Архитектура системы интерактивного обучения и тестирования алгоритмов и структур данных

Авторы: М. Миливоевич, Р. Джурджевич, Томашевич

Реферат: В данной работе мы предлагаем архитектуру новой системы программного обеспечения для самостоятельного интерактивного обучения и оценки в области алгоритмов и структур данных. Система расширяет возможности Визуального моделирования Алгоритмов (VSA), инструмент, ранее разработанных на факультете Электротехники, университет Белграда. Система контроля обеспечивает обучаемого независимости уровня, гибкий набор входных данных, назначение и настраиваемая автоматическая оценка тест-taker действия. В статье описывается архитектура предлагаемой системы и дает соответствующие детали реализации.

Ключевые слова: автоматическая оценка, интерактивное обучение, проверка знаний.

I. ВВЕДЕНИЕ

Программные комплексы для обучения и оценки представляют собой ценное средство в сфере образования. С одной стороны, они обеспечивают повторение материала (т.е. процедуры обучения) в объеме, необходимом для обучаемого. С другой стороны, после завершения тренировочного цикла, они обеспечивают проверку знаний, с дополнительной автоматической оценкой. Использование таких систем уменьшает среднее время инструктора, предназначенное для обучения или тестирования. Кроме того, инструктор может даже не присутствовать в момент обучения , что позволяет ему уделять больше времени для деятельности, которая требует повышенного внимания. Передовые системы предоставляют собой инструменты для анализа результатов тестирования, которые могут помочь инструктору сделать важные рекомендации о том, как изменить состав уроков и тестов.

Наиболее распространенные системы для обучения и тестирования знаний (виртуальная учебная среда) [1] основаны на вопросах с ответами, из которых субъект должен выбрать определенный. Тем не менее, такой подход годится только для вопросов в областях, где достаточно обычных знаний, так как не учитывается процедура получения ответа. Кроме того, подход с использованием ответов позволяет обучаемому угадать правильный .

Ранее обсуждался подход к предоставлению ответов не подходящих для задач, которые являются общими в области алгоритмов и структур данных. В этом домене, знание о конкретном алгоритме, а также его правильном применение, на самом деле представляет собой ответ. Следовательно, субъект, который обладает небольшими знаниями о конкретном алгоритме и субъект, который делает ошибки в последнем этапе алгоритма, будет оцениваться таким же образом.

В этой статье мы предлагаем архитектуру системы самостоятельного обучения и тестирования в тех областях, где процедура решения проблемы имеет большое значение. Хотя система разработана в основном в виде программного инструмента для улучшения качества курса в алгоритмах и структуры данных в Школе Электрической Инженерии, Университет Белграда, предлагаемая архитектура обеспечивает более широкий диапазон применения.

Остальная часть работы организована следующим образом. В разделе 2 излагаются проблемы, которые автоматизированная система обучения и тестирование должна решить. Мы также дали краткий обзор существующих систем и указали на некоторые свои нежелательные черты, которые предлагаемая архитектура намерена устранить. В разделе 3 обсуждается, желаемые функциональные возможности предлагаемой системы и представлены наиболее важные детали архитектуры. В разделе 4 мы дали некоторые детали реализации и прототип системы структуры, с кратким обзором используемых технологий. В разделе 5 мы сделаем выводы и укажем на некоторые направления дальнейших исследований.

II. Проблемы и существующие решения

В этом разделе мы даем обзор проблем в области автоматизированного обучения и тестирования знаний и даем краткий обзор существующих решений.

Использование программных комплексов для подготовки и тестирования знаний в образовании имеет несколько желательных последствий:

• Обучение проводится без или с минимальным участием инструктора, лишнее время и усилия направляются на деятельность, которая требует больше внимания

• обучение может быть настроено под уровень знаний стажера, практикант может повторить процедуру обучения столько раз, сколько требуется для того, чтобы освоить материал

• Результаты тестирования доступны сразу после окончания испытания, независимо от количества предметов,

• обучение может осуществляться на расстоянии, что эффективно снижает потребность в физическом присутствии (предмета и преподавателей) и увеличивает время и доступность учебного материала. Существующие системы тестирования знаний, с наиболее важными представителями [2], традиционно состоят из набора вопросов, подготовленных заранее, где субъект проходит испытания, выбирая один (а иногда и более) предлагаемый ответ.

Иногда в параметризации присутствуют вопросы, которые позволяют проверять уровень понимания, вместо простого воспроизводства знаний. Пример параметризации дано в [3]. Однако такой метод проверки знаний вызывает несколько важных проблем. Во-первых, этого недостаточно для оценки задач, которые требуют знания алгоритма, поскольку, в случае неправильного ответа, он не может принимать во внимание уровень осведомленности алгоритма предмета. Кроме того, в случае правильного ответа, невозможно сказать на самом деле ли объект знает ответ, угадал его или нашел ответ, сделав несколько ошибок (что так или иначе компенсируют друг друга). Во-вторых, ограниченные параметризации предполагают, что преподаватель должен создать большое число подобных вопросов на ту же тему. Наконец, при создании вопроса, и его задача дать

правильный ответ. Это значительно снижает возможность обучения, так как субъект может рассчитывать только на материал (вопросы), предоставленный инструктором. Он также значительно снижает возможность самотестирования, когда субъект определяет и решает проблему по-своему. Мотивация для проектирования архитектуры системы подготовки исходит из того, что существующая общеизвестная система не обеспечивает адекватное решение вышеуказанных проблем.

