УДК 004.424

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ПРАВИЛЬНОСТИ ВЫПОЛНЕНИЯ ПРИЛОЖЕНИЙ

Кравченко М.Ю.
Национальный технический университет Украины
«Киевский политехнический институт»
кафедра автоматизированных систем обработки информации и управления
E-mail: kravchenko568@gmail.com

Аннотация

Кравченко М. Ю. Автоматическая проверка правильности выполнения приложений*. Рассматривается практичность и простота проверки правильности выполнения программ с входящими и выходящими файлами. Показано, что применение данного программного средства позволяет определить правильность работы приложения с меньшими затратами интеллектуального ресурса человека.*

Общая постановка проблемы

Затраты человеческого ресурса на проверку и тестирование приложений растут с такой же огромной скоростью, как и развиваются компьютерные технологии. С развитием ИТ нагрузка на тестировщиков программных продуктов, а также на преподавателей ВУЗов и учителей информатики в школах и техникумах значительно увеличивается. Например, при анализе правильности 100 учебных приложений в группе из 20 студентов-программистов, временные затраты могут составить более 60 часов. Это то время, которое уходит на анализ и ввод входных данных, время выполнения программы и анализ полученных выходных результатов. Объем нагрузки возрастет в несколько раз в случае увеличения количества участников проверки приложений. Значит, вопрос уменьшения трудоемкости процессов контроля правильности разрабатываемых студентами приложений и сокращения времени на эту работу является актуальным. Разработанное программное средство позволяет значительно сократить время и уменьшить затраты интеллектуального ресурса человека на проверку правильности работы программы.

Анализ исследований и существующих решений

Рассмотрим некоторые из существующих программных средств контроля правильности выполнения приложений.

Программное средство контроля приложений на CD издания «Олимпиадные задачи по программированию (+CD)» [1] позволяет контролировать правильность выполнения приложений с входящими и выходящими файлами. На тестирование принимается файл с исходным текстом программы. Файл должен иметь расширение .pas и имя, соответствующее названию входного файла задачи. Чтобы отправить решение на тестирование, нужно перейти в каталог, где лежит файл с исходным текстом программы. Если тестирующая программа установлена правильно, система выдаст один из ответов: об ошибке компиляции, ошибке времени выполнения, о превышении предела времени, об отсутствии выходного файла, об ошибке представления или сообщение «Принято».

Данное приложение имеет ряд недостатков:

– проверка программ, написанных только на языке Pascal;
– отсутствие GUI, все управление происходит через ввод команд в консоль;
– отсутствие возможности добавлять свои входящие и исходящие файлы тестов;
– отсутствие возможности управлять количеством прогонов тестирования программы.

Еще одно из программных средств автоматического тестирования приложений – тестовый драйвер, который разработан в Санкт-Петербургском государственном институте

точной механики и оптики А. А. Сухановым. Программа написана для операционной системы DOS, значит, морально устарела.

На факультете ВМиК МГУ им. М.В. Ломоносова А. Черновым была создана система, работающая на операционных системах UNIX и Windows NT. В данной программе реализовано создание исчерпывающего протокола тестирования. Это дает возможность использовать её для проверки учебных задач. Программа может работать для проверки результатов учебных приложений учащимся и, как следствие, дает возможность проверить правильность выполнения написанного кода перед сдачей работы [4].

Программное средство Contester позволяет проводить турниры и индивидуально проверять решения задач по спортивному программированию. Программа содержит условия задач разной сложности. Также есть возможность тестирования учебных приложений, написанных на: C++, Object Pascal, C#, J# и Visual Basic [5].

Постановка проблемы

Целью данной статьи является анализ возможности использования компонента CLR платформы .NET как сервиса операционной системы Windows для разработки программного средства контроля правильности выполнения приложений с входящими и выходящими файлами для обеспечения быстрого и удобного тестирования программ с наименьшими затратами человеческого ресурса.

