УДК 004.89 *Научная статья* DOI: 10.18287/2223-9537-2025-15-1-67-81

Метод разработки интеллектуальных тренажёров на основе онтологии предметной области

© 2025, О.А. Сычёв ✉, Н.А. Пенской, Г.В. Терехов

Волгоградский государственный технический университет (ВолгГТУ), Волгоград, Россия

Аннотация

В учебных дисциплинах обучаемому необходимо усвоить много новых понятий, для чего требуется большой объём тренировки с обратной связью. Интеллектуальный тренажёр может позволить обучаемому научиться решать простые задачи и получать объяснения ошибок, а преподаватель на занятии может уделить время решению более сложных задач. В данной работе предлагается метод разработки интеллектуальных тренажёров на основе онтологии предметной области в виде веб-приложений, доступных для аудиторной и внеаудиторной работы. Использование формата *RDF* для представления задачи и модели предметной области позволяет осуществить логический вывод с помощью машины вывода *Apache Jena Reasoner*. Приведён пример разработки интеллектуального тренажёра для изучения порядка вычисления выражений с поддержкой языков программирования *C++, C#* и *Python*, способного объяснять ошибки, генерировать объясняющие подсказки и вести обучающий диалог с помощью наводящих вопросов. Тренажёр опробован при обучении бакалавров и магистров факультета электроники и вычислительной техники Волгоградского государственного технического университета. Большинство студентов отметили разработанный тренажёр как более полезный, чем обучающий тест. Тренажёр может применяться при самостоятельном изучении темы и при проведении занятий в рамках учебного процесса.

Ключевые слова: *интеллектуальный тренажёр, обучение, онтология предметной области, порядок вычислений выражений, вводные курсы, программирование.*

Цитирование: *Сычёв О.А., Пенской Н.А., Терехов Г.В.* Метод разработки интеллектуальных тренажёров на основе онтологии предметной области. *Онтология проектирования*. 2025. Т.15, №1(55). С.67-81. DOI: 10.18287/2223-9537-2025-15-1-67-81.

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Введение

При первоначальном обучении программированию освоение большого числа новых абстрактных понятий, необходимых для написания первой программы, может создавать большие трудности [1-4] и приводить к разочарованию в профессии [5]. На вводных курсах используются различные стратегии для помощи обучающимся: компьютерное тестирование, инструменты на основе мобильных приложений [6], инструментальные среды для создания электронных курсов [7] и т.п.

Отдельного внимания заслуживает та группа обучаемых, которые с трудом усваивают новые понятия. Преподавателю не всегда удаётся уделить достаточное внимание каждому обучаемому, поэтому получение обучаемыми обратной связи о причинах их ошибок целесообразно формализовать и автоматизировать.

Частыми случаями являются ошибки, связанные с освоением новых понятий предметной области (ПрО). В таких случаях полезной может быть учебная среда, в которой обучаемый, решая простые задачи, получает информацию о новых понятиях ПрО и связях между ними, а также объяснение своих ошибок каждый раз, когда он их совершает [8, 9]. Работа с такой учебной средой может быть не ограничена по времени, а опыт, полученный от самостоятельного решения задач, полезен для изучения новых тем.

Большинство программ для отображения работы программного кода показывает выполнение кода с множеством сложных аспектов, что может стать излишней когнитивной нагрузкой и помешать процессу обучения. Успех упражнений по моделированию визуального программирования зависит от согласования взаимодействия графического интерфейса с конкретными целями обучения [10], что легче сделать с помощью нескольких инструментов, подходящих для обучения по конкретным темам. В [11] отмечается, что некоторые обучающиеся считают системы онлайн-обучения слишком повторяющимся, что можно устранить с помощью адаптивного тестирования. Согласно [12], сложные программные анимации «подавляют» некоторых обучаемых, это приводит к выводу, что таким обучаемым нужны простые тренажёры для освоения базовых понятий.

Программы для обучения могут подсказать обучаемому следующее правильное действие, когда он ошибся или запросил подсказку. Например, такой подход можно встретить в предварительном типе вопроса *Preg* для системы дистанционного обучения *Moodle* [13], где в вопросах открытого типа обучаемый имеет возможность получить обратную связь в виде места первой ошибки и следующего правильного символа или лексемы, если поиск правильного продолжения ответа представляет сложности. Этого не всегда достаточно, т.к. обучаемые могут выполнить правильный шаг по подсказке, не понимая, почему он правильный. Подсказка следующего правильного шага позволяет успешно закончить упражнение, но не объясняет обучаемому причины его ошибки.

Полезной помощью обучаемому является обратная связь с сообщением о нарушенных правилах ПрО и о следующем правильном шаге с объяснением правил ПрО, которые делают этот шаг возможным. Такая обратная связь может быть основана на парадигме с ограничениями, где каждая ошибка связана с нарушением определённого ограничения. Такой подход был предложен для автоматизации изучения понятий ПрО в [14].

Существуют учебные среды, пользоваться которыми можно как при взаимодействии с преподавателем, так и при самостоятельной подготовке. К ним можно отнести «КЛИОС» [15], которая предназначена для изучения русского языка, а также интеллектуальную среду поддержки разработки интегрированных экспертных систем «АТ-ТЕХНОЛОГИЯ» [16]. В этой среде комментарии к решаемой задаче, подсказки и объяснения полученных результатов задаются преподавателем при создании задачи, что делает создание задач трудоёмким, а качество обучения зависимым от конкретной задачи.

Цель работы - улучшение понимания обучаемым содержания понятий ПрО, используя интеллектуальную обучающую среду на основе онтологии ПрО, содержащей аксиоматическое определение понятий ПрО. Такая среда (интеллектуальный тренажёр, ИТ) сможет генерировать правильные ответы при решении простых задач об изучаемых понятиях в совокупности с поясняющей обратной связью об ошибках обучаемого.

1 Построение ИТ на основе онтологии ПрО

Для разработки предлагаемого подхода выбрана задача определения порядка вычисления выражения, записанного на языке программирования. Эта задача необходима для правильного понимания и написания кода и служит основой для более сложных задач (например, проверки типов данных). Обратная связь должна сообщить обучающемуся об ошибке и дать объяснение причины. Для создания поясняющей обратной связи используются декларативные модели, которые выражают логику принятия решений без описания её алгоритма. Это позволяет определить все правильные решения. Важным является то, что в декларативной модели неправильные решения вызовут ошибки в условиях, которые определяют правильность ответа, при этом конкретное нарушенное условие позволяет определить тип ошибки.

Для продуктивного изучения программирования важен правильный порядок изучения понятий и выполнения задач для проверки отдельных понятий и их сочетаний [17]. Навык определения порядка вычисления выражений относится к базовым и необходим в самом начале изучения программирования. От его усвоения зависит понимание следующих тем, особенно включающих сложные типы данных (массивы, объекты, указатели и ссылки).

В современных языках программирования задача определения порядка вычисления выражения не является простой. Языки программирования имеют несколько уровней приоритета операций и содержат такие понятия, как ассоциативность операций и отношения «упорядочено до» и «упорядочено после». На вводных курсах по программированию большая часть аудиторного времени, как правило, уделяется более сложным темам, таким как изучение циклов и методов отладки, при этом тема вычисления выражений остаётся недостаточно раскрытой. Например, для ответа на вопрос: «Может ли на текущем шаге быть выполнен оператор X?», обучаемому необходимо проанализировать до четырёх ветвей рассуждений, определяя, полностью ли выполнится левый, центральный и правый операнды X, и есть ли какие-либо операторы со строгим порядком вычисления своих операндов, блокирующие вычисление X. При анализе каждой ветви требуется ответить на несколько вопросов. Анализ последней ветви включает два рекурсивных вопроса, подразумевающих возможность решения той же задачи для определения операндов операторов со строгим порядком вычисления.