Очень желательно для системы образования, позволить автономно и пассивно, управлять самостоятельной подготовкой. Пассивная подготовка предполагает, что субъект придерживается автоматизированной процедуре алгоритма выполнения, с шагом и возможностью контролировать скорость выполнения и перемещаться (возможно, обратно) к определенному шагу. Контролируемая подготовка предполагает, что субъект пытается решить проблему вручную указывая выполнение шагов, в то время даются подсказки, которые помогают перейти к следующему шагу. Самостоятельная подготовка специалистов предполагает, что никакая помощь не уделяется этому вопросу. Кроме того, за исключением пассивного обучения, выдается предупреждение, как только обнаружена ошибка при выполнении шага. По существу, испытание такое же, как самостоятельная подготовка, но без предупреждений.

В оставшейся части этого раздела мы сделаем краткий обзор системам, которые поддерживают QTI стандарт [4]. Как указано в [5], QTI данных модель, используется для представления вопросов и ответов, оценки, результатов подразделений в целях создания тестов. Спецификация предназначена для поддержки простых вопросов и испытания материалов. Обмен данными между различными оценками систем возможен (из-за схемы XMS).

Moodle (модульная объектно-ориентированная окружающая среда) [6] является свободным, на основе PHP с открытым исходным кодом веб-приложением для электронного обучения. С апреля 2011 года, в Moodle пользовательская база данных содержит более 54 000 подтвержденных сайтов, которые обслуживают более 42 миллионов пользователей более чем в

4,5миллионов курсов. Он в основном используется образовательными учреждениями. Dokeos [7] является другим открытым исходным кодом веб-электронного обучения приложений, также разработанный в PHP. В настоящее время он обслуживает более 3 миллионов пользователей по всему миру. Он используется образовательными учреждениями, транснациональными компаниями, мед учреждениями и государственными администрациями. Sakai [8] бесплатный проект, написанный на Java, предназначенный для обучения, исследований и сотрудничества. Был сделан для сотрудничества нескольких университетов США. В настоящее время он используется в более чем 160 образовательных учреждений примерно на 200 000 пользователей. OLAT [9] является бесплатным с открытым исходным кодом веб-приложением, написанные на Java. В настоящее время есть 150 проверенных установок, с более чем 170000 пользователей в 8 000 курсов.

Насколько нам известно, ни одна из упомянутых систем не обеспечивает все желаемые функциональные возможности. Система должна иметь, например, контролируемое или самостоятельное обучение и тестирование в тех областях, где процедуры для решения проблемы имеет большое значение. Авторы не знают, что архитектура такой системы общедоступна. Следовательно, она представляет собой мотивацию для разработки системы автономно –интерактивной, контролируемой или самостоятельной подготовкой и проверкой знаний в тех областях, где знания оцениваются с настраиваемым методом. Следует отметить, что последовательность действий, которые представляют правильный ответ, не должна быть уникальной.

III. Описание и Архитектура

В этом разделе мы представляем архитектуру и возможности системы для оценки и подготовки кадров в области алгоритмов и структур данных, что является темой данной статьи.

По сути, предлагаемая система обеспечивает средства проведения части практического обучения в качестве интерактивного. Компьютерные сессии, где имитатор, который используется студентами, выступает в качестве личного тренера. Обязательство учителя заключается в предоставлении студентам

комплекса проблем, над которыми они должны работать и проверять свои знания, а также предоставить им тест, используемый для оценки приобретенных знаний. Работа студента регистрируется и может дать учителю представление сегмента рабочих материалов, что студенты, как правило, принимают в более медленном темпе, что особенно важно в режиме обучения. В режиме тестирования знаний, тестовый доступ предоставляется только тем студентам, которые принадлежат к группе пользователей, для которых это испытание предназначено. Озвучивание работы студентов осуществляется путем сравнения данных, когда система приходит в действие ожидать ответ путем выполнения заданного алгоритма в соответствии с выбором входных параметров. На основе записанной оценки, итоговая оценка выставляется по компоненту классификации программного обеспечения.

Эта система состоит из сервера и на стороне клиента. Сторона сервера включает в себя одно приложение (сервер), в то время сторона клиента состоит из трех независимых приложений: студент, преподаватель и администратор приложения. Используются три независимых клиента приложения для уменьшения сложности осуществления и повышения общей системы безопасности. С учетом выбранной технологии для развития, обратное проектирование клиентского приложения может раскрыть некоторые детали реализации. С тремя отдельными приложениями, из которых только студент будет сделано общедоступными, то шансы на подрыв системы безопасности, потенциально опасных пользователей существенно уменьшена. Компонент-схема, показывающая основные части системы приведен на рис. 1.

