УДК 004.31

Интеграция инструментов контроля навыков программирования в среду интернет-обучения

Пантелеев Е.Р., д-р техн. наук, Архипов А.Л., асп., Второв А.В., инж., Ильина Е.В., студ.

Предложена модель интеграции системы интернет-обучения с системой автоматизированного тестирования программ. Обоснован выбор технологической платформы реализации предложенной модели. Рассмотрена архитектура программно-информационного комплекса интеграции. Приведен пример построения системы тестов, обеспечивающей тонкую диагностику ошибок для формирования обучающих воздействий.

Ключевые слова: компьютерное обучение, формирование навыков программирования.

Integration of Programming Skills Control Tools into Internet Learning Environment

E. R. Panteleev, Doctor of Engineering, A. L. Arkhipov, Post-Graduate Student, A. V. Vtorov, Engineer, E. V. Ilyina, Student.

Model of integration of the internet learning management system with automated program code tester is proposed. The choice of the implementation technological basis is justified. Software Architecture of integration data complex is considered. An example of test data construction is given. It provides fine error diagnostics to create training impact.

Keywords: computer learning, programming skills build-up.

Освоение навыков алгоритмизации и программирования стало неотъемлемой частью подготовки дипломированного специалиста высшей школы по любому из направлений (специальностей). Это связано с тем, что информационные технологии являются критически важным инструментом решения научных и практических задач в любой области знаний. Однако формирование таких навыков возможно лишь путем активного вовлечения студента в процесс обучения [1], целенаправленной и длительной тренировки, в ходе которой студент трансформирует постановку задачи в алгоритм ее решения, кодирует этот алгоритм на одном из языков программирования, выполняет отладку программы, предъявляет ее для контроля, анализирует ошибки и исправляет их. Слабым звеном в этом цикле формирования навыков алгоритмизации и программирования является процедура проверки правильности решения. Как показывает практика, трудоемкость подобной проверки с привлечением преподавателя в качестве эксперта в несколько раз выше, чем, например, при контроле правильности решения задач по физике или математике [2], хотя проблема трудозатрат, особенно при массовом контроле знаний, существует и в этих дисциплинах. При отсутствии устных комментариев автора, например, при заочной форме обучения, подобная проверка решения еще более затруднена, а иногда и невозможна. Поэтому в последнее время для контроля правильности решения применяется автоматизированное тестирование (АТ) программ методом «черного ящика». Идея АТ состоит в том, что независимо от выбранного разработчиком алгоритма решения и способа его программной реализации алгоритм должен правильно преобразовывать входные данные задачи в выходные. Следовательно, многократное исполнение программы с различными наборами входных данных и проверка совпадения результата с контрольными наборами выходных данных могут с большей или меньшей степенью вероятности подтвердить правильность (ошибочность) авторского решения. В простейших случаях (например, проверка принадлежности точки плоской фигуре в задании С1 ЕГЭ по информатике и ИКТ) АТ может быть исчерпывающим. Однако в подавляющем большинстве случаев разработчик тестов вынужден ограничивать количество проверок, оставляя лишь те из них, которые способствуют выявлению ошибок, типичных для данного класса задач, а также ошибок, проявляющихся на предельных значениях данных. Тем не менее использование АТ, по сравнению с «живой» экспертизой, позволяет получить более объективную оценку правильности решения, и соответствующее программное обеспечение широко применяется как при проведении олимпиад по информатике, так и в процессе подготовки к ним [3–5]. АТ в данном случае ориентировано на выявление победителя (участника, решения которого успешно прошли наибольшее количество тестов) и не преследует цели обучения (формирования навыков алгоритмизации и программирования). Хотя такая возможность, в принципе, существует, так как программное обеспечение АТ для каждой задачи обычно возвращает результаты прохождения всех тестов. Если тесты спроектированы таким

