Моделирование и анализ данных
2021. Том 11. № 1. С. 78–93
DOI: https://doi.org/10.17759/mda.2021110106
ISSN: 2219-3758 (печатный)
ISSN: 2311-9454 (online)

Modelling and Data Analysis
2021. Vol. 11, no. 1, pp. 78–93
DOI: https://doi.org/10.17759/mda.2021110106
ISSN: 2219-3758 (print)
ISSN: 2311-9454 (online)

УДК 371.3

Применение цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы

Попков С.И.*
Московский государственный психолого-педагогический университет,
г. Москва, Российская Федерация
ORCID: https://orcid.org/0000-0003-2566-1262
e-mail: rslw25@gmail.com

Проведено исследование способов применения цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы. Продемонстрирован интерфейс рабочей среды и основные компоненты, обеспечивающие адаптивность платформы. Детально описана работа служб платформы, отвечающих за функции информационной безопасности. Приведен пример предъявляемого задания.

Ключевые слова: адаптивность, цифровая платформа, sandbox, обучение программированию.

Для цитаты:
Попков С.И. Применение цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы // Моделирование и анализ данных. 2021. Том 11. № 1. С. 78–93. DOI: https://doi.org/10.17759/mda.2021110106

1. ВВЕДЕНИЕ

В статье «Особенности программной реализации цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы» [1] была представлена концепция, позволяющая обучать программированию учащихся разного уровня, в частности, студентов высших учебных заведений. Были исследованы аналоги, выявлены их недостатки и сформулированы критерии, которым должна удовлетворять цифровая платформа для преодоления

***Попков Сергей Игоревич,** кандидат физико-математических наук, доцент факультета информационных технологий Московского государственного психолого-педагогического университета, г. Москва, Российская Федерация, ORCID: https://orcid.org/0000-0003-2566-1262, e-mail: rslw25@gmail.com

этих недостатков. Было обосновано применение платформы в качестве инструмента для психолого-педагогических исследований в специфической востребованной области, связанной с обучением информационным технологиям и формированием навыков командной работы и взаимодействия обучающихся в группе. Материал упомянутой статьи позволяет ознакомиться с проблематикой, построить архитектуру цифровой платформы и проверить ее по составленным критериям на соответствие заявленной в работе концепции.

Однако, детали, связанные с подходом к обучению платформы, равно как и примеры реализации предъявляемого задания, а также демонстрация способов реализации принципов информационной безопасности в ходе работы платформы не была продемонстрирована в упомянутой выше статье.

Материал этой статьи призван восполнить этот недостаток и представить более полное изложение существенных деталей работы платформы и характер взаимодействия с обучающимся пользователем (студентом) в рамках поставленных выше задач.

Хотя скрупулезное описание конкретных деталей выбранного решения для построения цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы с точки зрения безопасности и обеспечения адаптивности (например, описание конкретной модели машинного обучения, используемой в процессе генерации адаптивного сценария, или конкретные версии операционных систем и других приложений и модулей, составляющих окружение клиент-серверной архитектуры) и выходит за рамки материала данной статьи, для полноты изложения и ответа на основные вопросы о принципах работы платформы необходимо в общих чертах, на уровне обобщенных алгоритмов и последовательности выполняемых шагов, описать структуры и процессы, отвечающие за информационную безопасность, за процедуру взаимодействия пользователя с рабочей средой и за механизмы адаптивности платформы.

Необходимо также предоставить информацию о деталях такого взаимодействия в виде снимков экрана в ходе исследования рабочей среды, деталей и особенностей платформы в аспекте предоставляемых пользователю интерфейсов для изменения состояний рабочей среды с целью выполнения индивидуальных заданий или части коллективного задания, описать предметный характер этих заданий, чтобы было понятно, о чем идет речь и как именно формируется единое задание, предъявляемое в качестве окружения рабочей среды.

Перейдем непосредственно к вопросам, связанным с интерфейсами среды на стороне клиента.

2. ОБЗОР ИНТЕРФЕЙСА ГРАФИЧЕСКОЙ СРЕДЫ АДАПТИВНОЙ ПЛАТФОРМЫ НА СТОРОНЕ КЛИЕНТА

Возможный вид графической среды после авторизации студента (обучающегося) продемонстрирован на рис. 1.

