УДК 004.89 doi: 10.15827/0236-235X.142.201-212 2024. Т. 37. № 2. С. 201–212

# Автоматическая генерация заданий по результатам анализа использования их банка в интеллектуальной обучающей системе

**О.А. Сычев [1]✉, А.А. Прокудин [1], М.Е. Денисов [1]**

[1] Волгоградский государственный технический университет, г. Волгоград, 400005, Россия

**Ссылка для цитирования**
Сычев О.А., Прокудин А.А., Денисов М.Е. Автоматическая генерация заданий по результатам анализа использования их банка в интеллектуальной обучающей системе // Программные продукты и системы. 2024. Т. 37. № 2. С. 201–212. doi: 10.15827/0236-235X.142.201-212
**Информация о статье**
Группа специальностей ВАК: 2.3.1
Поступила в редакцию: 18.10.2023 После доработки: 20.12.2023 Принята к публикации: 22.12.2023

**Аннотация.** Развитие методов генерации учебных заданий и их классификации для применения в учебном процессе позволило реализовать интеллектуальную обучающую систему, способную при необходимости автоматически генерировать задания в фоновом режиме. В статье описана архитектура интеллектуальной обучающей системы с фоновой генерацией заданий, рассмотрена система CompPrehension, реализующая этот принцип. Для решения проблемы использовались методы символьного искусственного интеллекта и онтологических рассуждений, объектно-ориентированного проектирования и программирования, а также системного анализа. В работе представлены основные функциональные элементы таких компонентов системы, как тренажер, банк заданий, генератор заданий и домен (предметная область), а также их взаимодействие с БД для хранения заданий и с машинами вывода для их решения. Апробация подхода на банке заданий по определению порядка выполнения операторов в выражении показала способность системы избирательно пополнять несбалансированный банк, исходя из его состава и истории запросов на поиск заданий в нем. Менее чем за пять суток было сгенерировано несколько тысяч заданий на каждый концепт с недостаточным числом заданий. Это дает низкую вероятность получения одинаковых заданий даже при одновременном тестировании сотен студентов. Результаты исследования имеют практическую значимость для учебных заведений, стремящихся совершенствовать обучающий процесс.

**Ключевые слова:** интеллектуальные обучающие системы, генерация вопросов, дистанционное обучение, разработка обучающих систем, CompPrehension, банк заданий

**Введение**. В процесс преподавания входят два фундаментальных цикла обратной связи – цикл обучаемого (малый) и цикл преподавателя (большой). Малый цикл включает попытку обучаемого применить изученный материал (ответ на тестовый вопрос, решение задачи), оценку этого решения преподавателем, выдачу обратной связи о качестве решения и причинах или характере ошибок (при наличии) и подбор следующего задания для наиболее эффективного обучения. Предназначение цикла – дать возможность научиться конкретному обучаемому. Большой цикл осуществляется преподавателем и подразумевает выдачу и контроль заданий (тестов), оценку эффективности их решения, оценку качества банка заданий (вопросов) и выявление недостатков, корректировку банка заданий. Он предназначен для совершенствования учебного процесса и повышения его качества. Однако многие задачи как в малом, так и в большом цикле являются рутинными и трудоемкими, что ограничивает возможности преподавателей работать с конкретными обучаемыми и совершенствовать свои курсы.

Автоматизированные обучающие системы призваны улучшить образовательный процесс за счет автоматизации части задач – от предоставления информации об изучаемой теме до проверки решения задач и объяснения причин ошибок [1]. При разработке систем адаптивного тестирования и *интеллектуальных обучающих систем* (ИОС) достигнут большой прогресс в автоматизации цикла обратной связи обучаемого, включая выдачу подсказок, объяснение сделанных ошибок и адаптивный подбор следующей задачи на основе решения обучаемым предыдущих.

Гораздо меньше усилий вложено в автоматизацию цикла обратной связи преподавателя, то есть в анализ и пополнение банка заданий (вопросов), хотя качество заданий является одним из ключевых факторов успешности ИОС. Небольшое количество заданий в банке приводит к тому, что многие обучаемые будут получать одинаковые задания. Это провоцирует обмен правильными ответами и списывание, так что, банки заданий, составленные вручную, плохо масштабируются при росте аудитории. Кроме того, адаптивные обучающие системы должны иметь широкий выбор возможных задач различного уровня сложности на все изучаемые понятия и законы предметной области,

чтобы раскрыть потенциал адаптивных алгоритмов.

Автоматизация замыкания большого цикла обратной связи в процессе преподавания включает автоматическую генерацию учебных заданий, анализ банка заданий и его пополнение. Однако большинство работ по автоматической генерации заданий являются экспериментальными; сгенерированные задания, как правило, невозможно использовать в учебном процессе без редактирования или отбора преподавателем [2]. К тому же в процессе генерации редко учитываются результаты анализа банка заданий.

Новые исследования в области автоматической генерации и классификации учебных заданий открывают возможности их прямого использования без отбора и проверки человеком [3], а также разработки архитектур ИОС. Цель авторов настоящей статьи – разработка и апробация архитектуры ИОС, способной по мере автоматизированного адаптивного обучения (с реализацией малого цикла обратной связи) автоматически замкнуть большой цикл обратной связи: анализировать недостатки существующего банка заданий в ходе его реального применения, создавать запросы на генерацию дополнительных заданий, автоматически генерировать учебные задания и выбирать подходящие для включения в банк и использования в дальнейших учебных сессиях.

## Обзор автоматизированных обучающих систем для обучения программной инженерии

Система обучения Problets [4] предоставляет несколько типов заданий на понимание принципов процедурного программирования для трех языков. Создание вопросов происходит на основе шаблонов с рандомизацией, а обратная связь представляет собой контекстные объяснения для любого шага решения. Объяснение ошибок осуществляется через комментарии о том, как должно быть. В распространяемом комплекте Problets содержится, по-видимому, фиксированный набор шаблонов для задач, который не подлежит модификации во время эксплуатации. Пополнение возможно через шаблоны в нотации BNF, вручную задаваемые преподавателем.

В работе [5] представлена ASSISTment Builder – среда создания тренажеров с линейными примерами, нацеленная на сокращение времени формирования навыков и создания вопросов. Они описывают функцию «вариабилизации», позволяющую создавать шаблоны с переменными, и отмечают неспособность «обобщать схожие проблемы», что приводит к большому количеству работы по принципу «скопируй и немного измени». Поскольку похожие друг на друга учебные задачи полезны для практики начинающим, использование одного и того же обучающего контента для создания группы схожих задач облегчило бы их составление.

