Интеллектуальный учёт учебных достижений в системе «Цифровой ассистент преподавателя»

А.В. Горчаков, Л.А. Демидова, П.Н. Советов

***Аннотация*—Продолжающийся процесс цифровизации экономики обуславливает необходимость массовой подготовки разработчиков программного обеспечения, с увеличением числа студентов растёт нагрузка на преподавателей университетских курсов по программированию. Автоматизация некоторых аспектов таких курсов позволяет избавить преподавателей от рутинной деятельности. В данной статье рассматривается система Цифровой ассистент преподавателя (ЦАП), автоматизирующая курс программирования на языке Python в МИРЭА – Российском технологическом университете. ЦАП состоит из ядра, обеспечивающего порождение и проверку уникальных задач по программированию, а также интеллектуального модуля учёта учебных достижений для мотивации студентов к решению задач различающимися способами, веб-приложения. В статье рассмотрена архитектура системы ЦАП, основными особенностями которой являются автоматическая генерация уникальных задач по программированию одиннадцати различных типов, включающая реализацию авторских алгоритмов, интегрированные методы интеллектуального анализа данных и машинного обучения для автоматизации выявления подходов к решению задач в текстах программ, присылаемых на проверку. Приведены алгоритмы, автоматизирующие определение способов решения уникальных задач по программированию для выявления и учёта учебных достижений. Приведена статистика внедрения ЦАП, собранная в процессе эксплуатации системы.**

***Ключевые слова*—онлайн-образование, генерация задач по программированию, информационная система, анализ текстов программ, анализ программного кода.**

I. Введение

При массовом характере университетских курсов по программированию для преподавателей трудно, а подчас и невозможно, уделить время всем студентам в равной степени, придумать достаточно много уникальных задач в условиях, когда списывание становится нормой, детально проверить программные решения и предоставить студентам подробную обратную связь. Описанные трудности приводят к повышенной нагрузке на преподавателей, что может спровоцировать их эмоциональное выгорание, снизить желание творчески подходить к преподавательской деятельности [1, 2].

Известно об использовании информационных систем с поддержкой автоматизированной проверки задач по программированию [3-4] для решения обозначенных выше проблем. Система, описанная в работе [3], автоматизирует такие виды преподавательской деятельности, как проверка решений студентов и учет статистики успеваемости, при этом задачи, которые предлагается решить студентам, создаются преподавателями вручную. В работе [4] предложен предметно-ориентированный язык для описания задач, которые студентам предлагается решать конструктивно-выборочным методом, переставляя местами заранее известные строки исходного кода.

Заимствование программных решений задач курсов программирования в высшей школе является серьезной проблемой, которую необходимо решать при автоматизации проверки выданных студентам задач. В работах [5, 6] описан опыт применения алгоритмов кластеризации к наборам текстов программ для выявления дубликатов кода, причём в [5] анализируются наборы ключевых слов языков программирования, а в [6] сравниваются деревья абстрактного синтаксиса (англ. abstract syntax tree, AST), и на основе результатов сравнений выполняется кластеризация. Для полного исключения списывания в системах, автоматизирующих дисциплины математического цикла в высшей школе, поддерживается генерация уникальных задач по математике [7].

В РТУ МИРЭА была разработана система «Цифровой ассистент преподавателя» (ЦАП), автоматизирующая такие виды преподавательской деятельности в курсе Программирования на языке Python, как порождение уникальных задач по программированию 11 различных типов для каждого студента для предотвращения списывания [1], проверка программных решений задач, присылаемых студентами [1, 2], ведение статистики успеваемости [2], интеллектуальный анализ проверенных и принятых текстов программ для определения использованного подхода к решению задачи [8, 9].

Система ЦАП состоит из трёх модулей – ядра, веб-приложения и аналитической подсистемы. Архитектура системы ЦАП представлена на рисунке 1. Ядро системы обеспечивает генерацию уникальных задач и наборов тестов для каждого студента [1], а также осуществляет проверку присылаемых студентами решений. Веб-приложение ЦАП обеспечивает возможность человеко-

Статья получена 08.01.2022.

Горчаков А.В., кафедра корпоративных информационных систем, институт информационных технологий, МИРЭА – Российский технологический университет (e-mail: worldbeater-dev@yandex.ru)

Демидова Л.А., кафедра корпоративных информационных систем, институт информационных технологий, МИРЭА – Российский технологический университет (e-mail: liliya.demidova@rambler.ru)

Советов П.Н., кафедра корпоративных информационных систем, институт информационных технологий, МИРЭА – Российский технологический университет (e-mail: sovetov@mirea.ru)

машинного взаимодействия с ядром, предоставляя пользовательский интерфейс [2] для просмотра списка групп, просмотра статуса выполнения задач студентами выбранной группы и отправки решения выбранной задачи. Веб-интерфейс используется преподавателями для отслеживания успеваемости и активности студентов в течение семестра. Аналитическая подсистема включает средства для интеллектуального определения способов решения задач в текстах программ, присланных студентами (см. рис. 1), для выявления пробелов в навыках и знаниях студентов, для оценки качества и совершенствования генераторов задач [6,7].

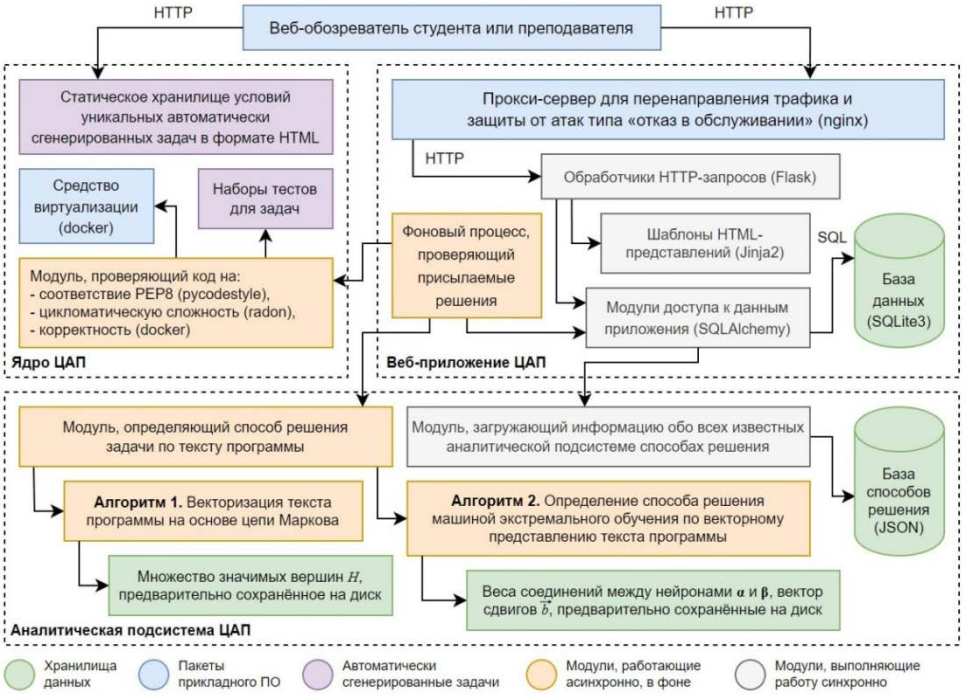


Рисунок 1. Архитектура системы «Цифровой ассистент преподавателя»

