Russian Technological Journal.
2022;10(3):7–23

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

УДК 004.02+004.4+004.891+004.91
https://doi.org/10.32362/2500-316X-2022-10-3-7-23

НАУЧНАЯ СТАТЬЯ

«Цифровой ассистент преподавателя» в массовом профессиональном обучении для цифровой экономики

Е.Г. Андрианова [@], **Л.А. Демидова**, **П.Н. Советов**

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] *Автор для переписки, e-mail: andrianova@mirea.ru*

Резюме

Цели. Активная цифровизация российской экономики вызывает дефицит ИТ-кадров и, в первую очередь, дефицит разработчиков программного обеспечения. Для российского университетского образования актуальной является задача массовой профессиональной подготовки таких специалистов. Цель работы – повышение качества массовой профессиональной подготовки программистов путем создания, внедрения и развития функциональности компьютерной системы «Цифровой ассистент преподавателя» (ЦАП). Эта система позволяет преподавателю в условиях массового обучения сконцентрироваться на функциях, требующих творческого подхода – составлении и обсуждении нетривиальных задач по программированию.

Методы. Использованы педагогические методы персонификации учебного процесса. Общий подход основан на удовлетворении ограничениям для создания генераторов задач по программированию. При генерации задач применены методы порождения случайных программ и данных на основе вероятностных контекстно-зависимых грамматик, а также методы трансляции с использованием дерева абстрактного синтаксиса. Для декларативного представления генератора задач применены методы функционального программирования, позволяющие создать предметно-ориентированный язык с помощью комбинаторов. Для проверки решений использованы методы автоматического тестирования.

Результаты. Разработана структура системы ЦАП. Рассмотрена автоматическая генерация задач по программированию, выделены классы практических задач, отражающие современную специфику разработки программного обеспечения, приведены примеры их генерации. Приведена схема генератора задач по программированию. Описана процедура автоматической проверки решения задач, осуществляемая с помощью набора программных тестов, сформированного генератором задач. Приведена процедура комплексной оценки решения обучающегося, включающая проверку корректности результата и проверку на плагиат решений в случае задач, созданных преподавателем вручную; соответствие стандарту стиля написания программы, метрикам оценки сложности программы и т.д. Рассмотрены ведение статистики успеваемости обучающихся и интерфейс взаимодействия обучающихся и преподавателей.

Выводы. Опыт внедрения ЦАП в учебный процесс курса «Программирование на языке Python» подтвердил возможность обеспечения персонификации учебного процесса для обучающихся в виде индивидуальных образовательных траекторий.

Ключевые слова: массовое обучение программированию, цифровой ассистент преподавателя, генерация заданий, индивидуальная траектория обучения, мотивация обучающихся

• **Поступила:** 29.12.2021 • **Доработана:** 03.03.2022 • **Принята к опубликованию:** 11.05.2022

Для цитирования: Андрианова Е.Г., Демидова Л.А., Советов П.Н. «Цифровой ассистент преподавателя» в массовом профессиональном обучении для цифровой экономики. *Russ. Technol. J*. 2022;10(3):7–23. https://doi.org/10.32362/2500-316X-2022-10-3-7-23

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

RESEARCH ARTICLE

Pedagogical design of a digital teaching assistant in massive professional training for the digital economy

Elena G. Andrianova [@], **Liliya A. Demidova, Petr N. Sovetov**

MIREA – Russian Technological University, Moscow, 119454 Russia
[@] *Corresponding author, e-mail: andrianova@mirea.ru*

Abstract

Objectives. The active digitalization of the Russian economy has resulted in a shortage of IT personnel; this is particularly true of software developers. Thus, the Russian university education is faced with the task of undertaking the large-scale professional training of such specialists. The purpose of the present work was to support the large-scale ("massive") professional training of programmers through the creation and implementation of Digital Teaching Assistant (DTA) computer system, allowing teachers working under stressful conditions to concentrate on functions that require a creative approach, namely, drawing up and discussing nontrivial programming tasks.

Methods. Pedagogical methods for the personification of learning processes were employed. The general approach was based on satisfying the constraints for creating programming task generators. Tasks were generated using methods for generating random programs and data based on probabilistic context-sensitive grammars, along with translation methods using an abstract syntax tree. The declarative representation of the task generator was performed using functional programming methods, allowing the creation of a domain-specific language using combinators. The solutions were checked using automated testing methods.

Results. The developed structure of the proposed DTA system was presented. Considering the automatic generation of programming tasks, classes of practical tasks that reflected the modern specifics of software development were identified along with examples of their generation. A diagram of the programming task generator was provided along with a description of the procedure for automatically checking the solutions of the tasks using a set of program tests generated by the task generator. The presented procedure for comprehensive assessment of a student's solution included verification of the correctness of the result and plagiarism checks in the case of tasks created manually by the teacher; this also involved validation for compliance with coding style standards, along with metrics for assessing program complexity, etc. The means for recording of statistics of academic achievement of students was characterized along with the interface of interaction between students and teachers.

Conclusions. The experience of introducing a DTA into the learning process of teaching programming in Python confirmed the possibility of personifying the learning process in the form of individual learning paths.

Keywords: massive programming learning, Digital Teaching Assistant, task generation, individual learning path, student motivation

• **Submitted:** 29.12.2021 • **Revised:** 03.03.2022 • **Accepted:** 11.05.2022

For citation: Andrianova E.G., Demidova L.A., Sovetov P.N. Pedagogical design of a digital teaching assistant in massive professional training for the digital economy. *Russ. Technol. J*. 2022;10(3):7–23. https://doi.org/10.32362/2500-316X-2022-10-3-7-23

Financial disclosure: The authors have no a financial or property interest in any material or method mentioned.

The authors declare no conflicts of interest.

ВВЕДЕНИЕ

Цифровая трансформация экономики коренным образом преобразует все сферы человеческой деятельности, порождая острую необходимость наличия большого количества профессионалов, обладающих ИТ-компетенциями или цифровыми компетенциями. В настоящее время актуальной является задача массовой подготовки профессиональных кадров для цифровой экономики, в первую очередь – разработчиков программного обеспечения (ПО). В среднесрочной перспективе на российском рынке труда ожидается рост спроса на разработчиков по таким перспективным направлениям, как искусственный интеллект (ИИ), анализ больших данных, робототехника, виртуальная реальность, Интернет вещей[1] [1, 2].

Технологии массового обучения разработчиков ПО требуют активного внедрения ИТ-технологий и интеллектуальных технологий в организацию учебного процесса и предполагают использование электронных образовательных ресурсов, поддержку удобной и быстрой коммуникации между участниками учебного процесса, применение передовых педагогических инноваций и практик.

Однако цифровизация учебного процесса для массового обучения разработчиков ПО не должна приводить к унификации подходов ко всем обучающимся. Наоборот, необходимо предоставлять обучающемуся широкие возможности в создании и поддержке персонификации обучения, в первую очередь, путем организации индивидуальных образовательных траекторий (ИОТ), предполагающих активную самостоятельную работу обучающегося. «Электронная культура создает условия доступности цифровых технологий для масс, повышая уровень их проникновения в социальную систему и все общественные отношения. Доступность этих технологий позволяет формировать новые картины мира и способы его познания. Человек в таких условиях формируется как постоянно развивающаяся, открытая антропосистема. Человек в цифровом пространстве становится стремительно расширяющей свои возможности интерактивной системой, которая формирует и соответствующие параметры будущего» [3]. При этом часть педагогических научных школ отмечает, что наблюдается снижение мотивации обучающихся к самостоятельной работе. Есть мнение, что причиной этого является тот факт, что «в настоящее время учебный труд имеет черты принудительного труда» [4]. Существующие стандарты и программы освоения учебного материала дают обучающемуся не личностно значимые образовательные результаты, а мотивируют выполнять заданные системой образования требования. Обучающиеся не обладают самостоятельностью в своих действиях: им навязывают цели, содержание и методы обучения, задают внешнюю систему контроля. «Мотивы к учению замещаются мотивами ответственности, социальной необходимости и принуждения: «ты должен», «обязан» [5].

Для повышения мотивации обучающихся существуют различные концепции типов обучения. Например, концепция эвристического обучения, базирующаяся на модели экстериоризации, ставит целью самореализацию обучающегося [6, 7]. Концепция развивающего обучения, использующая модель интериоризации, нацелена на развитие обучающегося [6, 7]. Задачей преподавателя при эвристическом или развивающем обучении является не столько разработка контента курса (содержания лекций и практических занятий), а поиск и мониторинг доступных образовательных ресурсов, где могут быть представлены материалы образовательного контента курса. Таким образом создается ИОТ обучающегося. Необходимо отметить, что поддержка мотивации обучающегося к получению компетенций в области разработки ПО становится важной задачей и не сводится только к материальной стороне вопроса.

В области подготовки разработчиков ПО технические университеты, использующие классическую форму обучения программированию, давно конкурируют с качественными массовыми открытыми онлайн-курсами от ведущих фирм и образовательных центров и со специализированными платформами онлайн-образования, такими, как Stepik [8] и JetBrains Academy [9], предоставляющими инструменты для

[1] Рынок труда в России. *Tadviser. Государство. Бизнес. Технологии*. https://clck.ru/ZD5AU, дата обращения 28.11.2021. [Rynok truda v Rossii (The labor market in Russia). *Tadviser. Gosudarstvo. Biznes. Tekhnologii* (in Russ.). https://clck.ru/ZD5AU. Accessed November 28, 2021.]