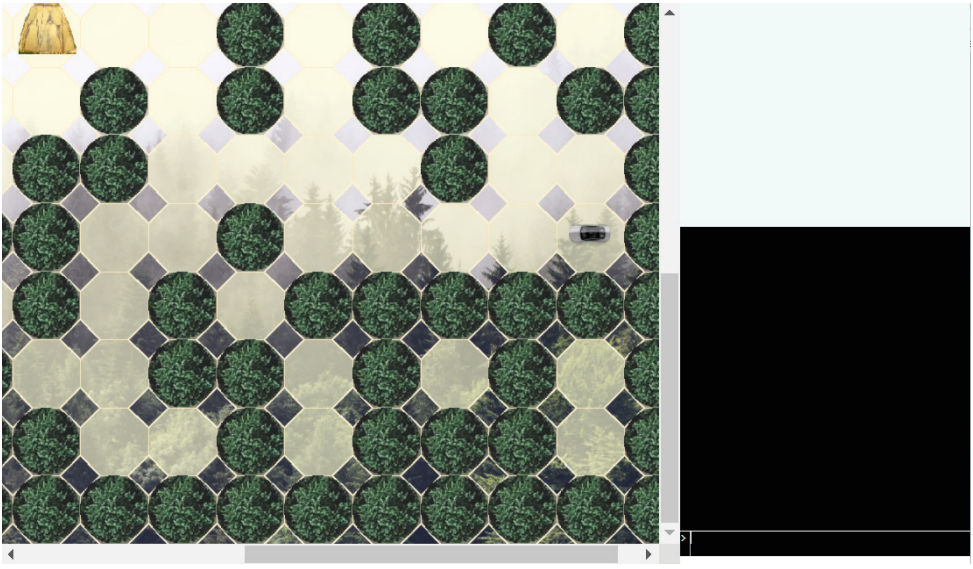

Рис. 1. Графический интерфейс для пользователя платформы (обучающегося) на стороне клиента

Слева располагается окно рабочей среды или карта лабиринта, указывающая местоположение агента (машины), которым управляет студент, в двумерном пространстве. Фон лабиринта, изображение агента и отображение элементов, блокирующих движение агента, зависит от деталей общего сценария, определяющих декоративные элементы. В данном случае в качестве фона служит лес, в качестве элементов, блокирующих движение агента – деревья.

Справа вверху располагается пространство информационной справки. Справка активируется при прохождении индивидуальных заданий для того, чтобы объяснять суть поставленной задачи, как с точки зрения сценария, так и с точки зрения применяемых технологий программирования. Также пространство информационной справки, по запросу студента, может содержать подсказки, если таковые имеются для текущего индивидуального задания.

Справа внизу располагается консоль, через которую осуществляются все функции по взаимодействию с рабочей средой, в том числе, перемещение агента по лабиринту, загрузка решений для индивидуальных и коллективных заданий, получение сведений о мире и статусе других участников и т.п. Взаимодействие осуществляется путем ввода набора команд, выполнение которых приводит к желаемым результатам.

Следует отметить, что «команды» берутся на основе функций и операторов из реально существующего языка программирования (в данной реализации – языка JavaScript), при этом функционал языка не урезается – в полной мере для взаимодействия с окружением рабочей среды и управлением агентом допускаются все стандартные конструкции применяемого языка, такие как циклы, условия, элементы объектно-ориентированной парадигмы и прочие возможности. Гипотетически,

это повышает риски с точки зрения информационной безопасности системы, однако в составе платформы реализованы специальные методы, предназначенные для защиты контекста рабочей среды и предотвращения использования потенциальной бреши в безопасности системы (например, попытки обойти предполагаемое решение, получить прямой доступ к инкапсулируемым атрибутам агента, модифицировать состояния лабиринта или иным образом нарушить целостность системы). Эти специальные методы будут рассмотрены далее в этой статье.

Помимо элементов, предусмотренных сценарием, на карте лабиринта отображаются элементы, связанные с прохождением индивидуальных заданий, а также в ячейках поля лабиринта могут находиться невидимые элементы, усложняющие решение общей групповой задачи для формирования навыков командной работы.

Групповое задание представляет собой построение маршрута в двумерном пространстве, обусловленном параметрами рабочей среды; маршрут должен быть построен таким образом, чтобы соединять поэтапно решенные элементы индивидуальных заданий – в контексте игры, данные передаются от одного элемента к другому, а решение индивидуальных заданий активирует элементы, то есть позволяет генерировать передаваемые данные). Групповое решение объединяет результаты работы всех участников и реализуется сообща всей группой.

К элементам, усложняющим решение группового задания (а также позволяющим преодолевать усложняющие факторы), относятся:

1) «Стиратель» – действует неограниченно, уничтожает данные, полученные в ходе посещения элемента индивидуального задания при построении маршрута.

   Возможное алгоритмическое решение: проверять переносимые данные на неопределенность, в случае обнаружения утери данных запомнить клетку, в которой это произошло, в дальнейшем обходить. Вернуться к последнему посещенному элементу с индивидуальным заданием, получить данные заново.
2) «Жук» – действует один раз, случайным образом перемешивает значения команд (Например, команда «car.left()», перемещающая агента влево, может начать перемещать его вверх).

   Возможное алгоритмическое решение: последовательно проверять команды на наличие противоречий, в случае обнаружения противоречий выполнять команды последовательно и по реакции среды определять их новое назначение.
3) «Искажатель» – действует один раз, перемешивает элементы индивидуальных заданий на карте.

   Возможное алгоритмическое решение: посетить все элементы индивидуальных заданий и по возвращаемым данным сопоставить ожидаемые значения с новым местоположением.
4) «Телепорт» – действует неограниченно, перемещает агента в произвольное свободное место в двумерном пространстве рабочей среды.

   Возможное алгоритмическое решение: проверять текущие координаты в соответствии с предпринятым ранее действием; если обнаружено несоответствие новых координат ожидаемым («ожидаемой» ситуацией с точки зрения агента является

корректное перемещение или случай неудачной попытки перемещения, когда агент остается на месте), то запомнить клетку, в которой это произошло, в дальнейшем обходить. Вернуться к последней посещенной ячейке и передать управление основному алгоритму.