Также аналитический модуль используется для учёта учебных достижений в ЦАП – студентам предлагается решать автоматически сгенерированные уникальные задачи всеми известными аналитическому модулю способами, решение одним из известных системе способов считается достижением. Согласно данным, представленным в работе [10], внедрение систем достижений в образовательный процесс положительно сказывается как на успеваемости студентов, так и на времени, проведённом за решением задач. В [11] показано, что внедрение игровых подходов в образовательный процесс позволяет повысить вовлеченность студентов.

Данная статья структурирована следующим образом. В секции II рассмотрено ядро системы ЦАП, обеспечивающее генерацию уникальных задач по программированию и их автоматизированную проверку. В секции III описана архитектура веб-приложения ЦАП, при помощи которого студенты и преподаватели взаимодействуют с системой. В секции IV описаны методы интеллектуального учёта учебных достижений в ЦАП. В секции V приведена статистика внедрения ЦАП в образовательный процесс.

II. Ядро для генерации задач и оценки решений

Задача генерации упражнений по программированию может быть представлена в виде задачи удовлетворения ограничениям и решаться с помощью метода «сгенерировать и проверить» (generate and test) [1, 2]. Типы генерируемых ядром системы ЦАП задач относятся к одному из двух классов: перевод некоторой нотации в программу, преобразование форматов данных. Перечень типов задач, которые предлагается решить студентам курса программирования, приведён в таблице 1.

Таблица 1. Перечень типов задач, поддерживаемых системой Цифровой ассистент преподавателя.

№	Краткое описание типа задачи	Класс задачи
1.	Реализовать функцию	Перевод нотации в программу
2.	Реализовать кусочную функцию	Перевод нотации в программу
3.	Реализовать итерационную функцию	Перевод нотации в программу
4.	Реализовать рекуррентную функцию	Перевод нотации в программу
5.	Реализовать функцию, оперирующую векторами	Перевод нотации в программу
6.	Реализовать функцию, вычисляющую дерево решений	Перевод нотации в программу
7.	Реализовать преобразование битовых полей	Преобразование форматов данных
8.	Реализовать разбор текстового формата	Преобразование форматов данных
9.	Реализовать конечный автомат Мили в виде класса	Перевод нотации в программу
10.	Реализовать преобразования табличных данных	Преобразование форматов данных
11.	Реализовать разбор двоичного формата данных	Преобразование форматов данных

Сборник задач для студентов генерируется *ядром* (см. рис. 1) заранее и представляет собой набор HTML-документов, созданных при помощи инструмента Pan-

doc [12] из файлов в формате Markdown, в которых используется язык LaTeX для описания формул, язык DOT [13] для описания графов или иной декларативный язык разметки для визуализации условия задачи.

Проверка решений состоит из двух этапов: статического анализа, включающего в себя проверку кода на соответствие стандарту PEP8, проверку цикломатической сложности кода, а также тестирования программы, осуществляемого в «песочнице», Docker-контейнере [14] со средой выполнения gVisor [15], обеспечивающей изоляцию для защиты от несанкционированного доступа. При несовпадении результатов тестирования веб-приложение ЦАП демонстрирует студенту только выходные данные теста, выполнение которого вызвало ответ, отличный от ожидаемого.

III. Веб-приложение системы ЦАП

Для обеспечения возможности человеко-машинного взаимодействия с ядром системы, осуществляющим приём и проверку присылаемых решений, используется *веб-приложение ЦАП* (см. рис. 1). Веб-приложением автоматизируется приём решений задач, проверка присланных решений, делегируемая ядру из фонового процесса, также веб-приложение визуализирует статистику выполнения задач студентами, что является полезным как для преподавателей для отслеживания активности обучающихся, так и для самоконтроля студентов.

Веб-приложение ЦАП реализовано согласно архитектуре «Модель-представление-контроллер» (англ. Model-View-Controller, MVC) [16], включает в себя обработчики HTTP-запросов (контроллеры), шаблоны HTML-страниц (представления), модули доступа к данным и функции обработки данных (модель). Диаграмма последовательностей для процесса взаимодействия студента с системой ЦАП представлена на рисунке 2.

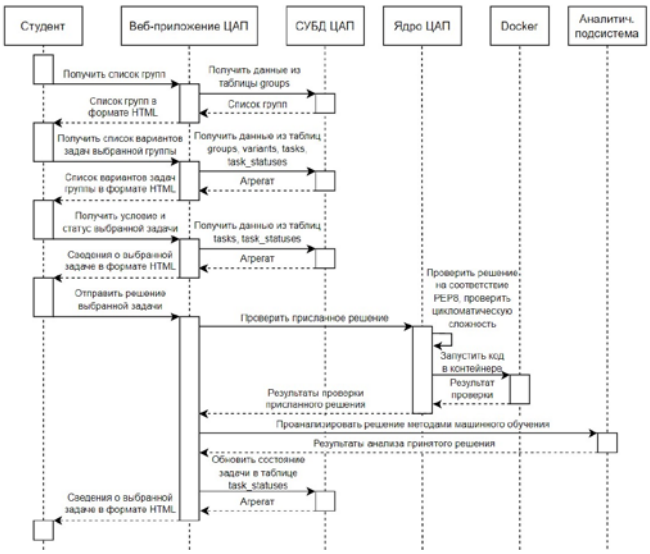


Рисунок 2. Диаграмма последовательностей системы «Цифровой ассистент преподавателя»

Для хранения данных используется реляционная система управления базами данных SQLite3. Модули доступа к данным содержат SQL-запросы, реализованные в соответствии с требованиями предметной области, и позволяют извлекать агрегаты из базы данных посредством объектно-реляционного отображения записей из базы данных в объекты Python с использованием библиотеки SQLAlchemy [17]. Диаграмма сущность-связь (англ. Entity relationship, ER) для схемы базы данных веб-приложения ЦАП показана на рисунке 3. Как показано на рисунке 3, база данных веб-приложения включает в себя 7 таблиц и обеспечивает хранение номеров задач, сведений о группах, номерах вариантов, соответствующих порядковым номерам студентов в списках групп, статусов задач, содержащих информацию о том, принято ли отправленное решение.