Система ASSISTment предлагает обучающемуся два типа направляющей обратной связи: серию текстовых подсказок о том, как решить задачу, и серию вспомогательных вопросов, отображаемых в случае неправильного ответа на «главную задачу». Каждый вспомогательный вопрос имеет несколько подсказок, и каждая задача (вопрос) может отображать заранее подготовленное сообщение как реакцию на известный неправильный ответ. Система ASSISTment предоставляет три варианта организации последовательности учебных задач: фиксированный, случайный и ограниченный заранее определенным набором задач (данные об успеваемости учащегося не используются). Предложенная авторами архитектура предполагает полный контроль над контентом со стороны преподавателя. Банк заданий обновляется вручную. Для помощи создателям контента добавлены возможность комментирования вопросов учащимися и простая статистика наиболее частых неправильных ответов.

Автоматически генерировать задачи на определение результатов выполнения кода, случайным образом изменяя константы и операторы в синтаксическом дереве фрагмента кода на языке Java, позволяет Unlimited Trace Tutor [1]. Для генерации используется фиксированный набор простых шаблонов, содержащих циклы for, while и развилки if (пополнение банка заданий не предполагается). Для получения более сложных фрагментов кода шаблоны могут комбинироваться. При запуске тренировки учащийся может выбрать алгоритмические структуры, которые он хочет изучать, а также уровень сложности. Продолжительность тренировки может быть намеренно ограничена по времени.

Система Automata Tutor v3, описанная в источнике [2], поддерживает автоматическую генерацию для пяти типов задач и нескольких уровней сложности. Для задач на основе графов, таких как регулярные выражения, уровень сложности влияет на структуру генерируемого графа. Для задач на основе кода некоторые слу-

чайные преобразования применяются к созданным вручную базовым программам, разработанным для разных уровней сложности, а затем задачи низкого качества отбрасываются. Составление базовых программ, видимо, является ответственностью разработчика новых видов задач.

Визуальная среда моделирования программ UUhistle [6] предоставляет богатые функциональные возможности для отслеживания каждого мелкого события, происходящего во время выполнения кода. Она может как демонстрировать действия, так и просить учащегося выполнить их вручную, предоставляя текстовые объяснения, когда учащийся ошибается. Система UUhistle содержит фиксированный набор встроенных примеров кода, а также позволяет использовать произвольный код на языке Python в качестве трассируемой программы. Система не осуществляет генерацию и пополнение банка заданий.

Почти все приведенные аналоги предоставляют преподавателю сводку по оценкам учащихся, и ни один из них не предполагает возможности использовать информацию о составе банка заданий и статистику его применения при генерации новых заданий.

**Генерация вопросов и учебных заданий**

Автоматическая генерация может создать десятки и сотни тысяч вопросов за относительно короткое время. Для сравнения: банк вопросов ASSISTment содержит всего около 3 000 вопросов, созданных вручную более чем 30 преподавателями, а наименьшее время, затраченное на создание учебного материала, составляет около 40 часов на каждый час обучения [5].

Автоматизация управления учебными заданиями затрагивает два аспекта – создание вопросов и оценка ответов. Наиболее популярный [7] и надежный способ генерации вопросов состоит в использовании шаблона почти готовой задачи с местами для вставки (плейсхолдерами) [8]. Их перечень фиксирован [5] или выбирается случайным образом из заданного диапазона (для числовых величин) [9]. Одни подстановочные параметры могут вычисляться из других в момент формирования вопроса. Этот подход позволяет кратно размножить задания, созданные человеком, но ограничен набором ручных шаблонов, создание которых все еще требует больших затрат. Существуют вариации шаблонного метода, например, в работе [10] предлагается автоматически извлекать компоненты задания из подготовленного вручную исходного кода.

Другой подход использует перечень правил или дерево решений со случайными переходами для генерации шаблона с последующим заполнением случайными величинами, например, синтез [11] или модификация [1] программ на основе грамматики, модели NLP-трансформеров [12]. Этот подход позволяет генерировать сотни тысяч заданий с приемлемой вариативностью, но преподаватель имеет меньше контроля над структурой и содержанием генерируемых вопросов, а правильная настройка генератора может оказаться сложной задачей. Недостатком этого подхода является то, что для генерации заданий на другую тему или даже другого типа в рамках одной темы требуется разработка новой конфигурации для генератора или создание совершенно нового генератора.

В сложных и/или менее детерминированных предметных областях популярен подход с использованием специальных датасетов [13], или больших баз знаний, онтологий. По теме программирования тоже существуют онтологии, например, CodeOntology для языка Java [14], но они могут отсутствовать для целевого языка программирования или быть недостаточно информативными. Так, в готовом датасете CodeOntology удалена информация о действиях внутри методов, которая была бы полезна в настоящем исследовании. Этот подход предоставляет так же мало контроля над фактическим содержимым вопросов, но, с другой стороны, базируется на разнообразных реальных данных и процесс генерации более прямолинейный и масштабируемый, чем в предыдущем подходе.

Подход с использованием открытых данных – текстов [15] или открытого кода [16], который используется в данной работе, применим ко многим видам учебных вопросов и имеет высокий потенциал благодаря большому объему и разнообразию доступных через Интернет открытых данных, которые могут стать хорошим источником для шаблонов вопросов. Трудности могут возникать из-за разрозненности источников данных, отсутствия в них подходящих метаданных для быстрого поиска и сложностей предобработки и фильтрации сырых данных, к тому же возможен риск потенциально недопустимого контента. С точки зрения наполнения банка вопросов вероятен дисбаланс в разрезе различных существенных харак-

теристик вопросов из-за объективного дисбаланса в реальных данных. Например, попытка найти много разных примеров программного кода с использованием вложенных тернарных операторов может не увенчаться успехом, так как такое редко встречается в профессиональном коде из-за снижения читаемости. Другой пример: оператор walrus в Python является допустимым вариантом оператора присваивания и даже рекомендуется к использованию, но в профессиональном коде редко используется из-за непопулярности. Поэтому также необходимо планировать выравнивание численности непредставленных групп примеров.

Автоматически сгенерированные задачи должны быть автоматически проверяемыми [17], что помогает масштабировать все этапы обучения. «Длинные» вопросы с открытым ответом (например, написание программ и эссе), которые включают элемент творчества, проверить полностью автоматически невозможно, так как перечень критериев для определения правильности ответа слишком широк. Хотя есть попытки применить к таким ответам машинное обучение, качество проверок далеко от 100 %, и потому они используются, как правило, только для помощи оценщику-человеку [18].