эффективного обучения языкам программирования, востребованным в текущий момент работодателями в области разработки ПО.

Реализация противоэпидемических мероприятий и использование онлайн-технологий в учебном процессе привели к тому, что «смешанная» форма организации обучения стала частью учебного процесса в университетах. Это изменило обязанности преподавателя добавлением функционала тьюторства – увеличился объем творческих функций преподавателя, связанных с наставничеством и необходимостью учета индивидуальных особенностей обучающегося при освоении практико-ориентированного курса по программированию. Очевидно, что нагрузка на преподавателя существенно увеличилась, и возникла необходимость автоматизации рутинной части его обязанностей путем создания и использования интеллектуальных цифровых помощников – или цифрового ассистента преподавателя (ЦАП). ЦАП может взять на себя функции выдачи и составления практических заданий по программированию, генерирования заданий и проверки результатов выполнения заданий, отслеживания посещаемости и работы обучающихся, сбора цифровых следов и т.п.

Разработка интеллектуальной обучающей системы – цифрового ассистента преподавателя, ее реализация и внедрение в преподавание учебного курса обучения программированию в техническом университете является актуальной задачей.

1. ПЕДАГОГИЧЕСКИЙ ДИЗАЙН И ОСНОВНЫЕ ЭЛЕМЕНТЫ КОМПЬЮТЕРИЗАЦИИ ПРЕПОДАВАНИЯ КУРСА ПО ОБУЧЕНИЮ ПРОГРАММИРОВАНИЮ

Современная ситуация в системе образования по обучению программированию предоставляет большие возможности преподавателю в постановке целей обучения и проектировании педагогического дизайна учебного курса, определяющего способы достижения поставленных целей. Педагогический дизайн – это систематизированный подход к созданию образовательных решений, использующий педагогические принципы и теории для обеспечения высокого качества обучения [10].

В контексте цифровизации обучения педагогический дизайн является эффективным инструментом разработки цифрового образовательного контента курса, призванного обеспечить разумное сочетание онлайн и офлайн обучения («смешанная» форма обучения), создания интеллектуальной среды курса обучения программированию, поддержки реализации ИОТ.

К базовым принципам педагогического дизайна относятся [10] следующие: наглядное объяснение целей и задач обучения, привлечение и удержание внимания обучающегося, пробуждение интереса к изучаемой теме или методам обучения, быстрая апробация полученных знаний на практике, получение обратной связи от обучающегося, оценка и самооценка успеваемости и общая оценка эффективности учебного курса, помощь обучающемуся в сохранении полученных знаний и их правильном использовании.

Применение принципов педагогического дизайна при разработке контента учебного курса позволяет определить оптимальные условия для достижения целей обучения: модель интерактивного взаимодействия участников образовательного процесса, формы представления учебного контента курса, методы повышения мотивации обучающихся к самостоятельной работе над материалами курса (когнитивная функция), Например, применение методов краудсорсинга позволяет выявить образовательные запросы и образовательные дефициты, актуальные для обучающихся по данному курсу обучения программированию.

В рамках работы над педагогическим дизайном курса обучения программированию определяется не только формат отображения учебного контента, но разрабатываются методические указания для обучающихся и преподавателей, задания и упражнения по разработанному контенту, тестовые задания и формы самооценки обучающихся. Тестовые задания, их выполнение и последующая самооценка обучающимся своего образовательного уровня по курсу частично могут быть основаны на геймификации. Самооценка по определению обучающимся своего уровня компетентности может соответствовать событиям в электронной системе, например, таким, как прием выполненного задания по шкале «зачтено/незачтено», оценивание в балльно-рейтинговой системе и т.п. Для качественного контента учебного курса обязательно включение мотивационных стимулов, например, в виде творческих заданий, определения проблемного поля курса и т.п.

Обязательными элементами педагогического дизайна при проектировании практико-ориентированного курса обучению программированию являются следующие элементы:

- интерактивные материалы в онлайн-доступе (обучающийся может запускать примеры непосредственно в учебнике, использовать элементы встроенного графического интерфейса для обновления содержимого графиков и т.д.);
- изложение материалов в персонализированном виде – в виде графа зависимостей (графа знаний);
- онлайн-система приема и проверки задач;
- автоматическая адаптация сложности задач и теоретических материалов к уровню обучающегося;

- автоматическая генерация задач;
- автоматическая генерация подсказок и комментариев по решениям задач.

2. СОВРЕМЕННЫЕ ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ

Перспективным направлением проектирования педагогического дизайна университетского курса массового обучения программированию является использование инструментальных средств и режима «смешанного» обучения («blended learning»), включающего элементы очного и онлайн-обучения. Асинхронность онлайн-части образовательного процесса дает обучающемуся свободу выбора материала и времени на его изучение. Представляет интерес, в частности, модель перевернутого класса («flipped classroom»), предполагающая самостоятельное усвоение основных теоретических знаний по курсу в онлайн-режиме. Очные лекции при этом становятся более интерактивными, включают в себя обсуждения по самостоятельно изученным темам, широко используют дополнительные материалы и совместное решение задач.

Для удобства усвоения обучающимися материал традиционной полуторачасовой лекции в онлайн-режиме все чаще оформляется в виде набора коротких (5–15 мин.) видеофрагментов [11]. Растет популярность использования технологий виртуальной реальности в учебном процессе [11]. Для своевременной оценки степени изученности материала прямо в процессе онлайн-лекции может быть реализована обратная связь с обучающимися в виде кратких тестовых вопросов и тестовых задач. Особую роль в обеспечении глубокого усвоения лекционного материала могут играть такие интерактивные элементы лекции, как вставки исполняемого кода и различного рода наглядные динамические модели, управляемые с помощью графического интерфейса («explorable explanation») [12]. В частности, веб-приложение Python Tutor [13] позволяет в динамическом режиме наглядно изобразить детали выполнения программы на языке Python.

При организации массового курса по обучению программированию важно обеспечить персонализацию обучения в виде ИОТ, предполагающую адаптацию образовательного процесса под нужды обучающегося. Для режима лекции может быть использовано представление материала курса в виде графа знаний («knowledge graph») с возможностью свободы выбора между изучаемыми узлами-темами с учетом ребер-зависимостей [14].

В режиме выполнения практических занятий и в условиях количественных ограничений на преподавательский состав интерес могут представлять интеллектуальные системы обучения (Intelligent Tutoring System, ITS), призванные обеспечить обратную связь с обучающимися или передачу индивидуальной инструкции или ответа на действия обучающихся в режиме реального времени без вмешательства преподавателя. Это компьютерные системы, включающие в себя модель области знаний, а также модели обучающегося и преподавателя. Такие интегрированные системы зачастую используют методы машинного обучения и отличаются высокой сложностью построения [15].

Сложность построения интеллектуальной системы обучения в виде единой системы, интегрирующей все ситуации и сценарии взаимодействия, привела к распространению компьютерных инструментов, решающих отдельные задачи, относящиеся к области интеллектуальных систем обучения. Это, в частности, системы автоматической проверки полученных от обучающихся решений [16], системы автоматического формирования подсказок в случае неправильного решения [17], а также системы автоматической генерации задач с заданной сложностью [18] и системы проверки решений на плагиат [19].

Среди недостатков подобных систем следует выделить их нацеленность исключительно на индивидуальную работу обучающегося, а также на ограниченный, формально определенный класс задач. В этой связи могут быть полезны специализированные форумы, такие, как Piazza[2], включающие средства поощрения обучающихся, помогающих своим коллегам. Важную роль также играет использование учебных проектов («project-based learning»), коллективно разрабатываемых обучающимися, и применение технологий геймификации учебного процесса [20].

3. ЦИФРОВОЙ АССИСТЕНТ ПРЕПОДАВАТЕЛЯ

3.1. Структура цифрового ассистента преподавателя

При проведении массовых университетских курсов по программированию большая часть учебной нагрузки приходится на ассистентов преподавателей, составляющих задачи по программированию и проверяющих правильность их решения обучающимися. В этой связи предлагается новый подход к организации учебного процесса подготовки разработчиков ПО, базирующийся на использовании компьютерной системы – цифрового ассистента преподавателя. Использование ЦАП позволяет

[2] https://piazza.com/, дата обращения 13.01.2022. [https://piazza.com/. Accessed January 13, 2022.]

преподавателям в условиях массового обучения сконцентрироваться на функциях, требующих творческого подхода и заключающихся, в частности, в составлении и обсуждении нетривиальных задач по программированию, а также – в развитии функциональности ЦАП.

К основным функциям ЦАП (рис. 1) относятся:

- генерация индивидуальных задач;
- проверка задач с информативным ответом для обучающегося;
- ведение статистики успеваемости обучающихся;
- интерфейс взаимодействия участников учебного процесса.