Таблица «messages» содержит сведения обо всех текстах программ, отправленных студентами в систему. Запущенный в фоне процесс время от времени анализирует таблицу «messages» и в случае наличия новых текстов программ последовательно выполняет их проверку, обновляя таблицу «task_statuses», в которой содержатся сведения об успеваемости (см. рис. 3).

Таким образом, состояние, хранящееся в «task_statuses» в момент времени T, определяется серией предшествующих событий, что позволяет симулировать работу системы с момента её запуска в эксплуатацию, воспроизводить состояние системы в прошедшие моменты времени. Данный архитектурный подход имеет название Event Sourcing (ES) [18].

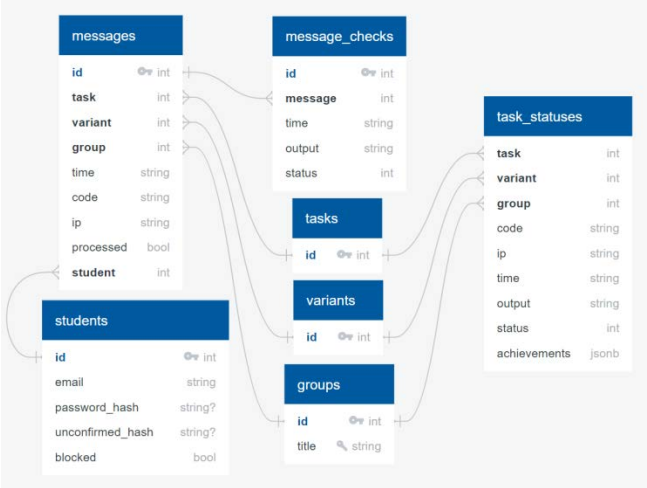


Рисунок 3. Диаграмма сущность-связь базы данных веб-приложения Цифровой ассистент преподавателя.

Результаты проверки текстов программ сохраняются в таблице «message_checks» для последующего анализа. Дополнительно в таблице «task_statuses», содержащей результаты последних проверок задач, хранится информация о достижениях студентов, под достижением понимается выявленный *аналитической подсистемой ЦАП* (см. рис. 1) способ решения задачи. Взаимодействие веб-приложения с ядром системы осуществляется посредством вызовов программных API, реализованных в ядре на языке Python, проверка решений осуществляются в фоновом процессе. В случае успешного принятия текста программы ядром осуществляется анализ кода методами интеллектуального анализа данных аналитической подсистемой (см. рис. 2).

Пользовательский интерфейс веб-приложения ЦАП для просмотра списка вариантов группы представлен на

рисунке 4. Веб-интерфейс для отправки решения задачи и отслеживания результатов проверки имеет вид, показанный на рисунке 5а, в случае, если последнее отправленное решение задачи принято системой и студентом задача была решена всеми способами, известными системе. В случае, если программа была отклонена ЦАП, в веб-интерфейсе приводятся сведения о причине отклонения программы, как показано на рисунке 5б.

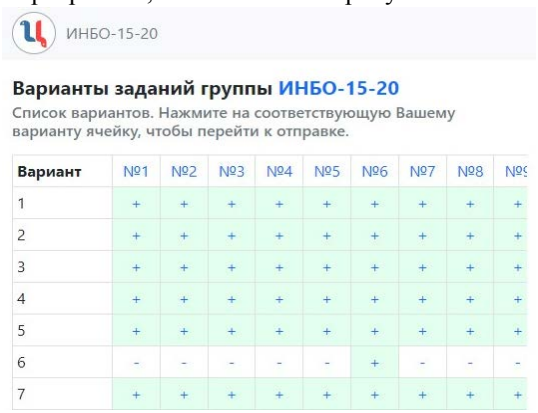
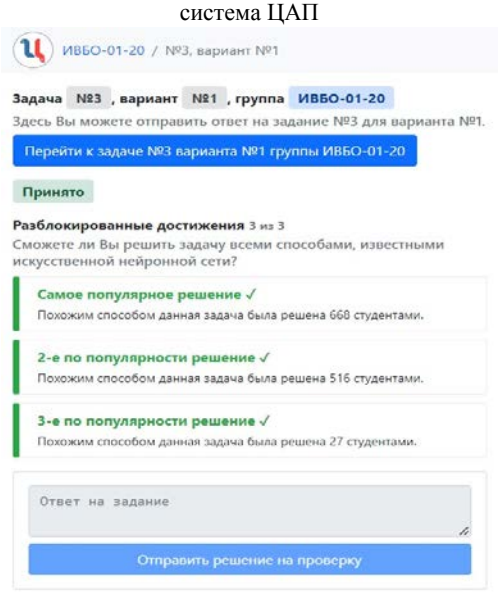


Рисунок 4. Пользовательский интерфейс перечня вариантов и задач студентов одной из групп курса, в котором применяется система ЦАП



(а)

(б)

Рисунок 5. Пользовательский интерфейс страницы отправки и отслеживания статуса задачи в ЦАП в случае: а) принятия решения системой, б) отклонения решения.

IV. Интеллектуальный учёт учебных достижений

A. Выявление основных подходов к решению задач

При разработке программного обеспечения с обширной кодовой базой нередко встречается дублирование подходов к решению задач, разработчиками программного обеспечения заимствуется публично доступный код или самостоятельно реализуется алгоритм, уже использующийся в программной системе, над которой ведётся работа. В результате последних исследований был предложен ряд методов и алгоритмов избавления от дубликатов кода, поиска заимствований, обнаружения ошибок, основанных на методах интеллектуального анализа данных и машинного обучения [5, 6, 19, 8, 9].

Проблема плагиата в ЦАП решена, поскольку ядро автоматически генерирует уникальные задачи для каждого студента. Однако, при решении уникальных задач студентами могут быть применены различающиеся подходы к решению задач. Выявление основных использованных подходов полезно как для оценки качества генераторов задач, так и при реализации системы учёта достижений для повышения мотивации студентов изучить разные подходы к решению задач [9].

В конце семестра курса Программирования на языке Python, в котором используется ЦАП, производится анализ полученных от студентов программ при помощи алгоритма кластеризации, рассмотренного в работе [8].

Результатами анализа являются кластеры из схожих решений задач. Программы, принадлежащие одному кластеру, содержат схожие, но не полностью идентичные, синтаксические конструкции, поскольку варианты уникальных автоматически сгенерированных задач значительно различаются. Таким образом, кластерный анализ позволяет выявить основные подходы к решению задач в ЦАП, использованные студентами в течение семестра.

