К ВОПРОСУ ИСПОЛЬЗОВАНИЯ ЯЗЫКА PYTHON ПРИ ИЗУЧЕНИИ МАТЕМАТИЧЕСКИХ ДИСЦИПЛИН СТУДЕНТАМИ ИТ-СПЕЦИАЛЬНОСТЕЙ

А. М. Кадан

Гродненский государственный университет имени Янки Купалы
Гродно, Беларусь
E-mail: *kadan@mf.grsu.by*

Представлены методология использования и компоненты интернет-системы тестового контроля знаний, применяемой при изучении естественнонаучных дисциплин студентами, специализирующимися в области ИТ и программирования. К особенностям предлагаемой системы относятся использование интерпретируемого языка программирования Python и отсутствие требования точного соответствия ответа испытуемого эталонному ответу.

There are presented the methodology of use and components of Internet-system of test knowledge control used in the study of science disciplines by students specializing in IT and programming. The special features of the proposed system include the use of the interpreted Python programming language and the absence of the requirement of exact compliance of the answer of examinee to the reference answer.

Ключевые слова: обучение программированию, удаленная проверка программ, Python, интерпретируемые языки программирования.

Keywords: programming training, remote verification of programs, Python, interpreted programming languages.

ВВЕДЕНИЕ

Не вызывает споров, что при подготовке специалистов в области ИТ, программирования, прикладной математики количество и качество задач, самостоятельно решенных студентом, в рамках учебной дисциплины определяет уровень качества усвоенного материала. Также важно, чтобы изучение теоретического материала сопровождалось демонстрацией приемов применения изучаемых положений для решения практических задач из различных предметных областей, с обязательным и постоянным контролем полноты и качества предлагаемых решений.

Опыт работы со студентами специальностей, в квалификации которых присутствует сочетание «инженер-программист», показывает, что усвоение материала и интерес предмету естественно-научного содержания существенно повышаются если преподавание материала ведется через использование элементов программирования: от решения отдельных независимых примеров и задач до написания небольших библиотек подпрограмм и исследовательских проектов, связанных с отдельными темами изучаемой дисциплины.

ПРИМЕР ПРЕДМЕТНОЙ ОБЛАСТИ

Традиционно для организации всестороннего тестирования большого числа учебных примеров целесообразно использование систем автоматизированной проверки ответов на основе заранее подготовленных контрольных тестов. Ситуация, когда изучение дисциплины предполагает контроль знаний учащихся не только в усвоении теоретического материала (который может быть реализован выбором правильных вариантов ответов), но требует анализа приведенного ответа, типична при изучении математических дисциплин, где важен как сам ответ,

так и правильность цепочки рассуждений, позволяющих его получить. Причем представление сделанных испытуемым рассуждений заранее не регламентировано.

К примеру, решение задачи нахождения энтропии из курса «Теория информации» с использованием языка Python, представленной в следующем виде:

```
# Вычислить энтропию дискретной случайной величины,
# представленной вектором вероятностей ее значений.
            # Результат - в бит/симв
   P = {‘x0’:1/2, ‘x1’:1/4, ‘x2’:1/8, ‘x3’:1/8}
                      H = ...
```

может быть дано несколькими принципиально разными способами, использующими различные вычислительные аспекты языка Python и различный формат представления результата.

Вариант правильного ответа 1

```
   P = {‘x0’:1/2, ‘x1’:1/4, ‘x2’:1/8, ‘x3’:1/8}
                    H = 1.75
```

Вариант правильного ответа 2

```
                from math import log
   P = {‘x0’:1/2, ‘x1’:1/4, ‘x2’:1/8, ‘x3’:1/8}
H = -(1/2*log(1/2,2)+ 1/4*log(1/4,2)+ 2*1/8*log(1/8,2))
```

Вариант правильного ответа 3

```
                from math import log
   P = {‘x0’:1/2, ‘x1’:1/4, ‘x2’:1/8, ‘x3’:1/8}
                  def entro(a):
                        s = 0
                     for x in a:
                  s -= a[x]*log(a[x],2)
                       return s
                   H = entro(P)
```