Построение ИТ включает:

1) разработку онтологической модели (ОМ) изучаемой ПрО, учитывающей возможные ошибки в процессе решения задач;
2) составление шаблонов обратной связи для обучаемого в соответствии с видами ошибок;
3) разработку интерфейсов обучаемого (минимизируя количество действий, необходимых для ввода ответа) и преподавателя (для ввода задач), в виде веб-интерфейсов;
4) разработку программного кода, связывающего интерфейсы пользователей с ОМ.

ОМ ПрО, используемая для разработки ИТ, должна удовлетворять условиям [14]. Принципиальным компонентом модели является система правил вывода для проверки решения задачи обучаемым и поиска смысловых ошибок в нём. ОМ может дополняться системой правил для пошагового решения задачи, позволяющей генерировать подсказки, если обучаемый затрудняется продолжить решение. Эти правила могут быть запущены на машине вывода для получения результатов. ОМ учебной задачи включает:

- объекты ПрО;
- классы объектов (категории или типы объектов ПрО), которые могут быть связаны отношением наследования;
- свойства классов (общие для всех объектов этого класса) и их объектов;
- отношения (связи) между объектами.

Для представления данных для машинного вывода используется стандарт *Resource Description Framework* (*RDF*). Единицей представления информации в *RDF* является триплет $T=<S,P,O>$, который включает три компонента: субъект S, предикат P и объект O. Множество триплетов представляют собой базу знаний (БЗ) или *RDF*-граф, которые выражают факты об объектах в моделируемой ПрО. Такой граф обычно рассматривается как совокупность: словаря, который служит для описания конкретных данных; свойств классов и свойств операторов. Словарь является частью графа знаний и описывает систему абстрактных понятий, например классы объектов и их возможные связи, и применяется для описания другой части графа - экземпляров, которые представляют конкретные сущности в моделируемой ПрО. Примером универсальной машины вывода является *Apache Jena* [18].

В качестве альтернативы можно использовать деревья мыслительных процессов как расширение деревьев решений $G=(V, L)$ [19]. Вершины V в таком дереве соответствуют ша-

гам в решении задачи, а рёбра *L* - результатам выполнения шагов. С их помощью можно описать процесс решения задачи для моделируемой ПрО. Поскольку для эффективной работы ИТ необходимы два набора правил (для решения задачи и для нахождения ошибок), то необходимо построить два дерева. Дерево для решения задачи содержит один вид узлов-листьев: позитивный (зелёный) узел с перечнем действий над графом знаний для запоминания найденного шага решения. Каждому такому узлу ставится в соответствие один вид подсказки, который параметризуется найденными данными. Дерево для проверки ответа и нахождения ошибок содержит два вида листьев: зелёные (по данной ветке ответ не содержит ошибок) и красные (обнаружена ошибка). Перечень красных узлов составляет основу списка возможных видов ошибок.

Составление системы правил вывода для определения ошибок позволяет классифицировать виды ошибок; в процессе поиска ошибки находятся данные, идентифицирующие объекты, связанные с ошибкой, которые необходимо включить в сообщение. На их основе составляются шаблоны обратной связи для обучаемого, которые используются для генерации объясняющих сообщений об ошибках. Разрабатываются необходимые шаблоны для вывода информации об объектах, значениях их свойств и т.д. Для набора правил решения задачи составляются шаблоны подсказок, объясняющих, почему в данных обстоятельствах подсказанная часть решения задачи является правильной.

Для ИТ необходимо разработать предметно-зависимый интерфейс моделируемой ПрО, который позволит наглядно отображать задачу и вводить результат решения в виде минимального количества действий. Поскольку задания, которые допускают интеллектуальную обратную связь, должны иметь закрытое множество ответов [14], то эффективность интерфейса обучаемого можно измерять количеством воздействий на компьютерную мышь, необходимых для решения задачи. Интерфейс преподавателя должен максимально упрощать ввод задачи и осуществлять перевод её условия в формат *RDF*, при этом минимизируя необходимость ознакомления с новыми языками записи заданий.

Современные ИТ, доступные через сеть Интернет, представляют собой клиент-серверные приложения, где взаимодействие между компонентами тренажёра может быть организовано посредством *HTTP Rest API* (см. рисунок 1).

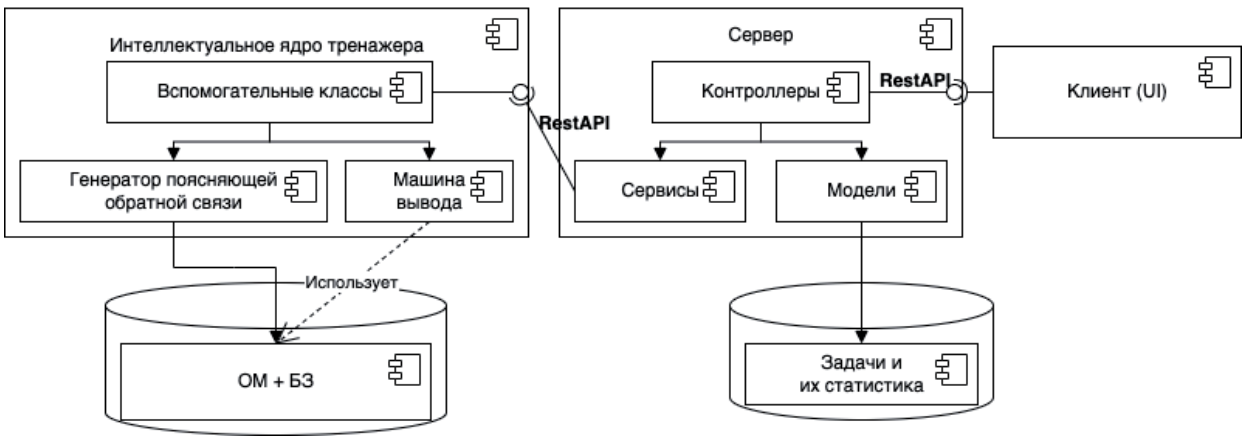


Рисунок 1 – Диаграмма компонентов интеллектуального тренажёра

Одним из архитектурных шаблонов, применяемых при создании веб-приложений, является модель-представление-контроллер (*Model-View-Controller, MVC*). На нём основаны веб-фреймворки *Ruby On Rails* [20], *Laravel* [21] и др. Серверная часть должна содержать сервисы для доступа к программной машине вывода: программный интерфейс приложения (*API, Application Programming Interface*), контроллеры для обработки запросов от клиента, модели

для взаимодействия с базой данных (БД), шаблоны для генерации пользовательского интерфейса и данных для клиентского компонента, например в виде формата *JSON* (*JavaScript Object Notation*). Правила ПрО хранятся в БЗ (например, в формате *XML*), отдельно в БД хранятся созданные преподавателями задачи и статистика их решения.

На каждом шаге решения задачи клиентская часть должна обновлять модель ответа обучаемого - представлять ответ обучаемого в виде *JSON* и отправлять его на сервер посредством *AJAX*-запроса (*Asynchronous Javascript and XML,* асинхронный *JavaScript* и *XML*). Вспомогательные классы интеллектуального ядра тренажёра получают и анализируют запрос, вызывают машину вывода для проверки полученной части ответа. Если текущий шаг решения задачи содержит ошибку, генератор поясняющей обратной связи создаст подсказку на естественном языке на основе дополнительной информации из ответа машины вывода.

