Мамаева Светлана Олеговна,
Степулёнок Денис Олегович

ТЕСТИРУЮЩАЯ СИСТЕМА СПбГЭТУ

ВВЕДЕНИЕ

Автоматизированные тестирующие системы получают все большее распространение. Изначально они создавались для автоматизации рутинного процесса тестирования программ. При ручном тестировании человек (тестер) многократно проверял программу на наличие ошибок (много раз проделывал одну и ту же последовательность действий), чтобы понять, исправлена ли найденная ранее ошибка, и убедиться, что программа не «испортилась» при исправлении других ошибок, при добавлении новых функций или переносе программы на другую платформу. Последовательность действий при тестировании очень часто повторяется, повторяющиеся действия можно автоматизировать (то есть сделать так, чтобы программу тестировала другая программа). При этом процесс тестирования может быть автоматическим (без участия оператора) и интерактивным (с участием оператора).

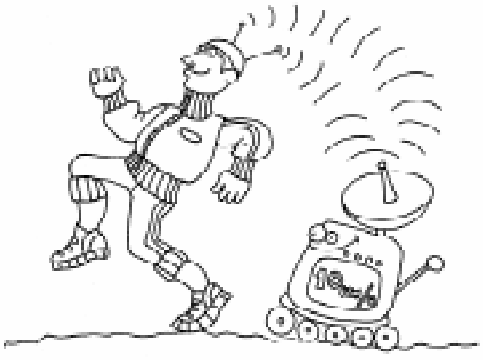
Под тестом подразумевается набор формализованных условий, которым должна удовлетворять реакция программы на входное воздействие. Очевидно, что система тестирования с подобными принципами может использоваться и для проверки знаний человека в любой области, допускающей формализацию задач.

Тестирующие системы применяются в физике, математике, программировании и даже иногда в гуманитарных науках. Наибольшее распространение они получили при проверке навыков программирования. Например, такие системы применяются на Международной олимпиаде по программированию по правилам организации ACM (Association for computer machinery).

В математике применять тестирующие системы тоже целесообразно – решение большинства задач представляет собой некоторое множество функций или чисел. Довольно утомительно проверять однообразные решения математических задач и в то же время довольно просто придумать формальные требования к решениям. При этом схема может быть следующей: человек предлагает системе ответ (некоторое множество чисел или функций), система либо принимает ответ («правильное» множество и ответ человека совпадают), либо указывает элемент правильного множества, который не упомянул человек, либо указывает на элемент, который является в ответе лишним. По этому принципу работает система Verifier, описание которой можно найти в «КИО» № 1, 1998 г. и «КИО» № 2, 2000 г.

ТЕСТИРОВАНИЕ ПРОГРАММ

Все существующие на данный момент системы для тестирования программ можно поделить на два класса – использующие стандартный ввод/вывод или файлы отчета и системы псевдо-пользовательского взаимодействия с обратной связью. Под последними понимаются системы, которые способны генерировать пользовательские действия (щел-



Автоматизированные тестирующие системы получают все большее распространение.

чок мыши, нажатие клавиши, перемещение курсора) и считывать полученные результаты, контролируя их с эталонными. Системы второго класса гораздо менее распространены, но существенно расширяют горизонты задач, предлагаемых в тестах.

Рассмотрим подробно современные тестирующие системы первого и второго вида. Системы класса STDIORPT (Standard Input/Output Or Report) варьируются от простейших скриптов на базе языков Unix-shell, Python, Perl до написанных на языке C распределенных систем (то есть систем, работающих на нескольких компьютерах одновременно), которые имеют связи с реляционными базами данных.

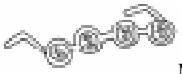
ТЕСТИРОВАНИЕ УЧАСТНИКОВ СОРЕВНОВАНИЙ

Тестирующие системы для проведения соревнований (Интернет – соревнований) по программированию с применением тестовых задач обычно используют автоматизированные системы проверки результатов и контроля командных баллов.