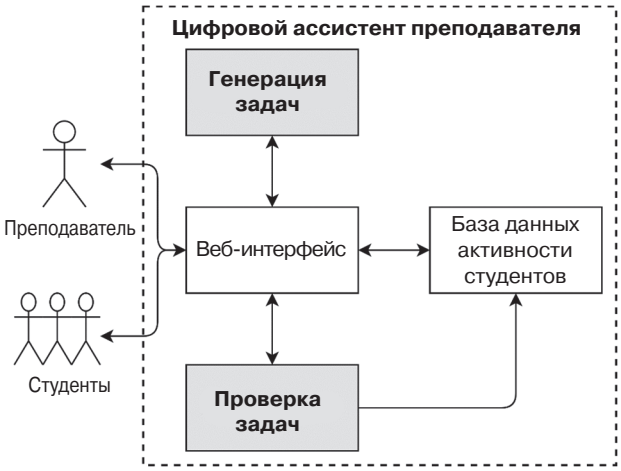

Рис 1. Структура цифрового ассистента преподавателя

В отличие от ИОС [21], ЦАП не предназначен для полной замены преподавателя и не является «черным ящиком», автоматически подбирающим задачи в процессе обучения и оценивающим решения в соответствии со скрытыми от обучающегося параметрами. Модель учителя, используемая в ИОС, заменена в ЦАП явными настройками со стороны учителя-человека.

3.2. Автоматическая генерация задач по программированию

Автоматическая генерация задач по программированию позволяет обеспечить каждого из множества обучающихся индивидуальным набором задач. Самостоятельно сделать это преподавателю затруднительно, а при большом количестве учащихся и невозможно. Также автоматическая генерация задач частично позволяет решить проблему плагиата решений.

Создание генераторов задач по программированию является трудоемким процессом.

К генератору задач предъявляются следующие требования:

- поддержка основных тем курса и конструкций языка;
- интересность, практическая полезность задач;
- ясность, однозначность текста условия и его схожесть с текстом, составленным людьми;
- одинаковая сложность всех генерируемых вариантов задачи при фиксированных настройках генератора задач;
- отсутствие явных шаблонных решений для различных вариантов задачи;
- возможность автоматической проверки решения.

В составе ЦАП имеется множество генераторов задач $G = \{g_1, \ldots, g_n\}$, каждый из которых генерирует варианты задач своего типа. Функция-генератор задач $g_i(h, s)$ принимает на вход хеш-значение индивидуальных данных обучающегося (h), параметр сложности или размера задачи (s).

Далеко не для всех типов задач по программированию известны подходы к их автоматическому генерированию. Далее рассматриваются те типы задач, для которых соответствующие генераторы были созданы.

В рамках курса обучения программирования на некотором языке можно выделить следующие классы практических задач, отражающие современную специфику разработки ПО:

- составление новой программы;
- анализ существующей программы;
- составление модульных тестов для существующей программы.

Среди задач, направленных на составление новой программы, известны [20] генераторы задач для таких видов задач, как:

- трансляция некоторого представления в код программы;
- преобразование форматов данных.

В задачах первого вида входным представлением может быть математическая формула. Таким образом обучающийся учится практически важному навыку перевода математического языка на язык программирования. Другим примером задачи первого вида является перевод программы с одного языка программирования на другой. В качестве входного представления может использоваться и графическая нотация. Примерами задач, использующих графическую нотацию, является перевод графа конечного автомата или UML-диаграммы в программный код.

В задачах преобразования входного формата данных в выходной формат могут использоваться как текстовые, так и двоичные форматы данных.

В общем виде генераторы задач рассмотренных видов могут быть реализованы с использованием подходов из области комбинаторных задач ИИ. В частности, перспективным является

использование подхода на основе удовлетворения ограничениям [22]. При использовании этого подхода множество переменных и их областей значений описывают пространство генерируемых программ или форматов данных, а множество ограничений позволяет отфильтровать только те результаты, которые соответствуют заданным характеристикам для выбранного типа задач.

Случайная конфигурация p (программа или описание формата данных) из пространства конфигураций P определяется следующим образом:

$$p \in \mathrm{P} \wedge \phi_c(p) \wedge \phi_t(p).$$

Здесь предикат-спецификация ϕ_c определяет условия отбора p, а предикат ϕ_t отражает возможность создания для p набора невырожденных (т.е. с разнообразными значениями в заданных пределах) программных тестов.

Порождение случайных конфигураций может быть осуществлено с помощью вероятностных грамматик, в которых каждое правило дополнено вероятностью его применения в процессе порождения результата. Этот же подход используется в области фаззинг-тестирования [23].

Особенный интерес представляют вероятностные контекстно-зависимые грамматики, позволяющие при порождении случайного результата добиться не только синтаксически корректного результата, но и осуществить ряд семантических проверок. Примером такой проверки является проверка на корректное использование ранее определенной переменной в своей области видимости.

При генерации задачи недостаточно сформировать ее условие. Необходимо также получить множество программных тестов, на которых будет проверяться корректность решений обучающихся. Этап генерации тестов является частью общего процесса генерации задачи. При неудовлетворительном качестве полученных тестов условие задачи необходимо изменить.

Помимо простых модульных тестов, содержащих множество пар $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, где x_i – вектор входных значений, а y_i – результат вычисления, могут использоваться тесты с отслеживанием состояния для реагирующих систем. Входные данные в этом случае – это трасса, состоящая из вызовов (методов, сообщений) и соответствующих каждому вызову реакций системы.

Среди типичных задач по программированию, связанных с анализом корректности существующих программ, интерес могут представлять генераторы, позволяющие по набору шаблонов программ создать их вариации, а также внести в программный код ошибку, исправление которой поручается обучающемуся.

Класс задач, связанных с созданием программных тестов, характеризуется использованием мутационного тестирования [24] для определения качества программных тестов, составленных обучающимся для автоматически сгенерированной программы.

3.3. Реализация генераторов задач

На рис. 2 показана общая схема генератора задач по программированию. На основе хеш-кода информации об обучающемся и параметра сложности задачи формируется условие задачи, а также программные тесты для автоматической проверки решений.

Генератор случайных конфигураций, возможная реализация которого изложена далее, формирует случайную программу или случайный формат данных. Конфигурация далее подвергается последовательности упрощений ее структуры. В случае порождаемой программы это могут быть традиционные

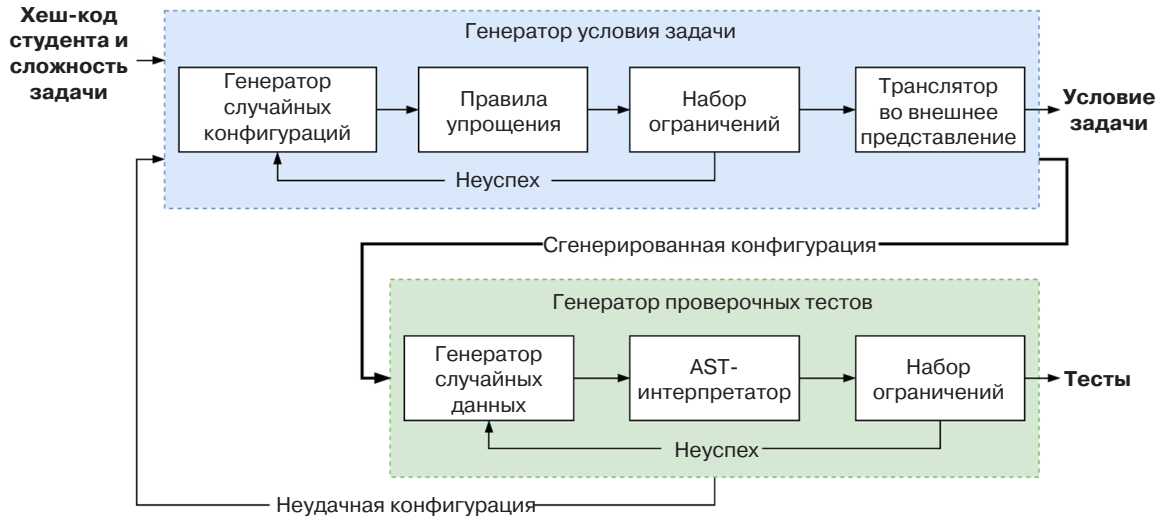

Рис. 2. Схема генератора задач по программированию

оптимизации со стороны компилятора, такие, например, как свертка констант и упрощение выражений. Характеристики конфигурации оцениваются с помощью набора ограничений. Эти ограничения, в частности, позволяют сделать однородной сложность генерируемых задач. Успешно сгенерированная конфигурация транслируется во внешнее представление, представляющее собой условие задачи для обучающегося.

Генератор проверочных тестов создает случайные данные в соответствии с типом формируемой задачи. Полученная ранее конфигурация оценивается с помощью набора тестов. В случае конфигурации-программы осуществляется интерпретация ее внутреннего представления в виде дерева абстрактного синтаксиса (Abstract Syntax Tree, AST) с использованием случайных данных в качестве входов программы. В случае конфигурации-формата данных соответствующие структуры заполняются случайными значениями. Если набор тестов с заданными характеристиками сформировать не удалось (например, после заданного числа попыток), то осуществляется возврат к порождению новой конфигурации.

Для реализации генератора случайных конфигураций предлагается использовать специальный язык, описывающий грамматику – правила порождения конфигураций. Этот язык является расширением вероятностных контекстно-зависимых грамматик и представляет собой набор комбинаторов в духе функционального программирования.

Комбинаторы – функции высшего порядка, использующиеся для создания встроенных предметно-ориентированных языков (Embedded Domain-Specific Language, eDSL). Известно, в частности, множество реализаций комбинаторов синтаксического разбора [25]. В рассматриваемом случае комбинатор после вычисления аргументов возвращает лексическое замыкание *f*:

$$f: c \rightarrow c'.$$

При этом используется модель вычислений, в которой комбинаторы принимают контекст c и возвращают его новую версию. Контекст $c = \{(a_1, v_1), \ldots, (a_n, v_n)\}$ является множеством, состоящим из n пар вида «атрибут-значение». В контексте всегда находятся следующие атрибуты: результат вычисления a_{val} и булево значение ошибки a_{err}. При возникновении ошибки в процессе вычислений на комбинаторах происходит локальный откат – возвращение к одному из вызывающих комбинаторов, который эту ошибку обрабатывает.