Новизна данной работы заключается в практической разработке автоматизированной обучающей системы с возможностью автоматической (без участия человека) генерации заданий для пополнения банка знаний по мере необходимости. Разработанные ранее системы позволяли пополнять банк заданий вручную или автоматизированно, но при участии человека.

Практическая значимость исследования заключается в существенном уменьшении трудозатрат преподавателей на поддержку банков заданий в автоматизированных обучающих системах, включая анализ повторений заданий и разработку новых.

## Метод построения ИОС на основе автоматической генерации заданий

Для эффективной работы ИОС должна обеспечивать учащихся заданиями для усвоения учебных тем, стремясь снизить до минимума частоту их повторения в рамках выполняемого упражнения, тем самым снижая вероятность списывания и запоминания уже решенного задания. Система, позволяющая автоматически замкнуть оба цикла обратной связи учебного процесса, должна решать в автоматическом режиме следующие задачи:

– подбор задания для решения на основе имеющихся данных о знаниях и умениях обучаемого;

– оценка решения задания;

– определение ошибок обучаемого и их возможных причин с генерацией объясняющих сообщений;

– определение классов заданий, которых недостаточно для эффективной работы;

– генерация учебных заданий, их классификация и отбор по заданным критериям;

– решение сгенерированных задач для определения множества правильных ответов.

Полезными, но необязательными являются генерация подсказок в процессе решения задания, демонстрация примеров решения заданий, ведение обучающего диалога с помощью наводящих вопросов.

При ручном планировании банка заданий в адаптивных обучающих системах очень сложно предсказать потенциальную потребность в заданиях различных категорий, поскольку это требует знания того, какие ошибки и трудности обучаемые будут испытывать чаще и задания какой сложности наиболее эффективны для обучения. Поэтому в процессе эксплуатации адаптивной системы возникает дефицит определенных видов задач. Это приводит к их частому повторению или подбору менее подходящих задач, что снижает качество обучения [19]. Так, при эксплуатации прототипа ИОС CompPrehension [20] с автоматически сгенерированным банком задач на тему «порядок вычисления выражений в алгоритмических языках высокого уровня» заметен значительный недостаток задач на выполнение операции внутри операций скобочного типа (вызовы функций, обращение к элементу массива) и на правую ассоциативность операций. При возникновении недостатка заданий должен создаваться запрос на генерацию новых заданий и активироваться механизм его выполнения. Ключевым фактором успеха является то, что ИОС обладает необходимой информацией о том, какие задания и насколько часто требуются, и способна самостоятельно пополнять банк заданий при достижении граничных условий, в роли которых могут выступать низкая точность выполнения запросов на поиск задач (нет задачи с подходящими свойствами, берем похожую) или высокая частота использования одной и той же задачи в одной группе студентов.

Запрос на генерацию заданий должен описывать ожидаемый уровень сложности, а также концепты и законы предметной области, необ-

ходимые для решения задачи. Он может также включать в себя перечень недопустимых концептов и законов в случае, если обучаемые изучили тему не полностью. Авторы предлагают абстракцию требований к заданию, которая определяет необходимые свойства задач с тем, чтобы во время проведения упражнения выполнить поиск в банке и выдать учащемуся задание, наиболее отвечающее указанным характеристикам. Это основной способ обеспечить генерацию уникальных упражнений из больших банков заданий, которые не являются обозримыми для человека-преподавателя.

Предлагаемая архитектура такой обучающей системы состоит из трех основных компонентов.

- Тренажер. Взаимодействует с обучаемыми и преподавателями и реализует выбор заданий упражнений, их проверку, выдачу объясняющей обратной связи, генерацию подсказок.
- Банк заданий. Хранит задачи и выбирает подходящую случайную задачу по запросу. Ведет статистику использования заданий по упражнениям и может генерировать запросы на добавление заданий определенного типа в случае их недостаточного количества.
- Генератор задач. Производит фоновую генерацию учебных заданий в соответствии с запросами. Построен на основе подхода, использующего открытый программный код для генерации заданий [21].

Возможны различные сценарии использования генератора задач в процессе работы ИОС, а также однократный запуск генератора задач перед началом эксплуатации тренажера для генерации полного банка заданий. Это позволяет при необходимости эксплуатировать тренажер отдельно от генератора, а также дает преподавателю большую уверенность в банке заданий. Такой сценарий поможет в простых предметных областях, где доля сложных заданий (с участием нескольких различных концептов или правил предметной области) невелика. При создании более комплексных ИОС при однократной генерации велик риск получить несбалансированный банк заданий, в котором некоторые темы будут обойдены вниманием.

Подход с обратной связью заключается в фоновой генерации заданий в зависимости от потребностей тренажера, в котором задания могут добавляться небольшими порциями, закрывая текущую потребность тренажера в задачах на плохо покрытые определенные темы (включая при необходимости сложные задания на комбинации концепций). На рисунке 1 показан процесс работы ИОС, основанной на генерации недостающих задач в фоновом режиме.

С практической точки зрения целесообразно применять оба подхода: для инициализации нового банка можно применить массовую генерацию без конкретных требований, а затем во время учебного процесса вести фоновую генерацию тех заданий, количество которых оказалось недостаточным для эффективного обучения.

Использующие предложенную фоновую генерацию заданий ИОС могут обеспечить следующие преимущества перед классическими:

– экономия времени на создании банка заданий («скажи системе, чему надо научить»);

– сокращение возможности обмена правильными ответами за счет большого количества непрерывно пополняющихся заданий;

– наполнение банка заданий на основе потребностей обучаемых (банк содержит преимущественно те задания, в которых есть потребность);

– улучшение точности подбора заданий под конкретные потребности обучаемых.

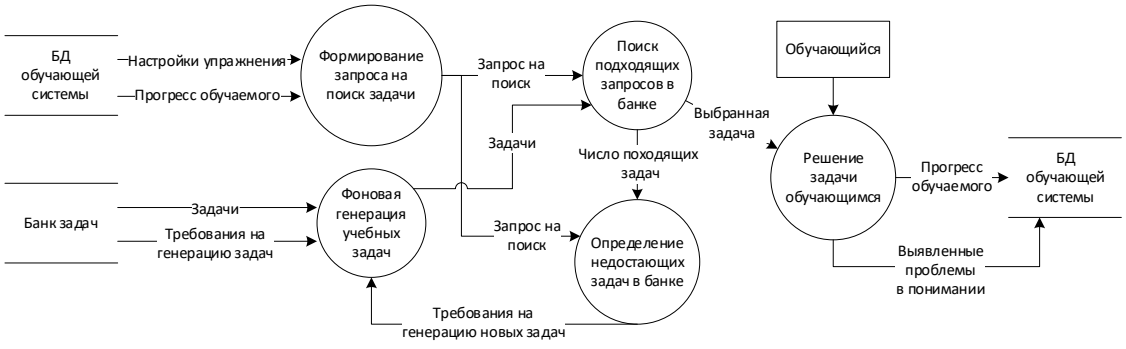


*Рис. 1. Схема использования автоматического подбора задач при выполнении упражнения в ИОС*

*Fig. 1. Scheme of using an automatic selection of tasks when performing exercises in ITS*

Этот подход не лишен недостатков. Например, можно выделить следующие:

– трудоемкость процесса написания генератора заданий для сложных предметных областей;

– отсутствие со стороны преподавателя контроля за автоматическим подбором заданий: недоверие преподавателя к действиям системы, которые он не контролирует, может тормозить ее внедрение;

– несоответствие стратегии подбора задания в ИОС стилю обучения преподавателя (в идеале ИОС должна поддерживать различные стратегии обучения).

