УДК 004.588+004.031.42

Автоматизированная система для проведения практических занятий по программированию

© С.Ю. Скоробогатов

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

В статье представлена автоматизированная система тестирования T-BMSTU, созданная в МГТУ им. Н.Э. Баумана для проверки программ, которые разрабатывают студенты в рамках проведения практических занятий по программированию. Описаны составные части системы, принципы их работы, их интеграция с внешним программным обеспечением. Уделено внимание вопросам развертывания системы. Рассмотрены методические и организационные аспекты ее внедрения в учебный процесс. Показаны преимущества и недостатки подготовки профессиональных программистов с использованием системы T-BMSTU. Обсуждаются возможные подходы к преодолению указанных недостатков, а также направления дальнейшего развития системы.

Ключевые слова: *автоматизированная система тестирования, наборы тестов, обнаружение некорректных заимствований.*

Введение. С каждым годом становится все больше российских и зарубежных университетов, внедряющих в образовательный процесс автоматизированные системы тестирования для подготовки профессиональных программистов. Этот процесс подстегивается развитием информационных технологий, увеличением доступности онлайн-сервисов, а также наметившейся в университетском образовании тенденцией к переносу акцента с привычных аудиторных занятий на самостоятельную работу студента, осуществляемую под контролем преподавателя.

В настоящей статье автоматизированной системой тестирования будем называть программно-аппаратный комплекс, выполняющий проверку программ, разработанных студентами IT-специальностей в рамках выполнения домашних заданий, лабораторных и практических работ.

Работа студента с автоматизированной системой тестирования заключается в решении предлагаемого системой набора задач. Решение задачи подразумевает составление программы на указанном языке программирования и отправку исходного текста этой программы в систему для автоматической проверки. Проверка правильности программы заключается в запуске ее на наборе тестов. В простейшем случае каждый тест представляет собой вариант входных данных для программы и правильный ответ, который ожидается на выходе программы. Если выходные данные программы не совпадают с правильным ответом, считается, что программа не прошла данный тест.

Развитие автоматизированных систем тестирования берет начало в олимпиадном программировании. Для проведения олимпиад были разработаны такие автоматизированные системы, как Ejudge [1], PCMS2 [2, 3], Contester [4], CATS [5]. Несмотря на успешное использование этих систем для проведения учебных занятий по программированию [6–8], отметим, что они ориентированы прежде всего на проведение соревнований и тренировок, в то время как системы, предназначенные для учебного процесса, должны быть ориентированы на учебные дисциплины и составляющие их модули.

Некоторые университеты применяют автоматизированные системы тестирования, специально созданные для проведения учебных занятий [9–11]. Так, в МГТУ им. Н.Э. Баумана была разработана и с 2010 г. используется и совершенствуется оригинальная система T-BMSTU, сочетающая автоматическую проверку студенческих программ с накоплением базы их исходных текстов и комментариев преподавателей. В отличие от олимпиадных систем база условий задач в системе T-BMSTU разбита по учебным дисциплинам и модулям и обеспечивает автоматический подсчет баллов, начисляемых студенту за успешно решенные задачи и формирующих его итоговую оценку по учебной дисциплине.

Структура автоматизированной системы тестирования T-BMSTU. Автоматизированная система тестирования T-BMSTU представляет собой распределенный программно-аппаратный комплекс, состоящий из трех компонентов: web-сервера, одного или нескольких серверов тестирования и сервера обнаружения некорректных заимствований.

Web-сервер является центральным компонентом автоматизированной системы. Он реализован в виде демона операционной системы Linux и предоставляет студентам и преподавателям онлайн-доступ к системе. Кроме того, web-сервер обеспечивает взаимодействие с серверами тестирования по специальному протоколу, реализованному поверх протокола WebSockets.

Условия задач в форматах HTML и MarkDown, а также наборы тестов для проверки решений располагаются в виде файлов и каталогов в файловой системе компьютера, на котором запущен web-сервер. Синхронизация условий и наборов тестов с компьютерами преподавателей, ответственных за их разработку, осуществляется через облачное хранилище Dropbox. Использование облачного хранилища также позволяет автоматически копировать наборы тестов на серверы тестирования.