образом, что каждый из них выявляет определенную ошибку, информация о типах обнаруженных ошибок может быть доведена до автора решения и/или использована для формирования соответствующего учебного воздействия (контекстная подсказка, ссылка на релевантный материал, новая задача, сложность которой определяется успехом или неудачей выполнения предыдущей). Однако функцию информирования программные средства АТ не поддерживают. С другой стороны, программные средства компьютерного обучения способны формировать учебное воздействие в контексте результата выполнения контрольного задания, но ограничены набором шаблонных схем формирования задания и проверки ответа (выбор варианта ответа, упорядочение или сопоставление элементов ответа, сравнение ответа с эталонным числом или строкой). В результате их возможности ограничены контролем усвоения фактического материала. Поэтому задача создания компьютерных средств обучения, объединяющих возможности контроля навыков алгоритмизации и программирования с формированием учебных воздействий в функции ошибок, допущенных при выполнении контрольного задания, исключительно актуальна, особенно при подготовке студентов по заочной форме обучения. Ниже рассматривается один из возможных подходов к ее решению, основанный на использовании сервис-ориентированной интеграции двух интернет/интранет приложений – среды интернет-обучения ГИПЕРТЕСТ [6] (далее – ГИПЕРТЕСТ) и системы автоматизированного тестирования программ eJudge [5] (далее – eJudge).

Основная идея предлагаемого подхода состоит в программном «склеивании» двух абсолютно независимых сетевых приложений при помощи web-сервиса – глобально доступной библиотеки подпрограмм со специфицированным стандартизованным интерфейсом [7]. Для того чтобы приложение-клиент могло воспользоваться web-сервисом, ему достаточно иметь в своем распоряжении описание этого сервиса на языке WSDL (web service description language – язык описания web-сервисов) и среду программирования, которая поддерживает вызовы web-сервисов (PHP, Java, ASP.NET, Delphi). Некоторые приложения, такие как, например, MATLAB, изначально предоставляют клиентам возможности доступа к своим ресурсам по сети Интернет. Обеспечивающий такой доступ сервис-посредник MATLAB Web Server позволяет разрабатывать интернет-приложения с использованием стандартных компонентов MATLAB [8, 9]. Однако eJudge подобных решений не предлагает, поэтому для «склеивания» необходимо разработать сервис интеграции (СИ), который обеспечит согласованную работу этих двух приложений в процессе прохождения теста[1] в рамках следующего сценария (табл. 1).

Анализ взаимодействия интегрируемых компонентов в процессе тестирования (табл. 1) показывает, что роль СИ заключается в обеспечении опосредованного взаимодействия ГИПЕРТЕСТ и eJudge в процессе отправки готового решения и его проверки с применением АТ. Контекст этого взаимодействия составляет пара идентификаторов «студент – задача».

С точки зрения ГИПЕРТЕСТ сервис-посредник должен обеспечивать интерфейсные функции выдачи контрольного задания и приема готового ответа. Выдача контрольного задания – рутинная операция, в результате выполнения которой ГИПЕРТЕСТ получает от СИ по заданному идентификатору задачи фрагмент html-кода, содержащий формулировку задания. СИ выбирает задание из базы, сформированной на подготовительном этапе. А вот функция приема готового ответа действительно принимает в качестве входных параметров код программы и идентификатор языка программирования, на котором она написана, но не возвращает, как можно предположить, оценку решения, так как проверка решения выполняется приложением eJudge, а СИ лишь запрашивает этот ресурс. Поэтому СИ возвращает приложению ГИПЕРТЕСТ только признак «ответ находится на обработке». Сама же оценка решения будет получена через неопределенный интервал времени, когда eJudge выполнит компиляцию кода программы и проверит ее правильность с помощью имеющихся в его базе тестов этой программы. Но так как функция приема ответа представляет собой выполняемый на сервере сценарий и время его исполнения ограничено, чтобы исключить возможность «зависания» сервера, реально выполняемое функцией приема ответа действие заключается в том, что полученный ответ ставится в очередь на обработку.