2 ИТ определения порядка вычисления выражений

ИТ[1] разработан с использованием описанного в разделе 1 метода. ОМ выражения на языке программирования разработаны для каждого поддерживаемого тренажёром языка. Каждая ОМ состоит из двух классов верхнего уровня - «Токен» и «Элемент». Дочерними классами для класса «Элемент» являются «Оператор» и «Операнд». В каждом языке программирования можно выделить множество свойственных ему операторов, которые в ОМ являются подклассами класса «Оператор». Для языка *C++* класс «Оператор» имеет 55 подклассов, отражающих конкретные операторы языка программирования. Конкретные операторы, встречающиеся в выражении на языке программирования, могут рассматриваться в качестве экземпляров классов операторов, например класса «Сложение». Пример фрагмента иерархии классов ОМ, составленный с помощью редактора *Protege* для выражения на языке программирования *C++,* представлен на рисунке 2.

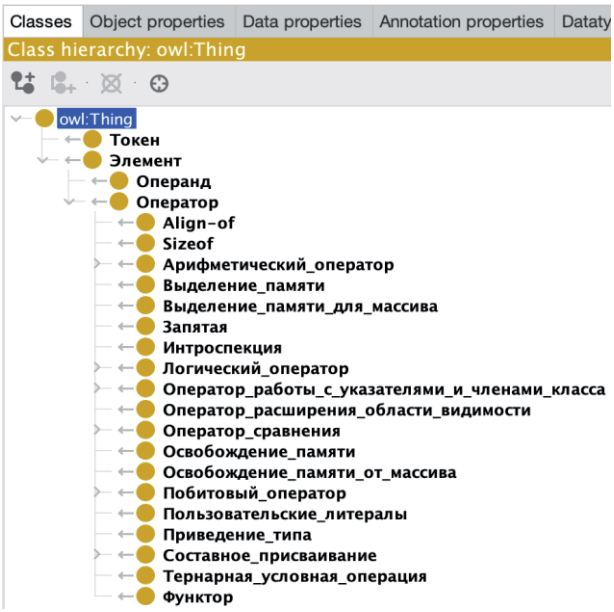


Рисунок 2 – Верхний уровень иерархии онтологии выражения на языке программирования на примере *C++* (выполнено в редакторе *Protege*)

Представление выражения на языке программирования в качестве онтологии включает задачу систематизации отношений. Объект класса «Элемент» должен быть связан с одним или несколькими объектами класса «Токен», что отражает отношение «имеет токен». Взаимное расположение токенов в выражении отражается при помощи отношения «находится непосредственно слева от» и его производных отношений. Связь между объектами классов «Элемент» и «Оператор» задаётся при помощи отношения «является операндом». Выделенные в ходе разработки ОМ отношения представлены на рисунке 3.

Первая версия ИТ разработана с использованием подготовленных правил для машины вывода [22] и переработана с использованием описанного в данной статье метода. Применение дерева мыслительного процесса позволило добавить возможность задавать обучаемому

[1] Интеллектуальный тренажер HOWITWORKS:EXPRESSIONS. *Сычёв О.А., Пенской Н.А., Терехов Г.В.* Свидетельство о государственной регистрации программы для ЭВМ. Номер свидетельства: RU 2022685372. 2022 Номер заявки: 2022684973.

наводящие вопросы, когда он совершил ошибку при выполнении упражнения, поскольку дерево явно описывает желаемый ход рассуждения обучаемого. Вопросы генерируются на основе шаблонов, приписанных к узлам дерева мыслительного процесса, и словарей. Это даёт возможность определить и исправить ошибки обучаемого, если сообщений от ИТ, связанных со сделанными им ошибками, оказывается недостаточно. В связи с особенностью ПрО составлено одно дерево мыслительного процесса, поскольку существует один способ решения этой задачи (перебор возможных вариантов и выбор подходящих).

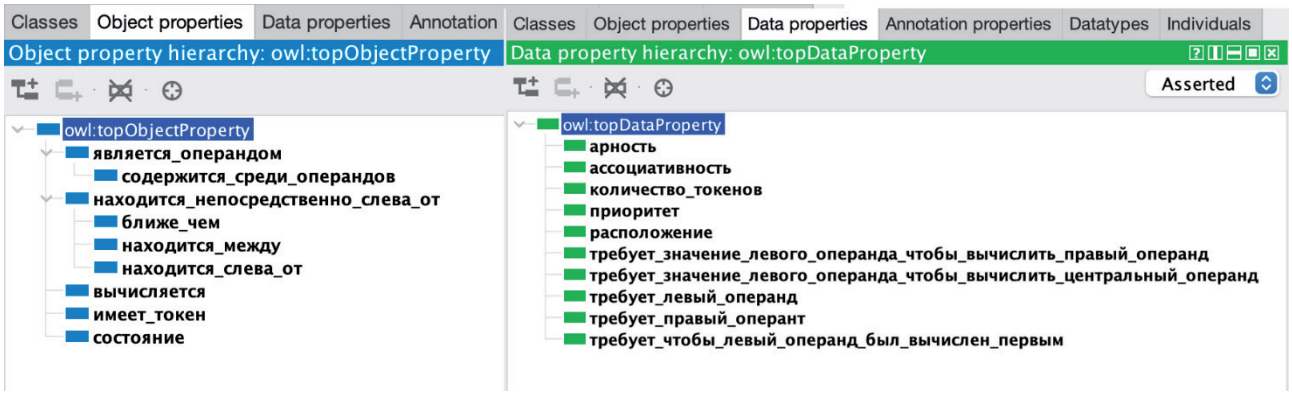


Рисунок 3 – Типы отношений для выражения на языке программирования на примере *C++* (выполнено в редакторе *Protege*)

Дерево мыслительного процесса основано на описании нескольких словарей: классы объектов, свойства класса, свойства объекта, связи, вопросы и действия. Классы определяют виды объектов, которые содержатся в формулировке упражнения и в его решении. Свойства класса - данные, значения которых одинаковы для всех объектов данного класса. Свойства объекта - данные, значения которых могут отличаться для разных объектов одного и того же класса. Связи предназначены для соединения двух объектов, на их основе можно вычислять новые связи. Словарь вопросов и словарь действий содержат виды вопросов и действий для конкретного дерева.

Дерево мыслительного процесса для задачи порядка вычисления выражения имеет четыре ветви (В1-В4), которые позволяют получить ответы на соответствующие вопросы: были ли вычислены полностью вложенные операторы (В1); были ли вычислены полностью операторы, являющиеся операндами слева (В2) и справа (В3); не влияют ли на вычисление вышестоящие операторы со строгим порядком вычисления операндов (В4). Разработанное дерево содержит 23 узла вопросов, 7 узлов действий, 3 узла ветвления, 2 узла цикла, 22 зелёных узла-листа (обозначающих корректность решения) и 12 красных (обозначающих различные способы сделать ошибки). Например, необходимые словари для рассмотрения ветви В2, которая определяет текущий конкурирующий оператор слева и взаимный порядок вычисления выбранного обучаемым оператора и конкурирующего оператора, показаны в таблице 1.