Структура учебных курсов, отправленные решения и комментарии преподавателей и серверов тестирования сохраняются на web-сервере в реляционной базе данных. Отметим, что база данных хра-

нит всю информацию за весь период эксплуатации системы, т. е. студент четвертого курса имеет возможность решать задачи, которые он не успел решить, например, на первом курсе, а сервер обнаружения некорректных заимствований может выявлять решения, списанные студентами у своих старших товарищей.

Развертывание web-сервера T-BMSTU на новом компьютере, работающем под управлением операционной системы Linux, не требует больших усилий благодаря тому, что web-сервер не зависит от установленного на компьютере программного обеспечения, а его конфигурирование сводится к указанию путей к базе данных и каталогу условий задач с наборами тестов. Можно предположить, что исходный код web-сервера можно откомпилировать и запустить в любой современной UNIX-системе, для которой существуют компилятор языка Go и библиотека sqlite3 для управления базами данных. Например, нет никаких сомнений, что web-сервер будет работать в системах FreeBSD и MacOS X, хотя соответствующих экспериментов не проводили. Что касается запуска web-сервера в операционной системе Windows, то для этого потребуется внести в его исходный код незначительные изменения, связанные с обработкой системных сигналов.

Проверку решений, отправляемых студентом в систему T-BMSTU, осуществляет *сервер тестирования*. Сервер тестирования реализован в виде скрипта на языке Ruby. Он аутентифицируется на web-сервере как преподаватель, имеющий право просматривать решения студентов, отклонять неправильные решения и добавлять к ним комментарии.

Одновременно могут работать несколько серверов тестирования. Балансировка нагрузки между ними осуществляется путем «голосования»: через определенный интервал времени каждый свободный сервер тестирования опрашивает web-сервер, пытаясь получить исходный текст решения, которое нужно проверить. Таким образом, серверы тестирования, занятые проверкой, автоматически исключаются на время проверки из процесса распределения отправленных решений.

В настоящий момент сервер тестирования поддерживает семь языков программирования: C, C++, Pascal, Java, Go, Ruby и Scheme. Если полученное решение написано на языке, требующем компиляции, сервер тестирования вызывает соответствующий компилятор и получает либо исполняемый файл, либо набор сообщений об ошибках. В случае интерпретируемых языков Ruby и Scheme сервер тестирования вызывает интерпретатор языка в режиме поиска синтаксических ошибок. В любом случае решение, содержащее синтаксические ошибки, сразу отклоняется, и web-серверу посылает-

ся описание ошибок, которое добавляется в базу в качестве комментария к решению.

Запуск решения на каждом тесте выполняется в защищенном изолированном окружении. Запущенное решение не имеет доступа ни к сетевым интерфейсам, ни к файловой системе сервера. Более того, сервер тестирования контролирует время работы и количество использованной памяти для процесса, в котором запущено решение, а также для всех дочерних процессов, которые может породить решение. Сервер тестирования знает предельные значения времени работы и объема памяти для каждой задачи. В случае превышения этих предельных значений он прерывает выполнение решения и всех его дочерних процессов и посылает web-серверу сообщение с указанием номера теста, на котором произошло превышение.

Решение, написанное на небезопасном с точки зрения работы с памятью языке, например C, C++ и Pascal, дополнительно запускается в отладчике valgrind. Это позволяет выявить такие трудноуловимые ошибки, как выход за границы массива, неправильная работа с указателями, утечки памяти, использование неинициализированных переменных. При обнаружении этих ошибок решение отклоняется сервером тестирования, даже если оно выдает правильный ответ.

Содержимое стандартного потока вывода, получаемое в результате проверки решения на очередном тесте, сравнивается сервером тестирования с правильным ответом. По умолчанию сравнение осуществляется с точностью до пробельных символов. Однако автор задачи может оформить сколь угодно изощренный способ сравнения в виде отдельной программы, и тогда сервер тестирования будет делегировать сравнение этой программе.

Для упрощения развертывания сервера тестирования он оформлен в виде виртуальной машины в формате VirtualBox. На этой виртуальной машине установлена операционная система Linux и создано защищенное окружение, содержащее все библиотеки, которые могут потребоваться для выполнения решений. Тем самым сервер тестирования можно за несколько минут запустить на любом компьютере под управлением любой операционной системы, если для них существует версия VirtualBox.