Дальнейшую обработку размещенных в очереди ответов производит другой компонент СИ – менеджер очереди (МО). Алгоритм его действий определяется спецификой интерфейса eJudge, ориентированного на взаимодействие с клиентской программой просмотра (браузером) по протоколу http (рис. 1). Особенность этого протокола в том, что сервер (eJudge), отправив клиенту (браузер или сценарий МО) ответ на его запрос, «забывает» о нем. Поэтому клиент вынужден напоминать о себе. Пользователь браузера обновляет страницу результатов турнира, периодически нажимая клавишу F5 (запрос на выгрузку страницы). Сценарий МО также выполняет этот запрос, предварительно выбрав еще не оцененные ответы, которые ожидают обработки в очереди. Затем МО выполняет разбор полученной по запросу в формате html таблицы результатов турнира. Оцененные ответы из очереди удаляются, а результат передается приложению ГИПЕРТЕСТ (табл. 1, п.5).

[1] Взаимодействие компонентов на этапе разработки теста в статье не рассматривается

Таблица 1. **Взаимодействие интегрируемых компонентов в процессе тестирования**

№ п/п	Операция	Функции участников			
		Браузер студента	ГИПЕРТЕСТ	Сервис интеграции	eJudge
1	Получение контрольного задания		Клиент	Сервер	
2	Отображение задания и подготовка решения	Клиент	Сервер		
3	Отправка решения на проверку	Клиент	Сервер/Клиент	Сервер	
4	Ожидание оценки решения			Клиент	Сервер
5	Формирование учебного воздействия		Сервер	Клиент	
6	Интерпретация учебного воздействия	Клиент	Сервер		

С точки зрения СИ, выступающего клиентом этой операции, ГИПЕРТЕСТ представляет собой сервис, обеспечивающий операцию сохранения оценки в базе результатов тестирования. Как серверный сценарий, МО выполняет описанные выше действия однократно. Однако, поскольку время обработки ответа сервером eJudge непредсказуемо, сценарий МО необходимо активизировать многократно. В качестве такого «будильника» в предлагаемой архитектуре СИ используется планировщик заданий операционной системы (WindowsScheduler в ОС Windows или cron в GNU-системах).

Планировщик активизирует МО через заданный временной интервал, обеспечивая таким образом эмуляцию «поведения» пользователя браузера. Представленный здесь сценарий взаимодействия иллюстрирует диаграмма потоков данных (рис. 2).

Помимо описанного сценария, интерес представляют также возможности настройки интегрируемых компонентов, обеспечивающие «тонкую» диагностику ошибок студента и формирование по результатам этой диагностики соответствующих учебных воздействий.

По умолчанию ejudge признает решение верным, если программа корректно работает на всех тестах, в противном случае решение считается ошибочным. Однако такое поведение системы – это лишь один из возможных вариантов. Для описания других вариантов рассмотрим используемый eJudge принцип проверки решений более подробно. При проверке решения в первую очередь выполняется компиляция исходного кода программы. Если компиляция завершается ошибкой, то ejudge возвращает пользователю код «Ошибка компиляции». Следующим шагом является проверка решения на заранее подготовленном наборе тестов. Каждый тест представляет собой набор входной информации, набор эталонной (правильной) выходной информации и набор дополнительной (при необходимости) информации для проверки теста. eJudge предлагает два способа проверки – либо до первой ошибки, после которой проверка прекращается и решение признается неверным, либо с независимой проверкой решения на всех тестах. Для формирования учебных воздействий по результатам проверки решения предпочтителен второй способ.

Проверка решения на каждом из тестов может завершиться либо одним из сообщений об ошибке («Превышение времени выполнения», «Превышение лимита памяти», «Ошибка шага выполнения», «Ошибка формата выходных данных»), либо результатом сравнения выходного файла программы с эталоном. Для этого ejudge запускает утилиту контроля, которая может быть как типовой, так и специальной (для каждой задачи). Если результат не совпадает с эталоном, возвращается код «Неверный ответ», иначе – код «ОК». При этом есть возможность вернуть не только код ответа, но и текстовое сообщение (в предположении, что каждый тест выявляет определенную ошибку), комментирующее эту ошибку (тип ошибки и способ ее исправления). Глубина методической помощи пользователю в данном случае ограничена лишь сложностью написания утилиты контроля.