На основе предложенного подхода доработана мультидоменная ИОС CompPrehension путем внедрения поддержки генерации заданий, в том числе на основании анализа потребностей системы. Архитектура разработанной системы показана на рисунке 2. Частичная диаграмма классов основных компонентов системы представлена на рисунке 3: ввиду большого количества классов в системе CompPrehension (208 классов) на диаграмме отражены только основные классы и методы компонентов, принимающие участие в процессах подбора вопросов, оценки ответов на вопросы и генерации новых заданий.

Центральный компонент системы, поддерживающий ее образовательные функции, – тренажер. Он организует выполнение всех ключевых процессов малого цикла обратной связи в процессе обучения. Тренажер использует четыре основных базовых компонента, для которых возможно подключение альтернативных реализаций в виде плагинов, – домен (предметная область), инкапсулирующий все, что связано с особенностями конкретной предметной области, педагогическая стратегия, определяющая порядок выдачи задания, виды обратной связи с обучаемым, а также способы реакции на успехи и неудачи обучаемого, бэкенд (используется для интеграции внешних машин вывода, рассуждающих при решении задачи) и фронтенд (способ взаимодействия с пользователем, как правило, обучаемым). Домен содержит также модули, необходимые для генерации заданий, три остальных базовых компонента используются только тренажером.

Кроме того, в домене, как в формальном представлении предметной области, содержится все необходимое для создания вопросов, оценки ответов на них и предоставления обратной связи (рис. 3, компонент «Абстрактный домен»). Принимая решение об оценке правильности ответа обучаемого, домен использует бэкенд для запуска машины вывода и формальных рассуждений на основе правил предметной области, которым обучает система. Домен генерирует обратную связь обучаемому, включающую объяснения ошибок и подсказки. В настоящее время в системе CompPrehension реализованы домены «Порядок вычисления выражения» и «Алгоритмические структуры»,

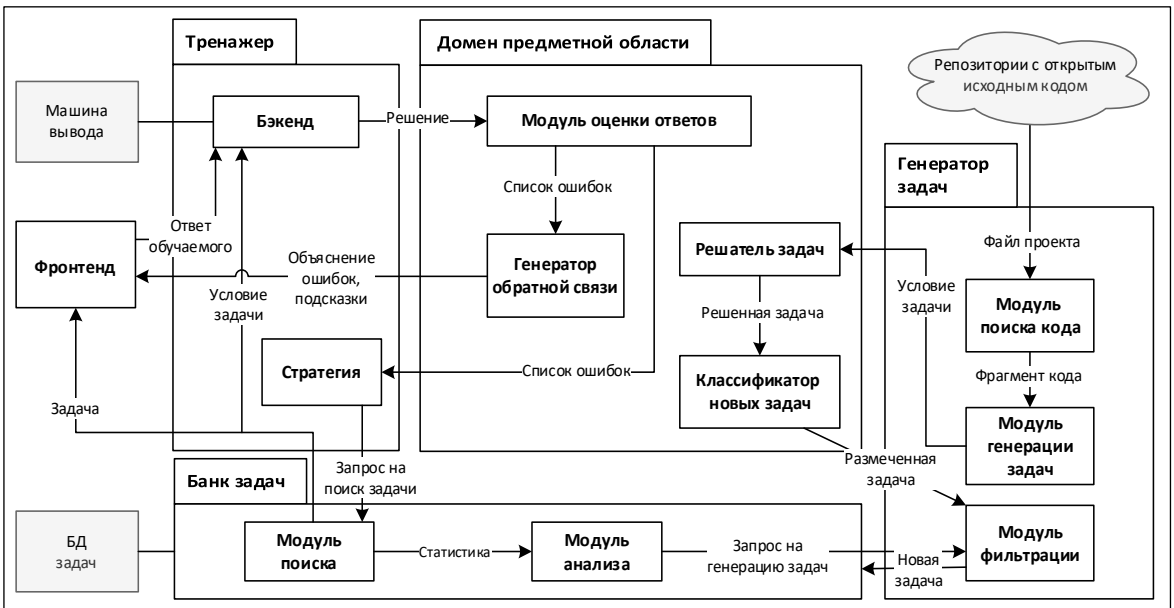


*Рис. 2. Основные компоненты обучающей системы CompPrehension с генератором заданий*

*Fig. 2. Main components of the CompPrehension tutoring system with a learning-problem generator*