Основой рассматриваемого языка комбинаторов являются комбинаторы, реализующие операции, известные из теории формальных языков: конкатенацию (*seq*) и объединение (*alt*). Базовый комбинатор *seq* вычисляет последовательность своих аргументов-комбинаторов, обновляя контекст. При возникновении ошибки происходит возврат из комбинатора с восстановлением состояния контекста на начало выполнения *seq*. Базовый комбинатор *alt* вычисляет случайно выбранный комбинатор среди своих аргументов. Для каждого из аргументов может быть задана вероятность его выбора. Комбинатор *check* проверяет выражение-аргумент на истинность, используя текущий контекст в качестве аргумента. Если выражение оказалось ложным, то единичное значение a_{err} в контексте сигнализирует об ошибке. Комбинатор *find* повторяет выполнение комбинатора-аргумента до тех пор, пока результат вычислений не оказался успешным.

Рассмотрим вариант комбинаторного языка, реализованного на Python. С использованием комбинаторов может быть разработан генератор случайных многочленов, показанный на рис. 3. Реализация этого генератора состоит из трех правил. Правило *coeff* на рис. 3а определяет значение коэффициента. Выбор осуществляется между случайным значением в заданном диапазоне и нулем

```
1 coeff = alt(randint(-100, 100), 0, prob=[0.3, 0.7])
2
3 term = λ k, n: mul(k, power('x', n))
4
5 main = find(seq(
6     let(a=coeff), let(b=coeff),
7     let(c=coeff), let(d=coeff),
8     add(
9         term(ref('a'), 3),
10        term(ref('b'), 2),
11        term(ref('c'), 1),
12        term(ref('d'), 0),
13    ),
14    check(λ c: c.a != 0 or c.b != 0 or c.c != 0)
15 ))
```

(а)

$$86x^3 + 20x^2 + -27$$
$$-85x$$
$$-98x$$
$$86x^3 + -47x^2 + 51x$$
$$40x + 44$$
$$93x$$
$$-63x + -66$$
$$-62x^3 + -17x$$
$$-97x^2$$
$$94x^2$$

(б)

Рис. 3. Генератор случайных многочленов:
(а) реализация на eDSL; (б) примеры результатов генерации в формате LaTeX

```
args = shuffle_list(
    decl_ptr(ref('ty'), ref('buf1')), decl_ptr(ref('ty'), ref('buf2')),
    decl('int', ref('size'))
)
for_array = λ *body: for_loop(
    eq(decl('int', ref('idx')), ref('mtap')), lt(ref('idx'), ref('size')),
    postop('++', ref('idx')),  block(*body)
)
memset = call('memset',
    lst(ref('buf2'), 0, mul(call('sizeof', ref('ty')), ref('size')))
)
loop_body = stmt(eq(load(ref('buf2'), ref('idx')), expr))
main = seq(
    let(ty=one_of('int float double')),
    let(name=one_of('data buf arr x y')),
    let(buf1=to(λ c: c.name + '1')), let(buf2=to(λ c: c.name + '2'))
    let(size=to(λ c: c.buf1 + '_size')),
    let(idx=one_of('i j k n')),
    let(taps=to(λ c: sorted(sample(range(1, 5), randint(1, 3))))),
    let(mtap=to(λ c: max(c.taps))),
    function(
        one_of('calculate process compute perform'),
        'void', args, block(stmt(memset),
            alt(for_array(loop_body), while_array(loop_body))
        )
    )
)
```

Рис. 4. Реализация генератора случайных функций обработки массива

с вероятностью, заданной в *prob*. Правило *term* формирует AST-представление для выражения вида kx^n. Правило *main* формирует AST-представление многочлена до третьей степени. Комбинатор *let* позволяет именовать результаты вычислений, а *ref* позволяет далее обратиться к этим результатам. С помощью комбинатора *check* установлено ограничение – порождаемый многочлен должен иметь хотя бы один ненулевой коэффициент при *x*.

Поиск корректного результата осуществляет комбинатор *find*. Таким образом реализована простая декларативная система программирования в ограничениях. Производительность этого подхода значительно уступает, например, SMT-решателям и специализированным решателям для задач программирования в ограничениях. Тем не менее, рассматриваемый декларативный язык позволяет использовать ограничения произвольной сложности. Кроме того, скорость генерации обычно не является наиболее важной характеристикой для создания учебных задач. Наконец, SMT-решатели и другие сторонние инструменты можно использовать совместно с рассматриваемым комбинаторным языком.

На рис. 4 приведен более развернутый пример, реализующий генерацию случайных функций на языке C++ для обработки массивов. Здесь имитируется задача в духе реализации фильтров с конечной импульсной характеристикой.

В порождаемых случайных функциях согласованы типы данных и имена переменных. Цикл может быть реализован как с помощью *for*, так и с помощью *while* (реализация которого аналогична *for*). Реализация комбинатора *expr* не показана, поскольку не представляет ничего нового по сравнению с предыдущим примером порождения многочленов.

На рис. 5 приведены некоторые примеры порожденных случайных функций.

```
void process(int x_size, double *y, double *x) {
  memset(y, 0, sizeof(double) * x_size);
  for (int i = 4; i < x_size; i++) {
    y[i] = 8 * x[i] + 2 * x[i - 1] + -4 * x[i - 3] + -5 * x[i - 4];
  }
}
void perform(float *arr1, int arr1_size, float *arr2) {
  memset(arr2, 0, sizeof(float) * arr1_size);
  int k = 2;
  while (k < arr1_size) {
    arr2[k] = 5 * arr1[k] + 8 * arr1[k - 2];
    k++;
  }
}
void compute(int y1_size, float *y2, float *y1) {
  memset(y2, 0, sizeof(float) * y1_size);
  for (int k = 3; k < y1_size; k++) {
    y2[k] = -6 * y1[k] + -7 * y1[k - 3];
  }
}
```

Рис. 5. Примеры сгенерированных случайных функций обработки массива на C++

3.4. Автоматическая проверка задач

Проверка корректности полученного решения осуществляется с помощью набора программных тестов, сформированного генератором задач. Для тестирования реализуется «песочница» – среда безопасного выполнения программного кода.

Целесообразно дать комплексную оценку решению обучающегося, не ограничиваясь только проверкой корректности результата. Элементами такой оценки, получаемой автоматически, могут быть:

- проверка на плагиат решений в случае задач, созданных преподавателями вручную;
- проверка на соответствие стандарту стиля написания программы;
- анализ метрик оценки сложности программы;
- оценки корректности результата, вычислительной сложности программы, объема используемой программой памяти.

Первые 3 элемента списка могут быть реализованы с помощью инструментов статического анализа, а последние – с помощью инструментов профилирования программы.

При некорректном решении ответ ЦАП должен включать в себя, как минимум, сообщение от компилятора и указание невыполненного теста.

Более информативный ответ включает в себя указание на типовую ошибку, совершенную обучающимся. Такой механизм может быть реализован с помощью поиска шаблонов в дереве абстрактного синтаксиса некорректной программы. Правила вида «шаблон-сообщение» добавляются в ЦАП преподавателем. Существуют также подходы, позволяющие генерировать на основе некорректного решения автоматические подсказки, направляющие обучающегося в сторону корректного результата [26].

3.5. Ведение статистики успеваемости обучающихся

Результаты полученных от обучающихся решений сохраняются в базе данных ЦАП. Это позволяет преподавателю отказаться от ручного заполнения журнала успеваемости. Кроме того, комплексная оценка решений со стороны ЦАП дает возможность отказаться от упрощенных формул расчета итоговой оценки, а также от готовых наборов задач. Преподаватель может, например, ограничиться указанием минимально необходимого количества решенных задач для каждой из тем курса. В то же время ЦАП предоставляет обучающемуся возможность неограниченной тренировки на задачах, сложность и тип которых можно варьировать.

3.6. Интерфейс взаимодействия обучающихся и преподавателей

Для ЦАП является предпочтительным веб-интерфейс, позволяющий осуществить доступ к системе с любого устройства, подключенного к интернету.

С точки зрения обучающегося интерфейс должен позволять осуществлять выбор типов и параметров задач, относящихся к текущей изучаемой теме. Данные комплексной оценки предоставляются вместе с историей предыдущих результатов, что позволяет обучающемуся оценить свой прогресс.

С точки зрения преподавателя важную роль играет анализ данных, связанных с активностью обучающихся. С использованием интерфейса ЦАП преподаватель может вовремя оценить ситуацию и принять нужное решение. Это касается, например, ситуации, когда задача оказалась слишком легкой или сложной для большинства обучающихся. Анализ данных может быть использован для определения отстающих или для получения прогноза итоговой оценки по обучающемуся.

4. ОПЫТ РАЗРАБОТКИ КУРСА ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON

Курс обучения программированию на языке Python в РТУ МИРЭА организован в режиме смешанного обучения. Онлайн-инструменты, используемые в этом курсе, представлены на рис. 6.