Сервер обнаружения некорректных заимствований выполняет интеграцию системы T-BMSTU с внешним программным обеспечением, которое ведет поиск некорректных заимствований. Сервер работает на том же компьютере, что и web-сервер, и реализован в виде скрипта на языке Ruby. Он запускается по расписанию и осуществляет выемку решений из базы данных. Если с момента последнего запуска сервера в базе данных появились новые решения некоторой задачи, то все решения этой задачи передаются на вход установленным

в системе программам поиска некорректных зависимостей. Отчеты программ поиска некорректных зависимостей сохраняются в каталоге файловой системы, который синхронизирован через Dropbox с компьютером преподавателя, проверяющего решения.

В настоящее время сервер обнаружения некорректных зависимостей интегрирован с сервисом MOSS [12], предоставляемым Стэндфордским университетом, и с программой обнаружения плагиата Platypus [13].

Преимущества использования системы тестирования T-BMSTU. С организационной и экономической точки зрения автоматизированная система тестирования позволяет снизить нагрузку на преподавателя. Качественная проверка решения студента – это итерационный процесс, который продолжается до получения корректной программы. При проверке вручную на каждой итерации преподаватель вынужден выявлять ошибки путем изучения исходного кода и ручного запуска программы на наборах входных данных. Автоматическая проверка не исключает чтения исходного кода преподавателем, но в несколько раз сокращает количество итераций и избавляет от рутинных операций, таких как компиляция исходных текстов программы и сравнение результатов работы программы с эталонными результатами.

Отметим, что сервер тестирования в составе системы T-BMSTU может отклонять решения студентов, но не имеет права их принимать. Если решение прошло все тесты, оно помечается как проверенное сервером тестирования и поступает на проверку к преподавателю. Преподаватель оценивает аккуратность оформления, наличие комментариев, правильность декомпозиции программы на классы или функции и выносит окончательный вердикт.

Как известно, чем полнее тематика лекций подкреплена практическими занятиями, тем лучше студент усваивает учебный курс. Обычно количество предлагаемых студенту задач ограничено возможностями преподавателя. Снижение нагрузки на преподавателя, достигаемое благодаря использованию автоматизированной системы тестирования, снимает это ограничение. С внедрением системы T-BMSTU в МГТУ им. Н.Э. Баумана количество задач стало лимитироваться уже не нагрузкой преподавателя, а способностью студента решить предложенные ему задачи.

Отметим, что автоматизированная система позволяет проверить студенческое решение более полно, чем это может сделать преподаватель. Например, в системе T-BMSTU каждая студенческая программа запускается в среднем на 50 тестовых наборах входных данных, причем составители тестовых наборов стараются обеспечить покрытие всех частных и предельных случаев работы программы.

Более того, благодаря точному измерению сервером требуемых решением ресурсов можно так составить тестовые данные, что решения, в которых реализован неэффективный алгоритм, будут автоматически отклоняться системой из-за превышения предельных значений времени выполнения или объема используемой памяти.

Эффективное использование автоматизированной системы тестирования подразумевает ее доступность через Интернет круглосуточно семь дней в неделю. При этом система может параллельно обслуживать большое число пользователей. Это уменьшает время получения студентом отклика на внесенные исправления, благодаря чему работа студента над решением многократно ускоряется. Кроме того, круглосуточная доступность системы тестирования через Интернет позволяет сглаживать неравномерность нагрузки преподавателя в течение семестра. Эта неравномерность связана с тем, что при приближении контрольных точек, таких как окончание дисциплинарного модуля или зачет по дисциплине, количество решений, поступающих на проверку, неизбежно возрастает.

Другим важным аспектом использования автоматизированной системы тестирования является накопление всех решений студентов в единой базе данных. Эти решения могут потребоваться, например, при проведении аккредитационной экспертизы университета, однако организовать их ручной сбор и надежное хранение сложно. Кроме того, именно наличие базы студенческих решений позволяет эффективно выявлять в них некорректные заимствования.

Полезным побочным эффектом использования автоматизированной системы тестирования является накопление в ней большого количества условий задач с тщательно составленными наборами тестов для каждой задачи. Сейчас в системе T-BMSTU собрано около 250 задач. Хотя этого количества еще недостаточно для появления эффекта повторного использования уже имеющихся задач при разработке новых учебных курсов, можно ожидать, что потенциально этот эффект позволит еще больше снизить нагрузку на преподавателя.

Негативные последствия применения системы T-BMSTU. В процессе эксплуатации системы T-BMSTU были выявлены негативные последствия ее применения, которые, видимо, в той или иной мере присущи всем системам автоматизированного тестирования.