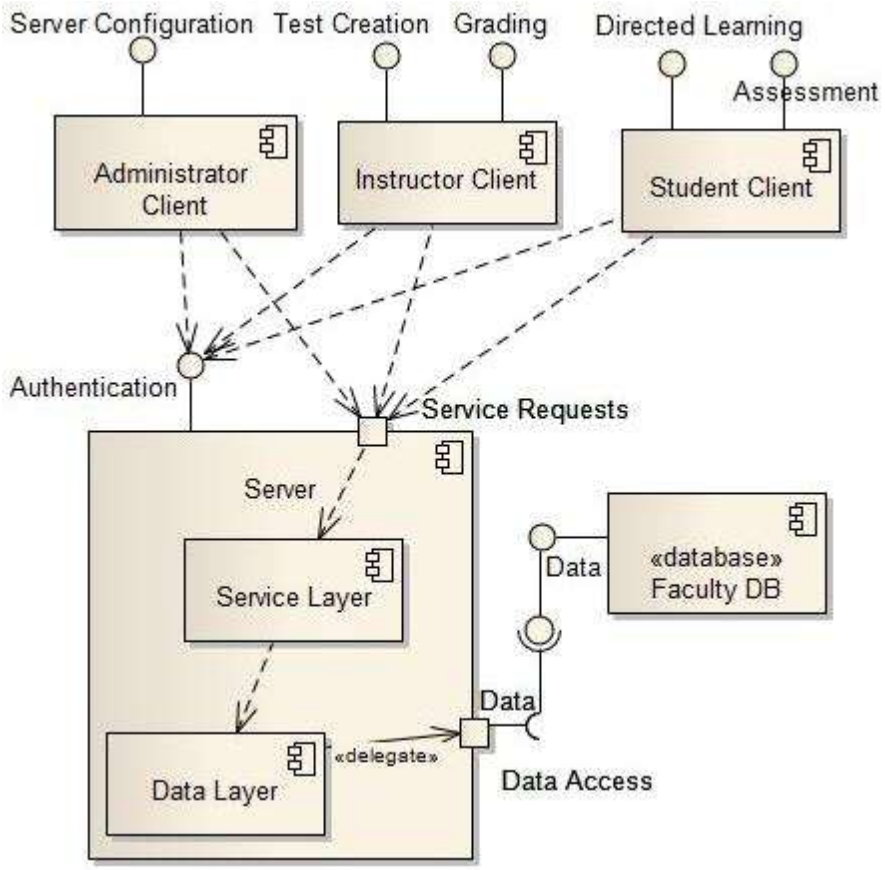


Рисунок 1 – Компонент-схема, показывающая основные компоненты системы - три клиента, сервер и факультет базы данных.

Клиентские компоненты обеспечивают конфигурацию сервера, создание и классификацию тестов, направленных на обучение и оценку. Сервер обеспечивает аутентификацию и обработку запросов на обслуживание, и связь с базой данных.

Серверное приложение используется для аутентификации и авторизации клиента и для обработки клиентских запросов. В случае длительного бездействия клиента, он может также прекратить сеанс клиента с предварительной отправкой предупреждения.

Для повышения общей безопасности и гибкости системы архитектура была организована на три слоя: уровень представления, уровень служб и доступа к данным. Уровень представления отвечает за отображение графического пользовательского интерфейса и реализована в компоненты (см.рис.1). Уровень служб определяет функциональность системы. Вводя службы, мы преодолеваем разрыв между представлением слоя и слоя доступа к данным, что приводит к лучшей развязки компонента и, соответственно, к большей гибкости и

безопасности системы. Уровень доступа к данным отвечает за сохранение информации через связь с базой данных.

База данных (см. рис. 1), с которой информация потока осуществляется исключительно через доступ к данным слоя внутри сервера приложений (см. раздел IV), содержит хранилище вопросов и алгоритм входа параметров, список всех активных тестов, информацию касающуюся студенческих групп, сведения о пользователях, статистические данные, и временного резервного копирования данных из тестов в настоящее время.

Администратор приложения отвечает за конфигурирование сервера приложений, а также организацию пользователей (например, добавление или удаление)и группы пользователей (например, добавление пользователей в группу "гость"). Существуют два типа гостей - анонимные и уполномоченные. Анонимные гости вошли в систему, и функционально доступным для них является пассивное обучение и обзор общественных тестов. Уполномоченный гости вошли в систему, и имеют доступ к большей части студентов с функциональными возможностями, за исключением, что те функции применяются только к общественным тестам.

Инструктор приложения отвечает за создание и техническое обслуживание студенческих групп (в первую очередь, связанных с курсами), вопрос создания тесов, публикацию тестов, а также для использования статистики и сортировки результатов тестов. Вопрос определен путем выбора алгоритма из желаемой области и установкой входных параметров алгоритма (либо выбрав их из хранилища или путем создания новых значений). Входные параметры включают в себя начальное состояние структуры данных, над которыми алгоритм выполняется, а также параметры, которые определяются самим алгоритмом. Например, для алгоритмов сортировки, соответствующим параметром будет порядок сортировки (по возрастанию или по убыванию). Следует отметить, что система также поддерживает традиционный способ определения вопросов (например, в форме множественного выбора текстовых вопросов), но, что не является предметом данной статьи. Тест состоит из любого ряда вопросов. Для добавления вопросов в тест учитель имеет право выбирать вопросы из хранилища существующих

вопросов, или создать новый. На каждый вопрос, система держит счет тестов, в которых он была использован, давая учителю представление о частоте использования какого-либо конкретного вопроса. При публикации тестов, учитель обозначает студента группы, к которой относится тест, длительность теста, дату активации или продолжительность, тип теста (для оценки целей студента или для самостоятельной практики). Учитель определяет метод оценивания теста, выбрав один из доступных модулей классификации. Тест считаются опубликованным после первого доступа пользователя к нему (Первая выборка из репозитория). После опубликования тест автоматически "заблокирован", и невозможно сделать любые последующие изменения к нему. Учитель может аннулировать тест, который. Кроме того, можно клонировать тест, который позволяет учитель, чтобы сделать подобные тесты, с незначительными изменениями ,содержанием и целями (кому они предназначены и каким образом).