Преподавательский состав курса представлен 4 лекторами и 13 преподавателями практических занятий. В обучении приняло участие свыше 1500 студентов, и для их своевременного оповещения о важных событиях курса был создан телеграм-канал. В качестве двустороннего канала общения была использована система отслеживания ошибок MantisBT, уже знакомая обучающимся по курсу конфигурационного управления, используемая как замена традиционным средствам взаимодействия (электронная почта и форум).

Лекции содержат базовую часть дисциплины, включающую в себя вопросы использования основных конструкций языка и введение в стиль объектно-ориентированного программирования (ООП), принятый в языке Python. В заключительных лекциях рассмотрен ряд специальных вопросов, в частности:

- автоматизация тестирования: метрики тестового покрытия кода, мутационное тестирование, контрактное программирование, тестирование на основе свойств и моделей,
- функциональное программирование: функции высших порядков, замыкания, декораторы, генераторы, неизменяемые типы данных, модули functools и itertools, библиотека NumPy.

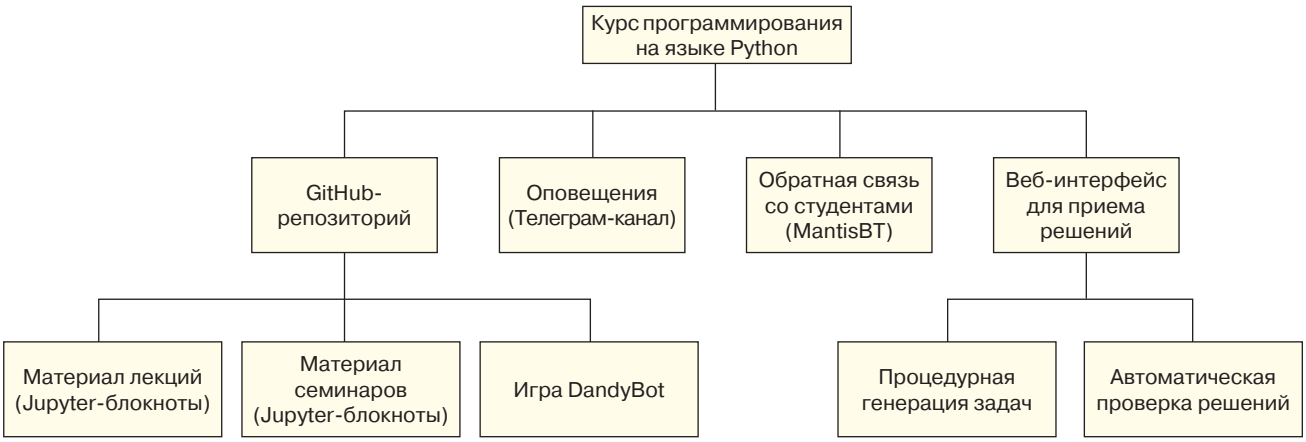

Рис. 6. Структура онлайн-части курса программирования на языке Python

Основные материалы курса размещены в отдельном GitHub-репозитории. При этом обучающимся обеспечиваются поощрения за найденные ошибки в материалах. Сообщения о найденных ошибках принимаются в режиме «pull request».

Использование Jupyter-блокнотов позволило сделать онлайн-лекции, проводимые в формате вебинара, более интерактивными. Непосредственно в процессе лекции обучающиеся экспериментируют с предложенными им короткими программами, а лектор, реагируя в том числе на вопросы обучающихся, имеет возможность модифицировать и запустить примеры программ в ячейках Jupyter-блокнота. Представление материала лекции в традиционном виде слайдов, но с добавленной интерактивностью ячеек, осуществляется с помощью расширения RISE для Jupyter-блокнотов.

Очное семинарское занятие включает в себя интерактивную теоретическую часть, которая дает возможность:

- оценить уровень усвоения обучающимися в режиме онлайн лекционного материала и, при необходимости, оперативно устранить пробелы в знаниях;
- ввести обучающихся в предметную область рассматриваемых на занятии задач.

На семинарских занятиях используется несколько типов задач:

- процедурно сгенерированные задачи;
- задачи из Jupyter-блокнотов;
- задачи, которые относятся к игре DandyBot;
- творческие проекты по темам преподавателей, индивидуальные и групповые.

Одной из основных проблем проектирования рассматриваемого курса были создание и организация оперативной проверки практических задач в условиях массового обучения с участием небольшого коллектива преподавателей. Было принято решение использовать подход на основе процедурной генерации: для каждого из обучающихся автоматически формировался свой, индивидуальный набор задач [18]. Такой подход позволил решить проблему плагиата среди обучающихся, а также снизил нагрузку на преподавателей, благодаря автоматической проверке решений.

Так, вариации следующих задач генерируются автоматически:

1) преобразование математической нотации в код: арифметические выражения;
2) преобразование математической нотации в код: ветвления;
3) преобразование математической нотации в код: циклы;
4) преобразование математической нотации в код: рекурсия;
5) преобразование дерева решений в код;
6) реализация перестановки битовых полей в двоичном слове;
7) преобразование табличного формата данных (рис. 7);
8) разбор двоичного файлового формата (рис. 8);
9) реализация конечного автомата средствами ООП.

Для генерируемой задачи может быть задана степень ее сложности, а также количество сформированных тестов, позволяющих обучающемуся проверить корректность своего решения.

Важную роль играет информативность сообщений, полученных обучающимся от системы, принимающей решения задач. Для некоторых типов задач реализована поддержка правил, позволяющих по найденному в коде решения шаблону выдать детальное сообщение об ошибке, сопровождаемое отсылкой к методам получения верного решения. Поиск по шаблону осуществляется на уровне представления программы студента в виде дерева абстрактного синтаксиса. В задаче на перестановку битовых полей такие правила позволяют выявить некорректный подход к решению (наличие функции перевода числа в строку) и подсказать правильное направление (использование побитовых операций).

Задача 3. Реализовать функцию преобразования табличных данных. Входная и выходная таблицы заданы в построчной форме. Заполненные ячейки имеют строковый тип данных. Пустые ячейки имеют значение None.

Над входной таблицей реализовать ряд преобразований:

– удалить пустые столбцы;

– разбить столбец № 1 по разделителю «!»;

– преобразовать содержимое ячеек по примерам;

– транспонировать таблицу.

Примеры табличных преобразований:

1. Исходная таблица:

		+78694947943!03-08-2003	Нет
		+79774177489!27-03-2002	Да
		+77571568485!05-12-2001	Да
		+73113372701!21-07-2003	Нет

Результат преобразования:

8694947943	9774177489	7571568485	3113372701
false	true	true	false
03-08-03	02-03-27	01-12-05	03-07-21

2. Исходная таблица:

		+76058136232!11-04-2001	Нет
		+72178434337!02-10-2000	Нет
		+76826857881!25-11-2003	Да

Результат преобразования:

6058136232	2178434337	6826857881
false	false	true
01-04-11	00-10-02	03-11-25

Рис. 7. Пример автоматически сгенерированной задачи преобразования табличных данных

Зачет по курсу языка Python также проходит с использованием процедурно сгенерированных задач. На рис. 9 представлен веб-интерфейс, с помощью которого обучающиеся присылают свои решения этого типа задач.

Основная часть семинарского занятия приходится на обсуждение и решение практических задач из Jupyter-блокнотов. Несколько таких задач разбираются «у доски»: преподаватель предлагает кому-то из обучающихся занять место у компьютера, подключенного к проектору аудитории. В таком формате остальные обучающиеся могут следить за ходом решения и комментариями преподавателя. Более подготовленные обучающиеся в процессе занятия самостоятельно решают задачи и присылают решения на проверку преподавателю через систему отслеживания ошибок. Такая схема работы позволяет преподавателю активно участвовать в занятии, не перемещаясь постоянно по аудитории от компьютера к компьютеру и, тем самым, более строго следовать ограничениям в условиях пандемии COVID-19. Внезапный переход на дистанционную форму обучения при такой схеме практического занятия

Задача 3.1. Реализовать разбор двоичного формата данных (в духе формата WAD игры Doom или графического формата PNG). Данные начинаются с сигнатуры 0 × 42, 0 × 58, 0 × 59, 0 × 9a, за которой следует структура A. Порядок байт: от старшего к младшему. Адреса указаны в виде смещений от начала данных. В решении разрешено использовать модуль struct.