Вариант правильного ответа 4

```
                from math import log
   P = {‘x0’:1/2, ‘x1’:1/4, ‘x2’:1/8, ‘x3’:1/8}
                  def entro(a):
      return sum(-a[x]*log(a[x],2) for x in a)
                   H = entro(P)
```

Вариант правильного ответа 5

```
                from math import log
   P = {‘x0’:1/2, ‘x1’:1/4, ‘x2’:1/8, ‘x3’:1/8}
      H = sum(-P[x]*log(P[x],2) for x in P)
```

Все приведенные ответы являются допустимыми и правильными; в то же время контролировать структуру каждого возможного ответа заранее предопределенными средствами не представляется ни целесообразным, ни возможным.

НЕКОТОРЫЕ ЗАМЕЧАНИЯ ПО ПОВОДУ РАБОТЫ СИСТЕМ АВТОМАТИЗИРОВАННОЙ ПРОВЕРКИ УЧЕБНЫХ ПРОГРАММ

Традиционно системы удаленного тестирования учебных программ имеют серверную компоненту, с которой пользователь взаимодействует посредством тонкого клиента. Серверная компонента фактически реализует все функции системы. Работа на клиентской машине предполагает лишь пересылку решений на сервер и получение сообщений о ходе тестирования. Поэтому представляется рациональным перенести часть функций тестирующей системы на сторону клиента, что расширило бы возможности системы. Это не должно вызывать значительных неудобств, связанных с инсталляцией дополнительного программного обеспечения, кроме среды программирования, в которой и разрабатываются тестируемые решения.

Следует отметить, что использование профессиональных сред программирования приветствуется в области разработки профессиональных приложений, но является излишним и обременительным при решении учебных примеров, приводит к значительной избыточности кода, не относящегося непосредственно к реализации необходимых алгоритмов и вычислений.

Поэтому актуален вопрос владения «вторым» языком программирования, предназначенным для выполнения относительно несложных расчетов и повседневной «автоматизации» различного вида работ. В качестве такого языка в настоящее время многими используется язык Python, весьма популярный в академической среде западных университетов.

Отметим, что при попытке реализовать систему удаленного компьютерного тестирования учебных программ, ориентированную на использование специализированной среды разработки или специализированных нотаций, возникает ряд проблем, в частности, необходимость:

- проверки правильности не только отдельных независимых программ, но и комплексных программных решений, последовательно разрабатываемых на основе собственных библиотек;
- предварительной обработки клиентской компонентой введенного ответа тестирующей системы с целью контроля правильности синтаксиса решений;
- сравнения ответа пользователя с эталонным ответом без требования точного их соответствия. Введенный испытуемым ответ считается правильным не только при точном совпадении с эталоном, но и в случае, если он может быть сведен к эталонному ответу путем эквивалентных преобразований в рамках системы аксиом, соответствующих предметной области.

ОСОБЕННОСТИ РЕАЛИЗАЦИИ СИСТЕМЫ ТЕСТИРОВАНИЯ ПРОГРАММ ПРИ РАБОТЕ С ИНТЕРПРЕТИРУЕМЫМ ЯЗЫКОМ

При использовании интерпретируемых языков, в частности Python, существенным недостатком является то, что фрагменты исходного кода приложения доступны для изучения, в отличие от компилируемых языков. Если в модуле, в котором осуществляется проверка решения, размещены эталонные функции или тесты, любой знающий студент сможет воспользоваться этими эталонами для решения своего задания.