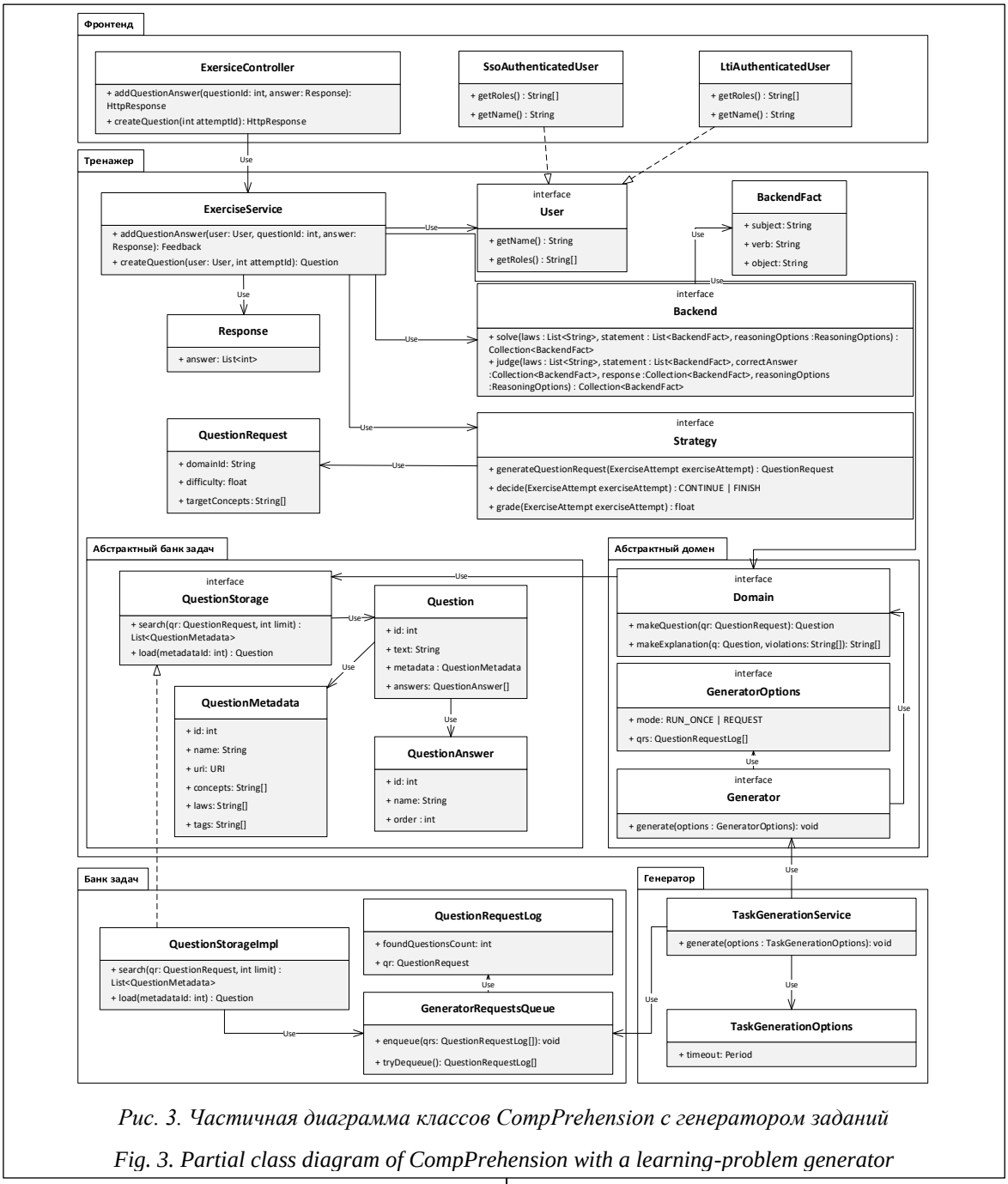


*Рис. 3. Частичная диаграмма классов CompPrehension с генератором заданий*

*Fig. 3. Partial class diagram of CompPrehension with a learning-problem generator*

идет разработка доменов «Области видимости», «Выражения доступа» и «Порядок прилагательных в английском языке».

Стратегия отделяет педагогический подход к обучению от особенностей предметной области. Она определяет требования к следующему заданию упражнения, учитывая как настройки этого упражнения, так и данные об успехах и ошибках обучаемого, а также решает, когда упражнение можно считать завершенным (рис. 3, интерфейс Strategy и класс QuestionRequest). Стратегия вычисляет оценку за ответ и упражнения на основе данных домена и определяет, какие виды обратной связи доступны обучаемому. Стратегии могут представлять фиксированное количество заданий (например, в целях оценки знаний на экзамене) или быть адаптивными к ответам обучаемого.

Интерфейс Backend (рис. 3) инкапсулирует взаимодействие с машинами вывода – логическими решателями, которые используют правила и факты, предоставляемые доменами. Такие интерфейсы используются для получения правильных ответов на задания домена и

определяют ошибки в ответах учащихся. Примерами подключаемых машин вывода могут служить Apache Jena, Prolog и др. Использование возможностей символьного искусственного интеллекта позволяет учить машину рассуждать так же, как и обучаемый, и находить объяснения ошибок, которые повышают доверие к решениям системы согласно концепции объяснимого искусственного интеллекта.

Публичный интерфейс тренажера – фронтенд – инкапсулирует в себе основные сценарии взаимодействия с ним пользователей. Он дает возможность организовать взаимодействие с тренажером как через собственный сайт, так и с помощью сторонних систем дистанционного обучения (поддерживается протокол LTI – Learning Tools Interoperability), а в перспективе позволит работать через мобильные приложения и чат-боты (рис. 2, компонент «Фронтенд»).

Банк задач позволяет хранить и наполнять базу учебных задач, а также выполнять поиск задач по определенным критериям. Концептуально банк задач состоит из модуля хранения и модуля анализа.

Модуль хранения содержит условия и метаданные для каждого учебного задания. Для удобства обновления банка данные каждого задания хранятся в одном файле, а его метаданные (учебные свойства и служебная информация) занесены в индексированную SQL-таблицу. Модуль позволяет выполнять быстрый поиск по большому количеству заданий средствами языка запросов SQL.

Модуль анализа проводит мониторинг статистики поисковых запросов и найденного для них количества подходящих вопросов. В случае нахождения запросов, число подходящих заданий для которых ниже порогового, а в результатах есть часто используемые задания, создается запрос на генерацию (рис. 3, классы GeneratorRequestQueue и QuestionRequestLog).

Генератор учебных заданий является отдельным компонентом и работает в фоновом режиме. Он предназначен для автоматизации большого цикла обратной связи обучающей системы и непосредственно взаимодействует с банком заданий и доменом. Реализованы два подхода к генерации – первоначальная генерация (по ручному триггеру) и автоматическая (по запросу на генерацию). Когда генератор получает запрос на генерацию, он передает его соответствующему домену. Домен анализирует запрос и самостоятельно принимает решение о начале генерации, процессом которой он управляет (рис. 2, компоненты «Генератор» и «Абстрактный домен»).

Для существующих доменов, посвященных изучению программирования, реализован процесс генерации, в котором в качестве источника данных для шаблонов задач используется открытый программный код, размещенный на портале github.com. Из загружаемых API-репозиториев отбираются файлы на целевом языке программирования, из которых извлекаются необходимые для генерации заданий фрагменты (например, отдельные выражения или функции). Фрагменты кода преобразуются в шаблоны заданий. Домен расширяет шаблон до готового задания или набора заданий, рассчитывает правильный ответ (множество ответов) для него и вычисляет учебные свойства задания, в частности, отделяет фактически используемые в процессе решения концепты от просто фигурирующих в условии.