Структура A:

1	Массив структур B, размер 4
2	int16
3	int16
4	int16
5	Структура C
6	Размер (uint32) и адрес (uint32) массива double
7	uint32
8	Структура D

Структура B:

1	uint32
2	double
3	Размер (uint32) и адрес (uint32) массива char

Структура C:

1	Массив int16, размер 5
2	uint64
3	int16
4	int8

Структура D:

1	int16
2	int8
3	int8
4	uint64
5	Размер (uint32) и адрес (uint32) массива uint16

Рис. 8. Пример автоматически сгенерированной задачи разбора двоичного файлового формата (тестовые данные не показаны)

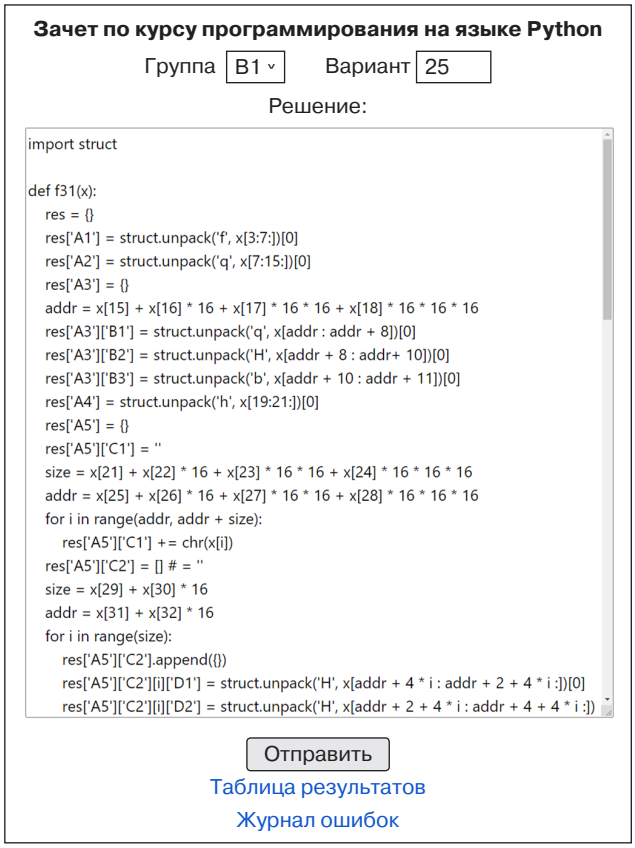

Рис. 9. Веб-интерфейс для приема решений автоматически сгенерированных задач

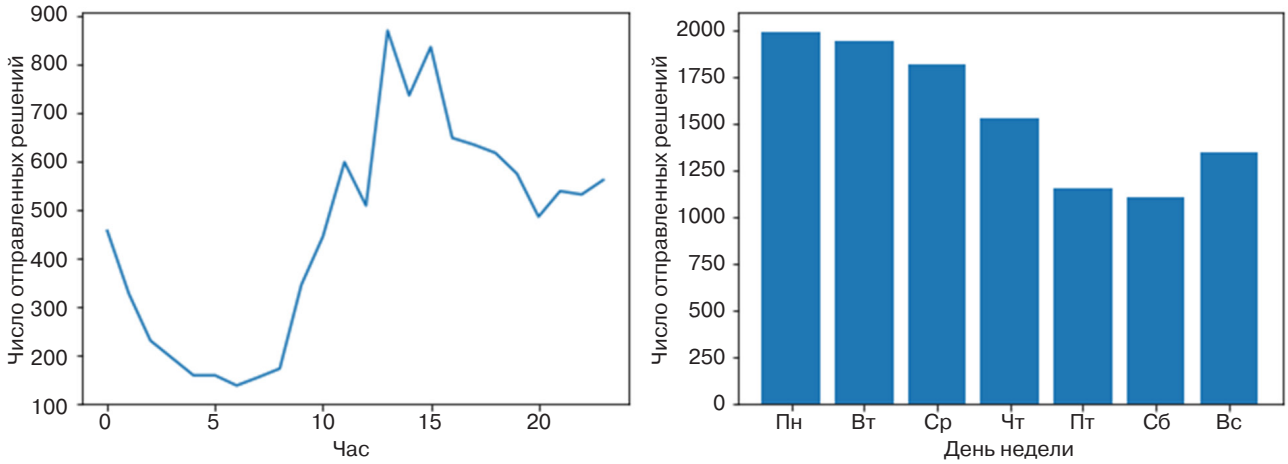

Рис. 10. Активность обучающихся в течение суток и дней недели

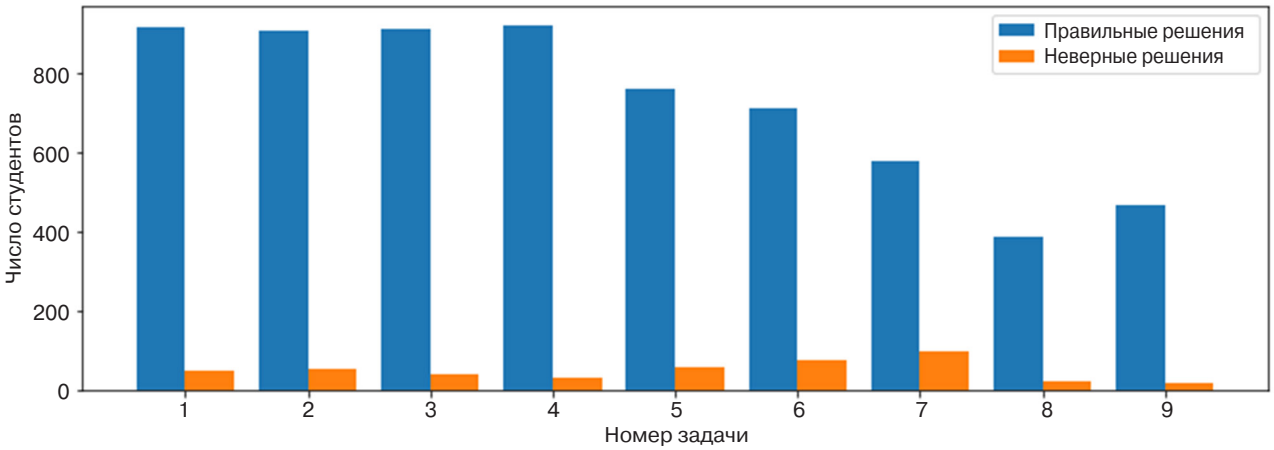

Рис. 11. Статистика по отправленным обучающимися решениям процедурно сгенерированных задач

не является серьезной проблемой, поскольку обучающиеся пользуются для отправки решений задач все той же системой отслеживания ошибок.

Практические задачи из Jupyter-блокнотов относятся к различным предметным областям и разработаны таким образом, чтобы представлять интерес для студентов, а также обеспечить возможность получить решение в течение одного практического занятия. В качестве тем практических задач используются следующие: генератор случайных докладов по цифровой экономике, преобразование Барроуза – Уиллера, генератор ASCII-баннера, модель сегрегации Шеллинга, изображение графа на основе физического моделирования, интерпретатор стекового языка, алгоритм иерархической кластеризации, формальная верификация головоломок из компьютерных игр, механизм undo/redo в графическом редакторе, язык запросов в духе SQL, фрактал Жюлиа, алгоритм Флойда – Стейнберга.

Отдельным видом практических задач являются задачи простого анализа данных на основе информации, собранной с помощью веб-инструмента для приема студенческих решений процедурно-сгенерированных задач. Здесь в режиме рефлексии обучающиеся изучают собственную активность на курсе. Примеры такого анализа показаны на рис. 10 и рис. 11.

Дополнительные задачи для практических занятий связаны со специально разработанной для рассматриваемого курса игрой DandyBot. Она относится к жанру игр для программистов, в которых для успеха в игре необходимо написать программный код. Игра DandyBot, пример уровня которой показан на рис. 12, выдержана в стилистике Roguelike-игр и позволяет запрограммировать на языке Python «интеллект» игровых персонажей.

На первых уровнях играющим требуется писать простейшие программы, что дает возможность привлечь к написанию кода на Python в игровой форме даже тех обучающихся, которые имеют минимальный опыт программирования.

Игра DandyBot в виде GitHub-репозитория, с которым работают обучающиеся, представляет собой заготовку, требующую многочисленных доработок. Это стимулирует обучающихся к развитию программного кода проекта DandyBot, а также к созданию дополнительных уровней.

