Д.А. Васенина, Л.Н. Лядова, М.А. Плаксин, С.И. Шарыбин

Россия, Пермь,

Государственный университет – Высшая школа экономики

(Пермский филиал),

Пермский государственный университет

LyadovaLN@hse.perm.ru, MAPL@list.ru

ПРОЕКТ «ЭЛЕКТРОННЫЙ ПРЕПОДАВАТЕЛЬ ПРОГРАММИРОВАНИЯ»

Описывается проект создания комплекса программ, предназначенного для поддержки обучения основам программирования. Комплекс включает несколько приложений, обеспечивающих снижение трудозатрат преподавателя в процессе обучения программированию, в частности, контролируется стиль программирования, полнота и неизбыточность наборов тестов, правильность выполнения программы.

The project of development of a program complex intended for support of training to bases of programming is described. This complex includes some applications providing decrease of labor expenditures of the teachers in the course of training to programming. For example style of programming, completeness and non-redundancy of tests sets, correctness of program execution results are controlled.

В процессе обучения основам программирования студент или школьник должен освоить не только теоретический материал, но и получить опыт решения задач с помощью компьютера, разработки программ. Часто обучение программированию сводится к изучению языка программирования и получению навыков работы в выбранной для обучения системе программирования, кодирования и выполнения программ. При этом упускается из виду то, что создание программы – сложный многоэтапный процесс, а обучение программированию предполагает получение многих навыков, выходящих за рамки написания программ в определенной среде. Студенты, разрабатывая программу, стремятся показать, что «она работает», подбирая для этого входные данные – «подгоняя под ответ», а преподаватели не успевают оценить представляемые решения, рассмотреть все аспекты выполненных студентами работ. В результате студенты не получают необходимых знаний, навыков грамотной разработки программ.

Особенно эта проблема остро стоит при подготовке специалистов в области информационных технологий (ИТ), разработчиков, программистов. Именно на младших курсах закладывается основа для изучения специальных дисциплин, инструментальных средств создания информационных систем (ИС) различного назначения, технологий, обеспечивающих поддержание всего жизненного цикла программного обеспечения (ПО).

Современные предметно-ориентированные технологии создания ИС позволяют привлекать экспертов, специалистов в различных предметных областях к разработке и сопровождению ПО на всех этапах жизненного цикла ИС. Поэтому не менее важно показать все стороны процесса создания программ студентам других специальностей, направлений подготовки – будущим специалистам, которые могут участвовать в управлении жизненным циклом ПО, не являясь профессиональными программистами.

Обучение программированию предполагает рассмотрение различных аспектов создания ПО, получение навыков постановки задачи, формализации требований к решению; составления алгоритма ее решения и выбора нужных алгоритмических приемов (итерация или рекурсия и т.п.); применения различных технологий программирования (нисходящего и восходящего программирования, модульного программирования и т.п.); подбора оптимальных структур, типов данных; записи алгоритма на языке программирования, применения нужных языковых конструкций (циклов с пред- или постусловием, *for* вместо *while* или наоборот, именованных констант и пр.); оформления программы в виде, удобном для чтения и дальнейшей переработки; построения набора критериев тестирования программы методами «черного ящика» и «белого ящика», составления тестов по сформированным наборам критериев, проверка полноты и неизбыточности составленного набора тестов; выполнения программ на разработанных наборах тестов и проверка правильности выполнения программ и пр. Качественное обучение всем перечисленным аспектам программирования требует больших трудозатрат как со стороны студента, так и со стороны преподавателя. Это послужило причиной для начала реализации проекта по автоматизации труда преподавателя – по созданию комплекса программ, которые в совокупности должны сыграть роль «электронного преподавателя программирования».

Исследование существующих решений показало, что создание такого комплекса программ является актуальной задачей, так как существующие системы поддержки обучения программированию, в основном, направлены на автоматизацию проверки программ с точки зрения корректности решения поставленной задачи. Косвенно такие системы могут проверить оптимальность выбора алгоритма решения задачи (через ограничения на использование ресурсов при выполнении программы). При этом практически не существует средств, которые обеспечивали бы контроль навыков, крайне важных для базового уровня обучения программированию (умение разрабатывать хорошо структурированный и оформленный код, тестировать программы и т.д.), и помощь в их освоении.