Функциональность системы

В процессе поставленной выше задачи разработано программное средство контроля правильности выполнения приложений, которое предусматривает реализацию таких бизнес-процессов:

– сбор первичных данных. Планируется проводить социологические опросы учителей средних школ, преподавателей техникумов и преподавателей ВУЗов для оптимизации визуального интерфейса программного средства;

– передача данных с помощью сети Интернет для удалённого тестирования программ;

– хранение данных для тестирования в базе данных: входящие файлы тестирования, исходящие файлы тестирования;

– статистическая обработка данных с целью повышения производительности тестирования;

– экспертный анализ и экспертное оценивание набора наиболее подходящих тестов

– формирование отчетов по результатам тестирования приложений

Технические особенности реализации и архитектура программного средства

Программное средство разработано на платформе Microsoft .NET с использованием среды выполнения Common Language Runtime (CLR), которая выполняет код и предлагает службы, облегчающие процесс разработки. CLR – виртуальная машина, интерпретирующая и исполняющая код на языке CIL, в который компилируется программа, написанная на C++. Основное назначение CLR – это выполнение приложений, соблюдение всех программных зависимостей, управление памятью, обеспечение безопасности, интеграция с языками программирования и пр. Среда выполнения обеспечивает множество сервисов, облегчающих создание и внедрение приложений, и существенно улучшает надежность последних. Все сервисы предоставляются унифицированной библиотекой классов. Эта библиотека содержит более 1000 классов для решения различных программных задач – от взаимодействия с сервисами операционной системы до работы с данными и XML-документами. Состав инструментария разработчика программного средства показан на рис.1

Рисунок 1 – Состав инструментария разработчика программного средства

Разработанное программное средство имеет одноуровневую архитектуру, представляющую собой модульное приложение. Модули, связанные с базой данных, обеспечивают возможность добавлять, редактировать и удалять тесты, а также передают тесты модулям проверки приложений. Модули проверки приложений осуществляют запрос тестов, запуск тестируемого приложения и проверку правильности результата, выдаваемого приложением. Модули передачи данных позволяют дистанционно проверять отчеты тестируемых приложений. Архитектура программного средства показана на рис 2.