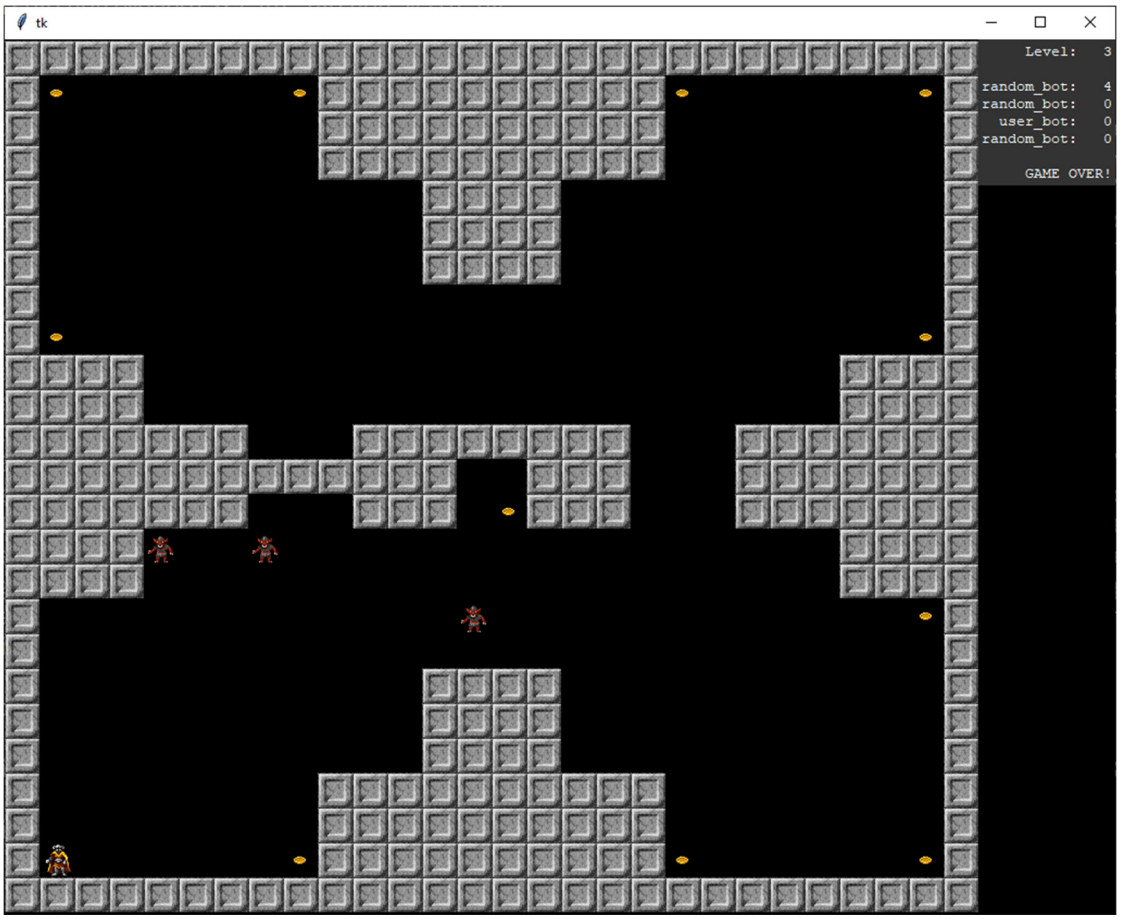

Рис. 12. Уровень игры DandyBot с управляемым программой на языке Python игроком

ЗАКЛЮЧЕНИЕ

В современных условиях (массовое обучение программированию, переход с очной формы обучения на дистанционное/смешанное и обратно в течение семестра, конкуренция со специализированными курсами и платформами обучения языкам программирования) преподаватели университетских курсов по обучению программированию сталкиваются с существенным увеличением нагрузки, связанным в большинстве случаев с выполнением рутинных действий.

Применение разработанного цифрового ассистента преподавателя, автоматизирующего такие процедуры, как генерация индивидуальных задач и проверка их решения с информативным ответом для обучающегося, ведение статистики успеваемости обучающихся, позволяет преподавателю уделить больше рабочего времени и внимания функциям, требующим творческого подхода, в частности, составлению и обсуждению нетривиальных задач по программированию.

Опыт внедрения ЦАП в учебный процесс курса «Программирование на языке Python» подтвердил возможность обеспечения персонификации учебного процесса для обучающихся в виде ИОТ.

Вклад авторов

Е.Г. Андрианова: Обоснована актуальность автоматизации процесса массового обучения программированию. Исследованы и адаптированы методы персонификации учебного процесса для массового обучения программированию, выделены классы практических задач, отражающих современную специфику разработки программного обеспечения.

Л.А. Демидова: Предложена процедура комплексной оценки решения обучающегося, включающая проверку корректности результата и проверку на плагиат решений в случае задач, созданных преподавателем вручную, а также соответствие стандарту стиля написания программы, метрикам оценки сложности программы и т.д. Описаны ведение статистики успеваемости обучающихся и интерфейс взаимодействия обучающихся и преподавателей.

П.Н. Советов: Разработаны алгоритмы, удовлетворяющие ограничениям для создания генераторов задач по программированию, построена архитектура системы «Цифровой Ассистент Преподавателя» и выполнена реализация данной системы. Разработана и описана процедура автоматической проверки решения задач, осуществляемая с помощью набора программных тестов, сформированного генератором задач.

Все авторы – разработка и внедрение системы «Цифровой Ассистент Преподавателя» в практику учебного процесса РТУ МИРЭА по дисциплине «Программирование на языке Python».

Authors' contributions

E.G. Andrianova: The urgency of automating the process of massive programming training was substantiated. The personification methods for the massive programming training process were investigated and adapted. Types of practical tasks reflecting the contemporary specifics of software development were highlighted.

L.A. Demidova: A procedure for a comprehensive assessment of the student's solution was proposed. This procedure included checking the correctness of the result and checking for plagiarism of solutions in the case of tasks created by the teacher manually, as well as compliance of the program style with the standard and metrics for assessing the program complexity, etc. The maintenance of statistics of students' progress and the interface of interaction between students and teachers are described.

P.N. Sovetov: Algorithms satisfying the constraints for creating generators of programming tasks were developed, the Digital Teaching Assistant computer system architecture was built, and the implementation of this system was completed. A procedure for automatic verification of task solutions was developed and described. This procedure is realized using software tests generated by the task generator. He

All authors took part in the development and implementation of the Digital Teaching Assistant computer system in the practice of the RTU MIREA educational process in the Programming in Python discipline.

СПИСОК ЛИТЕРАТУРЫ

1. Гудов М.М., Ермакова Э.Р. Структурные преобразования российской экономики в условиях форсированной цифровизации производственных отношений. *Теоретическая и прикладная экономика.* 2020;2:1–8. https://doi.org/10.25136/2409-8647.2020.2.32625
2. Новикова Е.С. Риски и перспективы трансформации высшей школы для российской экономики в условиях глобализации и цифровизации. *Международная торговля и торговая политика*. 2021;7(4):147–162. https://doi.org/10.21686/2410-7395-2021-3-147-162
3. Ярославцева Е.И. Гуманитарные аспекты цифровых технологий. *Вестник Российского философского общества*. 2020;1–2(91–92):248–251. URL: https://rfo1971.ru/wp-content/uploads/2020/03/09-03_248-251.pdf
4. Ярославцева Е.И. Потенциал цифровых технологий и проблемы творчества человека. *Вопросы философии*. 2020;11:58–66. https://doi.org/10.21146/0042-8744-2020-11-58-66
5. Строков А.А. Цифровизация образования: проблемы и перспективы. *Вестник Мининского университета*. 2020;8(2):15. https://doi.org/10.26795/2307-1281-2020-8-2-15
6. Хуторской А.В. Педагогические предпосылки самореализации ученика в эвристическом обучении. *Вестник Института образования человека*. 2020;1:1. URL: http://eidos-institute.ru/journal/2020/100/Eidos-Vestnik2020-101-Khutorskoy.pdf
7. Хуторской А.В. Интериоризация и экстериоризация – два подхода к образованию человека. *Народное образование*. 2021;1(1484):37–49.
8. Khalyapina L., Kuznetsova O. Multimedia professional content foreign language competency formation in a digital educational system exemplified by STEPIK framework. *Lecture Notes in Networks and Systems*. 2020;131:357–366. https://doi.org/10.1007/978-3-030-47415-7_38
9. Панова И.В., Коливнык А.А. Обзор содержания онлайн курсов по обучению основам программирования на языке Python. В: *Современные образовательные Web-технологии в реализации личностного потенциала обучающихся*. сборник статей участников Международной научно-практической конференции. Арзамас; 2020. С. 523–528.

REFERENCES

1. Gudov M.M., Ermakova E.R. Structural transformations of the Russian economy in the conditions of acceleration digitalization of industrial relations. *Teoreticheskaya i prikladnaya ekonomika = Theoretical and Applied Economics*. 2020;2:1–8 (in Russ.). https://doi.org/10.25136/2409-8647.2020.2.32625
2. Novikova E.S. Risks and perspectives of higher school transformation for the Russian economy in conditions of gobalization and digitalization. *Mezhdunarodnaya torgovlya i torgovaya politika = International Trade and Trade Policy*. 2021;7(4):147–162 (in Russ.). https://doi.org/10.21686/2410-7395-2021-3-147-162
3. Yaroslavtseva E.I. Humanitarian aspects of digital technologies. *Vestnik Rossiiskogo filosofskogo obshchestva = Russian Philosophical Society*. 2020;1–2(91–92):248–251 (in Russ.). Available from URL: https://rfo1971.ru/wp-content/uploads/2020/03/09-03_248-251.pdf
4. Yaroslavtseva E.I. The potential of digital technologies and the problems of human creativity. *Voprosy Filosofii*. 2020;11:58–66 (in Russ.). https://doi.org/10.21146/0042-8744-2020-11-58-66.
5. Strokov A.A. Digitalization of education: problems and prospects. *Vestnik Mininskogo universiteta = Vestnik of Minin University*. 2020;8(2):15 (in Russ.). https://doi.org/10.26795/2307-1281-2020-8-2-15
6. Khutorskoi A.V. Pedagogical prerequisites for student self-realization in heuristic learning. *Vestnik Instituta Obrazovaniya Cheloveka*. 2020;1:1 (in Russ.). Available from URL: http://eidos-institute.ru/journal/2020/100/Eidos-Vestnik2020-101-Khutorskoy.pdf
7. Khutorskoi A.V. Interiorization and exteriorization – two approaches to human education. *Narodnoe obrazovanie*. 2021;1(1484):37–49 (in Russ.).
8. Khalyapina L., Kuznetsova O. Multimedia professional content foreign language competency formation in a digital educational system exemplified by STEPIK framework. *Lecture Notes in Networks and Systems*. 2020;131:357–366. https://doi.org/10.1007/978-3-030-47415-7_38
9. Panova I.V., Kolivnyk A.A. An overview of the content of online courses on teaching the basics of programming in the Python language. In: *Sovremennye obrazovatel'nye Web-tekhnologii v realizatsii lichnostnogo potentsiala obuchayushchikhsya* (*Contemporary Educational Web-technologies in the Realization of the Personal Potential of Students*): Collection of research articles of international scientific and practical conference. Arzamas; 2020. P. 523–528 (in Russ).