Для построения рассматриваемой ветви дерева необходимо определить порядок действий, чтобы ответить на вопрос: может ли в данный момент на текущем шаге вычислен выбранный оператор? Пусть *X1* - токен, который выбран на текущем шаге решения задачи, *X* - оператор, который соответствует выбранному токену. Если выбранный оператор требует вычисления левого операнда, то в исходном выражении необходимо найти конкурирующий оператор *Y* - не вычисленный слева от *X* оператор, удовлетворяющий условию *Y : является(Y, оператор) ∧ состояние(Y, 'не_вычислен') ∧ слева_от(Y, X)*, и соответствующий ему слева от *X1* токен *Y1 : является(Y1, токен) ∧ относится_к(Y1, Y) ∧ слева_от(Y1, X1)*. Далее возможны четыре ситуации: либо *Y* охватывает *X* (В2.1), либо *X* находится в скобках, в кото-

рых нет *Y* (В2.2), либо, наоборот, *Y* в скобках, в которых нет *X* (В2.3), либо скобок вовсе нет (В2.4). Если для первых двух ситуаций (В2.1, В2.2) ответ положительный, то *X* может быть вычислен на текущем шаге решения задачи; в третьей ситуации (В2.3) положительный ответ означает, что *X* вычислен быть не может; в четвёртой ситуации (В2.4) решение принимается на основе приоритетов *X* и *Y*, а в случае одинаковых приоритетов - на основе ассоциативности *X*. Рассмотренная ветвь решения показана на рисунке 4.

Для каждого красного узла дерева составлены шаблоны обратной связи. Например, если на текущем шаге решения задачи из двух операторов с одинаковым приоритетом ошибочно выбран находящийся правее оператор до вычисления оператора, находящегося левее, шаблон обратной связи для такой ошибки имеет вид: «*${val('X','u')} не может быть вычислен, поскольку сначала должен быть вычислен ${val('Y','u')} слева от него - эти операторы имеют одинаковый приоритет и вычисляются слева направо (левоассоциативны)*». В данном шаблоне *«${val('X','u')}»* и *«${val('Y','u')}»* являются подстановками, на место которых при генерации обратной связи должна быть добавлена информация о названии оператора и его позиции для поиска соответствующего оператора в исходном выражении. Всего разработанный ИТ содержит 8 шаблонов обратной связи для ошибок, 14 для подсказок, 331 для наводящих вопросов на русском языке, столько же на английском.

Таблица 1 – Выдержка из словарей необходимых для рассмотрения ветви В2

Название словаря	Содержимое словаря
Действия	Найти <ограничения> <класс объекта>
Классы	- токен - элемент - операнд - оператор скобки (элемент)
Свойства класса	- количество токенов (элемент), целое число: 1-2 - требует левый операнд (оператор), логическое: да/нет - требует правый операнд (оператор), логическое: да/нет - требует внутренний операнд (оператор), логическое: да/нет - приоритет (оператор), линейная шкала: зависит от языка ассоциативность (оператор), перечисляемое: левая, правая, не ассоциативная
Свойства объекта	- состояние (элемент), перечисляемое: не вычислено, вычислено, использовано, пропущено вычисляется (элемент), перечисляемое: да, нет
Отношения между объектами	- <элемент> имеет токен <токен> (<токен> относится к <элементу>) - <токен1> находится слева (меньше)/справа(больше) <токен2> - <токен1> находится между <токен2> и <токен3> - <токен1> ближе <токен2> чем <токен3> <элемент> является операндом <оператора>

ИТ работает в режимах использования обучаемым и преподавателем. Для создания упражнения необходимо задать выражения (в виде строки, разделяя операторы и операнды пробелами) на одном из поддерживаемых ИТ языков программирования (см. рисунок 5).

Можно ограничить количество доступных обучаемому подсказок или полностью их отключить в зависимости от назначения упражнения: обучение или проверка знаний. После создания упражнения преподавателю доступен постоянный адрес (*URL*) этого упражнения, который можно отправлять обучаемым. Доступен просмотр статистики по выполнению созданных преподавателем упражнений. Преподавателю не нужно вводить правильные ответы, ИТ находит их, используя БЗ ПрО и машину вывода.

Работа с задачей начинается с обработки её условия. После разделения выражения, являющегося условием задачи, на токены машина вывода по правилам ПрО, хранящимся в БЗ, строит дерево выражения, указывающее операнды операций. На основе дерева, добавляя информацию о зависимости порядка выполнения операций, для операций со строгим порядком вычисления операндов строится граф отношения «должен быть вычислен перед», представляющий собой ориентированный ациклический граф, который описывает все правильные порядки вычисления и причины всех зависимостей, которые будут использованы для генерации поясняющей обратной связи.

При проверке ответа обучаемого ошибка определяется как нарушение отношения «должен быть вычислен перед» в графе, а тип правила ПрО, который использовался при генерации связи, указывает на необходимое

объяснение. Подсказки могут быть сгенерированы с помощью графа путём выбора операции, вычислению которой ничто не препятствует; объяснение в этом случае опирается на причины, почему соседние невычисленные операции не мешают вычислять данную операцию.

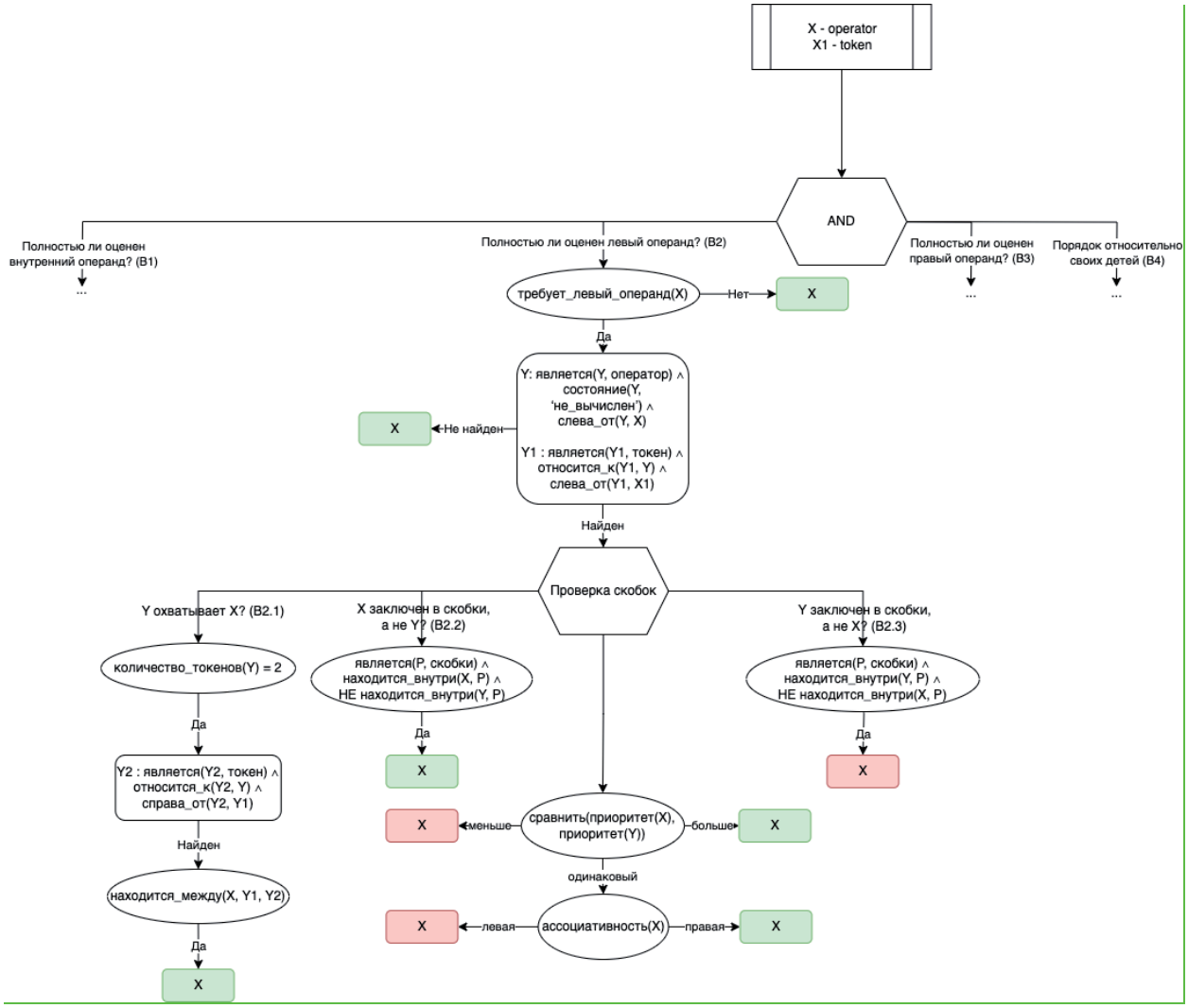


Рисунок 4 – Пример ветви дерева мыслительного процесса для определения «был ли вычислен конкурирующий оператор слева»