5) «Мина» – действует один раз, приводит агента в неработоспособное состояние.

   Возможное алгоритмическое решение: необходимо использовать «Защитника» (см. ниже).

6) «Защитник» – программируемый внешний агент, способный выполнять задания и передавать данные о своем состоянии, в частности, координаты в двумерном пространстве, характеристики работоспособности, хранимые данные, полученные от элемента индивидуального задания. Может использоваться как дополнительный помощник по преодолению влияния элементов, усложняющих решение (например, может «обезвредить мину», пожертвовав собой), обычно появляется в сложных сценариях.

Разберем элементы индивидуального задания на примере. На рис. 2 на полностью видимой ячейке слева изображен подобный элемент – палатка.

Рис. 2. Пример элемента индивидуального задания

Для осуществления взаимодействия с этим элементом агент должен переместиться в ту же ячейку, где находится в данный момент элемент. Процедура перемещения осуществляется с помощью консольных команд. Несмотря на то, что агент может находиться в единицу времени только в середине одной ячейки, анимация переходов отображает перемещение агента плавно. Благодаря внедренной в интерфейс технологии SVG [3], разработанной Консорциумом World Wide Web (W3C) [5], существует возможность масштабирования без значительной потери качества для векторных

элементов, а также предоставляется опция вертикальной и горизонтальной прокрутки окна рабочей среды с помощью элемента «колесо» манипулятора «мышь» (рис. 3).

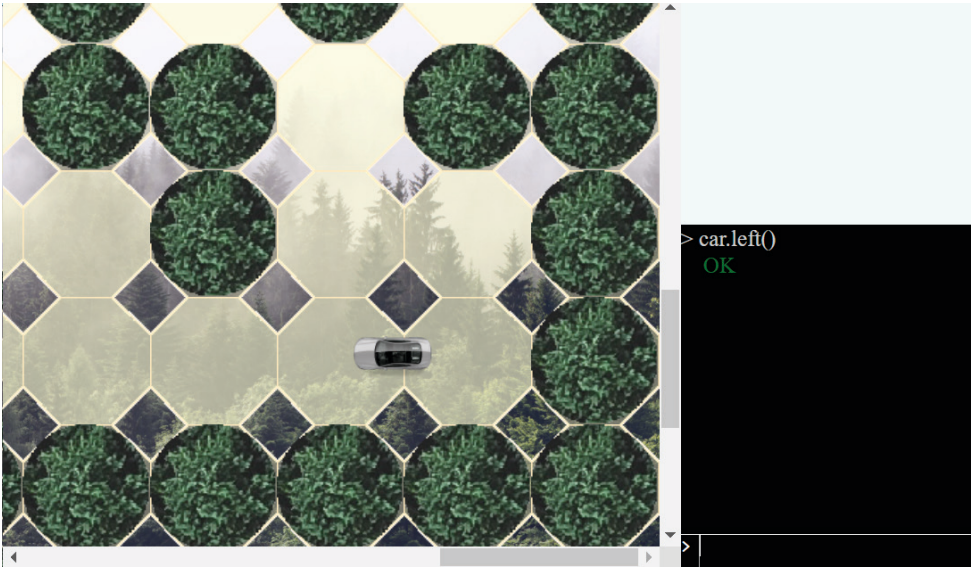

Рис. 3. Перемещение агента в двумерном пространстве рабочей среды в ходе выполнения консольной команды

Пример изменения графической среды на стороне клиента после осуществления взаимодействия агента с элементом показан на рис. 4.

Окно карты заменяется на изображение, демонстрирующее текущую стадию решения задачи для посещенного элемента. Текст слева вверху отражает суть задания, при этом задание генерируется с учетом индивидуальных особенностей студента, в зависимости от уровня его подготовки и допущенных ошибок в ходе решения предыдущих задач при взаимодействии с платформой. Консоль дает подсказку о команде для возврата в обычный режим.

Если студент выбирает это задание в качестве индивидуального (в данном случае – собрать дрова), он пишет файл с кодом, который позволяет решить задачу для произвольного набора входных данных (не только для предложенного в качестве примера в задаче). После этого код отправляется на сервер с помощью консоли, где проверяется на корректность. Если код признан верным, индивидуальное задание может поменяться до тех пор, пока не будет предложен последний этап решения (например, задача «собрать дрова» может быть заменена на «развести костер», и только после этого этапа индивидуальное задание будет считаться решенным в контексте игры).

Пример финального экрана, демонстрирующего завершение выполнения индивидуального задания, показан на рис. 5.

Выполнение индивидуального задания требует знаний в изучаемой области, а результат решения задания с точки зрения сценария приносит пользу группе, делая возможным решение группового задания.

Рис. 4. Последствия взаимодействия агента с элементом (палаткой)

Так, элемент «костер» может служить в связке с другими элементами, такими как «кузница» (разжечь кузнечные меха), «кухня» (приготовить провизию для похода), «воздушный шар» (обеспечить взлет) и т.д., в зависимости от сценария и совокупности индивидуально подобранных заданий. Например, построение маршрута от элемента «палатка», преобразованного корректным решением задачи в элемент «костер», до элемента «кузница» обеспечивает передачу данных между элементами, в семантическом контексте – обеспечение доставки огня в кузницу.