Преподаватель, использующий такую систему, может создать упражнение очень быстро, если для него разработан домен: достаточно указать свойства желаемого упражнения. Для этого все концепты и правила предметной области (домена) делятся на три класса: целевые (которые должны развиваться или оцениваться в рамках данного упражнения), запрещенные (обучаемый не должен их знать на данном этапе) и допустимые (не являются целью упражнения, но их знание ожидается от обучаемых – например, в результате изучения предыдущих тем). Задаются ожидаемая сложность упражнения, от которой зависит, например, среднее количество различных концептов и правил в одном задании, а также сложность структуры объектов в условии задачи. В общем случае простые упражнения будут чередоваться с заданиями на различные целевые концепты (правила), а сложные – комбинировать много концептов (правил) в каждом задании упражнения. Кроме того, выбирается желаемая педагогическая стратегия из доступных в системе (например, фиксированное число заданий или адаптивное выполнение до достижения системой уверенности в знаниях обучаемого). Преподаватель может опробовать различные режимы настроек с помощью просмотра созданного упражнения.

Таким образом, задав настройки на одном экране, преподаватель может получить упражнение с требуемыми свойствами, при этом, если в банке недостаточно задач с нужными свойствами, они будут добавлены по мере необходимости.

Замыкание большого цикла обратной связи требует серьезной работы от разработчиков домена и генератора задач для каждой предметной области, но эта работа проводится однократно для каждого нового вида задания. Она снимает большую нагрузку по составлению заданий с пользователей-преподавателей.

## Результаты апробации подхода и дискуссия

Для проверки предлагаемого подхода исследовано поведение генератора заданий. Основной целью эксперимента являлась проверка способности выполнять балансировку банка заданий. Для этого путем первичной генерации получен несбалансированный банк заданий, который затем был дополнен в процессе фоновой генерации при использовании в учебном процессе Волгоградского государственного технического университета.

В настройках поиска репозиториев генератора указан язык программирования Си, размер репозитория – от 50 КБ до 100 МБ, исключены клоны репозиториев и выполнена сортировка по убыванию пользовательского рейтинга.

В ходе начального заполнения получено 5 655 заданий на порядок вычисления выражений. Затем тренажер эксплуатировался в учебном процессе и накапливал статистику по видам требуемых заданий: сохранено 23 запроса на поиск задания, среди которых 11 существенно различных по составу целевых концептов, для двух из которых найдено менее 500 заданий, для девяти – менее 5 000 заданий. После этого была запущена фоновая генерация.

На первом этапе порог начала генерации и уровень остановки были заданы равными 500 заданиям. Согласно этой настройке в генерации участвовали только два запроса, которым соответствовало менее 200 заданий (см. таблицу). После работы генератора около суток для каждого из двух запросов найдено достаточно заданий, запросы помечены генератором как удовлетворенные и перемещены в архив, после чего генерация была остановлена.

На втором этапе, чтобы задействовать больше запросов, был увеличен порог генерации необходимых заданий в 10 раз, из-за чего девять запросов начали обрабатываться. Два запроса из предыдущего этапа в генерации не участвовали, так как они были архивированы, а новых запросов с недостатком не поступало (эти цели обучения не были актуализированы в процессе эксплуатации). Время работы генератора на втором этапе составило 4–5 суток. Для большинства запросов успешно сгенерировалось достаточно заданий, чтобы дополнить количество подходящих до 5 000. Эти запросы также были архивированы. Два запроса остались не полностью удовлетворенными, однако количество заданий по ним было сильно увеличено (см. таблицу). За более длительное время генерация может быть доведена до конца.

**Результаты дополнительной догенерации заданий**

**Results of additional learning-problem generation**

| Семейство целевых концептов | Заданий в банке до генерации | Количество обработанных репозиториев | Заданий добавлено в банк | Заданий в банке после генерации |
|---|---|---|---|---|
| *Первый этап (для порога в 500 заданий)* | | | | |
| Присваивания с обновлением (+= &= ) | 143 | 24 | 361 | 504 |
| Инкремент (++ --) | 168 | 32 | 351 | 519 |
| *Второй этап (для порога в 5 000 заданий)* | | | | |
| Оператор (=) | 2 073 | 37 | 2 985 | 5 058 |
| Оператор (,) | 972 | *114 | 2 960 | *3 932 |
| Арифметика (+ - * / %) | 2 086 | 33 | 2 937 | 5 023 |
| Побитовые (& \| >> <<) | 762 | *92 | 2 897 | *3 659 |
| Индексация ([ ]) | 2 520 | 33 | 2 508 | 5 028 |
| Сравнения (< == != >) | 2 824 | 35 | 2 278 | 5 102 |
| Доступ к полю (. ->) | 2 985 | 17 | 2 034 | 5 019 |
| Указатели (*_ ->) | 3 131 | 14 | 1 981 | 5 112 |
| Логические (! && \|\| ) | 3 407 | 43 | 1 597 | 5 004 |

Примечание: * – генерация была остановлена вручную.

За время работы генератора было скачано 1 398 репозиториев, 526 из которых генератор смог использовать, чтобы сгенерировать и отправить в банк тренажера 14 479 новых заданий на порядок вычисления выражений для шести запросов. Число заданий в банке возросло с 5 655 до 20 134. Из одного репозитория было сгенерировано от 1 до 640 заданий. Наиболее редко встречающиеся задания включали такие ситуации (правила), как наличие оператора внутри скобочного оператора (обращение к элементу массива, вызов функции) и двух подряд идущих операторов с одинаковым приоритетом и правой ассоциативностью.

## Заключение

В статье описан подход к построению ИОС. Проанализированы подходы к генерации учебных заданий и требования к системе, позволяющей автоматически генерировать учебные задания и использовать их в процессе обучения. Описана архитектура ИОС с фоновой генерацией заданий в соответствии с возникающей необходимостью, рассмотрена система CompPrehension, реализующая этот принцип на практике. В ходе апробации системы в учебном процессе университета изначально сгенерированный несбалансированный банк заданий был дополнен заданиями по темам, на которые заданий было мало, что позволило избежать выдачи одинаковых заданий даже при значительном количестве обучаемых.

В дальнейшем планируется расширить список поддерживаемых предметных областей, улучшить точность поиска заданий в банке и реализовать генерацию новых заданий в областях, не связанных с обучением программированию. Результаты исследования имеют практическую значимость для разработки ИОС и учебных заведений, которые стремятся повысить массовость и разнообразие учебных задач.

**Список литературы**

1. Qi R., Fossati D. Unlimited trace tutor: Learning code tracing with automatically generated programs. Proc. SIGCSE, 2020, pp. 427–433. doi: 10.1145/3328778.3366939.