В предложенном в работе [8] специализированном алгоритме для выявления подходов к решению задач предлагается сперва преобразовать набор текстов программ, решающих автоматически сгенерированные задачи заданного типа (см. таблицу 1), в векторные представления на основе цепей Маркова (см. рис. 7), полученные из деревьев абстрактного синтаксиса (см. рис. 6) после ряда упрощений [9].

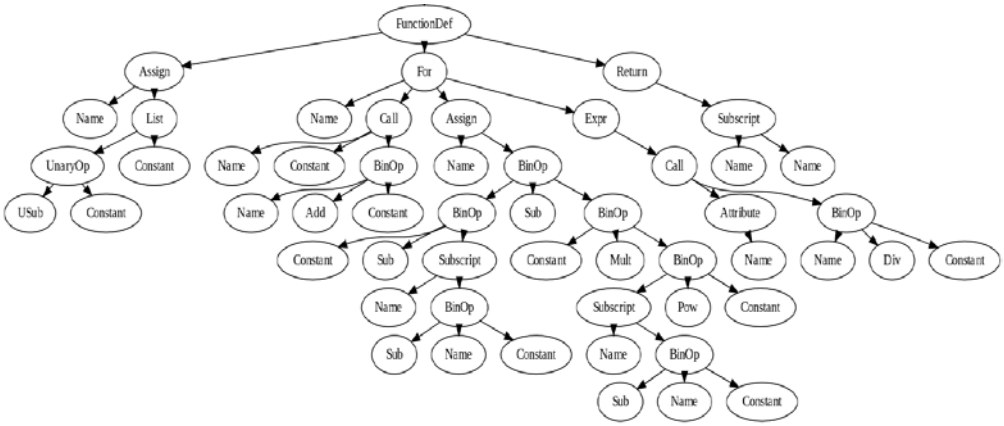


Рисунок 6. Дерево абстрактного синтаксиса для одного из решений уникальной задачи, относящихся к типу 4 (см. таблицу 1), построенное при помощи стандартной библиотеки Python

При преобразовании набора цепей Маркова в векторные представления происходит построение множества H, содержащего все различные типы вершин AST, встречающиеся программах из набора. Например, если в наборе присутствует цепь Маркова, показанная на рисунке 7, множество H имеет вид:

{FunctionDef, For, Assign, List, Op, Sub, Add, Mult, Pow, Div, Subscript, Name, …},

и может содержать иные типы вершин AST, встречающиеся в наборе цепей Маркова для анализируемых текстов программ. На основе множества H производится построение матриц смежности каждой из цепей Маркова, матрицы смежности принадлежат пространству $\mathbb{R}^m$, где m – мощность множества H.

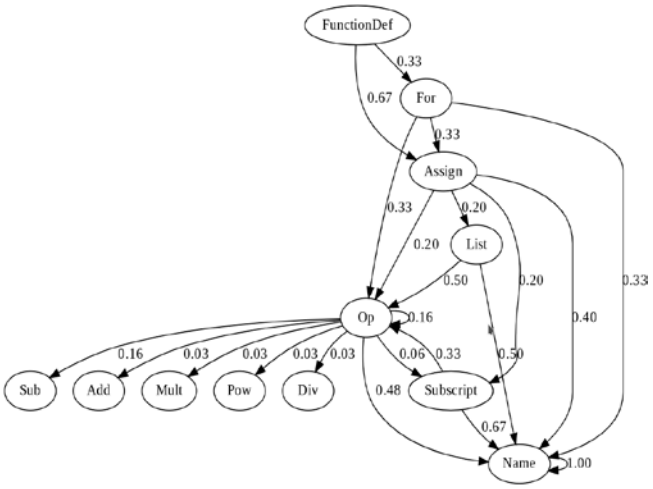


Рисунок 7. Граф переходов между состояниями цепи Маркова для дерева абстрактного синтаксиса, показанного на рис. 6.

Алгоритм 1. Векторизация программы на основе цепи Маркова

p – текст программы,
H – множество значимых вершин, построенное в процессе векторизации набора текстов программ и сохранённое на диск.

1. **Определить** R = {Import, Load, Store, arg, Module, keyword}.
2. **Построить** AST a для p средствами модуля «ast» Python.
3. **Удалить** из a вершины, принадлежащие множеству R.
4. **Заменить** узлы в a согласно [9].
5. **Построить** граф переходов цепи Маркова g для дерева a.
6. **Построить** матрицу смежности $\mathbf{M} \in \mathbb{R}^{|H|\times|H|}$ для графа g.
7. **Преобразовать** матрицу $\mathbf{M}$ к вектору $\vec{v} \in \mathbb{R}^m$, $m = |H|^2$.
8. **Возвратить** векторное представление $\vec{v}$ для текста программы p.

Множество значимых вершин H, являясь обучаемым параметром, после преобразования набора текстов программ в векторные представления сохраняется на диск и используется при векторизации новых программ, отсутствующих в исходном наборе (см. рис. 1), согласно Алгоритму 1 [9].

Затем к векторным представлениям текстов программ предлагается применить агломеративный алгоритм иерархической кластеризации с методом средней связи [8, 9], а число кластеров – выбрать на основе значений индекса кластерного силуэта [20]. Попарные расстояния между векторными представлениями текстов программ, на основе которых в [8] выполняется кластеризация, вычисляются при помощи меры Йенсена-Шеннона (англ. Jensen-Shannon Divergence, JSD) [21]:

$$\mathrm{JSD}(\vec{v}_i, \vec{v}_j) = \frac{1}{2}\sum_{k=1}^{m} v_{ik} \log_2 \frac{v_{ik}}{\frac{1}{2}(v_{ik}+v_{jk})} + \frac{1}{2}\sum_{k=1}^{m} v_{jk} \log_2 \frac{v_{jk}}{\frac{1}{2}(v_{ik}+v_{jk})}, \quad (1)$$

где $\vec{v}_i$ и $\vec{v}_j$ – сравниваемые векторные представления программ, v_{ik} – k-я компонента вектора $\vec{v}_i$, v_{jk} – k-я компонента вектора $\vec{v_j}$, m – число компонент векторного представления текста программы.

Б. Определение подходов для новых текстов программ

Кластеры, полученные в результате применения агломеративного алгоритма иерархической кластеризации к векторным представлениям текстов программ на основе цепей Маркова (см. Алгоритм 1), сортируются по уменьшению числа решений в кластерах [9]. Таким образом определяется популярность подходов к решению задач (см. рис. 5а). Векторным представлениям текстов программ ставится в соответствие метка класса – номер кластера, к которому принадлежит программа.

Размеченный таким образом набор текстов программ используется для обучения классификатора на основе машины экстремального обучения (англ. Extreme Learning Machine, ELM) – искусственной нейронной сети с эффективным алгоритмом расстановки весов [9, 22].

