РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ДЛЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

А.О. Голозубов

Введение.

Данная работа посвящена разработке пользовательского интерфейса для системы автоматизированного тестирования параллельных программ [1]. Рассматривается графический редактор описания формата файлов данных и возможности управления системой тестирования с помощью языка Python.

В настоящее время известен ряд систем, предназначенных для автоматического тестирования программного обеспечения: HP Quality Center [7], IBM Robot [8], IBM Functional Tester [9], AutomatedQA TestComplete [10], Borland SilkTest [11], UniTESK [12].

Система автоматизации тестирования по принципу «черного ящика»

Группой разработчиков Открытой распараллеливающей системы [5] была создана система автоматизации тестирования по принципу «черного ящика» [1].

Организация данной системы тестирования предъявляет следующие требования к тестируемой программе:

- Программа не является интерактивной, т.е. не содержит графического интерфейса пользователя и не считывает данные из консоли.
- Программа работает с заранее определенными входными и выходными файлами, имеющими фиксированный формат.

Рассматриваемая система тестирования в начале своей работы получает тестируемую программу и некоторую информацию о параметрах тестирования. Затем, в соответствии с заданными параметрами, система генерирует входные данные для тестируемой программы, запускает ее (возможно, несколько раз) со сгенерированными входными данными, а также заданными параметрами конфигурации архитектуры. В случае успешного запуска тестируемого ПО, система анализирует полученные выходные данные и выносит решение о том, пройден ли тест.

Формат входных и выходных файлов данных описывается с помощью языка описания формата файлов данных IODataDescription (расширение языка XML) [1]. С помощью описания формата файлов данных генерируются входные файлы для тестируемой программы, а также проверяются на соответствие описанию формата выходные файлы.

Тестирующая система может выполнять два вида тестов: «простой» тест – тестируемая программа запускается заданное количество раз, каждый раз с новыми входными данными, а результаты работы проверяются на соответствие формату выходных файлов; «сложный» тест – тестируемая программа запускается на одних и тех же входных данных, но с разными параметрами конфигурации архитектуры, затем результаты работы сравниваются.

В качестве параметров архитектуры, на которой будет запускаться тестируемая программа, тестирующая система поддерживает задание количества процессоров (для программ, использующих средства OpenMP), а также количества вычислительных узлов (для программ, использующих MPI).

Язык описания формата файлов данных IODataDescription

С помощью языка описания формата файлов данных IODataDescription можно описать два типа файлов данных: текстовый и бинарный.

В языке IODataDescription файл с данными представляется как последовательность строк, а строки - как последовательность токенов (ячеек с данными).

Токен может быть одного из пяти типов:

- Integer – целое число
- Double – число с плавающей запятой
- String – произвольная текстовая строка
- Const – константная строка (разделитель)
- Enum – элемент перечисления

Токен каждого типа имеет настраиваемые параметры:

1. Integer

- Минимальное значение
- Максимальное значение
- Имя
- Количество повторений

2. Double
- Минимальное значение
- Максимальное значение
- Точность
- Количество повторений

3. String
- Минимальная длина
- Максимальная длина

4. Const
- Значение
- Количество повторений

5. Enum
- Ссылка на имя перечисления

Токен типа Integer имеет необязательный параметр «имя». В случае, если этот параметр задан, токен становится именованным, и на него можно ссылаться в нижеследующих токенах, в параметре «количество повторений». В результате, при обработке файла данных тестирующая система повторит нужный токен или строку столько раз, какое значение содержится в том месте файла данных, которое соответствует именованному токену.

Язык IODataDescription позволяет описывать сложные межфайловые зависимости по именованным токенам, за исключением перекрестных ссылок между файлами. Тестируемая программа может принимать на вход несколько файлов данных, элементы которых могут быть логически связаны друг с другом, например, в файле А содержится размерность квадратной матрицы, а в файле Б – содержится сама матрица. В этом случае между двумя файлами данных возникает зависимость, которая отражается в описании формата файлов данных. Описания формата файлов данных будут таковы:

Файл A.xml
```
<configuration>
	<filetype>text</filetype>
	<row>
		<datatoken>
			<type>integer</type>
			<min>2</min>
			<max>5</max>
			<name>m_size</name>
		</datatoken>
	</row>
</configuration>
```

Файл B.xml
```
<configuration>
	<filetype>text</filetype>
	<row>
		<times>m_size</times>
		<datatoken>
			<type>integer</type>
			<times>m_size</times>
		</datatoken>
	</row>
</configuration>
```

Особенности реализации графического пользовательского интерфейса системы тестирования

В ходе эксплуатации тестирующей системы выяснилось, что при редактировании больших файлов описания формата данных для сложных программ с помощью текстового редактора пользователи допускают много ошибок. Для решения этой проблемы был создан редактор описания формата файлов данных с графическим интерфейсом пользователя, основанным на средствах кроссплатформенной библиотеки QT [6].