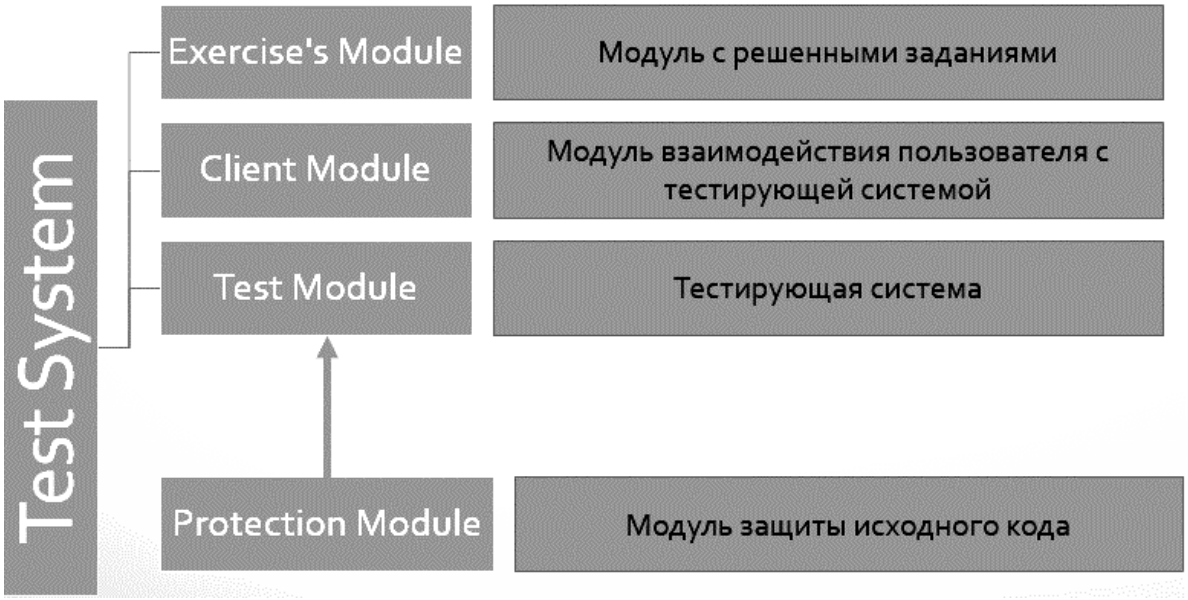
Поэтому необходимо выполнять сокрытие исходного кода от пользователей такой системы, что может быть реализовано комбинацией таких методов, как обфускация, замена опкодов, компилирование исходных кодов, шифрование.

Основные требования к реализации программного комплекса, использующего интерпретируемый язык программирования:

- наличие клиентской и серверной части;
- исходный код «важных» частей клиентского приложения не должен быть доступен в открытом виде конечному пользователю;

- открытыми для изучения входными данными приложения должны быть только решения заданий обучающимся;
- клиентское приложение должно выдавать удобочитаемые выходные данные для того, чтобы пользователь мог понять в чем ошибка в его работе.

Основываясь на положенных требованиях к системе, можно выделить следующие архитектурные компоненты такой тестирующей системы. На рисунке ниже представлены три основных модуля и один вспомогательный, предназначенный для защиты исходного кода клиентской компоненты. Первый модуль (Exercise's Module) содержит решения задач, предложенные пользователем. Второй модуль (Client Module) выполняет проверку решений и запускает третий модуль (Test Module), в котором хранятся эталонные решения заданий и специально подобранные тесты, которые будут применяться к решениям, приведенным студентом.



ЗАКЛЮЧЕНИЕ

К настоящему времени спроектирован и частично реализован прототип представленной в работе системы. Прототип функционирует в рамках корпоративной образовательной среды кафедры системного программирования и компьютерной безопасности Гродненского госуниверситета и используется автором при чтении дисциплины «Теория информации» студентам специальности «Программное обеспечение информационных технологий». Весьма активно изучаются вопросы безопасности его использования и устойчивости к атакам.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1. Moodle. Система управления обучением [Электронный ресурс]. URL: https://moodle.org/ (дата обращения: 15.08.2014).
2. Contester. Система для проведения турниров и индивидуального решения задач по олимпиадному программированию [Электронный ресурс]. URL: http://www.contester.ru/ (дата обращения: 15.08.2014).
3. *Кадан А. М., Курило С. В.* Использование системы автоматизированного тестирования учебных программ для обучения программированию // Современные информационные компьютерные технологии: сб. науч. ст. в 2 ч. / ГрГУ им. Янки Купалы ; редкол.: Е. А. Ровба, А. М. Кадан (отв.ред.) [и др.]. Гродно, ГрГУ, 2008. Ч. 1. С. 232–235.