Приложение студент предназначено для обучения и оценки, и обеспечивает интерактивную работу (на дисплее, выполнение тестов) в графическом пользовательском интерфейсе в которой работают приложения, разработанные в [12], [13]в качестве своей основы. После аутентификации, Приложение получает список доступных тестов с сервера, из которых студент может выбрать один, чтобы решить. Приложение студент автоматически загружает все данные, необходимые для выполнения теста, включая реализацию алгоритмов. Следует отметить, что Приложение может работать самостоятельно, без подключения к серверу. Приложение предназначено для работы в двух режимах: обучение и оценка. Как отмечено в разделе II, обучение может быть пассивным, контролируемое или направленное на самого себя. Тестирование может быть как самотестирование или градуированное тестирование. Самотестирования отличается от градуированного продолжительностью. В этом режиме, студент постепенно решает тес, без помощи, а затем, в конце теста, он показывает результат испытаний в виде процентов , а количество сделанных ошибок, сгруппировано по их тяжести.

Компонентом, ответственным за определение процентного соотношения между действиями пользователя и искомым является реализация алгоритма, указанная при создании вопроса, и компонент, отвечающий за оценку серьезности ошибки является реализация данных структур над которым выполняется алгоритм. Результаты тестов хранятся в базе данных, где они будут доступны для учителей, ответственных за их классификацию.

IV. Детали реализации

В этом разделе мы рассмотрим детали реализации прототипа приложения, которое используется для проверки архитектуры системы предлагаемой в данной работе, и рассмотрим используемые технологии.

Подключение клиентских приложений с сервером выполняется на основе встроенной библиотеки программирования языка Java, предназначенного для вызовов удаленных(RMI). В RMI имеется удобная функция, которая позволяет установить связь между двумя классами, расположенными на разных компьютерах (и методы которого, таким образом, выполняют различные виртуальные Java-машины) должны быть выполнены прозрачно, таким же образом, как если бы были классы на одном компьютере.

Для работы с базой данных доступа и передачи данных мы использовали Hibernate (версия 3.6.4), популярное с открытым исходным кодом программное обеспечение, основа для объектно-реляционного отображения. Это улучшение по сравнению с традиционным подходом в первую очередь позволяет легко и просто отображать модель базы данных (реляционная модель) в объектной модели(классы, которые используются в системе), с возможностью написания запросов программно с использованием Фреймворка собственные классы для выражения различных критериев запроса.

Вопросы интеграции, безопасности и управления транзакциями реализованы с использованием Spring Framework (версии 3.0.5),с открытым исходным кодом рамки для Java платформы. Spring Framework включает в себя несколько модулей, которые предоставляют широкий спектр услуг, из которых после проявляют

интерес в этой статье: инверсии управления, аспектно - ориентированного программирования, доступ к базам данных, транзакции управления, удаленные вызовы методов, аутентификации и авторизации и тестирование

SE2 Java 1.6 и NetBeans (версия 7.0), с открытым исходным кодом интегрированная среда разработки для Java платформы, были использованы для разработки прототипа приложения. На Рис.2 показана упрощенная диаграмма класса UML включающая в себя ключ классов сервера приложений. Сервер интерфейс предоставляет API связи, которые должны быть учтены в реализации сервера и классов. ServerImpl класс, который реализует интерфейс сервера. Его роль заключается в аутентификации и авторизации приложений, а также обеспечить им доступ к функциональности с помощью сервиса классов.