Прежде всего, работая с системой, студент не обучается самостоятельной проверке решений. Эта проблема может быть решена двумя путями: во-первых, сокрытием тестовых данных для части задач; во-вторых, с помощью «обратных» задач, в которых от студента требуется написать такой генератор тестов, чтобы правильное решение задачи проходило полученные тесты, а неправильное — не проходило.

Кроме того, оказалось, что небольшая часть студентов пренебрегает инструментальными средствами разработки, предпочитая редактирование и запуск решения непосредственно через web-интерфейс системы тестирования. Возможное решение проблемы — снижение оценки, если студент превысил определенное число попыток отправки решения.

Заключение. Опыт четырех лет эксплуатации системы T-BMSTU показал, что при внедрении автоматизированной системы в учебный процесс увеличивается количество предлагаемых в рамках учебного курса задач, повышается тщательность проверки решений, взаимодействие преподавателя и студента становится более удобным, а также появляется возможность обнаружения некорректных заимствований в текстах решений. Учитывая, что использование автоматизированной системы кардинально снижает нагрузку на преподавателя, можно утверждать, что она позволяет вывести практические занятия по программированию на качественно новый уровень без увеличения числа занятых преподавателей.

Дальнейшее развитие системы T-BMSTU предполагает добавление следующих отсутствующих в текущей версии возможностей:

- поддержка решений, исходные тексты которых состоят из нескольких файлов, и интеграция с такими онлайн-сервисами хостинга IT-проектов, как github;
- показ результатов обнаружения некорректных зависимостей непосредственно в web-интерфейсе системы;
- введение вариативности в условия задач;
- проверка правильности расстановки отступов в исходных текстах решений и доступное только преподавателю автоматическое форматирование исходных текстов;
- автоматическая генерация учебно-методических пособий из условий задач, представленных в системе.

ЛИТЕРАТУРА

[1] *Ejudge Contest Management System*. URL: https://ejudge.ru (дата обращения 12.10.2014).

[2] *Система PCMS2 проведения соревнований по спортивному программированию Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики*. URL: http://neerc.ifmo.ru/trains/information/software.html (дата обращения 12.10.2014).

[3] Станкевич А.С. Общий подход к проведению итогов соревнований по программированию при использовании различных систем оценки. *Компьютерные инструменты в образовании,* 2011, № 2, с. 27–38.

[4] *Система для проведения турниров и индивидуального решения задач по олимпиадному программированию Contester*. URL: http://www.contester.ru (дата обращения 12.10.2014).

[5] *Система автоматического тестирования программ и организации соревнований по программированию*. URL: http://imcs.dvfu.ru/works/work?wid=1203 (дата обращения 12.10.2014).

[6] Филинов А.Н. Система автоматического тестирования Ejudge. *Информатика и образование,* 2012, № 9, с. 63–64.

[7] Попова Е.Н., Трошина Е.Н. Автоматизация контроля знаний. *Вестник Моск. гос. открытого ун-та. Сер. Общественно-политические и гуманитарные науки*, 2010, № 2, с. 69–73.

[8] Пантелеев Е.Р., Архипов А.Л., Второв А.В., Ильина Е.В. Интеграция инструментов контроля навыков программирования в среду интернет-обучения. *Вестник ИГЭУ*, 2010, № 3, с. 104–108.

[9] Алыкова А.Л. Особенности автоматизированного тестирования знаний студентов в области программирования. *Вестник ИГЭУ*, 2005, вып. 4, с. 4–7.

[10] Верещагин А.Г. Автоматизация тестирования программ как средство повышения эффективности учебного процесса. *Известия МГИУ*, 2012, № 3 (27), с. 28–41.

[11] Лаптев В.В., Морозов А.В. Автоматизированная система для контроля лабораторных работ по программированию. *Известия ВолгГТУ*, 2011, т. 11, № 12, с. 92–95.

[12] Schleimer S., Wilkerson D.S., Aiken A. Winnowing: Local Algorithms for Document Fingerprinting. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data 2003*. ACM Press, 2003, pp. 76–85.

[13] *Platypus. On plagiarism or similarity detection*. URL: https://code.google.com/p/ouspg/wiki/Platypus (дата обращения 12.10.2014).

Статья поступила в редакцию 03.10.2014

Ссылку на эту статью просим оформлять следующим образом:

Скоробогатов С.Ю. Автоматизированная система для проведения практических занятий по программированию. *Инженерный журнал: наука и инновации*, 2014, вып. 11.

