляет таких возможностей.

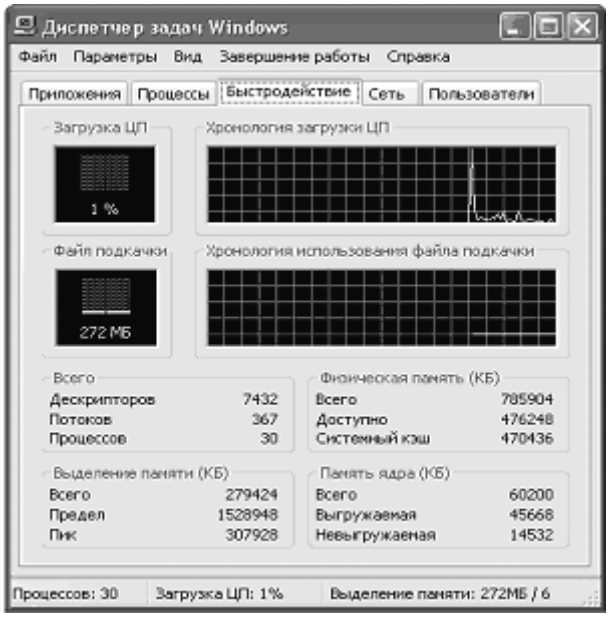
Отражение также не позволяет получить из метаданных некоторые определения. Так, нельзя определить значения по умолчанию для необязательных аргументов (имеющихся в Visual Basic, но отсутствующих в C#). В реальности приложениям редко требуются типы отражения. Обычно отражение используется в библиотеках классов, которым нужно понять определение типа, чтобы дополнить его. Так, механизм сериализации из FCL применяет отражение, чтобы выяснить, какие поля определяет тип. Объект форматирования из механизма сериализации получает значения этих полей и записывает их в поток байтов для пересылки через Интернет. Аналогично создатели Visual Studio используют отражение, чтобы определить, какие свойства показывать разработчикам при размещении элементов на поверхности Web-формы или Windows-формы во время ее создания.

Отражение используется, для решения некоторой задачи во время выполнения, когда приложению нужно загрузить определенный тип из некоторой сборки. Например, приложение может попросить пользователя предоставить имя сборки и типа, чтобы явно загрузить ее, создать экземпляр заданного типа и вызывать его методы. Концептуально подобное использование отражения напоминает вызов функций Win32 LoadLibrary и GetProcAddress. Часто привязку к типам и вызываемым методам, осуществляемую таким образом, называют поздним связыванием (late binding). Раннее связывание (early binding) имеет место, когда используемые приложением типы и методы известны при компиляции.

Все приведенные ниже примеры приложений используют отражение. Каждое демонстрирует особый способ применения отражения и иллюстрирует некоторый момент, важный для эффективного и рационального использования отражения.

Отражение чаще всего используется для манипулирования объектами при помощи сведений, которые можно получить при выполнении, но не при компиляции. Очевидно, что такой динамический анализ и манипулирование типами снижает быстродействие. Кроме того, при использовании отражения компилятор не поможет найти и исправить в программе ошибки, связанные с безопасностью типов.

Однако проведенная практическая оценка анализа быстродействия даёт вполне удовлетворительные результаты. Для тестирования использовались три различных динамически подключаемых библиотеки. Как известно, DLL представляет собой библиотеку функций (ресурсов), которыми может пользоваться любой процесс, загрузивший эту библиотеку. В то же время процесс отражения приводит к тому, что наименьшее количество памяти будет затрачиваться на самую сложную, в смысле сложности кода, программу. На рисунке приведены некоторые результаты работы программы.



Загрузка памяти без включения программы

Если загрузить программу, то ей надо сразу около 10 мегабайт оперативной памяти, но через тридцать секунд она уже потребляет всего около 7 мегабайт.

Если подключить первую DLL-библиотеку, то объем памяти уменьшится до – 6828 КБ; если вторую, до – 5720 КБ; если же третью, до – 6348 КБ; если подключить все DLL-библиотеки, то объём оперативной памяти, занимаемой программой, будет равен 17350 КБ.

Литература

1. Лабор, В. В. Си Шарп: Создание приложений для Windows/ В. В. Лабор. – Минск : Харвест, 2003.
2. Programming C#, 2nd Edition / J. Liberty. – O'Reilly; 2002.
3. Программирование для Microsoft Windows на C#: в 2-х т.; пер. с англ. Москва : Русская Редакция, 2002.
4. Рихтер, Дж. Программирование на платформе Microsoft.NET Framework /Дж. Рихтер; пер. с англ. – 2-е изд., испр. – Москва : Русская редакция, 2003.

ВЛИЯНИЯ ПАРАМЕТРОВ НЕЙРОСЕТЕВОЙ СИСТЕМЫ НА ТОЧНОСТЬ ИДЕНТИФИКАЦИИ ЭПИЛЕПСИИ

С. В. Безобразова

Учреждение образования «Брестский государственный технический университет», Беларусь

Научный руководитель В. А. Головко

Внешне эпилепсия проявляется по-разному – это могут быть судорожные припадки, изменение личности, потеря сознания. Эпилепсия – хроническое заболевание головного мозга человека, характеризующееся повторными припадками, которые возникают в результате чрезмерных нейронных разрядов (эпилептические припадки) и сопровождаются разнообразными клиническими и параклиническими симптомами [1].