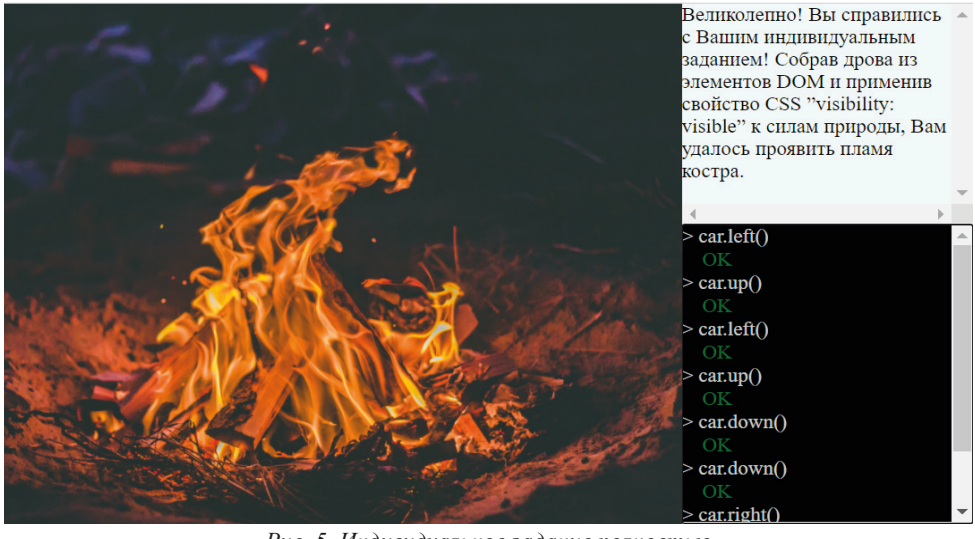

Рис. 5. Индивидуальное задание полностью решено и готово к отправке данных для группового задания

3. ОБЩАЯ СХЕМА КОМПОНЕНТОВ, ОБЕСПЕЧИВАЮЩИХ АДАПТИВНОСТЬ ПЛАТФОРМЫ

Схема, раскрывающая общий принцип адаптивности, показана на рис. 6.

Обычные стрелки указывают на вызов блока (в направлении вызова), номера стрелок указывают последовательность вызовов между блоками (блок, из которого исходит стрелка с номером 1, является входной точкой, то есть с него начинается работа алгоритма, обеспечивающего адаптивность платформы). Пунктирные линии со стрелками указывают на вызов блока из другого процесса, пунктирные линии с точками указывают на взаимодействие блока с хранилищем, точка по направлению к блоку указывает на операцию чтения, точка по направлению к хранилищу – на операцию записи.

Несмотря на распределенный характер адаптивной платформы и большое количество серверов и баз данных разного назначения, а также компьютеров в локальной сети, описанная схема позволяет абстрагироваться от программно-аппаратных особенностей реализации платформы и сосредоточиться на взаимосвязях и логике компонентов. С этой точки зрения, существуют 2 взаимодействующих между собой процесса.

Процесс адаптивной модификации задания, который объединяет между собой блоки «Загрузка адаптивного тестирования» и «Подготовка динамических элементов» и осуществляет загрузку и подбор динамических компонентов задания для формирования единого задания, предъявляемого участникам группы в рабочей среде.

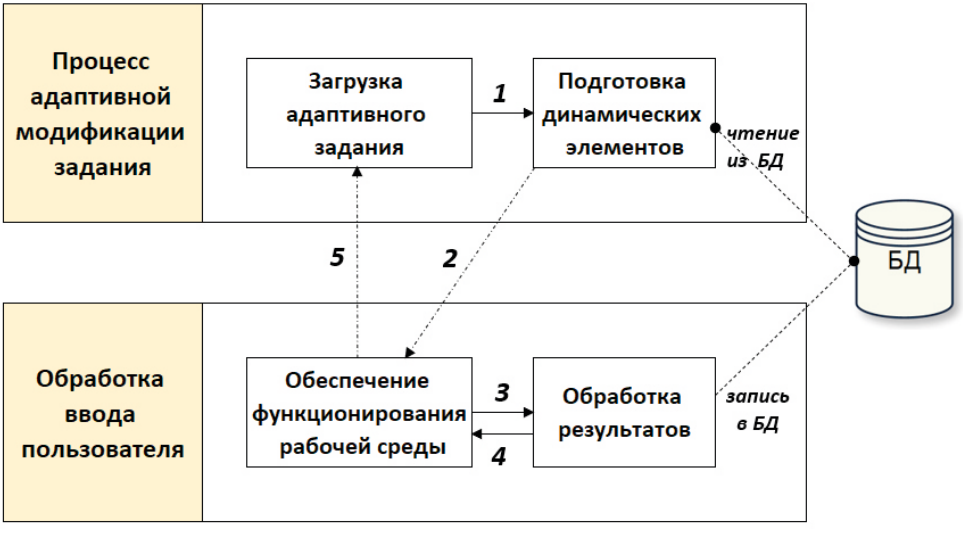

Рис. 6. Общая схема принципа адаптивности платформы

Процесс обработки ввода пользователя, который объединяет между собой блоки «Обеспечение функционирования рабочей среды» и «Обработка результатов» и осуществляет меры по обеспечению безопасного ввода, проверку корректности решений,

предлагаемых пользователем и учет его уровня навыков, а также реализацию процедуры машинного обучения для подготовки новых заданий для участников группы.