Для иллюстрации описанных выше возможностей АТ рассмотрим набор тестов для проверки одного из заданий ЕГЭ по информатике уровня С [10] (рис. 3, условие адаптировано для проверки с помощью автоматизированной системы).

В данной программе необходимо исправить две ошибки. Во-первых, есть область, для которой программа отвечает «принадлежит», тогда как точка не принадлежит указанной области, а во-вторых, есть две области, для которых программа ничего не отвечает, – одна задается невыполненным первым условием, а вторая – невыполненным вторым условием. Кроме того, необходимо проверить, что доработанная программа не стала работать неправильно там, где до доработки работала правильно. Исчерпывающее АТ доработанной программы можно выполнить при помощи набора тестов, представленного в табл. 2.

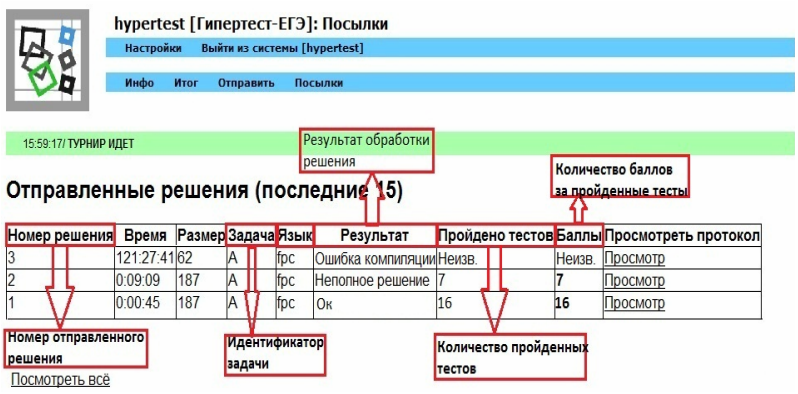

Рис. 1. Протокол результатов соревнований eJudge

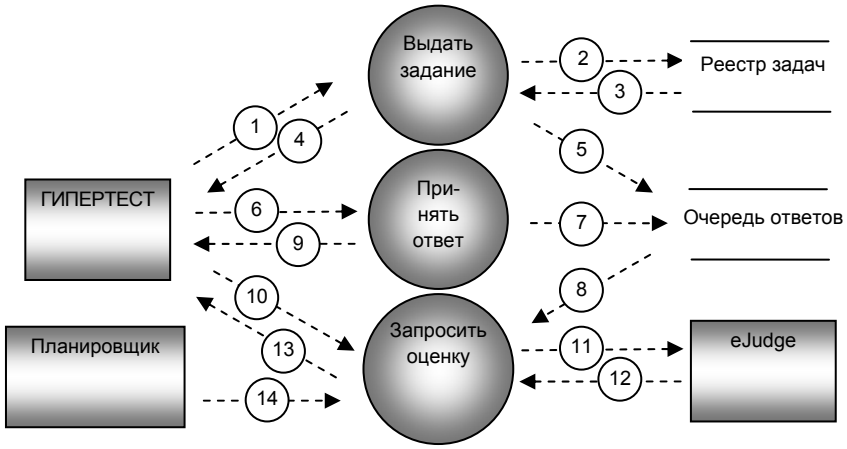

Рис. 2. Взаимодействие интегрируемых компонентов в рамках сценария формирования и оценки ответа: 1 – идентификация, данные теста; 2 – данные теста; 3 – задача; 4 – идентификатор задания, задание; 5 – идентификатор задачи; 6 – идентификация, идентификатор задания, ответ; 7 – идентификатор задания, ответ; 8 – идентификатор задания, ответ; 9 – подтверждение получения ответа; 10 – идентификатор задания, ответ; 11 – идентификация, № задачи, ответ; 12 – результат обработки; 13 – идентификация, идентификатор задания, оценка; 14 – время запуска