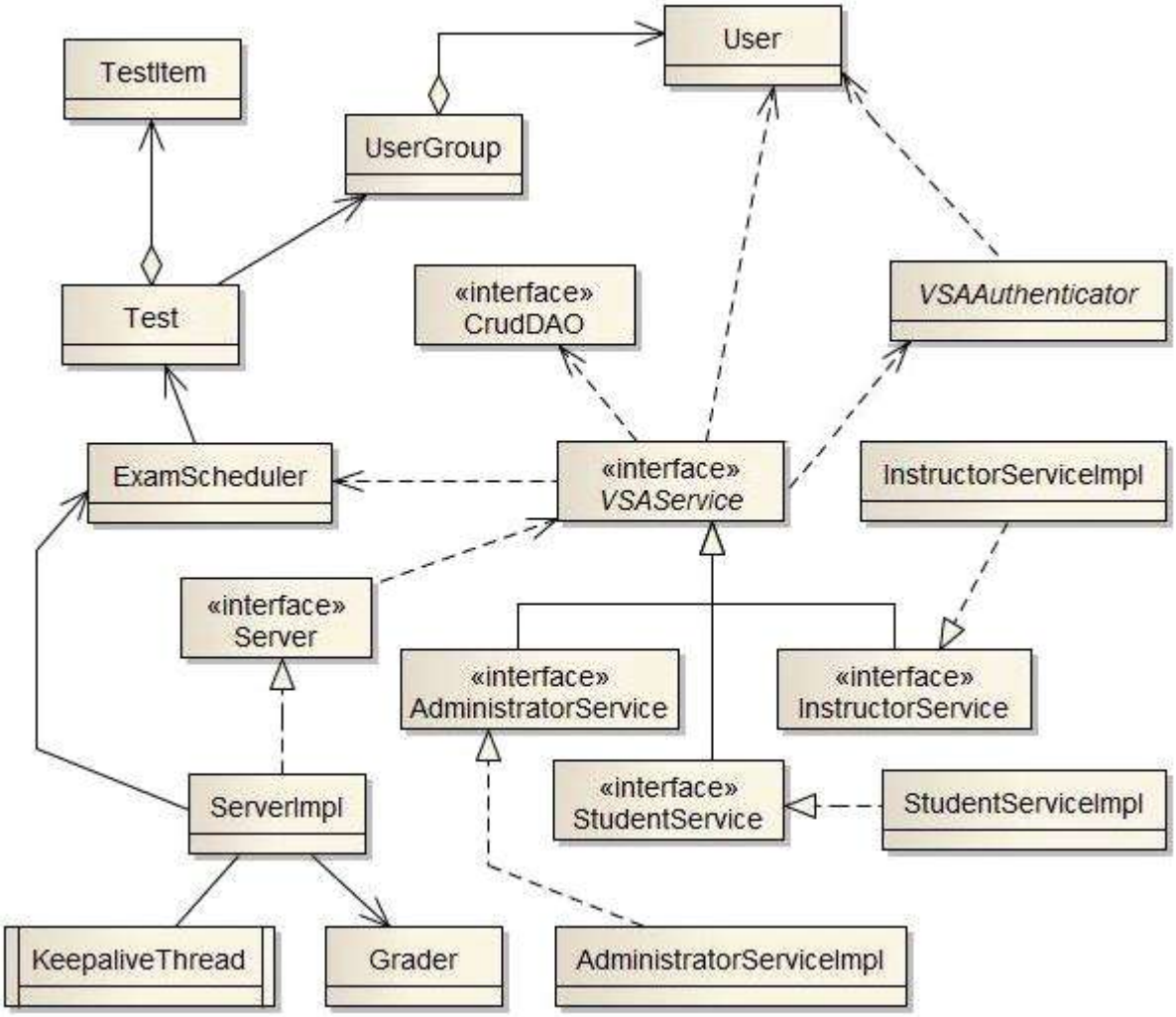


Рисунок 2 – Упрощенная схема, показывающая основные классы сервера приложений.

VSA Service класс и интерфейс иерархии образуют уровень услуг для приложений, в то время остальные классы либо представляют собой абстракции домена или служат для интеграции с API службами в остальной части системы.

Функциональные возможности системы реализованы в классе иерархии, состоящей либо в прямой реализации из VSA Service интерфейса или из классов, которые оставляют своих наследников (классы Administrator Service Impl, и Instructor Service Impl Student Service Impl), и сервера приложений, экземпляр после успешной аутентификации в соответствии с типом клиентского приложения. Каждый из этих классов реализует соответствующий интерфейс –Administrator Service, и Instructor Service Student Service - которая обеспечивает API для удаленной связи с клиентскими приложениями. Вышеупомянутые классы и интерфейсы, образуют сервер приложение службы слоя (см. рис. 1). Каждый клиент приложений имеет доступ к точно к определенному набору через свою функциональность VSAService реализации. StudentService интерфейс предоставляет функциональност, для доставки нужного теста из базы данных или ввода ответов респондента в базу данных. С другой стороны, Instructor Service интерфейс, который обеспечивает функциональность, необходимую для учителей – создание вопросов и размещение их в репозитории, создания и публикации тестов, сортировки результатов тестирования и т.д.

Роль KeepaliveThread, который является активным классом, заключается в принятии во внимание длительности клиентских приложений "бездействия периодов и, при необходимости, прекращение клиентских сессий.

Каждый из классов, который реализует некоторые службы интерфейсов (AdministratorService, InstructorService,StudentService) использует CrudDAO интерфейс, где сохраняет свои данные в базу данных. CrudDAO интерфейс обеспечивает минимальный набор методами, которые связаны с основными операциями с базами данных - создание, чтение, ввод и удаление (часто сокращенно CRUD).Иерархия классов, реализующих интерфейс CrudDAO представляет собой слой доступа к данным (см. рис. 1). Для отображения классов

объекта домена в таблицах реляционных доменов мы использовали ранее упомянутые объектно-реляционные отображения рамки, Hibernate.