Подробное представление схемы, описывающей принцип адаптивности, применяемый в ходе работы цифровой адаптивной платформы, занимает слишком внушительный размер, чтобы ее можно было отобразить в одном наглядном изображении без сопутствующих проблем масштабирования.

Каждый из представленных четырех блоков будет раскрыт в нотации BPMN далее. Схема компонентов показана в максимально общем виде, без раскрытия конкретной модели машинного обучения, значений гиперпараметров, особенностей процедуры оптимизации и прочих технических деталей, выходящих за рамки материала данной статьи. Основное внимание уделено общей процедуре, обеспечивающей взаимодействие компонентов, влияющих на обучение модели и определение уровня навыков пользователей с компонентами, отвечающими за безопасное функционирование рабочей среды и проверку данных, вводимых пользователем.

Структура блока «Загрузка адаптивного задания» приведена на рис. 7.

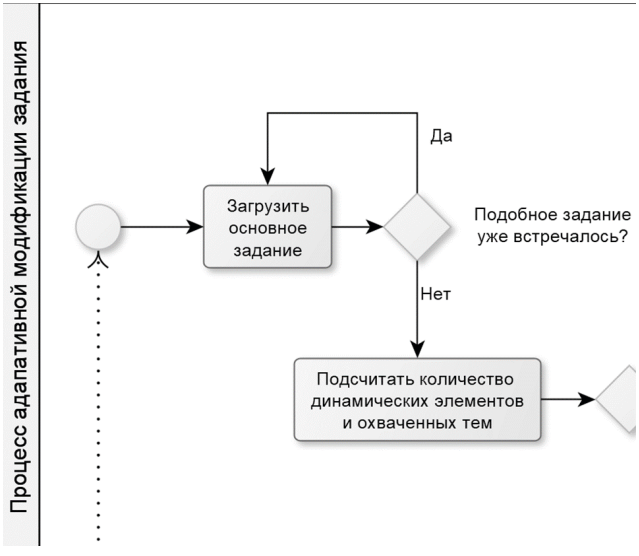

Рис. 7. Структура блока «Загрузка адаптивного задания»

Процесс адаптивной модификации задания начинается с загрузки основного задания, задающего детали сценария и декоративных элементов для всей группы. Обеспечивается уникальность предъявляемого задания (подразумевается, что банк достаточно большой, чтобы количество формулировок и скриптов для различных заданий и сценариев во много раз превышало количество возможных занятий студентов с платформой – то есть число проводимых пар с участием цифровой платформы).

После этого основное задание подвергается семантическому разбору и выявлению потенциально изменяемых фрагментов (слотов), которые можно заменить на

динамические элементы, чтобы обеспечить студентов индивидуальными заданиями с учетом их опыта работы с платформой и общего уровня навыков.

После этого управление передается группе операций, формирующих блок «Подготовка динамических элементов», описываемые операции схематически изображены на рис. 8.

На этом этапе все динамические элементы, которые соответствуют обнаруженным в основном задании слотам, загружаются из базы данных вместе с сопровождающими их данными – описаниями заданий для разных уровней, декоративными элементами, а также серией тестов для запуска в интерактивной среде, обеспечивающей безопасность выполнения кода – «песочнице».

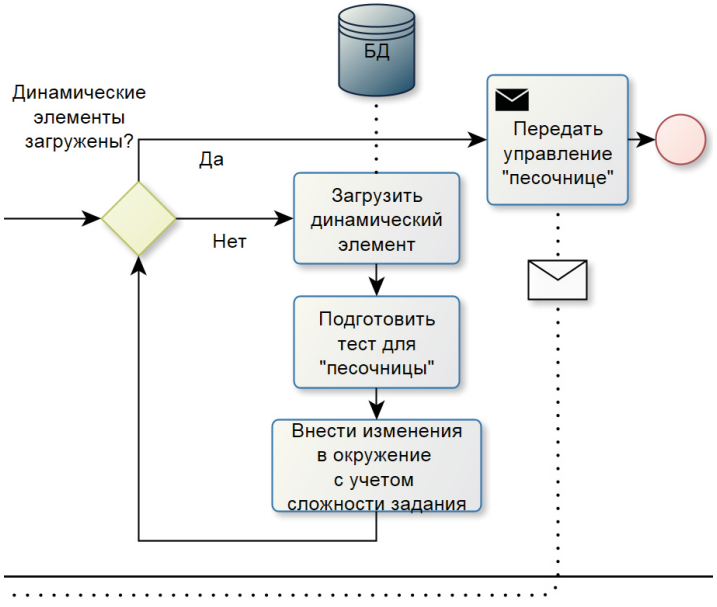

Рис. 8. Структура блока «Подготовка динамических элементов»

В зависимости от опыта и уровня навыков студента динамический элемент, предъявляемый в качестве индивидуального задания, подбирается в соответствии с критерием оптимальной сложности для выполняющего его студента. Этот критерий вычисляется на основе совокупности параметров, включая общий уровень группы, успеваемость по исследуемой в задаче теме, общее количество ошибок, допущенных студентом при выполнении предыдущих заданий и т.п.