Наиболее известными системами проверки решений задач по программированию, применяемыми в учебном процессе, являются системы дистанционного проведения олимпиад по программированию. Были рассмотрены такие системы как ejudge, Sphere Online

Judge, система, применяемая на Timus и TopCoder [3 и др.]. Все они выполняют запуск решений (программ), получаемых для проверки от учащихся или участников олимпиад, на авторском наборе тестов с целью проверки полноты и оптимальности решения поставленной задачи. При этом, к сожалению, не гарантируется полная верификация полученной программы. Кроме того, в полученных решениях не учитываются и не оцениваются используемые структуры данных и реализуемые программами алгоритмы, стилистика программного когда, умение студентов тестировать свои программы, самостоятельно разрабатывая критерии и тесты. Еще один недостаток использования подобных систем – их ориентация на проведение соревнований, на проверки и контроль, но не на обучение (консультирование, пояснение результатов проверок, объяснение выявленных ошибок и т.п.). Еще одно ограничение, связанное с использованием этих систем, – ориентация в большинстве случаев на конкретные системы программирования и сложность расширения их функциональности, интеграции с внешними системами.

Существует ряд исследовательских работ на тему автоматизации процесса верификации программ, организации эффективного процесса обучения программированию ([1], [2] и др.), но они также не затрагивают перечисленные выше аспекты. Результаты этих исследований можно использовать в будущем для реализации дополнительных средств проверки полноты решений, контроля авторства программ и т.д. Предлагаются интересные подходы к созданию адаптивных обучающих систем, которые в автоматическом режиме производят обучение большой аудитории студентов с сильно различающимся уровнем навыков и знаний, оптимальным образом выстраивая образовательные траектории.

При реализации разрабатываемого комплекса было принято решение об исполнении его подсистем в виде модулей, которые могут использоваться как самостоятельно, так и интегрироваться с другими системами. Такое решение позволяет совместить процесс проверки решений с возможностями обучения, использовать единую базу пользователей (учащихся) и агрегировать статистику по каждому пользователю не только в рамках системы обучения программированию, но и другим дисциплинам, где используются средства автоматизации обучения и контроля знаний (например, средства компьютерного тестирования, применяемые в учебном процессе при проведении контрольных мероприятий).

Программный комплекс делится на две части: клиентская, с которой взаимодействуют пользователи, и серверная, которая производит проверки решений.

Было принято решение о реализации серверных компонентов комплекса на базе семейства операционных систем (ОС) GNU/Linux. Выбор обусловлен тем, что 1) имеются механизмы, позволяющие получать о процессах информацию, которой манипулирует ядро ОС, что важно для профилирования процессов; 2) есть возможность запуска приложе-

ний в изолированных пространствах, что обеспечивает надежность и независимость проверки получаемых решений; 3) реализованы механизмы, позволяющие управлять возможностью использования приложениями того или иного системного вызова. Кроме того, в образовательных учреждениях в качестве серверной ОС чаще всего используется Linux.

В настоящее время реализованы серверные компоненты верификации решений (программ, разработанных учащимися), а также автоматизации проверки корректности построения учащимися таблиц минимального грубого тестировании (МГТ).

Основа разработки комплекса – система проведения олимпиад по программированию WebTester/Gate (рис. 1).