URL: http://engjournal.ru/catalog/pedagogika/hidden/1330.html

Скоробогатов Сергей Юрьевич родился в 1977 г., окончил МГТУ им. Н.Э. Баумана, в 2003 г. Заместитель заведующего кафедрой «Теоретическая информатика и компьютерные технологии» МГТУ им. Н.Э. Баумана. Автор 10 публикаций. Специализируется в области разработки компиляторов и интерпретаторов языков программирования. e-mail: skorobogatov@bmstu.ru

Automated system for practical classes on programming

© S.Yu. Skorobogatov

Bauman Moscow State Technical University, Moscow, 105005, Russia

The article presents an automated testing system T-BMSTU, created in BMSTU to check programs that students develop as part of the practical training in programming. System components, their working principles, and their integration with external software are described. Attention is paid to the deployment of the system. Methodical and organizational aspects of its introduction to the learning process are considered. Since use of the framework in professional programmers' training has both advantages and disadvantages, we offer possible approaches to overcoming of the disadvantages as well as directions of further development.

Keywords: *automated testing system, test kits, detection of incorrect borrowings.*

REFERENCES

[1] *Ejudge Contest Management System*. Available at: https://ejudge.ru (accessed on 12.10.2014).

[2] *Sistema PCMS2 provedeniya sorevnovaniy po sportivnomu programmirovaniyu Sankt-Peterburgskogo natsionalnogo issledovatelskogo universiteta informatsionnykh tekhnologiy, mekhaniki i optiki* [System PCMS2 to hold competitions for sports programming of St. Petersburg National Research University of Information Technologies, Mechanics and Optics]. Available at: http://neerc.ifmo.ru/trains/information/software.html (accessed on 12.10.2014).

[3] Stankevich A.S. Станкевич А.С. Kompyuternye instrument v obrazovanii — Computer Tools in Education, 2011, no. 2, pp. 27–38.

[4] *Sistema* Contester *dlya provedeniya turnirov i individualnogo resheniya zadach po olimpiadnomu programmirovaniyu* [System Contester for Tournaments and Individual Solution of Problems in the Olympiad Programming].. Available at: http://www.contester.ru (accessed on 12.10.2014).

[5] *Sistema avtomaticheskogo testirovaniya program i organizatsii sorevnovaniy po programmirovaniyu* [System of program automatic testing and of competition organization on Programming]. Available at: http://imcs.dvfu.ru/works/work?wid=1203 (accessed on 12.10.2014).

[6] Filinov A.N. *Informatika i obrazovanie — Informatics and Education*, 2012, no. 9, pp. 63–64.

[7] Popova E.N., Troshina E.N. *Vestnik Moskovskogo gosudarstvennogo otkrytogo universiteta. Seriya Obschestvenno-politicheskie I gumanitarnye nauki — Bulletin of the Moscow State Open University. Series Social, Political and Human Sciences*, 2010, no. 2, pp. 69–73.

[8] Panteleyev E.R., Arkhipov A.L., Vtorov A.V., Ilina E.V. *Vestnik IGEU — Vestnik IGEU*, 2010, no. 3, pp. 104–108.

[9] Alykova A.L. *Vestnik IGEU — Vestnik IGEU*, 2005, no. 4, pp. 4–7.

[10] Vereschagin A.G. Izvestiya MGIU — *Proceedings of MGIU*, 2012, no. 3 (27), pp. 28–41.

[11] Laptev V.V., Morozov A.V. *Izvestiya VolgGTU — Proceedings of V*STU, 2011, vol. 11, no. 12, pp. 92–95.

[12] Schleimer S., Wilkerson D.S., Aiken A. Winnowing: Local Algorithms for Doc-ument Fingerprinting. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data 2003*. ACM Press, 2003, pp. 76–85.

[13] *Platypus. On plagiarism or similarity detection*. Available at: https://code.google.com/p/ouspg/wiki/Platypus (accessed on 12.10.2014).

Skorobogatov S.Yu. (b. 1977) graduated from the Bauman Moscow State Technical University in 2002. Deputy Head of the Department "Computer Science and Technologies" at the Bauman Moscow State Technical University. Author of 10 publications. Specializes in the field of compiler construction. e-mail: skorobogatov@bmstu.ru