Каждому элементу в слоте сопоставляется и загружается из базы данных сопровождающая его серия проверок, входящих в единый протокол теста, определяющий правильность решения для данного элемента задания и характер сделанных ошибок для более точного вычисления критерия оптимальной сложности.

После того, как все слоты перезагружены и заменены актуальными для данной группы участников динамическими элементами, обновленное задание задает окру-

жение рабочей среды и передает управление «песочнице», реализуемой в процессе обработки ввода пользователя. Данный процесс инициируется в ходе передачи управления в блок «Обеспечение функционирования рабочей среды», структура которого представлена на рис. 9.

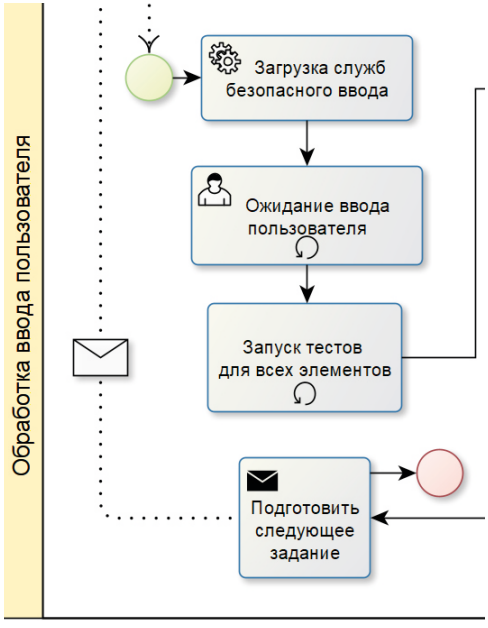

Рис. 9. Структура блока «Подготовка динамических элементов»

Основное предназначение системы – изоляция ввода пользователя и проверка на валидность для предотвращения атак со стороны потенциальных злоумышленников, пытающихся скомпрометировать систему и получить доступ к банку заданий. Также система направлена на предотвращение вредоносных изменений в рабочей среде под воздействием ошибочного кода.

Например, часть заданий подразумевает написание функций, осуществляющих модификацию дерева HTML DOM согласно заданным параметрам. Неосторожный запуск этих функций в рабочей среде может повредить ее окружению и графическому интерфейсу, получив доступ непосредственно к иерархии элементов HTML, составляющих интерфейс. В другом случае, преднамеренно с вредоносной целью написанная функция может получить доступ к объекту, представляющему агента, и передать ложные данные на сервер, сфабриковав выигрышную для студента ситуацию.

Для того, чтобы избежать подобных ситуаций, создается виртуализированная изолированная среда – «песочница» (англ. “sandbox” [4]), предотвращающая попытки получения доступа к вводу-выводу со стороны выполняемых в ней программ.

Система активирует в фоновом режиме обеспечивающие информационную безопасность службы, применяя шаблон проектирования «Наблюдатель». Службы, запущенные в фоновом режиме, подготавливают «песочницу» и параметры изолирован-

ного окружения для проверки ввода пользователя. Пользователь (студент) создает программу на стороне клиента и с помощью консольных команд загружает ее на сервер. Пользователь получает оповещение о том, что задание находится на проверке.

Сервер подтверждает передачу файла и проверяет валидность формы отправки запроса на сервер – то есть полное соответствие полей данных ожидаемым значениям, в числе проверяемых на корректность могут быть такие выражения, как формат файла, идентификаторы авторизации и скрытые токены сессии для определения личности отправителя, а также служебные флаги, определяющие режимы и условия, для которых требуется произвести проверку (групповой или индивидуальный вид задания, тип задания, используемый язык программирования и т.п.).

Если валидность не удалось подтвердить, сервер возвращает сообщение об ошибке в рабочую среду. Если валидность подтверждена, сервер отправляет сигнал фоновым службам, отвечающим за информационную безопасность, и переводит поток, отвечающий за обработку данных, в режим ожидания ответа от служб.

Как только службы получают сигнал от сервера о наступлении события, связанного с передачей файла, они загружают его в «песочницу» и подготавливают тесты, связанные с соответствующим заданием. «Песочница» в изолированной среде запускает программный код и возвращает результат выполнения каждого теста. Если все тесты пройдены успешно, считается, что студент выполнил задание. Если нет, и для не пройденного теста имеются комментарии для пользователя, то они опционально передаются через сервер пользователю в качестве подсказок. В любом случае, полностью успешно пройденные тесты переводят индивидуальное задание и связанный с ним элемент рабочей среды в решенное состояние, если же хотя бы один из тестов не успешен, задание считается не выполненным.

Поток, отвечающий за обработку данных, в обоих случаях получает ответ о результате тестирования от служб и выходит из режима ожидания, передавая информацию на сторону клиента, а пользователь получает подтверждение о том, что его задание успешно или неуспешно прошло процедуру проверки.