Рис. 3 показывает упрощенный класс UML диаграмму, основные классы из клиентских приложений. Клиентский интерфейс обеспечивает базовый API для связи с сервером, в то время как класс, реализующий его, ClientImpl, реализует поведение, общие для всех клиентских приложений - аутентификация и авторизация, конфигурация параметров связи (например, адрес IP-сервера) и т.д. Студент, преподаватель и администратор клиенты приложений реализованы как классы StudentClientImpl, и InstructorClientImpl, AdministratorClientImpl соответственно. Как можно видеть в диаграмме, каждый из клиентских приложений использует надлежащее осуществление VSAService интерфейсов.

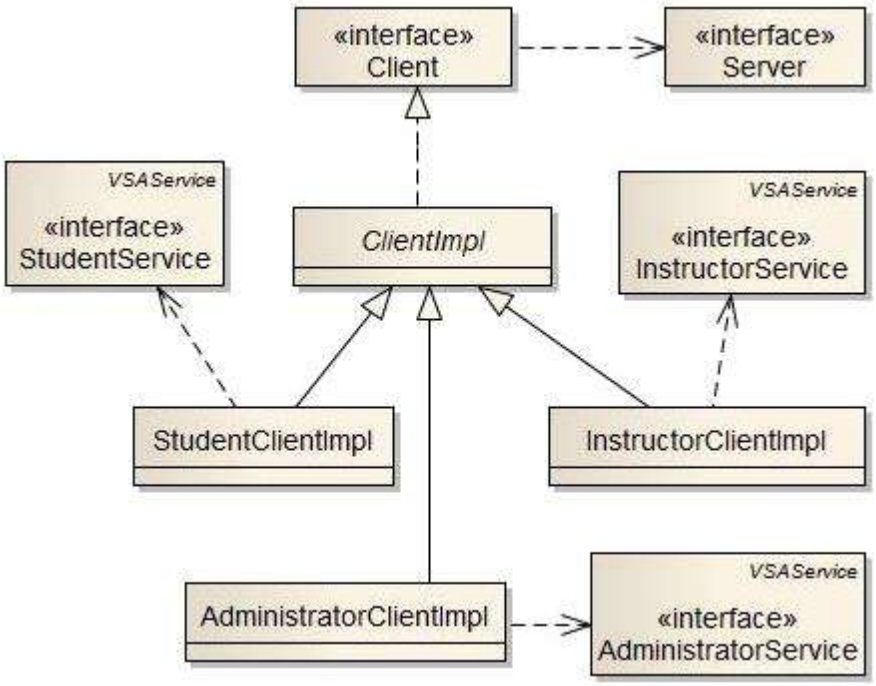


Рисунок 3 – Упрощенная схема основных классов клиентских приложений.

Клиентский интерфейс обеспечивает API связи, реализует ClientImpl функции, общие для всех клиентов, а клиент Приложения используют надлежащую реализацию VSAService интерфейсов для достижения функциональности.

Внедрение системы, представленной в данном разделе опирается на инструмент под названием Визуальный тренажер Алгоритмов (VSA). Этот инструмент является результатом серии диссертаций и дипломных работы студентов Школы электротехники в Белграде [10] - [13].Презентация слоя VSA, используется в качестве основы для новой презентации системы слоя. В связи с этим, в данном разделе мы кратко опишем ядро инструмента VSA.

VSA ядро состоит из трех ключевых основных классов и их специализациями: класс алгоритмов, который определяет сам алгоритм, т.е. выполнение процедуры; ObservableVariable класс, который является базовым классом всех структур данных классов; и используемый алгоритм и действий, который представляет собой абстракцию действия, предпринятых алгоритмом для его структуры данных. Действие класса содержит все необходимую информацию для определения участников акции, с одной стороны, и состояние использованной структуры данных после выполнения действий, с другой стороны. Например, в случае перекачки, два элемента массива, экземпляр класса действий, что представляет этот шаг в алгоритме будет содержать идентификатор массив, элементы которого копируют массив перед своп, и массив позиций(индексы) элементов, меняющих местами их значения.