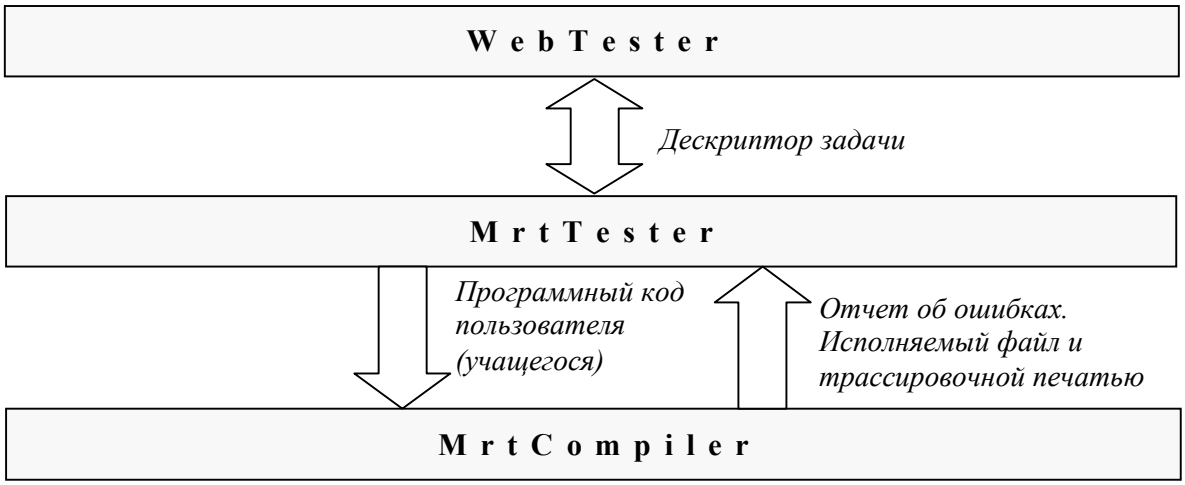

Рис. 1 . Общая архитектура серверной части

WebTester (рис. 2) – серверная часть системы проведения олимпиад (верификации решений). Пользователи взаимодействуют с сервером через клиентскую часть приложения (Gate). Для работы пользователю необходимо установить Web-браузер. Цикл проверки решений отвечает за прием непроверенных решений из Web-интерфейса; запуск соответствующего модуля проверки; выдачу результатов проверки через Web-интерфейс; запуск процесса закачки новых задач из Web-интерфейса в серверную часть. Под профилированным запуском подразумевается запуск процесса и полное слежение за тем, сколько и каких ресурсов он потребил, немедленное прерывание исполнения в случае нарушения ограничений на ресурсы и правила доступа к ним. Библиотека LibRun обеспечивает запуск процесса от имени указанного пользователя и с указанными ограничениями, позволяет запускать приложения в изолированных пространствах файловой системы, что позволяет повысить безопасность проверки. Библиотека API LibWebtester реализована для абстрагирования ядра WebTester от реализации механизмов ОС (работа с потоками, надстройками (plug-ins), низкоуровневых POSIX Sockets). Таким образом, система приобретает возможность перенесения на другие POSIX-совместимые ОС (Solaris и FreeBSD).

Библиотека модуля тестирования WebTester реализует следующие функции: инициализации/выгрузки модуля; активации/деактивации модуля; активации/деактивации процесса проверки решений; приёма новой задачи из интерфейса; приёма решения для проверки.