ELM представляет собой персептрон с единственным скрытым слоем (см. рис. 8), в котором веса соединений между входными и скрытыми нейронами $\boldsymbol{\alpha}$, а также сдвиги скрытого слоя $b_1, b_2, \ldots, b$ инициализируются случайно, а веса соединений между скрытыми и выход-

ными нейронами $\boldsymbol{\beta}$ вычисляются как:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y}_L, \quad (2)$$

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H} + \gamma\mathbf{E})^{-1}\mathbf{H}^T, \quad (3)$$

$$\mathbf{H} = \sigma(\mathbf{X}_L\boldsymbol{\alpha}^T + \mathbf{b}), \quad (4)$$

где $\mathbf{X}_L \in \mathbb{R}^{s\times n}$ – матрица, в которой s строк кодируют s векторов размерности n для объектов из обучающего набора; $\boldsymbol{\alpha}^T \in \mathbb{R}^{n\times d}$ – транспонированная матрица весов соединений между нейронами входного и скрытого слоя, причём n – число входных нейронов, d – число скрытых нейронов; $\mathbf{b} \in \mathbb{R}^{s\times d}$ – матрица, полученная преобразованием вектора сдвигов скрытого слоя вида $\mathbb{R}^d$ в матрицу $\mathbb{R}^{1\times d}$, в которой первая строка затем дублируется s раз; σ – функция активации, поэлементно применяемая к матрице; $\mathbf{H} \in \mathbb{R}^{s\times d}$ – выходная матрица скрытого слоя; $\mathbf{H}^{\dagger} \in \mathbb{R}^{d\times s}$ – псевдо обратная матрица для $\mathbf{H}$; $\mathbf{E} \in \mathbb{R}^{d\times d}$ – единичная матрица; $\mathbf{Y}_L \in \mathbb{R}^{s\times m}$ – матрица, в которой s строк кодируют s векторов размерности m для ответов из обучающего набора прецедентов; скаляры $d \in \mathbb{N}$ и $\gamma \in \mathbb{R}$ являются гиперпараметрами алгоритма ELM.

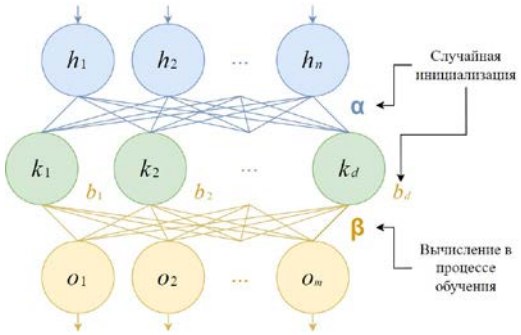


Рисунок 8. Машина экстремального обучения.

Алгоритм 2. Определение способа решения машиной экстремального обучения по векторному представлению текста программы.

$\mathbf{X}_T \in \mathbb{R}^{s\times n}$ – матрица из s строк, кодирующих n-мерные векторные представления текстов программ на основе цепей Маркова,
$\gamma \in \mathbb{R}$ – параметр регуляризации,
$d \in \mathbb{N}$ – число скрытых нейронов,
σ – функция активации скрытого слоя,
$\boldsymbol{\alpha} \in \mathbb{R}^{d\times n}$ – входные веса,
$\boldsymbol{\beta} \in \mathbb{R}^{d\times m}$ – выходные веса,
$\vec{b} \in \mathbb{R}^d$ – сдвиги скрытого слоя.

1. **Вычислить** матрицу $\mathbf{b} \in \mathbb{R}^{s\times d}$ клонированием $\vec{b} \in \mathbb{R}^d$ s раз.
2. **Вычислить** матрицу $\mathbf{H} = \sigma(\mathbf{X}_T\boldsymbol{\alpha}^T + \mathbf{b})$.
3. **Вычислить** матрицу ответов $\mathbf{Y}_T = \mathbf{H}\boldsymbol{\beta}$, причём $\mathbf{Y}_T \in \mathbb{R}^{n\times m}$.
4. **Возвратить** $\mathbf{Y}_T$, содержащую s строк, каждая из которых содержит m-мерный унитарный код способа решения.

При классификации векторных представлений текстов программ в качестве функции активации σ в формуле (4) в аналитической подсистеме ЦАП используется логистическая сигмоида:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \quad (5)$$

где x – элемент выходной матрицы скрытого слоя $\mathbf{H}$ (4).

Как показано в работе [23], алгоритм ELM значительно превосходит метод опорных векторов (англ. support vector machine, SVM) и метод случайного леса (англ. random forest, RF) по скорости обучения и выполнения предсказаний в задаче многоклассовой классификации векторных представлений текстов программ, также ELM превосходит метод k ближайших соседей (англ. k-nearest neighbor, KNN) по скорости выполнения предсказаний, не уступая в точности. В аналитической подсистеме ЦАП (см. рис. 1) важно быстро предоставить студентам обратную связь о выявленном способе решения [9].

После обучения классификатора на основе ELM веса $\boldsymbol{\alpha}$ и сдвиги скрытого слоя $\vec{b}$, а также вычисленные по формуле (2) веса $\boldsymbol{\beta}$ сохраняются на диск, для выполнения предсказаний в аналитической подсистеме ЦАП используется Алгоритм 2 (см. рис. 1).

Значения параметров d и γ выбираются для каждой задачи поиском по сетке и также сохраняются на диск, в качестве функции активации используется сигмоида (5). Строки матрицы $\mathbf{X}_T$, подаваемой на вход Алгоритму 2, являются векторными представлениями текстов программ (см. Алгоритм 1), для которых необходимо определить метку класса – номер подхода к решению.

V. Статистика внедрения

Система ЦАП (см. рис. 1) использовалась для автоматизации создания и проверки домашних заданий курса Программирования на языке Python в РТУ МИРЭА в весеннем семестре 2022 года. В течение семестра студентами курса было решено более 14000 уникальных автоматически сгенерированных задач по программированию. Всего системой ЦАП было получено более 60000 текстов программ, из которых более 50000 неправильных решений задач было автоматически отклонено, студентам было показано сообщение об ошибке и было предложено загрузить исправленную версию (см. рис. 5а). Сравнение числа принятых и отклонённых программ приведено на рис. 9.