Взаимосвязь служб может быть реализована как с помощью механизма межпроцессного взаимодействия, так и с применением интерфейса FFI [2].

Эта процедура повторяется для всех тестов для каждого элемента, для которого обнаружен запрос на проверку решения. Проиллюстрируем этот механизм на конкретном примере.

В примере, приведенном в первом разделе, мы осуществили взаимодействие с элементом «палатка». Скомпилированное индивидуальное задание не учло наш предыдущий опыт, поскольку его не было – это было первое задание, которое выполнялось в рамках цифровой адаптивной платформы. Для того, чтобы выполнить его («собрать дрова»), потребовалось написать функцию, показывающую уровень наших знаний (в данном случае, навыки работы с HTML DOM). Функция реализовала задачу (выборку «подходящих для дров», т.е. соответствующих определенному классу, элементов HTML). Файл с функцией был отправлен на сервер и проверен в «песочнице». Все тесты оказались успешно пройдены, и элемент «палатка» пре-

вратился в элемент «дрова», а индивидуальное задание сменилось на следующее – «зажечь костер». Применяя навыки и знания из смежной области (работа с правилами CSS), повторяем похожую последовательность действий с новой функцией-решением, и после успешного прохождения заданием тестов в изолированной среде сервер меняет элемент «дрова» на «костер». С этим элементом связанных заданий нет, это означает, что индивидуальное задание решено, а элемент «костер» становится элементом для группового задания.

Несмотря на то, что данный блок реализует безопасную взаимосвязь между решенными заданиями и состояниями рабочей среды, решенные и не решенные задания никак не влияют на знания платформы о навыках студентов и не делают платформу адаптивной; поэтому, когда блок «Обеспечение функционирования рабочей среды» завершает свою работу, он передает управление блоку «Обработка результатов», схема которого представлена на рис. 10.

Тесты содержат специальный маркер-идентификатор для дополнительной индексации тестов в базе данных по группам, имеющим одинаковый маркер. Маркер выставляется одинаковым для группы, если потенциальные ошибки, на которые проверяют тесты в группе, имеют схожий характер – например, синтаксические ошибки, отсутствие проверки граничных значений, некорректное условие завершения цикла и т.п. Маркер позволяет, таким образом, объединять задания в базе не только по тематике, но и по характеру ошибок, которые могут проявляться вне зависимости от изучаемой темы (например, синтаксическую ошибку можно допустить и в описании массивов JavaScript, и в описании правил CSS). Кроме того, даже в рамках одной темы, это позволяет детально проработать конкретные недочеты, допущенные студентом в ходе выполнения заданий. Таким образом, осуществляется поиск похожих ошибок, допущенных студентом.

Обработка результатов происходит следующим образом: проверяется общая степень корректности выполнения заданий по каждой теме (отношение правильно решенных заданий к общему числу попыток; для любого учтенного задания число попыток всегда не меньше 1, так как если группе или студентам не удалось предъявить ни одного решения за ограниченное время, то все тесты считаются проваленными). Далее проводится поиск по индексированным тестам (поиск похожих ошибок). Для низких значений корректности выполнения принимается решение, что вся тема не проработана участником; для высоких значений исследуется характер ошибок по маркерам групп проиндексированных тестов, принимается решение о предъявлении заданий на конкретную ошибку. Значения степени корректности влияют и на то, с какой вероятностью уже на следующем занятии (на следующей практической паре с применением платформы) будет предъявлено соответствующее задание, а именно, с тем большей вероятностью, чем больше ошибок и недочетов было допущено при проработке соответствующей темы. Но для этого показателя вероятность считается по всей совокупности пройденных занятий таким образом, что, чем больше ошибок допускается на всей совокупности пройденных занятий в заданиях на одну тему, тем большее значение принимает соответствующая вероятность.

Пороговые значения, которые обеспечивают принятие решения, задаются моделью машинного обучения, которая обучается на данных статистики об успехах группы и каждого студента. Эта же модель корректирует оцениваемый уровень навыков каждого студента и группы в целом. Это необходимо для загрузки адаптивно подобранных под уровень навыков заданий из базы для следующего занятия на основе прогноза, формируемого в качестве результата работы модели. Учитываются отобранные задания для профилактики совершенных ошибок. Промежуточные результаты фиксируются в базе данных.

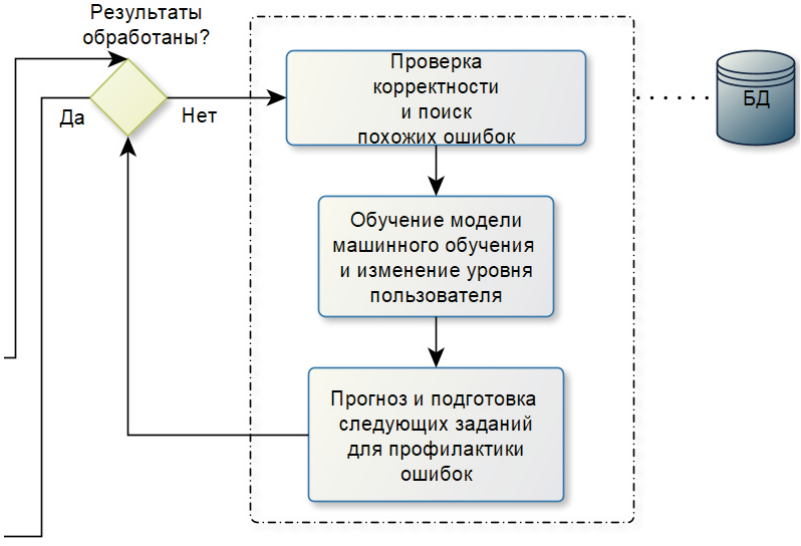

Рис. 10. Структура блока «Обработка результатов»