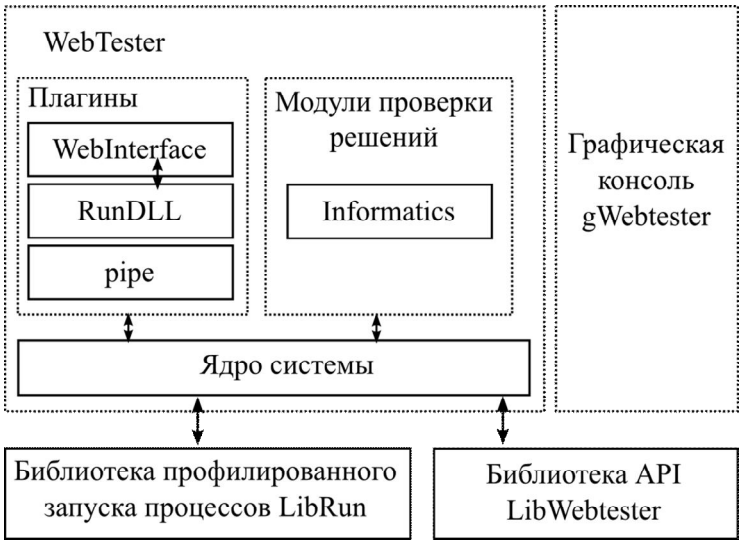

Рис. 2. Основные компоненты системы WebTester

MrtTester и MrtCompiler – программы автоматизации проверки таблиц МГТ. MrtTester реализует основную логику процесса проверки таблиц МГТ. MrtCompiler используется для вставки трассировочной информации в исходный программный код и его компиляции. В компонент MrtCompiler передаётся имя файла, содержащего программный код, переданный для проверки, и имя бинарного файла, в который помещается результат работы компонента. Основой программы проверки является компонент трассировки, т.е. сбора статистики об исполняемых участках кода проверяемой программы. Этот подход оказался наиболее подходящим для установленных условий. Компонент предобработки исходного кода основывается на лексическом и синтаксическом анализе программного кода. Семантический анализ требует более полной спецификации языка, что влечёт необходимость адаптации компонента предобработки исходного кода с одного диалекта языка на другой. В ходе проверки таблиц МГТ осуществляется анализ трассировочных данных после запуска проверяемой программы. Упрощённо процесс проверки МГТ происходит по следующей схеме: компиляция проверяемого решения; запуск проверяемого решения на наборе тестов, подготовленном студентом; запуск проверяемого решения на эталонном наборе тестов; возврат результата проверки. Отчёт об ошибках, возвращаемый в MrtTester, разделяется на две части: коды завершения процесса компилятора и буфера stdout/stderr этого процесса. Если в ходе проверки найдены ошибки или несоответствия, то генерируется подробное их описание и проверка прекращается, в противном случае проверка прекращается с результатом «Решение зачтено».

Выделение этих компонентов обеспечивает упрощение добавления новых языков и диалектов, поддерживаемых системой, возможность интеграции с другими программными средствами автоматизации проверки решений.

Кроме серверных компонентов, описанных выше, реализована программа StyleChecker, предназначенная для проверки текста проверяемых программ на соответствие заданным правилам стилевого оформления. Реализовано автономно работающее приложение, которое устанавливается на рабочих местах пользователей (учащихся), выполняющее проверку программ, написанных на языке Pascal. Осуществляется, в частности, проверка следующих правил: программа должна быть оформлена в одном стиле; каждый новый оператор должен находиться в своей строке; внутри одной конструкции не должно быть переносов, не предусмотренных соответствующим правилом для данной конструкции; все вложенные конструкции должны иметь отступы; количество строк с комментариями должно составлять не менее трети от всего количества строк в программе и пр. Проверка программы включает два основных этапа: на первом этапе выполняется обработка текста программы с целью превращения его в цепочки лексем определенного вида, которые можно обрабатывать независимо друг от друга; на втором – обработка полученных цепочек, сравнение их на соответствие заданным правилам. На следующем этапе предполагается интеграция StyleChecker с серверными компонентами комплекса.

Приложения комплекса прошли апробацию в учебном процессе в Пермском филиале ГУ-ВШЭ (на факультете бизнес-информатики) и в ПГУ (на механико-математическом факультете), в МОУ СОШ № 9 г. Перми, а также при проведении нескольких олимпиад по программированию, летних школ для учащихся Пермского края.

Литература

1. *Касьянова Е.В*. Адаптивные методы и средства поддержки дистанционного обучения программированию : дис. ... канд. физ.-мат. наук : 05.13.11 Новосибирск, 2006. – 181 с. РГБ ОД, 61:07-1/295.

2. *Промский А.В*. Формальная семантика C-LIGHT программ и их верификация методом Хоара : Дис. ... канд. физ.-мат. наук : 05.13.11 : Новосибирск, 2004. – 157 с. РГБ ОД, 61:05-1/309.

3. *Чернов А*. Система проведения турниров по программированию ejudge: [Электронный документ] (http://ejudge.ru/).