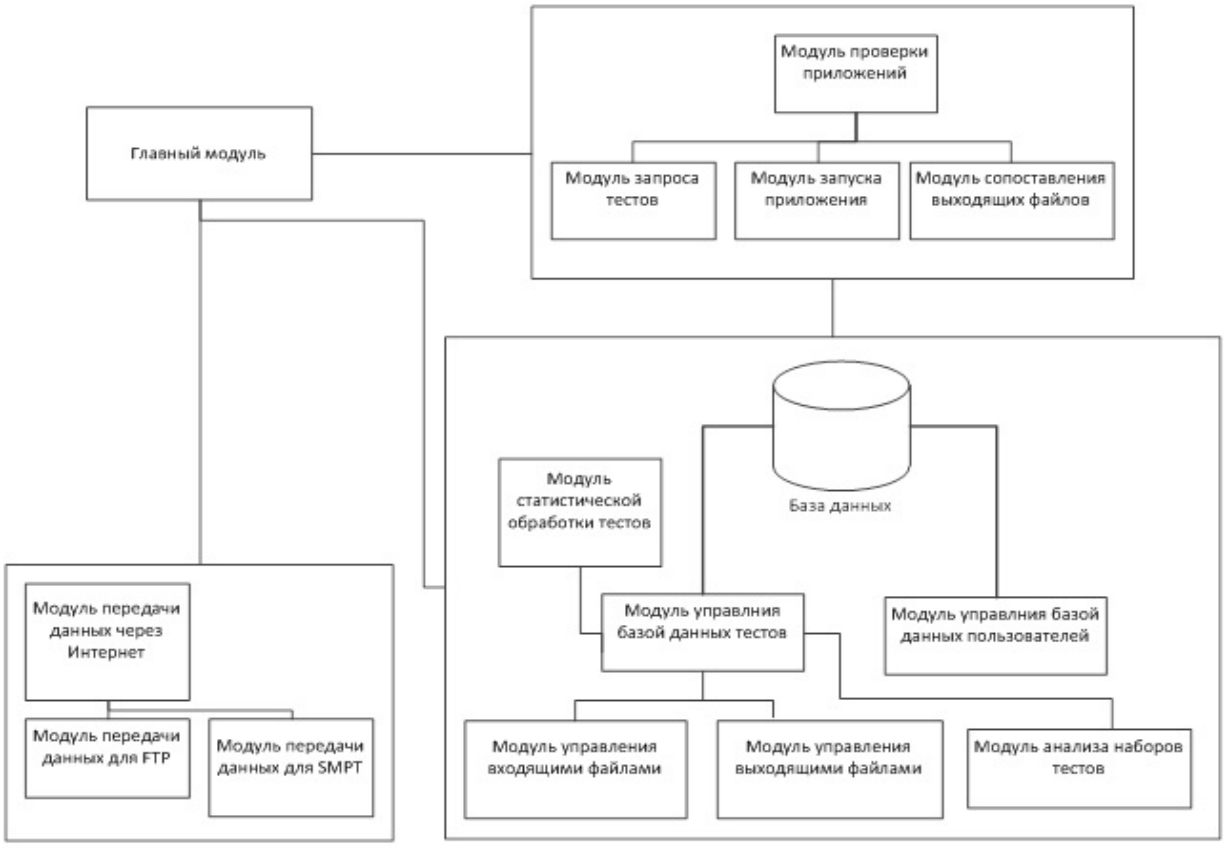

Рисунок 2 –. Структурная схема модулей программного средства

Для реализации GUI был использован шаблон приложенийWindows Forms. Процесс проверки приложений реализован на основании побитового сравнения выходных файлов. Сравнивается выходной файл приложения с соответствующим выходным файлом в базе данных. Если они идентичные, то приложение считается правильно работающим для данного теста. В противоположном случае тестирование приложения заканчивается, и оно определяется как неправильно работающее.

Для тестирования приложения необходимо, чтобы оно получило входной файл. База данных файлов тестов может храниться как на компьютере пользователя, так и на FTP-сервере. Передача данных через FTP реализуется соответствующим модулем. Он получает строку адреса на FTP сервере входных и выходный файлов и копирует их в нужную директорию. Если данная строка пуста, то программное средство копирует файлы тестов из директории по умолчанию.

После проверки приложения пользователь может отправить файл по электронной почте. Передача отчетов о работе приложений осуществляется по SMPT-протоколу.

Алгоритм тестирования приложения программным средством

При выборе набора тестов для программы нужно определить соответствующие файлы. Данная процедура осуществляется посредством выбора лабораторной работы в соответствующем окне программного средства. После этого, нажимая кнопку «перейти к сдаче лабораторной», нужные наборы входящих и выходящих файлов передаются в директорию выполнения приложения.

Следующее окно предусматривает загрузку архива с программой, где находяться все её файлы, и .exe-файл программы. После чего кнопка «перейти к тестированию лабораторной» становится активной. Нажав на неё, пользователь осуществляет тестирование программы в соответствии с количеством тестов. Алгоритм тестирования приведен на рис.3.

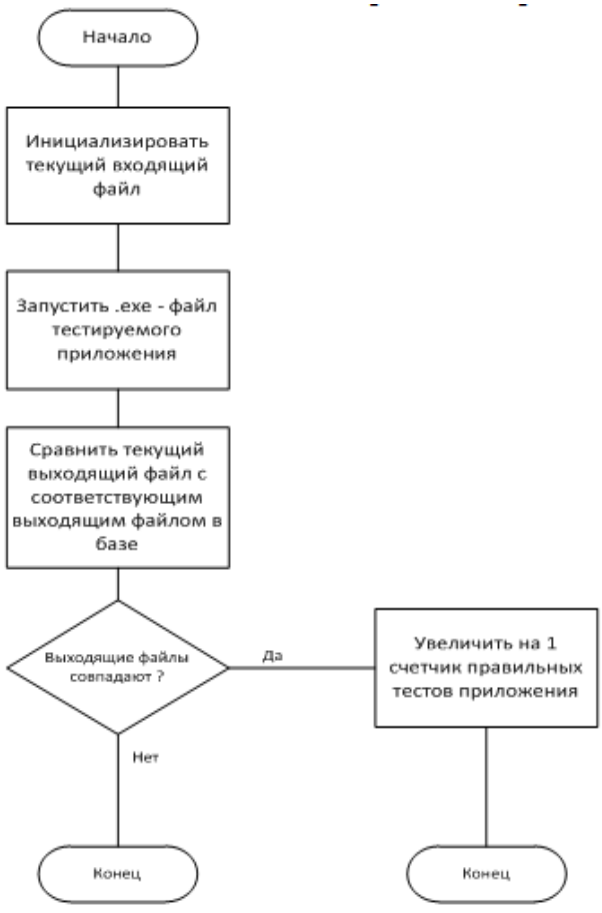

Рисунок 3 – Алгоритм тестирования:

После этого выводиться результат тестирования для данного приложения.