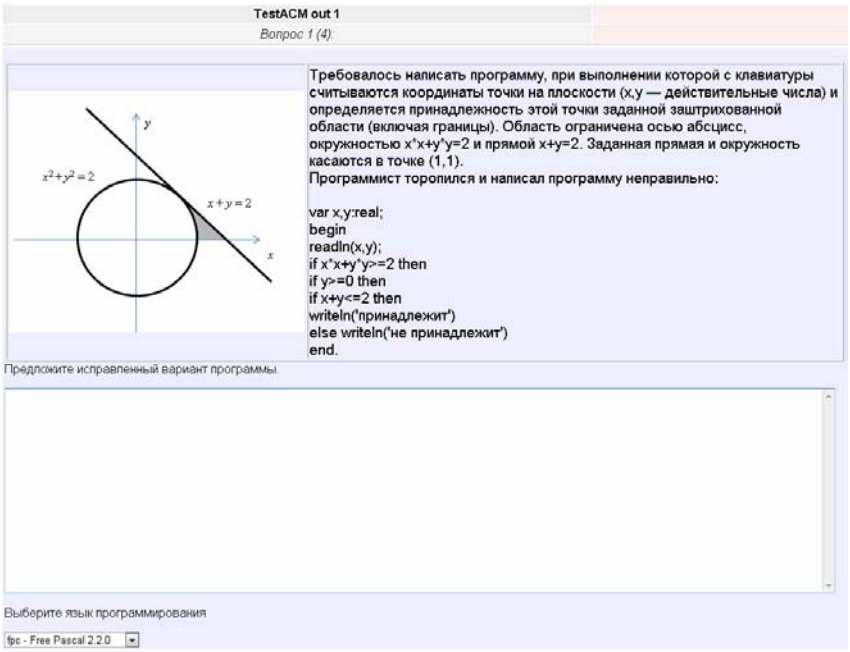

Рис. 3. Представление задания на разработку программы клиенту ГИПЕРТЕСТ

Таблица 2. **Исчерпывающий набор диагностических тестов**

№ п/п	Группа тестов	Тип выявляемой ошибки
1	(1,1), (2,0), (1.5,0), (1.5, 0.3)	Для точек на границе заданной области и внутри нее программа продолжает отвечать «принадлежит» (как и исходная)
2	(1,5), (10,10), (0,10)	Для точек, для которых исходная программа отвечала «не принадлежит», доработанная отвечает то же самое
3	(0,2), (0,1.5), (0.3, 1.5)	Для точек, для которых исходная программа ошибочно отвечала «принадлежит», доработанная отвечает «не принадлежит»
4	(2,-10), (10,-10), (-2,-10)	Для точек, для которых исходная программа ничего не отвечала (второе условие), доработанная отвечает «не принадлежит»
5	(0,0), (0,1), (1,0)	Для точек, для которых исходная программа ничего не отвечала (первое условие), доработанная отвечает «не принадлежит»

Чтобы констатировать отсутствие ошибки определенного типа, необходимо правильно пройти все тесты соответствующей группы. По итогам АТ можно:

- определить количество набранных первичных баллов за решение;
- сформировать учебное воздействие в соответствии с типом ошибки.

Для подсчета количества набранных первичных баллов в данном случае используется следующий алгоритм. Если не пройдены тесты групп 1 или 2, то задача признается решенной полностью неверно (0 баллов). Если не пройдены тесты группы 3, но пройдены тесты групп 4, 5 или пройдены тесты группы 3, но не пройдены тесты группы 4 или 5, то задача признается частично решенной (1 балл). Если пройдены все группы тестов, то задача признается полностью решенной (2 балла).

Очевиден в данном случае и алгоритм формирования учебного воздействия в форме контекстной подсказки (в ГИПЕРТЕСТ предусмотрена возможность подсказки при прохождении теста в тренировочном режиме). Например, если не пройдены тесты группы 5, возможна подсказка типа «Для точек, лежащих вне окружности, программа должна выдать сообщение «не принадлежит»».