10. Воробьева Н.А., Обоева С.В., Бернадинер М.И. Использование технологий педагогического дизайна в условиях цифровизации образования. *Вестник Московского городского педагогического университета. Серия: Информатика и информатизация образования*. 2020;1(51):34–37.
11. Guo P.J., Kim J., Rubin R. How video production affects student engagement: An empirical study of MOOC videos. In: *Proceedings of the First ACM Conference on Learning @ Scale Conference*. 2014. P. 41–50. https://doi.org/10.1145/2556325.2566239
12. Lau S., Guo P.J. Data theater: a live programming environment for prototyping data-driven explorable explanations. *Workshop on Live Programming* (*LIVE*). 2020. 6 p. URL: https://www.samlau.me/pubs/Data-Theater-prototyping-explorable-explanations_LIVE-2020.pdf
13. Guo P.J. Online python tutor: embeddable web-based program visualization for cs education. In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. 2013. P. 579–584. https://doi.org/10.1145/2445196.2445368
14. Miller H., Willcox K., Huang L. Crosslinks: Improving course connectivity using online open educational resources. *The Bridge*. 2016;43(3):38–45. URL: http://hdl.handle.net/1721.1/117022
15. Utterberg M.M., et al. Intelligent tutoring systems: Why teachers abandoned a technology aimed at automating teaching processes. In: *Proceedings of the 54th Hawaii International Conference on System Sciences.* 2021. P. 1538. URL: http://hdl.handle.net/10125/70798
16. Sherman M., et al. Impact of auto-grading on an introductory computing course. *J. Comput. Sci. Coll.* 2013;28(6):69–75.
17. Rivers K., Koedinger K.R. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *Int. J. Artif. Intell. Educ*. 2017;27(1):37–64. https://doi.org/10.1007/s40593-015-0070-z
18. Sovietov P. Automatic generation of programming exercises. In: *2021 1st International Conference on Technology Enhanced Learning in Higher Education* (*TELE*). IEEE. 2021. P. 111–114. https://doi.org/10.1109/TELE52840.2021.9482762
19. Schleimer S., Wilkerson D.S., Aiken A. Winnowing: local algorithms for document fingerprinting. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (*SIGMOD'03*). 2003. P. 76–85. https://doi.org/10.1145/872757.872770
20. Rogers M., et al. Exploring personalization of gamification in an introductory programming course. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (*SIGCSE'21*). 2021. P. 1121–1127. https://doi.org/10.1145/3408877.3432402
21. Putnam V., Conati C. Exploring the need for explainable artificial intelligence (XAI) in intelligent tutoring systems (ITS). *IUI Workshops*. 2019. V. 19. URL: https://explainablesystems.comp.nus.edu.sg/2019/wp-content/uploads/2019/02/IUI19WS-ExSS2019-19.pdf
22. Щербина О.А. Удовлетворение ограничений и программирование в ограничениях. *Интеллектуальные системы*. 2011;15(1–4):53–170.

10. Vorob'eva N.A., Oboeva S.V., Bernadiner M.I. Using pedagogical design technologies in the context of digitalization of education. *Vestnik Moskovskogo gorodskogo pedagogicheskogo universiteta. Seriya: Informatika i informatizatsiya obrazovaniya = The academic Journal of Moscow City University, series Informatics and Informatization of Education*. 2020;1(51):34–37 (in Russ).
11. Guo P.J., Kim J., Rubin R. How video production affects student engagement: An empirical study of MOOC videos. In: *Proceedings of the First ACM Conference on Learning @ Scale Conference*. 2014. P. 41–50. https://doi.org/10.1145/2556325.2566239
12. Lau S., Guo P.J. Data Theater: A live programming environment for prototyping data-driven explorable explanations. *Workshop on Live Programming* (*LIVE*). 2020. 6 p. Available from URL: https://www.samlau.me/pubs/Data-Theater-prototyping-explorable-explanations_LIVE-2020.pdf
13. Guo P.J. Online python tutor: embeddable web-based program visualization for cs education. In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. 2013. P. 579–584. https://doi.org/10.1145/2445196.2445368
14. Miller H., Willcox K., Huang L. Crosslinks: Improving course connectivity using online open educational resources. *The Bridge*. 2016;43(3):38–45. Available from URL: http://hdl.handle.net/1721.1/117022
15. Utterberg M.M., et al. Intelligent tutoring systems: Why teachers abandoned a technology aimed at automating teaching processes. In: *Proceedings of the 54th Hawaii International Conference on System Sciences.* 2021. P. 1538. Available from URL: http://hdl.handle.net/10125/70798
16. Sherman M., et al. Impact of auto-grading on an introductory computing course. *J. Comput. Sci. Coll.* 2013;28(6):69–75.
17. Rivers K., Koedinger K.R. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *Int. J. Artif. Intell. Educ*. 2017;27(1):37–64. https://doi.org/10.1007/s40593-015-0070-z
18. Sovietov P. Automatic generation of programming exercises. In: *2021 1st International Conference on Technology Enhanced Learning in Higher Education* (*TELE*). IEEE. 2021. P. 111–114. https://doi.org/10.1109/TELE52840.2021.9482762
19. Schleimer S., Wilkerson D.S., Aiken A. Winnowing: local algorithms for document fingerprinting. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (*SIGMOD'03*). 2003. P. 76–85. https://doi.org/10.1145/872757.872770
20. Rogers M., et al. Exploring personalization of gamification in an Introductory programming course. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (*SIGCSE'21*). 2021. P. 1121–1127. https://doi.org/10.1145/3408877.3432402
21. Putnam V., Conati C. Exploring the need for explainable artificial intelligence (XAI) in intelligent tutoring systems (ITS). *IUI Workshops*. 2019. V. 19. Available from URL: https://explainablesystems.comp.nus.edu.sg/2019/wp-content/uploads/2019/02/IUI19WS-ExSS2019-19.pdf

23. Wang J., Chen B., Wei L., Liu Y. Skyfire: Data-driven seed generation for fuzzing. In: *2017 IEEE Symposium on Security and Privacy* (*SP*). IEEE. 2017. P. 579–594. https://doi.org/10.1109/SP.2017.23
24. Papadakis M., et al. Mutation testing advances: an analysis and survey. *Adv. Comput*. 2019;112:275–378. https://doi.org/10.1016/bs.adcom.2018.03.015
25. Hutton G., Meijer E. *Monadic Parser Combinators*. Technical Report NOTTCS-TR-96-4. Department of Computer Science, University of Nottingham. 1996. 38 p. URL: https://www.cs.nott.ac.uk/~pszgmh/monparsing.pdf
26. Phothilimthana P.M., Sridhara S. High-coverage hint generation for massive courses: Do automated hints help CS1 students? In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (*ITiCSE'17*). 2017. P. 182–187. https://doi.org/10.1145/3059009.3059058

22. Shcherbina O.A. Constraint satisfaction and constraint programming. *Intellektual'nye sistemy = Intelligent Systems*. 2011;15(1–4):53–170 (in Russ.).
23. Wang J., Chen B., Wei L., Liu Y. Skyfire: Data-driven seed generation for fuzzing. In: *2017 IEEE Symposium on Security and Privacy* (*SP*). IEEE. 2017. P. 579–594. https://doi.org/10.1109/SP.2017.23
24. Papadakis M., et al. Mutation testing advances: an analysis and survey. *Adv. Comput*. 2019;112:275–378. https://doi.org/10.1016/bs.adcom.2018.03.015
25. Hutton G., Meijer E. *Monadic Parser Combinators*. Technical Report NOTTCS-TR-96-4. Department of Computer Science, University of Nottingham. 1996. 38 p. Available from URL: https://www.cs.nott.ac.uk/~pszgmh/monparsing.pdf
26. Phothilimthana P.M., Sridhara S. High-coverage hint generation for massive courses: Do automated hints help CS1 students? In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (*ITiCSE'17*). 2017. P. 182–187. https://doi.org/10.1145/3059009.3059058

Об авторах

Андрианова Елена Гельевна, к.т.н., доцент, заведующий кафедрой корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: andrianova@mirea.ru. Scopus Author ID 57200555430, ResearcherID T-7908-2018, SPIN-код РИНЦ 9858-3229, http://orcid.org/0000-0001-6418-6797

Демидова Лилия Анатольевна, д.т.н., профессор, профессор кафедры корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: demidova@mirea.ru. Scopus Author ID 56406258800, ResearcherID R-6077-2016, SPIN-код РИНЦ 9447-3568, http://orcid.org/0000-0003-4516-3746

Советов Петр Николаевич, к.т.н., доцент кафедры корпоративных информационных систем Института информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: sovetov@mirea.ru. Scopus Author ID 57221375427, SPIN-код РИНЦ 9999-1460, http://orcid.org/0000-0002-1039-2429

About the authors

Elena G. Andrianova, Cand. Sci. (Eng.), Associated Professor, Head of the Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: andrianova@mirea.ru. Scopus Author ID 57200555430, ResearcherID T-7908-2018, SPIN-code RSCI 9858-3229, http://orcid.org/0000-0001-6418-6797

Liliya A. Demidova, Dr. Sci. (Eng.), Professor, Professor of the Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: demidova@mirea.ru. Scopus Author ID 56406258800, ResearcherID R-6077-2016, SPIN-code RSCI 9447-3568, http://orcid.org/0000-0003-4516-3746

Petr N. Sovetov, Cand. Sci. (Eng.), Associated Professor, Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: sovetov@mirea.ru. Scopus Author ID 57221375427, SPIN-code RSCI 9999-1460, http://orcid.org/0000-0002-1039-2429