Результаты экспериментов

В данной работе было проведено два эксперимента. Первый – преподаватель во время проверки 100 лабораторных работ не использовал программное средство. Второй – преподаватель проверил 100 лабораторных работ с использованием программного средства.

Как и в первом, так и во втором эксперименте, каждое учебное приложение прошло 5 наборов тестов.

Результаты тестирования представлены на рис. 4

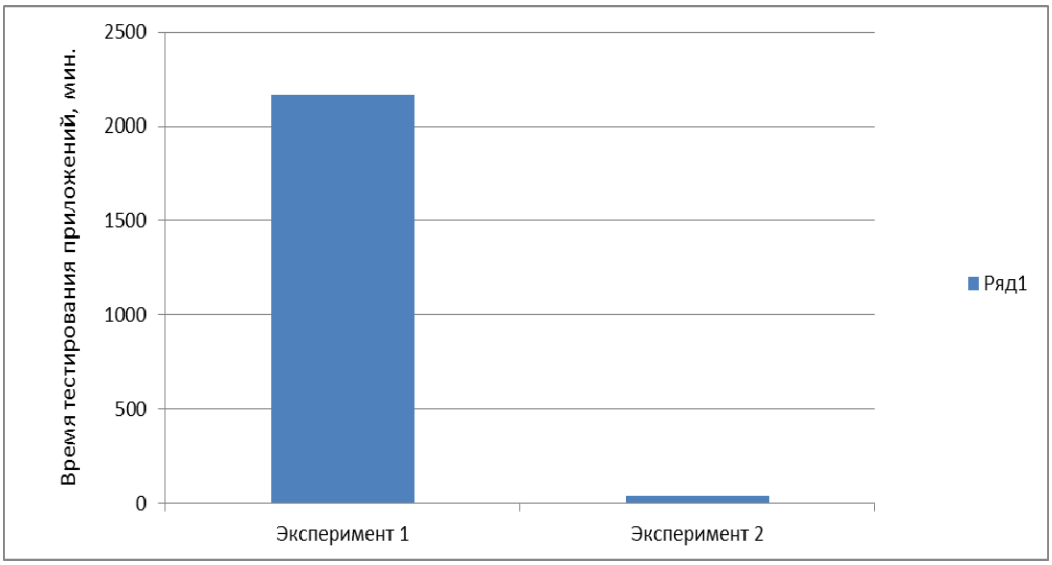

Рисунок 4 – Сравнение времени тестирования

Тестирование в первом эксперименте заняло 2164 мин., а во втором – 37 мин. Следовательно, разработанное программное средство позволяет значительно сократить время на проверку приложений с входящими и выходящими файлами.

Заключение

Применение среды выполнения CLR платформы Microsoft .NET открывает широкий спектр возможностей для создания программных средств и позволяет реализовать удобный пользовательский интерфейс. Созданное программное средство позволяет:

– тестировать практические навыки школьников и студентов;
– сокращать затраты времени на тестирование;
– сокращать затраты интеллектуального ресурса;
– создавать, редактировать и удалять наборы тестов для приложений;
– тестировать любые приложения формата .exe, которые работают с входными и выходными файлами.

Список литературы

1. Меньшиков, Ф. В. Олимпиадные задачи по программированию (+CD) [Текст] СПб.: – Питер, 2007. – С.22-23.

2. Common Language Runtime .[Електронный ресурс]/Режим доступа: http://ru.wikipedia.org/wiki/Common_Language_Runtime. Загол. с экрана.

3. CLR – Введение. [Електронный ресурс] /Режим доступа: http://www.mista.ru/net/clr.htm. Загол. с экрана.

4. Принципы проверки учебных и олимпиадных задач по информатике. [Електронный ресурс]/Режим доступа: http://g6prog.narod.ru/180803.html. Загол. с экрана.

5. Contester. [Електронный ресурс]/Режим доступа http://www.contester.ru/. Загол. с экрана.