Таким образом, в результате проведенных исследований выполнено следующее:

- разработана архитектура интегрированного комплекса, обеспечивающего компьютерный контроль правильности решения задач по программированию и формирование учебных воздействий в функции допущенных студентом ошибок;
- реализован web-сервис интеграции систем интернет-обучения ГИПЕРТЕСТ и автоматизированного тестирования программ eJudge, обеспечивающий согласованную работу этих двух систем в процессе выполнения сценария контроля навыков программирования;
- web-сервис интеграции испытан на примере контроля правильности решения задач уровня С ЕГЭ по информатике и ИКТ.
- продемонстрирован пример построения пакета тестов, обеспечивающих тонкую диагностику ошибок и формирование контекстной подсказки.

Список литературы

1. **Eman M. El-Sheikh.** Techniques for engaging students in an online computer programming course // Journal of Systemics, Cybernetics and Informatics Volume 7 – Number 1 – Year 2009 Pages: 1-12 (http://iiisci.org/journal/CV$/sci/pdfs/ZE036MI.pdf)

2. **Веретенников М.В.** Автоматизация проверки компьютерных программ в технических дисциплинах: Сб. науч. тр. «Дистанционные образовательные технологии. Пути реализации». Вып. 1. – Томск: Изд-во ТУСУР, 2004 (http://www2.tcde.ru/?43680).

3. **Васильев В.Н., Парфенов В.Г.** Командный чемпионат мира по программированию ACM 1998/1999. – СПб.: СПбГИТМО(ТУ), 1998.

4. **Чернов А.В.** Система проведения соревнований ejudge: Справочное руководство. Версия ejudge – 2.2.0 Версия документации – 20060304. – М., 2006.

5. **Чернов А.В.** Система тестирования ejudge. конференция «Свободное программное обеспечение в высшей школе» 27–28 января 2007 года. – Переславль-Залесский, 2007.

6. **Среда** разработки программ дистанционного обучения и профильного тестирования ГИПЕРТЕСТ: инструментальные средства / Е.Р. Пантелеев, И.А. Ковшова, И.В. Малков и др. // Информационные технологии. – 2001. – № 8. – С. 34–40.

7. **Эрик Ньюкомер**. Веб-сервисы: XML, WSDL, SOAP и UDDI Understanding Web Services: XML, WSDL, SOAP and UDDI Серия: Для профессионалов. – СПб.: Изд-во: Питер, 2003.

8. **Котельников И.А., Черкасский В.С.** MATLAB Web Server: вычисления в Интернете // Exponenta Pro. – 2004. – № 1 (5). – С. 4–9.

9. **Пантелеев Е.Р., Пекунов В.В., Первовский М.А.** Распределенная компонентная модель тестов в СДО ГИПЕРТЕСТ // Информационные технологии. – 2004. – № 8. – С. 41–46.

10. **ЕГЭ 2010.** Информатика. Типовые тестовые задания / П.А. Якушкин, В.Р. Лещинер, Д.П. Кириенко. – М.: Изд-во «Экзамен», 2010 (Серия «ЕГЭ 2010. Типовые тестовые задания»).

Пантелеев Евгений Рафаилович (контактное лицо),
ГОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
доктор технических наук, профессор кафедры программного обеспечения компьютерных систем,
адрес: 153021, Иваново, ул. Ясной Поляны, д. 11, кв. 6,
телефон/факс: (4932) 26-98-26,
E-mail: erp@poks.ispu.ru

Архипов Ален Леонидович,
ГОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
программист,
адрес: 153022, Иваново, ул. Лежневская, д. 116а, кв. 40,
телефон/факс: (4932) 26-98-26,
E-mail: alain@cmko.ispu.ru

Второв Алексей Владимирович,
ГОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
программист,
телефон/факс: (4932) 26-98-26,
E-mail: wert@ionb.ru

Ильина Евгения Вадимовна,
ГОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
студентка,
адрес: 153048, Иваново, ул. Ульяновская, д. 58, кв. 113,
телефон/факс: (4932) 26-98-26,
E-mail: ilevva@mail.ru