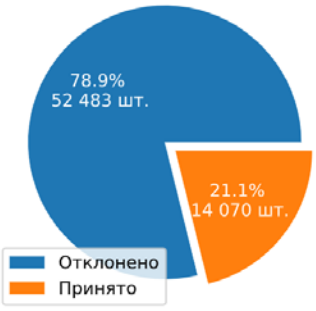


Рисунок 9. Сравнение числа принятых и отклонённых ЦАП программ в весеннем семестре 2022 года в РТУ МИРЭА

Решение, присланное студентом в ЦАП, может работать неправильно из-за синтаксических ошибок, из-за отсутствия функции main, содержащей решение задачи, из-за возникшего в процессе проверки исключения или из-за неправильного ответа, возвращаемого функцией, решающей задачу. На рис. 10 приведены гистограммы, показывающие, сколько текстов программ было отклонено в зависимости от типа задачи, с детализацией по причинам отклонения.

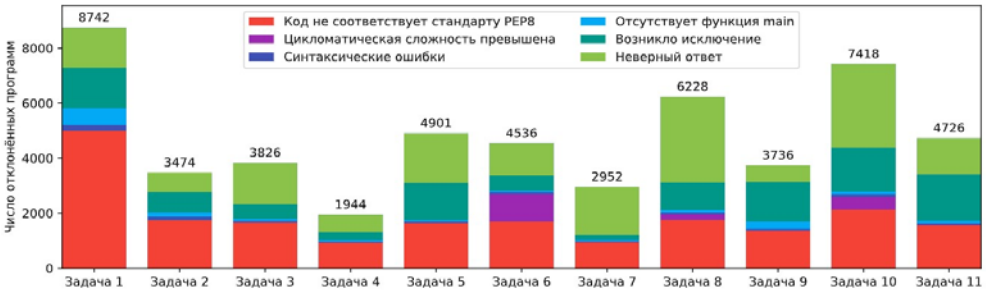


Рисунок 10. Причины отклонений присланных студентами решений в зависимости от типа задачи.

Согласно статистике, представленной на рисунке 10, у первой задачи, считавшейся наиболее простой из предложенных (см. таблицу 1), среднее число попыток решения является наибольшим. Это связано с обучением студентов работе с ЦАП, а также с ознакомлением с требованиями к отправляемым решениям методом проб и ошибок.

Из рисунка 10 заметно, что задачи на преобразования табличных данных (номер 8 в таблице 1) и на разбор текстового формата (номер 10 в таблице 1) также вызывают серьёзные трудности у студентов. Самыми простыми задачами оказались задача на реализацию рекурентной функции (номер 4 в таблице 1) и задача на преобразование битовых полей (номер 7 в таблице 1).

Уникальные задачи каждого типа из представленных в таблице 1 могут быть решены различающимися способами, студенты не ограничены в выборе подходов к решению задач. В процессе эксплуатации аналитической подсистемы ЦАП, выполняющей учёт учебных достижений студентов, было установлено, что различающиеся подходы к решению имеют разную популярность. Решения задач 6, 9, 11 наиболее вариативны.

Присланные студентами тексты программ, реализующие вычисления деревьев решений (номер 6 в таблице 1) содержали как решения при помощи условных операторов (самый популярный способ), так и решения при помощи словарей, сопоставления с образцом, циклов. Тексты программ, решающих задачи на разбор двоичного формата данных (номер 11 в таблице 1), содержали как решения при помощи классов с внутренним состоянием, так и решения при помощи функций (самое популярное решение), словарей, декораторов, лямбда-выражений.

Некоторые примеры решений задачи на реализацию конечного автомата Мили в виде класса (номер 3 в таблице 1) приведены в таблице 2: это решения при помощи условных операторов, словарей, сопоставления с образцом. Решения при помощи списков и кортежей опущены для краткости.

VI. Заключение

В данной работе рассмотрена архитектура системы ЦАП, основными особенностями которой являются автоматическая генерация уникальных задач по программированию 11 различных типов, основанная на алгоритмах, предложенных в [1], а также интегрированные методы интеллектуального анализа данных и машинного обучения для автоматизации выявления подходов к решению задач в текстах программ, присылаемых на проверку.

При получении от студента текста программы, решающего уникальную задачу, выполняется статический

Таблица 2. Примеры подходов к реализации конечного автомата (задача 9 в таблице 1).

```
class Milli:
    def __init__(self):
        self.sost = 'A'

    def sway(elf):
        if self.sost == 'A':
            self.sost = 'B'
            return 0
        if self.sost == 'C':
            self.sost = 'D'
            return 2
        if self.sost == 'D':
            self.sost = 'E'
            return 5
        raise KeyError
    ...
```

```
class StateMachine:
    state = State.A

    def update(self, transitions):
        self.state, out = transitions[self.state]
        return out

    def send(self):
        return self.update({
            State.A: [State.B, 0],
            State.B: [State.C, 3],
            State.C: [State.D, 4],
            State.D: [State.B, 6],
        })
    …
```

```
class FSM:
    state = "A"

    def join(self):
        match self.state:
            case "A":
                self.state = "B"
                return 0
            case "B":
                self.state = "C"
                return 2
            case _:
                raise KeyError
    …
```

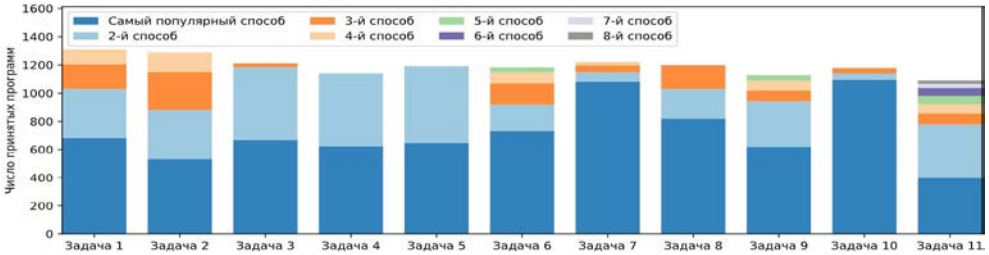


Рисунок 11. Популярность способов решения задач в зависимости от типа задачи

анализ кода, включающий проверку форматирования, оценку цикломатической сложности кода. Затем выполняется тестирование кода на соответствующем задаче наборе тестов [2]. В случае успешного прохождения тестов производится преобразование текста программы в вектор на основе цепи Маркова для соответствующего программе дерева абстрактного синтаксиса, полученный вектор подаётся на вход классификатору на основе машины экстремального обучения, определяющего подход к решению задачи [9]. Сведения о подходах решения, которыми была решена задача, сохраняются в базу данных системы ЦАП, за каждый открытый способ решения студенту начинаются дополнительные баллы для повышения мотивации к обучению.

В процессе эксплуатации системой ЦАП было обработано более 60000 текстов программ, из которых более 10000 решений было принято. Для каждой задачи было выявлено несколько способов решения. Полученная статистика использовалась преподавателями для оценки пробелов в навыках и знаниях студентов разных групп, а также для мотивации студентов изучать различные способы решения одной и той же задачи.