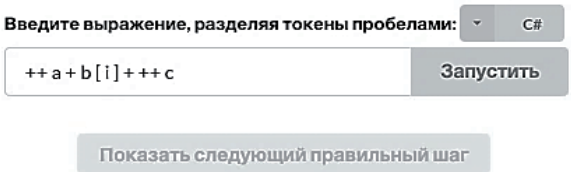


Рисунок 5 – Пример создания упражнения в интерфейсе преподавателя

Для решения задачи необходимо выбирать операции выражения в том порядке, в котором они вычисляются. Если операция выбрана правильно, она выделяется зелёным фоном и помечается порядком её выполнения в выражении, чтобы облегчить дальнейшее решение задачи. Если операция выбрана раньше, чем она может быть вычислена, она выделяется красным цветом, а под выражением отображаются сообщения с поясняющей обратной связью (см. рисунок 6). Под каждой операцией указана её позиция в выражении и в поясняющих сообщениях так, что определить, о какой конкретно операции идёт речь, не составит труда. ИТ распознаёт случаи, когда выражение имеет несколько возможных порядков вычисления.

ИТ может определить шесть видов ошибок:

1) вычисление операции первой с более низким приоритетом;
2) вычисление двух операций с одинаковым приоритетом и левой ассоциативностью справа налево;

3) вычисление двух операций с одинаковым приоритетом и правой ассоциативностью слева направо;
4) вычисление операции с центральным операндом (т.е. операции с двумя токенами, между которыми располагается её операнд, например, квадратных скобок, вызова функции или условной операции) до завершения вычисления операций внутри неё;
5) вычисление операции, имеющей операнд в скобках, до полного вычисления подвыражения в этих скобках;
6) неправильный порядок вычисления операндов операции со строгим порядком вычисления операндов (например, операции логического ИЛИ, логического И, запятая и условная операция в языке программирования *C++* всегда заканчивают вычисление своего левого операнда до начала вычисления остальных операндов).

Для каждой ошибки ИТ генерирует сообщение на основе шаблона с позициями всех соответствующих операторов (см. рисунок 6). Ошибки с номерами 4 и 5 являются похожими с точки зрения формальной модели ПрО, но для большинства начинающих программистов такое определение является слишком сложным, поэтому сообщения об ошибках для них выводятся разными. На рисунке 6 первая и вторая выбранные операции (операция «++» в позиции 1 и квадратная скобка в позиции 5) оказались правильными, но затем ошибочно выбрана операция «+» в позиции 8. Два оставшихся оператора должны быть вычислены до неё по разным причинам: операция «++» в позиции 9 должна быть вычислена ранее, потому что её приоритет выше чем у «+», а операция «+» в позиции 3 должна быть вычислена перед операцией «+» в позиции 8, потому что операция «+» является левоассоциативной. ИТ показывает обучаемому все причины, чтобы он мог проанализировать свою ошибку.

Рисунок 6 – Пример поясняющей обратной связи в интерфейсе обучаемого после совершения ошибки

Полезной функцией ИТ является отображение следующего правильного шага с пояснением, почему этот шаг является правильным. Отсутствие подсказок может приводить к попытке решить задачу перебором возможных вариантов, что не является конструктивным для освоения материала. Для получения подсказки обучаемый должен выбрать соответствующую кнопку. Поясняющее сообщение подсказки принимает во внимание две ближайших операции, которые ещё не были вычислены (слева и справа от той, которая была выбрана) и описывает, почему выбранная операция может быть вычислена на текущем шаге.

Пример использования подсказки показан на рисунке 7 - в данном случае операция «+» справа ещё не должна выполниться, потому что операция «+» является левоассоциативной. Ситуация до выбора подсказки имеет два правильных варианта следующего шага: операция «+» в позиции 3 и операция «++» в позиции 9.

Рисунок 7 – Пример использования подсказки в интерфейсе обучаемого при решении задачи

При совершении ошибки ИТ предлагает помочь разобраться в её причинах с помощью наводящих вопросов. Модуль наводящих вопросов реализован в виде чат-бота, который ведёт диалог с обучаемым. Тексты наводящих вопросов генерируются на основе шаблонов, аналогичных шаблонам обратной связи. Шаблоны соответствуют узлам дерева мыслительных процессов и переходам между ними, т.е. проверяются как знания метода решения задачи (отражённые в переходах дерева), так и фактические знания по теме (выполнение действий в узлах). Обучаемому предлагается на выбор несколько вариантов ответа. В зависимости от вопроса в качестве ответа можно выбрать один вариант из предложенных либо выбрать несколько вариантов. ИТ анализирует полученный ответ и может продолжить диалог и задать следующий наводящий вопрос или дать итоговый вывод. В любой момент можно завершить диалог с чат-ботом и продолжить решать задачу.

Для ошибочной ситуации, показанной на рисунке 6, вариант диалога, когда обучаемый верно ответил на вопрос чат-бота, может выглядеть, как показано на рисунке 8. В этом случае ИТ оценивает данный ответ на наводящий вопрос как верный и сообщает об этом. Наводящие вопросы и ответы на них генерируются на основе соответствующих узлов дерева мыслительного процесса (первый вариант ответа на рисунке 8 соответствует узлу с проверкой ассоциативности на рисунке 4 и т.д.).