2. D'Antoni L., Helfrich M., Kretinsky J., Ramneantu E., Weininger M. Automata tutor v3. In: LNTCS. Proc. CAV, 2020, vol. 12225, pp. 3–14. doi: 10.1007/978-3-030-53291-8_1.

3. Sychev O., Prokudin A., Denisov M. Generation of code tracing problems from open-source code. Proc. SIGCSE, 2023, pp. 875–881. doi: 10.1145/3545945.3569774.

4. Kumar A.N. Long term retention of programming concepts learned using a software tutor. In: LNPSE. Proc. ITS, 2020, vol. 12149, pp. 382–387. doi: 10.1007/978-3-030-49663-0_46.

5. Razzaq L., Patvarczki J., Almeida S.F., Vartak M., Feng M., Heffernan N.T., Koedinger K.R. The ASSISTment builder: Supporting the life cycle of tutoring system content creation. IEEE Transactions on Learning Tech., 2009, vol. 2, no. 2, pp. 157–166. doi: 10.1109/TLT.2009.23.

6. Juričić V. Software visualization in education. INFuture2019. Proc. INFuture2019: Knowledge in the Digital Age, 2020, pp. 216–220. doi: 10.17234/INFUTURE.2019.26.

7. Kurdi G., Leo J., Parsia B., Sattler U., Al-Emari S. A systematic review of automatic question generation for educational purposes. IJAIED, 2019, vol. 30, pp. 121–204. doi: 10.1007/s40593-019-00186-y.

8. Rusak G., Yan L. Unique exams: Designing assessments for integrity and fairness. Proc. SIGCSE, 2021, pp. 1170–1176. doi: 10.1145/3408877.3432556.

9. Mosbeck M., Hauer D., Jantsch A. VELS: VHDL E-learning system for automatic generation and evaluation of per-student randomized assignments. Proc. NORCAS Conf.: SoC, 2018, pp. 1–7. doi: 10.1109/NORCHIP.2018.8573455.

10. Жуков И.А., Костюк Ю.Л. Предметно-ориентированный язык для генерации заданий из исходных текстов программ // Вестн. НГУ. Сер. Информационные технологии. 2022. Т. 20. № 1. С. 18–27.

11. Ade-Ibijola A. Syntactic generation of practice novice programs in python. In: CCIS. Proc. SACLA, 2019, vol. 963, pp. 158–172. doi: 10.1007/978-3-030-05813-5_11.

12. Kulshreshtha D., Shayan M., Belfer R., Reddy S., Serban I.V., Kochmar E. Few-shot question generation for personalized feedback in intelligent tutoring systems. In: PAIS, 2022, vol. 351, pp. 17–30. doi: 10.3233/FAIA220062.

13. Das B., Majumder M., Phadikar S., Sekh A.A. Automatic question generation and answer assessment: A survey. RPTEL, 2021, vol. 16, art. 5. doi: 10.1186/s41039-021-00151-1.

14. Atzeni M., Atzori M. CodeOntology: RDF-ization of source code. In: LNISA. Proc. ISWC, 2017, vol. 10588, pp. 20–28. doi: 10.1007/978-3-319-68204-4_2.

15. Maheen F., Asif M., Ahmad H. et al. Automatic computer science domain multiple-choice questions generation based on informative sentences. PeerJ Comput. Sci., 2022, vol. 8, art. e1010. doi: 10.7717/PEERJ-CS.1010.

16. Sychev O., Penskoy N., Prokudin A. Generating expression evaluation learning problems from existing program code, Proc. ICALT, 2022, pp. 183–187. doi: 10.1109/ICALT55010.2022.00061.

17. Hahn M.G., Navarro S.M.B., De La Fuente V., Burgos D. A systematic review of the effects of automatic scoring and automatic feedback in educational settings. IEEE Access, 2021, vol. 9, pp. 108190–108198. doi: 10.1109/ACCESS.2021.3100890.

18. Bernius J.P., Krusche S., Bruegge B. Machine learning based feedback on textual student answers in large courses. Computers and Education: Artificial Intelligence, 2022, vol. 3, art. 100081. doi: 10.1016/j.caeai.2022.100081.

19. Narayanan S., Kommuri V.S., Subramanian S.N., Bijlani K. Question bank calibration using unsupervised learning of assessment performance metrics. Proc. ICACCI, 2017, pp. 19–25. doi: 10.1109/ICACCI.2017.8125810.

20. Sychev O., Anikin A., Penskoy N., Denisov M., Prokudin A. CompPrehension – model-based intelligent tutoring system on comprehension level. In: LNPSE. Proc. ITS, 2021, vol. 12677, pp. 52–59. doi: 10.1007/978-3-030-80421-3_6.

21. Prokudin A., Sychev O., Denisov M. Learning problem generator for introductory programming courses. Software Impacts, 2023, vol. 17, art. 100519. doi: 10.1016/j.simpa.2023.100519.

Software & Systems doi: 10.15827/0236-235X.142.201-212 2024, 37(2), pp. 201–212

## Intelligent tutoring system with automatic assignment generation according to the results of using an assignment bank

**Oleg A. Sychev [1]✉, Artem A. Prokudin [1], Mikhail E. Denisov [1]**

[1] Volgograd State Technical University, Volgograd, 400005, Russian Federation

**For citation**

Sychev, O.A., Prokudin, A.A., Denisov, M.E. (2024) 'Intelligent tutoring system with automatic assignment generation according to the results of using an assignment bank', *Software & Systems*, 37(2), pp. 201–212 (in Russ.). doi: 10.15827/0236-235X.142.201-212

**Article info**

Received: 18.10.2023 After revision: 20.12.2023 Accepted: 22.12.2023

**Abstract.** An educational process includes two feedback loops: learner feedback directs actual learning for a given individual (small loop), while teacher feedback about the quality of the bank of learning problems (big loop) allows improving a learning process in time, increasing the number of assignments which are often used repeatedly and so learners can share solutions. Automating the big feedback loop is an important step in developing intelligent tutoring systems, which grants course authors relief from routine work on creating and enhancing learning assignment banks. Modern advances in methods of learning assignment generation and their classification for using in a learning process allows implementing an intelligent tutoring system, which can generate new learning assignments in the background mode if necessary. The paper describes the architecture of an intelligent tutoring system with background learning assignment generation and the CompPrehension system, which implements the mentioned architecture. We describe all major functional system components: a training aid, a learning assignment bank, an assignment generator, a domain; and their interaction with the assignment database and external semantic reasoners for solving learning assignments. Practical evaluation of the described approach showed that the system can selectively improve the initial unbalanced learning assignment bank, taking into account the history of assignment requests. In less than 5 days, the system has generated several thousand additional learning assignments for each concept with low learning assignment count. This gives low probability of receiving the same assignments even when testing hundreds of students. The study results are significant for educational institutions seeking to improve the learning process.