В новой версии ЦАП для усложнения задачи на реализацию рекуррентной функции (номер 4 в таблице 1) планируется сделать обязательным разворачивание рекурсии в цикл, а также повысить вариативность задачи на преобразования битовых полей (номер 7 в таблице 1), оказавшимися самыми простыми для студентов.

С целью обучения классификаторов на присылаемых студентами программах в систему ЦАП в течение семестра, а не после его окончания, планируется применить и интегрировать в аналитическую подсистему алгоритм онлайн машины экстремального обучения (англ. Online sequential Extreme learning machine, OS-ELM) [24]. Классификация текстов программ позволяет автоматически обнаруживать редкие, уникальные решения. Результаты классификации могут быть в дальнейшем использованы для начисления студентам дополнительных баллов [9].

Библиография

[1] E.G. Andrianova, L.A. Demidova, P.N. Sovetov, “Pedagogical design of a digital teaching assistant in massive professional training for the digital economy,” In *Russian Technological Journal*, 2022, vol. 10, No. 3, pp. 7–23.

[2] P.N. Sovietov, A.V. Gorchakov, “Digital Teaching Assistant for the Python Programming Course,” In *Proceedings of the 2022 2nd International Conference on Technology Enhanced Learning in Higher Education (TELE)*, IEEE, 2022, pp. 272–276.

[3] М.Л. Симуни, А.Ю. Соловьев, В.И. Шайтан, “Автоматизированная проверка задач в курсе «Функциональное программирование»,” *Современные информационные технологии и ИТ-образование*, 2016, Т. 12, № 3-1, с. 90–96.

[4] И.А. Жуков, Ю.Л. Костюк, “Предметно-ориентированный язык для генерации заданий из исходных текстов программ,” *Вестник Новосибирского государственного университета. Серия: Информационные технологии*, 2022, Т. 20, № 1, с. 18–27.

[5] L. Moussiades, A. Vakali, “PDetect: A clustering approach for detecting plagiarism in source code datasets,” In *The computer journal*, 2005, vol. 48, no. 6, pp. 651–661.

[6] L. Jiang, G. Misherghi, Z. Su, S. Glondu, “Deckard: Scalable and accurate tree-based detection of code clones,” In *Proceedings of the 29-th International Conference on Software Engineering (ICSE'07)*, IEEE, 2007, pp. 96–105.

[7] П.А. Гилев, В.К. Казанков, А.В. Табиева, “Автоматическая генерация и проверка задач по дисциплинам математического цикла в высшей школе,” *Современное педагогическое образование*, 2022, № 11, с. 142–147.

[8] Л.А. Демидова, П.Н. Советов, А.В. Горчаков, “Кластеризация представлений текстов программ на основе цепей Маркова,” *Вестник Рязанского государственного радиотехнического университета*, 2022, № 81, с. 51–64.

[9] А.В. Горчаков, Л.А. Демидова, П.Н. Советов, “Автоматизированный анализ текстов программ при помощи представлений на основе цепей Маркова и машин экстремального обучения,” *Вестник Рязанского государственного радиотехнического университета*, 2022, № 82, с. 162–163.

[10] L. Hakulinen, T. Auvinen, A. Korhonen, “The Effect of Achievement Badges on Students' Behavior: An Empirical Study in a University-Level Computer Science Course,” In *International Journal of Emerging Technologies in Learning*, 2015, vol. 10, № 1, pp. 18–29.

[11] C.H. Su, C.H. Cheng, “A mobile gamification learning system for improving the learning motivation and achievements,” In *Journal of Computer Assisted Learning*, 2015, vol. 31, № 3, pp. 268–286.

[12] T. Mailund, *Introducing Markdown and Pandoc: Using Markup Language and Document Converter* Berkeley: Apress, 2019, 139 p.

[13] E. Gansner, E. Koutsofios, S. North, “Drawing graphs with dot,”, 2015, URL: https://www.graphviz.org/pdf/dotguide.pdf (accessed on 01.28.2023).

[14] M. Peveler, E. Maicus, B. Cutler, “Comparing jailed sandboxes vs containers within an autograding system,” In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 139–145.

[15] X. Wang, J. Du, H. Liu, “Performance and isolation analysis of RunC, gVisor and Kata Containers runtimes,” In *Cluster Computing*, 2022, Vol. 25, № 2, pp. 1497–1513.

[16] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford, *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional, 2002, ch. 14.

[17] M. Bayer, A. Brown, G. Wilson, “SQLAlchemy,” In *The architecture of open-source applications*, 2012, vol. 2, p. 20.

[18] S. Lima, J. Correia, F. Araujo, J. Cardoso, “Improving observability in event sourcing systems,” In *Journal of Systems and Software*, 2021, vol. 181, p. 111015.

[19] A. Marcus, J.I. Maletic, “Identification of high-level concept clones in source code,” In *Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, IEEE, 2001, pp. 107–114.

[20] M. Shutaywi, N.N. Kachouie, “Silhouette analysis for performance evaluation in machine learning with applications to clustering,” In *Entropy*, 2021, vol. 23, no. 6, p. 759.

[21] I. Dagan, L. Lee, F. Pereira, “Similarity-based methods for word sense disambiguation,” In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, 1997.

[22] G.B. Huang, Q.Y. Zhu, C.K. Siew, “Extreme Learning machine: Theory and applications,” In *Neurocomputing*, 2006, vol. 70, pp. 489–501.

[23] L.A. Demidova, A.V. Gorchakov, “Classification of Program Texts Represented as Markov Chains with Biology-Inspired Algorithms-Enhanced Extreme Learning Machines,” In *Algorithms*, 2022, vol. 15, № 9, p. 329.

[24] W. Guo, T. Xu, K. Tang, J. Yu, S. Chen, “Online sequential extreme learning machine with generalized regularization and adaptive forgetting factor for time-varying system prediction,” In *Mathematical Problems in Engineering*, 2018, vol. 2018, p. 6195387.