... большинство тестирующих систем обладают следующими типовыми блоками: блок принятия и проверки результатов от участника, блок подсчета командных очков, блок отображения статистики, блок точной синхронизации времени между командами.

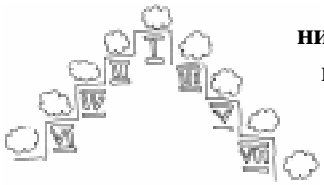
В настоящее время большинство тестирующих систем обладают следующими типовыми блоками: блок принятия и проверки результатов от участника, блок подсчета командных очков, блок отображения статистики, блок точной синхронизации времени между командами.

Блок принятия результатов в системах STDIORPT принимает результаты тестирования в виде текстового файла для тестов с фиксированным заданием или путем запуска исполняемой программы (решения) в своем собственном адресном пространстве, выдачи исходных данных на стандартный ввод и получения результатов со стандартного вывода. Для предотвращения неправомочных действий со стороны участников тестирования системы с выполнением исполняемого файла принимают чаще всего исходный код программы, после чего производят автосборку и тестирование в автономном режиме. Кроме того, применяются специальные средства защиты операционной системы для предотвращения запрещенных действий программой участника. Для распределенных командных тестирований широко используется сеть Internet, где данный блок обычно реализуется при помощи браузеров, где в одном окне выдается тестовое задание, а в другое предлагается скопировать полученный результат. Завершая обзор этого блока, отметим, что он должен быть обязательно выполнен на параллельной основе (то есть возможна параллельная обработка результатов большого числа участников с помощью отдельных потоков или процессов). Кроме того, такие блоки должны обязательно взаимно обмениваться информацией для синхронизации результатов и учета разности часовых поясов участников (если это необходимо). При выставлении результатов блок обязан действовать по принципу FIFO, то есть участник, приславший на обработку результат первым, и обрабатывается первым.

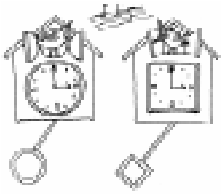


Блок учета командных очков получает информацию от потоков/процессов блоков тестирования и проверки, фиксиру-

ет время и полученный результат в базе данных, контролируя успешность выполнения данной операции. Вместо БД в старых системах тестирования может использоваться просто текстовый файл с текущими командными результатами.



Блок отображения статистики чаще всего тоже работает с базой данных, в которой хранится статистика по командам. Современная реализация данного блока представляет собой либо непрерывно обновляемую на тестовом сервере информацию статистики, либо HTML страницу, генерируемую динамически и выдаваемую участникам по запросу. Естественно, только второй вариант применим для распределенных тестов.



Блок синхронизации времени между командами обеспечивает синхронизацию между командами, находящимися в разных часовых поясах при использовании тестов на время (все команды начинают выполнение тестов в одно и то же время по своему часовому поясу) или при одновременном тестировании большого количества команд. Данный блок используется при промежуточных обработках тестовых данных и до записи информации в базу данных, где время чаще всего хранится в формате UTC или GMT.

Системы второго класса ATGUI (Automatic Testing Graphics User Interfaces), как правило, включают в себя все возможности систем первого класса, но добавляют к ним дополнительные возможности автотестирования графических интерфейсов. На данный момент известны системы для тестирования результатов, отображаемых в графических интерфейсах сред Unix и Windows. Некоторые системы могут включать в себя судейские терминалы ручного контроля, наряду с автоматизированными проверками. Недостатком таких систем является то, что они обрабатывают результаты последовательно и достаточно медленно, кроме того, сложность обрабатываемых ими интерфейсов, как правило, ограничена. Еще один недостаток таких систем состоит в необходимости программирования под конкретную платформу и графический интерфейс. Из несомненных достоинств данных систем можно отметить широкое разнообразие тестовых заданий, которые можно реализовать с их помощью. В последнее время широкую популярность завоевали системы автотестирования на базе платформы Java, которая изначально включает в себя средства поддержки автоматизированного тестирования интерфейсов.