**Keywords:** intelligent tutoring systems, question generation, online learning, learning system development, CompPrehension, learning assignment bank

### References

1. Qi, R., Fossati, D. (2020) 'Unlimited trace tutor: Learning code tracing with automatically generated programs', *Proc. SIGCSE*, pp. 427–433. doi: 10.1145/3328778.3366939.

2. D'Antoni, L., Helfrich, M., Kretinsky, J., Ramneantu, E., Weininger, M. (2020) 'Automata tutor v3', in *LNTCS. Proc. CAV*, 12225, pp. 3–14. doi: 10.1007/978-3-030-53291-8_1.

3. Sychev, O., Prokudin, A., Denisov, M. (2023) 'Generation of code tracing problems from open-source code', *Proc. SIGCSE*, pp. 875–881. doi: 10.1145/3545945.3569774.

4. Kumar, A.N. (2020) 'Long term retention of programming concepts learned using a software tutor', in *LNPSE. Proc. ITS*, 12149, pp. 382–387. doi: 10.1007/978-3-030-49663-0_46.

5. Razzaq, L., Patvarczki, J., Almeida, S.F., Vartak, M., Feng, M., Heffernan, N.T., Koedinger, K.R. (2009) 'The ASSISTment builder: Supporting the life cycle of tutoring system content creation', *IEEE Transactions on Learning Tech*., 2(2), pp. 157–166. doi: 10.1109/TLT.2009.23.

6. Juričić, V. (2020) ‘Software visualization in education. INFuture2019’, *Proc. INFuture2019: Knowledge in the Digital Age*, pp. 216–220. doi: 10.17234/INFUTURE.2019.26.

7. Kurdi, G., Leo, J., Parsia, B., Sattler, U., Al-Emari, S. (2019) ‘A systematic review of automatic question generation for educational purposes’, *IJAIED*, 30, pp. 121–204. doi: 10.1007/s40593-019-00186-y.

8. Rusak, G., Yan, L. (2021) ‘Unique exams: Designing assessments for integrity and fairness’, *Proc. SIGCSE*, pp. 1170–1176. doi: 10.1145/3408877.3432556.

9. Mosbeck, M., Hauer, D., Jantsch, A. (2018) ‘VELS: VHDL E-learning system for automatic generation and evaluation of per-student randomized assignments’, *Proc. NORCAS Conf.: SoC*, pp. 1–7. doi: 10.1109/NORCHIP.2018.8573455.

10. Zhukov, I.A., Kostyuk, Yu.L. (2022) ‘A domain-specific language for generating tasks from programs source code’, *Bull. of NSU. Ser.: Inform. Tech*., 20(1), pp. 18–27 (in Russ.). doi: 10.25205/1818-7900-2022-20-1-18-27.

11. Ade-Ibijola, A. (2019) ‘Syntactic generation of practice novice programs in python’, in *CCIS. Proc. SACLA*, 963, pp. 158–172. doi: 10.1007/978-3-030-05813-5_11.

12. Kulshreshtha, D., Shayan, M., Belfer, R., Reddy, S., Serban, I.V., Kochmar, E. (2022) ‘Few-shot question generation for personalized feedback in intelligent tutoring systems’, in *PAIS*, 351, pp. 17–30. doi: 10.3233/FAIA220062.

13. Das, B., Majumder, M., Phadikar, S., Sekh, A.A. (2021) ‘Automatic question generation and answer assessment: A survey’, *RPTEL*, 16, art. 5. doi: 10.1186/s41039-021-00151-1.

14. Atzeni, M., Atzori, M. (2017) ‘CodeOntology: RDF-ization of source code’, in *LNISA. Proc. ISWC*, 10588, pp. 20–28. doi: 10.1007/978-3-319-68204-4_2.

15. Maheen, F., Asif, M., Ahmad, H. et al. (2022) ‘Automatic computer science domain multiple-choice questions generation based on informative sentences’, *PeerJ Comput. Sci*., 8, art. e1010. doi: 10.7717/PEERJ-CS.1010.

16. Sychev, O., Penskoy, N., Prokudin, A. (2022) ‘Generating expression evaluation learning problems from existing program code’, *Proc. ICALT*, pp. 183–187. doi: 10.1109/ICALT55010.2022.00061.

17. Hahn, M.G., Navarro, S.M.B., De La Fuente, V., Burgos, D. (2021) ‘A systematic review of the effects of automatic scoring and automatic feedback in educational settings’, *IEEE Access*, 9, pp. 108190–108198. doi: 10.1109/ACCESS.2021.3100890.

18. Bernius, J.P., Krusche, S., Bruegge, B. (2022) ‘Machine learning based feedback on textual student answers in large courses’, *Computers and Education: Artificial Intelligence*, 3, art. 100081. doi: 10.1016/j.caeai.2022.100081.

19. Narayanan, S., Kommuri, V.S., Subramanian, S.N., Bijlani, K. (2017) ‘Question bank calibration using unsupervised learning of assessment performance metrics’, *Proc. ICACCI*, pp. 19–25. doi: 10.1109/ICACCI.2017.8125810.

20. Sychev, O., Anikin, A., Penskoy, N., Denisov, M., Prokudin, A. (2021) ‘CompPrehension – model-based intelligent tutoring system on comprehension level’, in *LNPSE. Proc. ITS*, 12677, pp. 52–59. doi: 10.1007/978-3-030-80421-3_6.

21. Prokudin, A., Sychev, O., Denisov, M. (2023) ‘Learning problem generator for introductory programming courses’, *Software Impacts*, 17, art. 100519. doi: 10.1016/j.simpa.2023.100519.

**Авторы**
**Сычев Олег Александрович** [1], к.т.н.,
доцент, oasychev@gmal.com
**Прокудин Артем Александрович** [1],
аспирант, prokudin@vstu.ru
**Денисов Михаил Евгеньевич** [1],
аспирант, denisov@vstu.ru

[1] Волгоградский государственный
технический университет,
г. Волгоград, 400005, Россия

**Authors**
**Oleg A. Sychev** [1], Dr.Sc. (Engineering),
Associate Professor, oasychev@gmal.com
**Artem A. Prokudin** [1],
Postgraduate Student, prokudin@vstu.ru
**Mikhail E. Denisov** [1],
Postgraduate Student, denisov@vstu.ru

[1] Volgograd State Technical University,
Volgograd, 400005,
Russian Federation