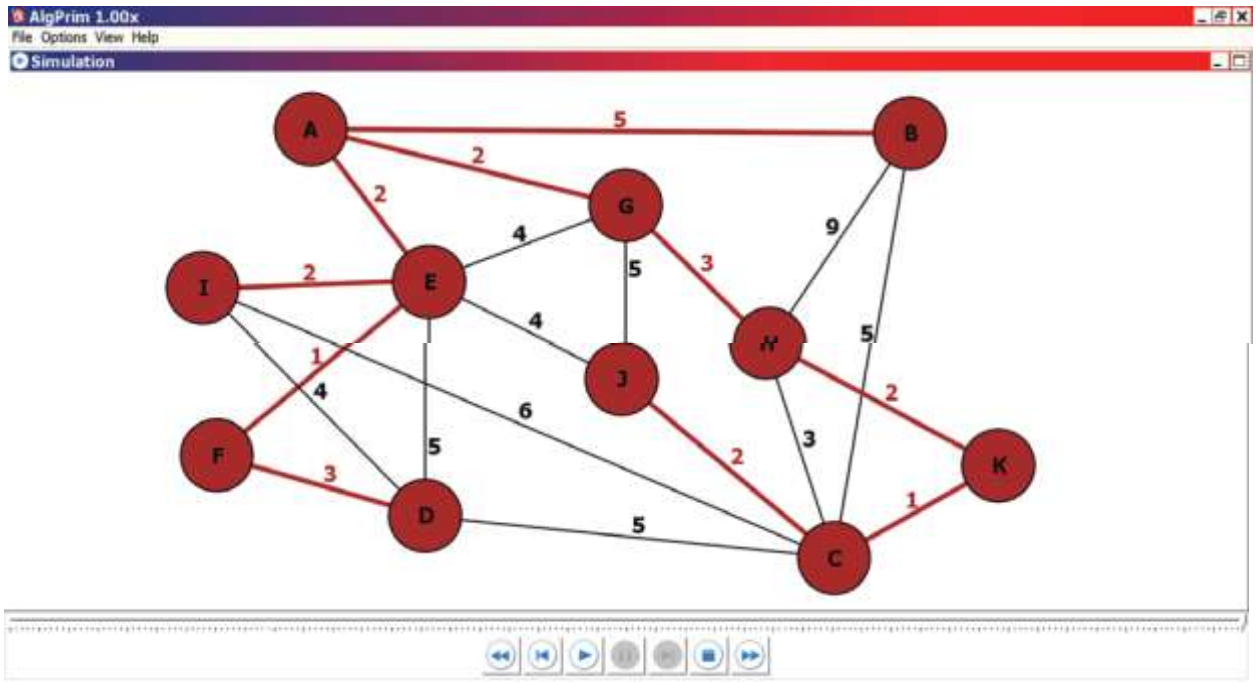
Выполнение алгоритма над структурами данных приведет список соответствующих действий, которые впоследствии интерпретируются презентацией слоя с образованием визуального представления. Структура данных класса ObservableVariable, помимо представления самих структур, также предоставляет утилиты методов, которые являются абстракции из основных операций по соответствующие структуры и также несут ответственность за генерации приведенный перечень действий.

Важной частью презентации слоя представлен двумя основными классами - класс представления(AbstractView) и просмотр классов (AbstractViewer).Просмотр классов определяет графическое представление компонентов и используется для визуализации алгоритма, а зритель классов определяет, что, когда и где рисовать. Класс представления имеет жизненно важное значение для реализации, проявляет

гибкость и представляет способ просмотра процесса выполнения алгоритма. Этот класс определяет стандартный интерфейс для обработки действий, которые создаются с помощью алгоритма, а также методы для отображения абстрактных структур данных, связанных с выполнением. Класс представления отвечает за список действий и образуются в результате выполнения алгоритма, который сделан путем перебора списка действий и отображения состояние используемых структур данных после каждого шага алгоритма, возможно, с анимацией действий, которые привели к этому состоянию.

Класс зритель (AbstractViewer) представляет собой активный класс, который сотрудничает с классом так, что он обеспечивает необходимый контекст для рисования. В его обязанности входит называть соответствующие методы класса представления своевременно.

На Рис. 4 показаны примеры использования инструмента для VSA визуализации некоторых алгоритмов на графах и двоичных деревьях. Описанная иерархия классов инструмента VSA играет ключевую роль в системе, предложенной в данной работе. Проверка ответов респондентов на основе расширений алгоритма и ObservableVariabl еклассов. Алгоритм класс отвечает за сравнение ряда действий ответчика с серией прошедших действий. Алгоритм может принять решение остановить сравнение действий после первого несоответствия или попытки восстановления. Это обеспечивает более точную оценку респондентами знания. С другой стороны, это ObsevableVariable ответственность за оценку тяжести ошибки, которая является частично на основе полученных данных структурами. Например, если ряд действий студента привела к плохо сформированной структуре, это указывает на серьезную ошибку.