После того, как все результаты для всех участников и группы обработаны, управление возвращается блоку «Обеспечение функционирования рабочей среды» для подготовки следующего задания. Этот блок, в свою очередь, возвращает управление процессу адаптивной модификации задания для следующего проводимого занятия, и так повторяется цикл взаимодействия компонентов, обеспечивающих адаптивность платформы.

Когда число занятий исчерпано или преподаватель принимает решение о завершении работы платформы, цикл прекращается, и работа адаптивной платформы останавливается.

4. ЗАКЛЮЧЕНИЕ

В статье были подробно описаны способы применения цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы. Проиллюстрирован характер взаимодействия системы с обучающимся пользователем на примере предъявляемого индивидуального задания и интерфейсов

рабочей среды, предоставляющей необходимый набор функций для обеспечения процесса обучения пользователя. Показано отличие индивидуального и группового задания, их взаимосвязь, а также особенности выполнения группового задания.

Исследованы компоненты, обеспечивающие адаптивность платформы. Продемонстрирован подход к формированию единого задания с учетом уровня навыков обучающихся и характера допущенных ими ошибок в ходе обучения с использованием платформы. Описан подход к обучению платформы, определено место алгоритмов машинного обучения в процедуре адаптивной модификации предъявляемых заданий.

Отдельно описаны аспекты, связанные с обеспечением информационной безопасности при работе с вводимыми обучающимся данными. Детально исследована работа служб платформы, отвечающих за безопасный ввод данных и изолированную обработку результатов.

Таким образом, в статье были изложены существенные детали, связанные с практическим применением цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы.

Литература

1. *Попков С.И.* Особенности программной реализации цифровой адаптивной платформы для обучения программированию с функцией формирования навыков командной работы – Моделирование и анализ данных. Том 10. № 3. 2020. С. 85–100.
2. *Попков С.И.* Программная реализация межъязыкового взаимодействия на базе динамических библиотек. – Нейрокомпьютеры: разработка, применение. № 3. 2018. С. 39–49.
3. W3C SVG Working Group – URL: https://www.w3.org/Graphics/SVG/ (дата обращения 10.12.2020 г.)
4. What is a Sandbox (in Software Testing)? – Definition from Techopedia – URL: https://www.techopedia.com/definition/27681/sandbox-software-testing (дата обращения 10.12.2020 г.)
5. World Wide Web Consortium (W3C) – URL: https://www.w3.org (дата обращения 10.12.2020 г.)

Application of the Digital Adaptive Platform for Learning Programming with the Teamwork Skills Forming Function

Sergey I. Popkov*
Moscow State University of Psychology & Education, Moscow, Russia
ORCID: https://orcid.org/0000-0003-2566-1262
e-mail: rslw25@gmail.com

Research is carried out on approaches for the digital adaptive platform for learning programming with the teamwork skills forming function application. The work environment interface and key components that make the platform adaptable are demonstrated. The platform services responsible for information security functions are detailed. An example of the presented task is given.

Keywords: adaptability, digital platform, sandbox, learning computer programming.

For citation:
Popkov S.I. Application of the Digital Adaptive Platform for Learning Programming with the Teamwork Skills Forming Function. *Modelirovanie i analiz dannykh = Modelling and Data Analysis*, 2021. Vol. 11, no. 1, pp. 78–93. DOI: https://doi.org/10.17759/mda.2021110106 (In Russ., abstr. in Engl.).

References

1. Popkov S.I. Features of the software implementation of the digital adaptive platform for learning programming with the teamwork skills forming function – *Modelling and Data Analysis*. Vol. 10. № 3. 2020. pp. 85–100 (In Russ.).
2. Popkov S.I. Software implementation of interlingual programming communication based on dynamic link libraries. – *Neurocomputers: development and application*. № 3. 2018. pp. 39–49. (In Russ.)
3. W3C SVG Working Group – URL: https://www.w3.org/Graphics/SVG/ (req. date 10.12.2020 г.)
4. What is a Sandbox (in Software Testing)? – *Definition from Techopedia* – URL: https://www.techopedia.com/definition/27681/sandbox-software-testing (req. date 10.12.2020 г.)
5. World Wide Web Consortium (W3C) – URL: https://www.w3.org (req. date 10.12.2020 г.)

***Sergei I. Popkov,** PhD in Physical and Mathematical Sciences, Assistant Professor of the Computer Science Faculty, Moscow State University of Psychology and Education, Moscow, Russia, ORCID: https://orcid.org/0000-0003-2566-1262, e-mail: rslw25@gmail.com