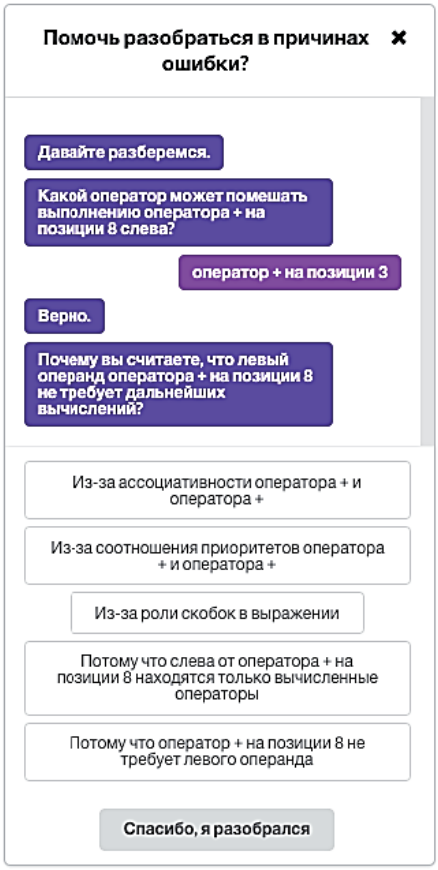


Рисунок 8 – Пример верного ответа на наводящий вопрос

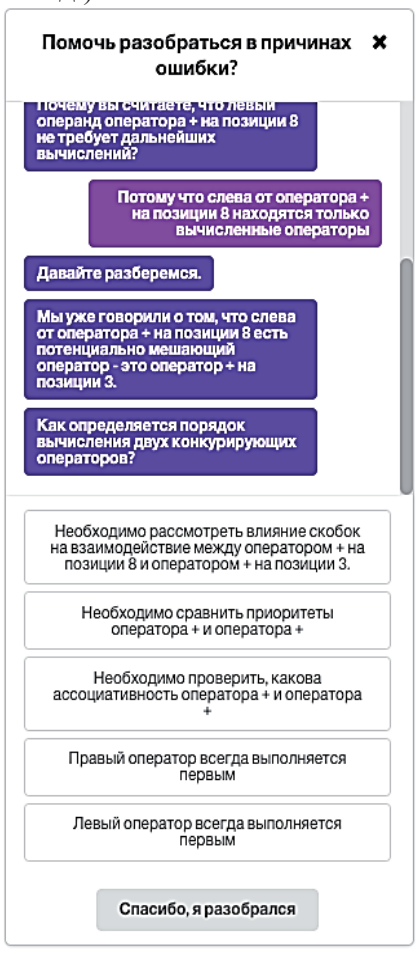


Рисунок 9 – Пример неверного ответа на наводящий вопрос

Если обучаемый неверно ответил на наводящий вопрос, ИТ распознает это и генерирует следующий наводящий вопрос в зависимости от совершённой ошибки (см. рисунок 9). Предложенные ИТ варианты ответов соответствуют новому вопросу.

Разработанный программный код связывает интерфейс пользователя с ОМ. ИТ представляет собой веб-приложение, базирующееся на веб-фреймворке *Ruby On Rails*, а интерфейс разработан с использованием библиотеки *Semantic UI*. Разработанное приложение доступно в сети Интернет. Клиент обменивается с сервером данными в формате *JSON* посредством *AJAX-запросов*. Созданные упражнения и статистика их решений хранятся на сервере в БД *PostgreSQL*.

3 Оценка эффективности разработанного ИТ

Для оценки эффективности разработанного ИТ при изучении программирования проведён эксперимент среди бакалавров и магистров факультета информатики и вычислительной техники Волгоградского государственного технического университета. В эксперименте приняли участие 30 студентов. Участникам было предложено самостоятельно воспользоваться обучающим тестом и ИТ в любом порядке. Обучающий тест содержал 12 вопросов с различными по сложности выражениями на языке *C++*, где необходимо было выбрать следующий невыполненный оператор из списка предложенных. Для сравнения с обучающим тестом были предоставлены десять упражнений в ИТ, содержащих выражение на языке *C++*.

Первые пять задач в ИТ простые:

(a + b) * c + d (требуется только знание приоритетов операций),
(a + b) * c * d (требуется знание приоритетов и ассоциативности операций),
a = b = 0 (вводит правоассоциативную операцию),
a < 0 && b > 0 (операция логического И имеет строгий порядок вычисления операндов),
a < 0 ? b = 0 : c = 0 (тернарная условная операция).

Следующие пять выражений сложные:

* ++ a - b (вводит унарный оператор разыменования указателя и префиксный инкремент),
a ++ || b > 0 && c -- || d < c (требуется знать порядок вычисления логических операторов),
a = & b -> c (требуется знать приоритеты операторов обращение по адресу и взятия адреса),
! * a ++ (требуется знать порядок вычисления префиксных и постфиксных унарных операторов),
a > b || c < d && (e - f > g || h > e - f) (требуется знать порядок вычисления логических операторов).

Студенты успешно выполнили все упражнения без помощи преподавателя. После выполнения упражнений студентам было предложено заполнить анкету. Большинство из участников отметили разработанный ИТ более полезным (см. рисунок 10) и хотели бы использовать аналогичные инструменты при изучении других тем. К положительным сторонам разработанного ИТ студенты отнесли возможность получить поясняющую обратную связь сразу после совершения ошибки. Основными пожеланиями участников эксперимента было добавление в ИТ теоретического материала по изучаемым понятиям ПрО.

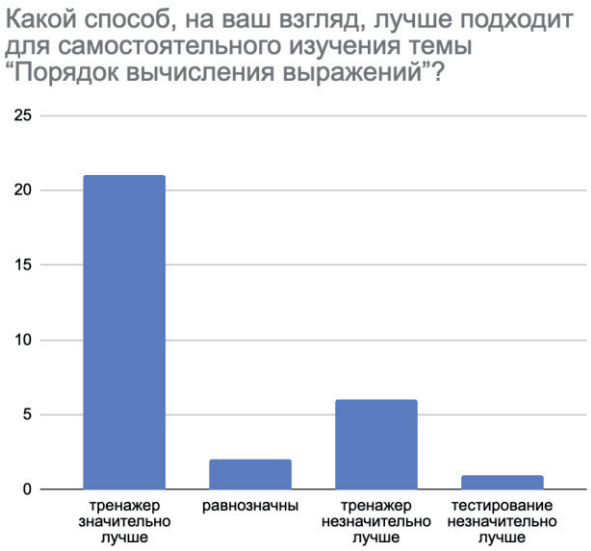


Рисунок 10 – Отзывы обучающихся о работе с интеллектуальным тренажёром

Заключение

Разработанный ИТ может быть полезен изучающим программирование. В текущей версии ИТ поддерживает освоение выражений на языках программирования *C++, C#* и *Python*. Достоинством ИТ является анализ каждого шага решения задачи и генерация объяснений совершённых обучаемыми ошибок; доступны подсказки следующего правильного шага, если обучаемый находится в затруднении.

Метод построения ИТ на основе онтологии ПрО позволяет разработать тренажёр, который автоматически генерирует поясняющую обратную связь обучаемому в зависимости от правил ПрО, нарушенных в процессе решения задачи. Применение деревьев мыслительного процесса при разработке ИТ позволило добавить возможность задавать обучаемому наводящие вопросы, когда он совершил ошибку. Система наводящих вопросов и переходов между

ними генерируется из ОМ решения задачи, что освобождает разработчиков тренажёра от ручной работы по составлению вопросов.

Изложенный в статье метод может быть использован для разработки ИТ для различных ПрО. Помимо описанного в статье разработаны тренажёры по изучению таких тем, как управляющие операторы, выражения доступа к данным и области видимости объектов.

Авторский вклад

Сычёв О.А. - концептуализация, управление проектом, проектирование и тестирование ПО, валидация данных, редактирование текста статьи. Пенской Н.А. - разработка ПО, сбор и верификация данных, редактирование текста статьи. Терехов Г.В. - разработка и тестирование ПО, сбор данных, визуализация, написание исходного текста статьи.

Список источников

[1] ***Papadakis S., Kalogiannakis M., Zaranis N.*** Developing funda-mental programming concepts and computational thinking with scratchjr in preschool education: a case study. *International Journal of Mobile Learning and Organisation*. 2016; 10(3): 187–202. DOI:10.1504/IJMLO.2016.077867.

[2] ***Rahmat M., Shahrani S., Latih R.*** Major Problems in Basic Programming that Influence Student Performance. *Procedia - Social and Behavioral Sciences*. 2012; 59: 287–296. DOI:10.1016/j.sbspro.2012.09.277.

[3] ***Basu S., Biswas G., Sengupta P., Dickes A., Kinnebrew J.S., Clark D.*** Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*. 2016; 11(1): 1-35. DOI:10.1186/s41039-016-0036-2.