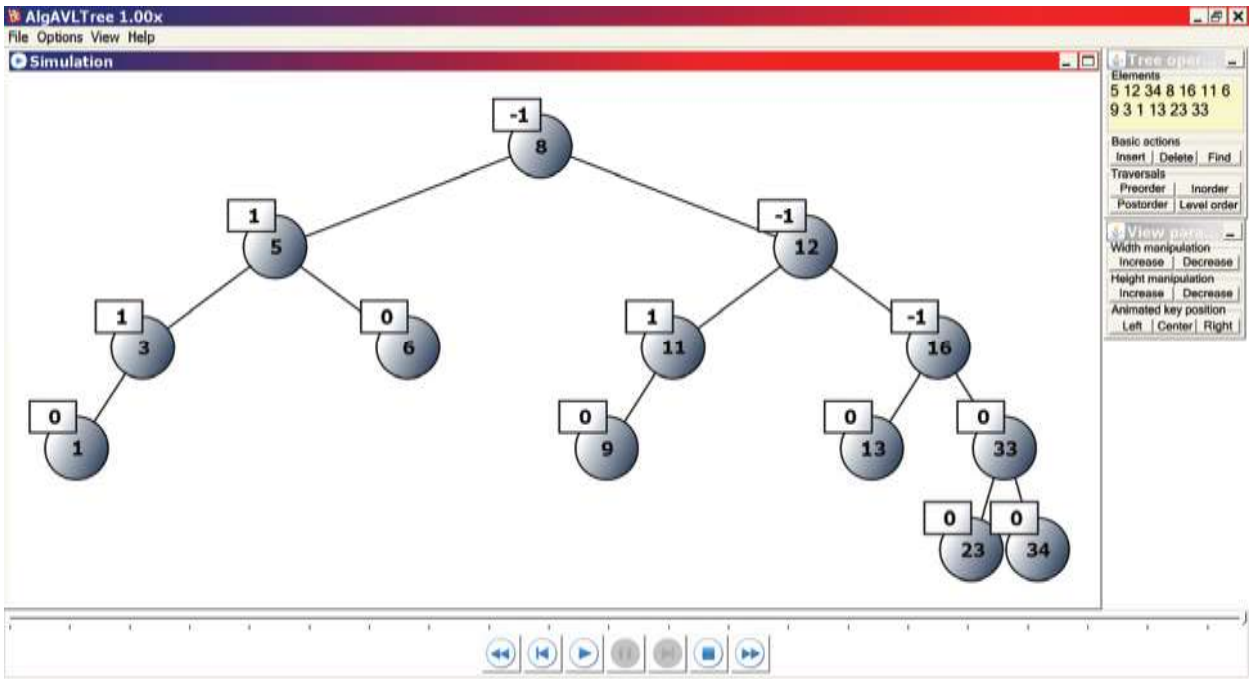


Рисунок 4 – Примеры с помощью инструмента VSA для отображения выполнения некоторых алгоритмов, а) нахождение минимального связующего дерева в данный граф, б) Встраивание ключей в AVL дерево.

V. ЗАКЛЮЧЕНИЕ

В этой статье мы представили архитектуру программного комплекса для обучения и тестирования знаний в областях, где решение проблем процедура очень важная, в отличие от традиционных систем тестирования. Предлагаемая система обеспечивает самостоятельное, контролируемое, и пассивное обучение, позволяющее пользователям задать параметры и данные для алгоритма исполнения. Следовательно, пользователи также имеют возможность проверить свои знания и понимание в конкретных ситуациях, которым преподаватель не уделял (достаточно) внимания. В тестовом режиме, система может оценить последовательность действий, указанных пользователем, которые лишь частично согласованы по заданному алгоритму, и оценить вероятность успеха этих действий. Кроме того, это дает улучшенные знания и понимание продемонстрированные пользователем.

Кроме запланированного введения системы в процессе алгоритма и структуры данных в школы электротехники в Белграде, далее совершенствование системы влечет развитие следующих компонентов: производительность оценки алгоритма (включая возможность сравнения производительности связанных алгоритмов), Статистический анализ результатов испытаний, индивидуальный прогресс мониторинга для каждого студента. Для того чтобы завершить системы, все соответствующие алгоритмы должны быть реализованы, а так же должны быть созданы в тестовом хранилище.

Указанные особенности мониторинга прогресса могут быть реализованы посредством соединения с существующими виртуальными средами обучения (Moodle), что уже обеспечивает соответствующую поддержку. Такое соединение, со стабильной, гибкой и широкодоступной виртуальной окружающей средой, будет обеспечивать несколько преимуществ, наиболее важными из которых повысить рейтинг системы и используемости, и мониторинга прогресса. Развитие соединительного модуля включает в себя реализацию IMS QTI стандарта внутри системы, а также публикация ключевых функциональных особенностей системы как веб-услуг.

Ссылки

[1]П. Dillenbourg, Д. К. Шнайдер, П. Synteta, "виртуальное обучениеСреды ", в кн. 3-й Греческой конференции по информационным и коммуникационные технологии в образовании, стр. 3-18, 2002.

[2]http://en.wikipedia.org/wiki/Virtual_learning_environния# List_of_some_virtual_learning_environments

[3]Moodle Тип вопроса - Java Molecule редактор,http://moodle.org/mod/data/view.php?d=13&rid=296

[4] QTI IMS, IMS Global Learning Consortium Вопрос и испытания Взаимодействие Спецификация (QTI), 2005,http://www.imsglobal.org/question/.

[5] WM Дэвис, HC Дэвис, "Проектирование инструментов оценки Сервис-ориентированная архитектура ", в сб. 1-я Международная ELeGIКонференция по современным технологиям для расширенного обучения Неаполь, Италия, 2005 год.

[6] Moodle.org: с открытым исходным кодом общинные инструменты для обучения,http://moodle.org/

[7] Dokeos - Open Source E-Learning, http://www.dokeos.com/.

[8]Сакаи проекта - пакета с открытыми источник знаний, портфолио, библиотека и средства управления проектами, http://sakaiproject.org/

[9] OLAT - Ваш источник открытого LMS, http://www.olat.org/

[10] И. Mićanović, "Разработка и внедрение VSA ядро,« бакалавр диссертация (на сербском), Школа электротехники, Университет Белград, 2009 год.

[11]M. Bjegović, "Проектирование графического интерфейса и представления данных для VSA" бакалавра диссертации (на сербском), Школа электротехники, Белградский университет, 2009.

[12]Д. Джуришич "Разработка и внедрение основных передовых VSA "бакалавра диссертации (на сербском), Школа электротехники, Белградский университет, 2009.

[13]M. Milivojević, "Разработка и внедрение передовых графических интерфейс для VSA, «бакалавр диссертации (на сербском), Школа Электротехника, Белградский университет, 2009.