В завершение отметим, что одним из самых важных факторов при реализации тестирующих систем является отказоустойчивость и надежность. Критический сбой системы может повлечь за собой сбой статистики команд, времени и т. д. Именно поэтому любая тестовая система должна быть обязательно оснащена средствами восстановления в рабочее состояние, предшествующее останову в случае критического сбоя. Для распределенных систем необходимы централизованные средства восстановления и мониторинга удаленных терминалов. Также обязательны средства проверки и мониторинга канала связи.

Теперь мы опишем наш опыт создания тестирующей системы, новой тестирующей системы СПбГЭТУ. Будем надеяться, что это может помочь при проектировании собственной системы.

НАЗНАЧЕНИЕ СИСТЕМЫ

Назначение системы (первоначальные предполагаемые варианты применения):

1. Проведение тренировок студентов для подготовки к командным олимпиадам по программированию по правилам ACM.
2. Проведение подготовки школьников к школьным городским и всероссийским олимпиадам по программированию.
3. Проведение различных олимпиад. Например, проведение Ленинградской областной олимпиады.

НЕДОСТАТКИ СТАРОЙ СИСТЕМЫ

Старая система обладает рядом существенных недостатков:

1. Система была написана для MS-DOS и не могла использоваться вместе с компиляторами под Windows. Сейчас большинство олимпиад перешло с Borland C++ 3.0, Borland Pascal 7.0, QBasic на Delphi, Visual Age for C++, Visual Age for Java.

2. Большинство современных пользователей привыкли к интерфейсу Windows, и «старый» интерфейс «под MS-DOS» для них неудобен.

3. Под операционными системами Windows 2000 и Windows XP старая система часто работать отказывается. Это опять же связано с тем, что система откомпилирована не под Windows.

4. Система не выдает сообщения об ошибках в конфигурационных файлах. Если, например, указан неправильный путь к каталогу тестов, система просто «вылетает», и понять, где произошла ошибка, очень сложно.

СТРУКТУРА СЕРВЕРА

Структура сервера изображена на рисунке 1.

В новой системе в качестве языка реализации выбран Borland Delphi 6.0, так как он обладает быстрым компилятором и удобным отладчиком.

Программы сервера не оснащены пользовательским интерфейсом. Он там и не нужен – весь интерфейс сосредоточен в программе участника. Администраторская утилита контроля является отдельной программой.

ФОРМАТ КОНФИГУРАЦИОННЫХ ФАЙЛОВ

В качестве альтернативы старому формату конфигурационных файлов был предложен формат на основе XML (см. таблицу 1). Это позволит упростить процедуры чтения/записи. Недостатком является необходимость создания конвертера из старо-