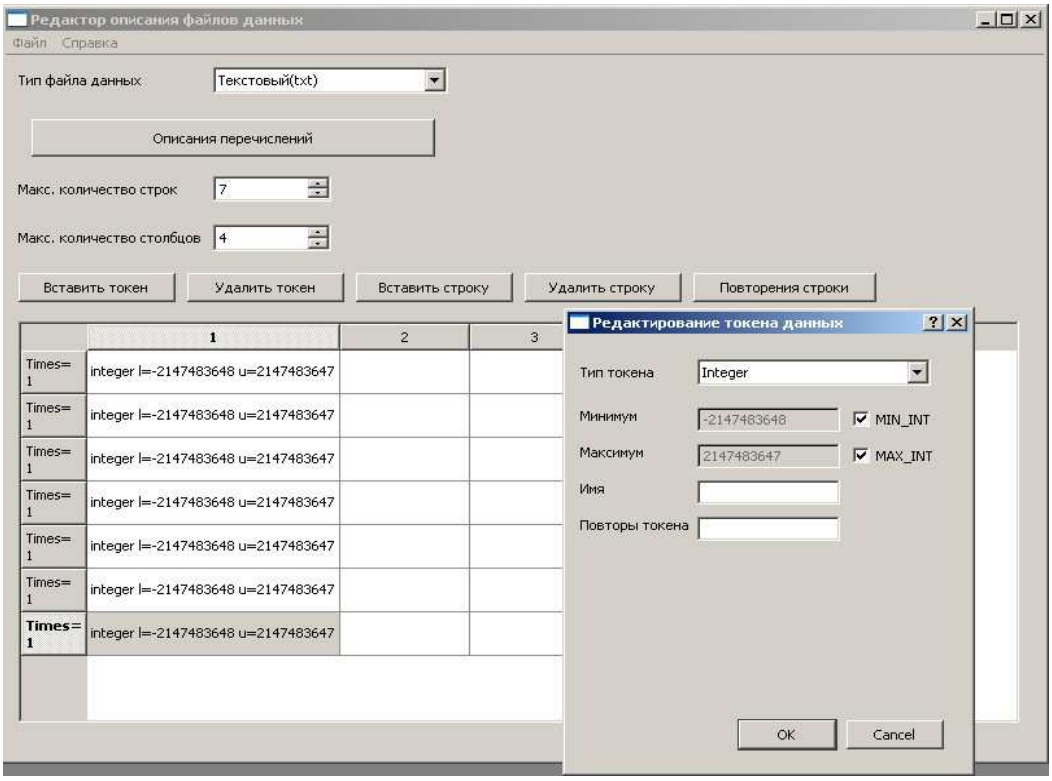

Рис.1. Редактор описания формата файлов данных.

Редактор предназначен для работы с одним файлом описания формата файла данных; он позволяет создавать, открывать, изменять и сохранять файл описания формата данных. Окно редактора состоит из двух частей: верхняя позволяет редактировать параметры, относящиеся ко всему файлу описания IODataDescription, а в нижней находится таблица, отражающая структуру файла описания формата данных. Редактирование и добавление новых токенов осуществляется в отдельном диалоговом окне. При работе редактора проверяются все ограничения языка описания формата данных IODataDescription, как на параметры токенов, так и на взаимное расположение токенов друг относительно друга.

Графический редактор использует общее внутреннее представление файлов описания данных с системой тестирования [1], кроме того, является кроссплатформенным приложением, благодаря использованию для реализации графического редактора библиотеки QT.

Проблема сложных входных данных

При тестировании арифметической библиотеки для работы с большими числами в кольцах вычетов по модулю n, возникла проблема сложных входных данных: с помощью языка IODataDescription невозможно было описать условие обратимости элемента в кольце вычетов по модулю n. Система тестирования генерировала случайные числа, подавала их на вход тестируемой программе, которая вызывала функцию обращения элементов в кольце вычетов по модулю n из арифметической библиотеки для работы с большими числами в кольцах вычетов по модулю n. Большинство вызовов функции обращения элементов в кольце вычетов по модулю n завершалось неудачно, так как автоматически сгенерированные случайные числа не являлись обратимыми элементами в заданном кольце вычетов.

Пользователю предоставлена возможность самостоятельно настроить тестирующую систему под свою программу, а именно: определить поведение модулей генератора случайных входных данных, подсистемы сравнения и анализа выходных данных с помощью языка программирования Python. Каждый файл описания данных можно связать с особым образом оформленным файлом программы на языке Python. Каждому токену можно дополнительно поставить в соответствие три параметра – имя функции в соответствующей программе на Python, которая будет выполнять генерацию данного токена, имя функции, которая будет проверять полученное значение токена на соответствие допустимым значениям, и имя функции, которая будет проверять полученные два значения на равенство.

Заключение

В результате создания пользовательского интерфейса для системы автоматизированного тестирования параллельных программ достигнуто улучшение управляемости процессом тестирования. Пользовательский интерфейс успел себя хорошо зарекомендовать при тестировании оптимизирующих преобразований Открытой

распараллеливающей системы, а также при тестировании арифметической библиотеки для работы с большими числами в кольцах вычетов по модулю n.

Автор выражает благодарность Татьяне Мухутдиновой, оказавшей большую помощь в подключении интерпретатора языка Python к тестирующей системе.

ЛИТЕРАТУРА:

1. Б.Я. Штейнберг, Е.В. Алымова, А.П. Баглий, Р.И. Морылев, З.Я. Нис , В.В. Петренко, Р.Б. Штейнберг «Автоматизация тестирования элементов высокопроизводительного программного комплекса». Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийской суперкомпьютерной конференции (21-26 сентября 2009 г., г. Новороссийск). - М.: Изд-во МГУ, 2009.
2. Г. Майерс «Искусство тестирования программ» - М.: Финансы и статистика, 1982г.
3. Википедия. Автоматическое тестирование.http://ru.wikipedia.org/wiki/Автоматическое_тестирование
4. Элфрид Дастин, Джефф Рэшка, Джон Пол «Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация» - М.: «ЛОРИ», 2003.
5. Открытая распараллеливающая система http://ops.rsu.ru
6. http://qt.nokia.com
7. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24_4000_100__
8. http://www-01.ibm.com/software/awdtools/tester/robot/index.html
9. http://www-01.ibm.com/software/awdtools/tester/functional/index.html
10. http://www.automatedqa.com/products/testcomplete/
11. http://www.borland.com/us/products/silk/silktest/index.html
12. http://www.unitesk.ru/