[4] ***Ramabu T., Sanders I., Schoeman M.*** Manipulatives for Teaching Introductory Programming to Struggling Students: A Case of Nested-decisions. *In Proceedings of the 13th International Conference on Computer Supported Education* (CSEDU 2021). 2021; 1: 505–510. DOI: 10.5220/0010477505050510.

[5] ***Pierrakeas C., Xenos M., Panagiotakopoulos C., Vergidis D.*** A comparative study of dropout rates and causes for two different distance education courses. *International Review of Research in Open and Distance Learning*. 2004; 5(2): 1-15. DOI: 10.19173/irrodl.v5i2.183.

[6] ***Хаблиева С.Р., Каргиева З.К.*** Анализ мобильных технологий для организации учебного процесса школ. *Вестник Северо-Осетинского государственного университета имени К. Л. Хетагурова*. 2022; (4): 190-197.

[7] ***Соловов А.В.*** Электронное обучение: проблематика, дидактика, технология. Самара: «Новая техника», 2006. 462 с.

[8] ***Mitrovic A., Koedinger K.R., Martin B.*** A comparative analysis of cognitive tutoring and constraint-based modeling. *Lecture Notes in Artificial Intelligence* (Subseries of Lecture Notes in Computer Science). 2003;2702: 313–322. DOI: 10.1007/3-540-44963-9_42.

[9] ***Aleven V., Koedinger K.*** An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*. 2002: 26(2): 147–179. DOI: 10.1207/s15516709cog2602_1.

[10] ***Sorva J., L¨onnberg J., Malmi L.*** Students' ways of experiencing visual program simulation. *Computer Science Education*. 2013; 23(3): 207–238. DOI:10.1080/08993408.2013.807962.

[11] ***Kollmansberger S.*** Helping students build a mental model of computation. In Proceedings of the Fifteenth Annual *Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10. 2010;15:128–131. DOI: 10.1145/1822090.1822127.

[12] ***Levy R., Ben-Ari M., Uronen P.*** The jeliot 2000 program animation system. *Computers & Education*. 2003; 40(1): 1–15. DOI: 10.1016/S0360-1315(02)00076-3.

[13] ***Сычёв О.А., Стрельцов В.О.*** Использование шаблонов в виде регулярных выражений в тренировочных и контрольных тестовых вопросах с открытым ответом. *Открытое образование*. 2015; 2(109): 38-45. DOI: 10.21686/1818-4243-2015-2(109-38-45.

[14] ***Углев В.А., Сычёв О.А., Аникин А.В.*** Интеллектуальный анализ цифрового следа при оценке контрольно-измерительных материалов для поддержки принятия решений в образовательном процессе. *Журнал Сибирского федерального университета. Серия: Техника и технологии*. 2022; 15(1): 121-136. DOI: 10.17516/1999-494X-0378.

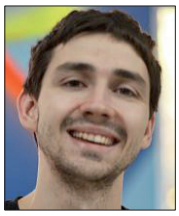
[15] ***Горисев С.А., Койнов А.В., Куземчик В.Д.*** Интеллектуальный лингвопроцессорный комплекс "КЛИОС" для обучения РКИ. *Современные проблемы науки и образования*. 2013; 6: 23-31.

[16] ***Рыбина Г.В.*** Интеллектуальная технология построения обучающих интегрированных экспертных систем: новые возможности. *Открытое образование*. 2017; 21(4): 43-57. DOI: 10.21686/1818-4243-2017-4-43-57.

[17] ***Hosseini R., Brusilovsky P.*** Javaparser: A fine-grain concept indexing tool for java problems. In Workshops Proceedings of AIED. 2013; 1009: 60–63.
[18] ***McBride B.*** Jena: a semantic web toolkit. *IEEE Internet Computing*. 2002: 6(6): 55–59. DOI: 10.1109/MIC.2002.1067737.
[19] ***Крыгин А.И., Гумеров М.Р., Москаленко Н.А., Сычёв О.А.*** Фреймворк для разработки интеллектуальных обучающих систем на основе моделей предметных областей в виде деревьев решений. *Двадцать первая Национальная конференция по искусственному интеллекту с международным участием* (КИИ-2023). 2023; Т.2: 206-217.
[20] ***Хотелов Д.А., Радыгин В.Ю., Меркушева А.С.*** Разработка системы мониторинга безопасности для кластера информационных систем, базирующихся на платформе RUBY ON RAILS. *Безопасность информационных технологий*. 2018; 25(3): 88-100. DOI: 10.26583/bit.2018.3.09.
[21] ***Мизюков Г.С.*** Проектирование LC/NC платформы на базе фреймворка Laravel. *Инженерный вестник Дона*. 2022; 11(95): 200-207.
[22] ***Sychev O.A., Penskoy N.A., Terekhov G.V***. A Tool to Teach Expressions with Feedback About Broken Laws // SIGCSE 2022 : Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (Providence, RI, USA, March 3-5, 2022) Association for Computing Machinery (ACM). [USA], 2022. Vol.2. P.1158. DOI: 10.1145/3478432.3499082.

Сведения об авторах

Сычёв Олег Александрович, 1979 г. рождения. Окончил Волгоградский государственный университет (ВолГУ) в 2002 г., к.т.н. (2005), доцент кафедры программного обеспечения автоматизированных систем ВолгГТУ. В списке трудов около 200 работ в области электронных образовательных систем и искусственного интеллекта. ORCID: 0000-0002-7296-2538; Author ID (РИНЦ): 447111; Author ID (Scopus): 57203852158; Researcher ID (WoS): M-4897-2015. *oasychev@gmail.com* ✉.



Пенской Никита Андреевич, 1995 г. рождения. Окончил ВолГУ в 2018 г. Аспирант кафедры программного обеспечения автоматизированных систем ВолгГТУ. В списке трудов около 10 работ в области интеллектуальных обучающих систем. ORCID: 0000-0002-4443-3399. *nik95penik@yandex.ru.*

Терехов Григорий Владимирович, 1993 г. рождения. Окончил ВолГУ в 2016 г. Старший преподаватель кафедры программного обеспечения автоматизированных систем ВолгГТУ. В списке трудов более 30 работ в области электронных образовательных систем. ORCID: 0000-0002-0289-1834; Author ID (РИНЦ): 1050906; Author ID (Scopus): 57224992574; Researcher ID (WoS): GWC-8187-2022. *grvlter@gmail.com.*

Поступила в редакцию 09.02.2024, после рецензирования 2.12.2024. Принята к публикации 17.12.2024.

Scientific article DOI: 10.18287/2223-9537-2025-15-1-67-81

A method for developing intelligent simulators based on ontology of the subject domain

© 2025, O.A. Sychev ✉, N.A. Penskoy, G.V. Terekhov

Volgograd State Technical University (VolSU), Volgograd, Russia

Abstract

In academic disciplines, students are required to learn many new concepts, which necessitates extensive training with feedback. An intelligent simulator enables students to practice solving simple problems while receiving explanations for their mistakes, allowing teachers to focus on addressing more complex issues during lessons. This paper presents a method for developing intelligent simulators based on the ontology of the subject domain, implemented as web applica-

tions suitable for both classroom and extracurricular use. Representing the problem and the subject area model in RDF format enables logical inference using the Apache Jena Reasoner inference engine. An example is provided of an intelligent simulator designed for learning the order of operations in expressions, supporting the C++, C#, and Python programming languages. The simulator can explain errors, generate explanatory hints, and engage in educational dialogue through guiding questions. The simulator was tested with undergraduate and graduate students of the Faculty of Electronics and Computer Engineering at Volgograd State Technical University. Most students found the simulator to be more useful than traditional training tests. It can be employed both for independent study and as part of the educational process in classroom settings.