Горчаков А.В., аспирант, ассистент кафедры корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: worldbeater-dev@yandex.ru. Scopus Author ID 57215001290. https://orcid.org/0000-0003-1977-8165

Демидова Л.А., 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: liliya.demidova@rambler.ru. Scopus Author ID 56406258800. https://orcid.org/0000-0003-4516-3746

Советов П.Н., к.т.н., доцент, 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. https://orcid.org/0000-0002-1039-2429

Intelligent Accounting of Educational Achievements in the "Digital Teaching Assistant" System

Artyom V. Gorchakov, Liliya A. Demidova, Peter N. Sovietov

***Abstract*—The ongoing process of digitalization of the economy necessitates the mass training of software developers. With an increase in the number of students, the load on teachers of university programming courses is growing. Automation of programming courses makes it possible to relieve teachers from routine activities. This article discusses the Digital Teaching Assistant (DTA) system that automates the Python programming course at RTU MIREA. The article considers the architecture of the DTA system, the main features of which are the automatic generation of unique programming tasks of eleven different types, including the implementation of author's algorithms, integrated methods of data mining and machine learning to automate the identification of approaches to solving problems in texts of programs sent for verification. The DTA system consists of a core that generates and checks unique programming exercises, an intelligent module for educational achievements accounting designed to motivate students to solve problems using different high-level concepts and ideas, and a web application. Algorithms are presented that automate the determination of methods for solving unique programming exercises in order to identify and record achievements. The statistics collected during the operation of the DTA system are presented.**

***Keywords*— online education, programming task generation, information system, program text analysis, program code analysis.**

REFERENCES

[1] E.G. Andrianova, L.A. Demidova, P.N. Sovetov, "Pedagogical design of a digital teaching assistant in massive professional training for the digital economy," In *Russian Technological Journal*, 2022, vol. 10, No. 3, pp. 7–23.

[2] P.N. Sovietov, A.V. Gorchakov, "Digital Teaching Assistant for the Python Programming Course," In *Proc. 2022 2nd Inter. Conf. on Technology Enhanced Learning in Higher Education (TELE)*, IEEE, 2022, pp. 272–276.

[3] M.L. Simuni, A.Y. Soloviov, V.I. Shaitan, "Automated problem checking for Functional programming course," In *Modern Information Technologies and IT-Education*, 2016, Vol. 12, No. 3-1, pp. 90–96.

[4] I.A. Zhukov, Y.L. Kostyuk, "A Domain-Specific Language for generating tasks from Programs Source Code," In *Vestnik NSU. Series: Information Technologies*, 2022, Vol. 20, No. 1, pp. 18–27.

[5] L. Moussiades, A. Vakali, "PDetect: A clustering approach for detecting plagiarism in source code datasets," In *The computer journal*, 2005, vol. 48, no. 6, pp. 651–661.

[6] L. Jiang, G. Misherghi, Z. Su, S. Glondu, "Deckard: Scalable and accurate tree-based detection of code clones," In *Proc. of the 29-th Inter. Conf. on Software Engineering (ICSE'07)*, 2007, pp. 96–105.

[7] P.A. Gilev, V.K. Kazankov, A.V. Tabieva, "Automatic Generation and Verification of Tasks in the Disciplines of the Mathematical Cycle in Higher Education," In *Modern Pedagogical Education*, 2022, No. 11, pp. 142–147.

[8] L.A. Demidova, P.N. Sovietov, A.V. Gorchakov, "Clustering of Program Source Text Representations Based on Markov Chains," In *Vestnik of RSREU*, 2022, No. 81, pp. 51–64.

[9] A.V. Gorchakov, L.A. Demidova, P.N. Sovietov, "Automated Program Text Analysis Using Representations Based on Markov Chains and Extreme Learning Machines," In *Vestnik of RSREU*, 2022, No. 82, pp. 162–163.

[10] L. Hakulinen, T. Auvinen, A. Korhonen, "The Effect of Achievement Badges on Students' Behavior: An Empirical Study in a University-Level Computer Science Course," In *International Journal of Emerging Technologies in Learning*, 2015, vol. 10, № 1, pp. 18–29.

[11] C.H. Su, C.H. Cheng, "A mobile gamification learning system for improving the learning motivation and achievements," In *Journal of Computer Assisted Learning*, 2015, vol. 31, № 3, pp. 268–286.

[12] T. Mailund, *Introducing Markdown and Pandoc: Using Markup Language and Document Converter* Berkeley: Apress, 2019, 139 p.

[13] E. Gansner, E. Koutsofios, S. North, "Drawing graphs with dot,", 2015, URL: https://www.graphviz.org/pdf/dotguide.pdf (accessed on 01.28.2023).

[14] M. Peveler, E. Maicus, B. Cutler, "Comparing jailed sandboxes vs containers within an autograding system," In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 139–145.

[15] X. Wang, J. Du, H. Liu, "Performance and isolation analysis of RunC, gVisor and Kata Containers runtimes," In *Cluster Computing*, 2022, Vol. 25, № 2, pp. 1497–1513.

[16] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford, *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional, 2002, ch. 14.

[17] M. Bayer, A. Brown, G. Wilson, "SQLAlchemy," In *The architecture of open-source applications*, 2012, vol. 2, p. 20.

[18] S. Lima, J. Correia, F. Araujo, J. Cardoso, "Improving observability in event sourcing systems," In *Journal of Systems and Software*, 2021, vol. 181, p. 111015.

[19] A. Marcus, J.I. Maletic, "Identification of high-level concept clones in source code," In *Proc. of the 16th Annual International Conf. on Automated Software Engineering (ASE 2001)*, IEEE, 2001, pp. 107–114.

[20] M. Shutaywi, N.N. Kachouie, "Silhouette analysis for performance evaluation in machine learning with applications to clustering," In *Entropy*, 2021, vol. 23, no. 6, p. 759.

[21] I. Dagan, L. Lee, F. Pereira, "Similarity-based methods for word sense disambiguation," In *Proc. of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conf. of the Europ. Chap. of the Association for Computational Linguistics*, 1997.

[22] G.B. Huang, Q.Y. Zhu, C.K. Siew, "Extreme Learning machine: Theory and applications," In *Neurocomputing*, 2006, vol. 70, pp. 489–501.

[23] L.A. Demidova, A.V. Gorchakov, "Classification of Program Texts Represented as Markov Chains with Biology-Inspired Algorithms-Enhanced Extreme Learning Machines," In *Algorithms*, 2022, vol. 15, № 9, p. 329.

[24] W. Guo, T. Xu, K. Tang, J. Yu, S. Chen, "Online sequential extreme learning machine with generalized regularization and adaptive forgetting factor for time-varying system prediction," In *Mathematical Problems in Engineering*, 2018, vol. 2018, p. 6195387.

Artyom V. Gorchakov, Postgraduate Student, Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadsky pr., Moscow, 119454). worldbeater-dev@yandex.ru. https://orcid.org/0000-0003-1977-8165

Liliya A. Demidova, Dr. Sci. (Eng.), Professor, Professor of the Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadsky pr., Moscow, 119454). liliya.demidova@rambler.ru. https://orcid.org/0000-0003-4516-3746

Peter N. Sovietov, Cand. Sci. (Eng.), Associate Professor, Department of Corporate Information Systems, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadsky pr., Moscow, 119454). sovetov@mirea.ru https://orcid.org/0000-0002-1039-2429