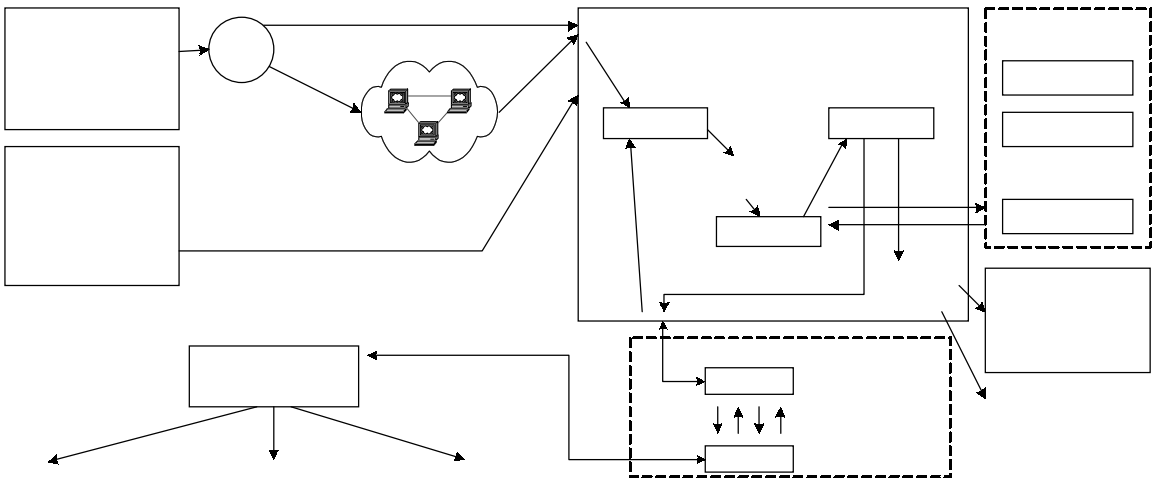


Рисунок 1.

Таблица 1.

Формат старого конфигурационного файла	Формат нового конфигурационного файла

`.USER`
`  .ID = "03"`
`  .Name = "SPb LEEI Team"`
`  .Queue = %E\03\`
`  .Status = %E\03\Stat.txt`
`  .Synchronize = No`
`  .Table = %E\03\Table.usr`
`  .Send Data = 1:".03.0005.STUD.ETU"`
`  .Disqualificated = NO`
	`<user>`
	`  ID = "03"`
	`  Name = "SPb LEEI Team"`
	`  Queue = "03\"`
	`  Status = "03\Stat.xml"`
	`  Synchronize = No`
	`  Table = "03\Table.xml"`
	`  Send_Data = ".03.0005.STUD.ETU"`
	`  Disqualificated = NO`
	`</user>`

го формата в новый. Впоследствии следовало бы встроить конвертер в программу «сервер», то есть сделать так, чтобы старые конфигурационные файлы можно было бы использовать без использования конвертора.

ОРГАНИЗАЦИЯ ИНТЕРНЕТ-АРХИВА ТРЕНИРОВОК

Интернет-архив нужен для поддержания в порядке коллекции тренировок в клубе программистов, а также для предоставления доступа к пакетам для будущих пользователей тестирующей системы вне университета.

ПРОГРАММА ПОДГОТОВКИ ПАКЕТА ЗАДАЧ И ОРГАНИЗАЦИОННЫХ ДАННЫХ

Для максимального упрощения центральной программы сервера и в то же время обеспечения наиболее дружественной среды для жюри олимпиады присутствует отдельная утилита, создающая файлы настроек для конкретной олимпиады и упаковывающая все для олимпиады в единый файл архива.

ТЕСТЕРЫ

Для создания программ, проверяющих ответы, используется аналог библиотеки TestLib, разработанной в ЛИТМО.

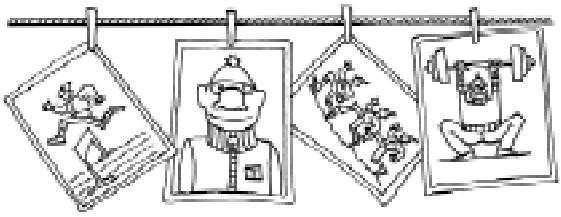
ПОДСИСТЕМА ПЕРЕДАЧИ ДАННЫХ

Выделена отдельно для унификации работы с различными протоколами (с разными типами сетей): TCP/IP, IPX/SPX.

ПРОГРАММА УЧАСТНИКА

Представляет собой самую «красивую» компоненту, которую видит участник и жюри. Она позволяет следить за соревнованием, просматривать текущие результаты.

Для каждого клиента программа-сервер устанавливает определенный уровень привилегий, допускающий получение определенной информации о соревновании.



...Интернет-архив нужен для поддержания в порядке коллекции тренировок ...

Мамаева Светлана Олеговна, студентка VI курса СПбГТУ,

Степулёнок Денис Олегович, студент V курса СПбГЭТУ, руководитель студенческого клуба программистов, методист ЗСШП.