Keywords: *intelligent simulator, training, domain ontology, order of expression evaluation, introductory courses, programming.*

For citation: *Sychev O.A., Penskoy N.A., Terekhov G.V.* A method for developing intelligent simulators based on ontology of the subject domain [In Russian]. *Ontology of designing*. 2025; 15(1): 67-81. DOI:10.18287/2223-9537-2025-15-1-67-81.

Conflict of interest: The authors declare no conflict of interest.

List of figures

References

[1] ***Papadakis S, Kalogiannakis M, Zaranis N.*** Developing funda-mental programming concepts and computational thinking with scratchjr in preschool education: a case study. *International Journal of Mobile Learning and Organisation*. 2016; 10(3): 187–202. DOI:10.1504/IJMLO.2016.077867.

[2] ***Rahmat M, Shahrani S, Latih R.*** Major Problems in Basic Programming that Influence Student Performance. *Procedia - Social and Behavioral Sciences*. 2012; 59: 287–296. DOI:10.1016/j.sbspro.2012.09.277.

[3] ***Basu S, Biswas G, Sengupta P, Dickes A, Kinnebrew JS, Clark D.*** Identifying middle school students' challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*. 2016; 11(1): 1-35. DOI:10.1186/s41039-016-0036-2.

[4] ***Ramabu T, Sanders I, Schoeman M.*** Manipulatives for Teaching Introductory Programming to Struggling Students: A Case of Nested-decisions. *In Proceedings of the 13th International Conference on Computer Supported Education* (CSEDU 2021). 2021; 1: 505–510. DOI: 10.5220/0010477505050510.

[5] ***Pierrakeas C, Xenos M, Panagiotakopoulos C, Vergidis D.*** A comparative study of dropout rates and causes for two different distance education courses. *International Review of Research in Open and Distance Learning*. 2004; 5(2): 1-15. DOI: https://doi.org/10.19173/irrodl.v5i2.183.

[6] ***Khablieva SR, Kargieva ZK.*** Analysis of mobile technologies for organizing the educational process of schools [In Russian]. *Vestnik of North Ossetian State University named after K.L. Khetagurov*. 2022; (4): 190-197.

[7] ***Solovov AV.*** E-learning: issues, didactics, technology [In Russian]. Samara: New technology. 2006. 462 p.

[8] ***Mitrovic A, Koedinger KR, Martin B.*** A comparative analysis of cognitive tutoring and constraint-based modeling. *Lecture Notes in Artificial Intelligence* (Subseries of Lecture Notes in Computer Science). 2003: 2702: 313–322. DOI: https://doi.org/10.1007/3-540-44963-9_42.

[9] ***Aleven V, Koedinger K.*** An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*. 2002: 26(2): 147–179. DOI: https://doi.org/10.1207/s15516709cog2602_1.

[10] ***Sorva J, L¨onnberg J, Malmi L.*** Students' ways of experiencing visual program simulation. *Computer Science Education*. 2013; 23(3): 207–238. DOI:10.1080/08993408.2013.807962.
[11] ***Kollmansberger S.*** Helping students build a mental model of computation. *In Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10. 2010; 15: 128–131. DOI: https://doi.org/10.1145/1822090.1822127.
[12] ***Levy R, Ben-Ari M, Uronen P.*** The jeliot 2000 program animation system. *Computers & Education*. 2003; 40(1): 1–15. DOI: https://doi.org/10.1016/S0360-1315(02)00076-3.
[13] ***Sychev OA, Streltsov VO.*** Use of regular expressions as templates in formative and summative open answer questions [In Russian]. *Otkrytoe obrazovanie*. 2015; 2(109): 38-45. DOI: https://doi.org/10.21686/1818-4243-2015-2(109-38-45.
[14] ***Uglev VA, Sychev OA, Anikin AV.*** Intelligent analysis of digital footprint during assessment of control and measuring materials to support decision-making during learning process [In Russian]. *Journal of Siberian Federal University Engineering & Technologies*. 2022; 15(1): 121-136. DOI: 10.17516/1999-494X-0378.
[15] ***Gorisev SA, Koynov AV, Kuzemchik VD.*** Intelligent tutoring system "KLIOS" based on the linguistic processor for studing russian as a foreign language [In Russian]. *Modern Problems of Science and Education*. 2013; (6): 23-31.
[16] ***Rybina GV.*** Intelligent technology for construction of tutoring integrated expert systems: new aspects [In Russian]. *Otkrytoe obrazovanie*. 2017; 21(4): 43-57. DOI: https://doi.org/10.21686/1818-4243-2017-4-43-57.
[17] ***Hosseini R, Brusilovsky P.*** Javaparser: A fine-grain concept indexing tool for java problems. *In Workshops Proceedings of* AIED. 2013; 1009: 60–63.
[18] ***McBride B.*** Jena: a semantic web toolkit. IEEE Internet Computing. 2002; 6(6): 55–59. DOI: 10.1109/MIC.2002.1067737.
[19] ***Krygin AI, Gumerov MR, Moskalenko NA, Sychev OA.*** A framework for developing intelligent tutoring systems based on domain models in the form of decision trees [In Russian]. *XXI Russian conference on artificial intelligence* (RCAI-2023). 2023; 1: 206-217.
[20] ***Khotelov DA, Radygin VY, Merkusheva AS.*** Development of the security monitoring system for cluster of information systems based on the ruby on rails framework [In Russian]. *Bezopasnost` Informatsionnykh Tekhnologiy*. 2018; 25(3): 88-100. DOI: http://dx.doi.org/10.26583/bit.2018.3.09.
[21] ***Miziukov GS.*** Designing an LC/NC platform based on the Laravel framework [In Russian]. *Engineering journal of Don*. 2022; 11(95): 200-207.
[22] ***Sychev OA, Penskoy NA, Terekhov GV***. A Tool to Teach Expressions with Feedback About Broken Laws // SIGCSE 2022 : Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (Providence, RI, USA, March 3-5, 2022) Association for Computing Machinery (ACM). [USA], 2022. Vol.2. P.1158. DOI: 10.1145/3478432.3499082.

About the authors

Oleg Aleksandrovich Sychev (b. 1979), graduated from the VolSU in 2002, Candidate of Technical Sciences (Engineering) (2005), Associate Professor at Software Engineering Department, VolSU. Author of more than 200 scientific papers in the field of learning management systems and AI. ORCID: 0000-0002-7296-2538; Author ID (RSCI): 447111; Author ID (Scopus): 57203852158; Researcher ID (WoS): M-4897-2015. *oasychev@gmail.com.* ✉.

Nikita Andreevich Penskoy (b. 1995) graduated from the VolSU in 2018, a postgraduate student of the Department of Software Automated Systems, VolSU. He is a co-author of about 10 scientific papers in the field of intelligent tutoring systems. ORCID: 0000-0002-4443-3399. *nik95penik@yandex.ru.*

Grigory Vladimirovich Terekhov (b. 1993) graduated from the VolSU in 2016, a postgraduate student of the Department of Software Automated Systems, VolSU. Author of more than 30 scientific papers in the field of learning management systems. ORCID: 0000-0002-0289-1834; Author ID (RSCI): 1050906; Author ID (Scopus): 57224992574; Researcher ID (WoS): GWC-8187-2022. *grvlter@gmail.com.*

Received February 9, 2024. Revised December 2, 2024. Accepted December 17, 2024.