ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У «Б е л Г У»)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА АДАПТИВНОГО ПЕДАГОГИЧЕСКОГО ТЕСТИРОВАНИЯ

Магистерская диссертация
обучающейся по направлению подготовки 02.04.01 Математика и компьютерные науки очной формы обучения, группы 07001531
Атрахименок Яны Михайловны

Научный руководитель
к.т.н., доцент
Румбешт В.В.

Рецензент
к.т.н., доцент
Заливин А.Н.

БЕЛГОРОД 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях современной образовательной деятельности высших учебных заведений преподавателям затруднительно применять индивидуальный подход к каждому студенту, т.к. численность академической студенческой группы, в среднем, около 20 человек, а для применения индивидуального подхода группа обучающихся должна составлять около 5-7 человек. В настоящее время, одним из способов реализации индивидуально-ориентированного обучения в образовательных учреждениях является использование систем электронного обучения (СЭО), работающих на основе адаптивных алгоритмов, способных формировать индивидуальные образовательные траектории за счет учета расширенного набора параметров обучаемых (психофизиологических характеристик, уровня фундаментальных базовых знаний, механизмов адаптивного тестирования, специфики восприятия информации и т.д.).

В настоящее время дистанционное образование охватывает все более широкие массы учащихся. В этих условиях особую актуальность приобретает применение информационных технологий в сфере образования и, в частности, средств автоматизации контроля знаний студентов.

Преимущество адаптивного теста перед традиционной формой тестирования - его эффективность. Адаптивный тест может определить уровень знаний тестируемого с помощью меньшего количества вопросов. При выполнении одного и того же адаптивного теста тестируемые с высоким уровнем подготовки и тестируемые с низким уровнем подготовки увидят совершенно разные наборы вопросов: первый увидит большее число сложных вопросов, а последний - легких. Доли правильных ответов у обоих могут совпадать, но так как первый отвечал на более сложные вопросы, то он наберет большее количество баллов.

Одной из важнейших задач в области контроля знаний является автоматизация синтеза и оценки сложности тестовых заданий. В соответствии с этим, было принято решение по разработке автоматизированной системы адаптивного педагогического тестирования, являющейся средством генерации различного рода тестов из прикрепленных баз данных. Основным требованием к разработанной системе являлось ее интеллектуальность, достигающаяся за счет организации адаптивности процесса тестирования.

Актуальность темы магистерской работы характеризуется тенденцией применения индивидуально-ориентированных методик в образовательном процессе, что в свою очередь приводит к необходимости совершенствования алгоритмического обеспечения систем педагогического тестирования.

Целью работы является повышение эффективности образовательного процесса, посредством разработки и реализации метода синтеза и оценки сложности тестовых заданий и построения индивидуального сценария тестируемого.

Задачи:

- провести анализ существующих систем автоматизированного тестирования;
- разработать математический аппарат решения адаптивного тестирования;
- создать модель системы адаптивного тестирования, написать ее программную реализацию и обеспечить эффективность функционирования.

Магистерская работа состоит из введения, трех глав, заключения, используемой литературы и приложения.

В первой главе приводится анализ проблемы автоматизации педагогического тестирования, а также существующих систем электронного обучения, в ходе которого рассматриваются различные адаптивные технологии и методики, использующиеся в них.

Во второй главе диплома осуществляется разработка методики разбиения банка тестовых задания по уровням сложности, используя такой математический аппарат, как теория нечетких множеств Также одной из основных наработок является создание методики построения траектории тестируемого.

В третьей главе выпускной квалификационной работы рассматривается реализация автоматизированной системы адаптивного педагогического тестирования, основанной на созданных методиках магистерской работы.

В заключении подводятся итоги исследования, формируются окончательные выводы и рекомендации по применению разработанных алгоритмов.

Данная работа состоит из 58 страниц и 1 приложения, содержащих 21 рисунок и 2 таблицы.

1. КОМПЬЮТЕРНОЕ АДАПТИВНОЕ ТЕСТИРОВАНИЕ

Стимулом к появлению адаптивного тестирования стала нацеленность на повышение эффективности педагогических измерений, которая обычно была связана с сокращением количества заданий, времени, стоимости тестирования, а также с увеличением процента точности оценок учащихся. Адаптивный подход основывается на уникальности процедуры отбора заданий теста, которая обеспечивает синтез эффективных тестов благодаря оптимизации трудности заданий применительно к уровню подготовленности обучаемых.

Совершенствование (оно же оптимизация) трудности заданий, как правило, осуществляется пошагово. Если учащийся выполняет задание правильно, то после него ему дается более трудное. В случае неправильного выполнения задания происходит откат назад к более легким заданиям банка. Если не выполнено три задания подряд, процесс прекращается и специальными методами (чаше всего с помощью теории IRT) выводится балл учащегося за выполненные задания по сформированному специально для него адаптивному тесту. То есть в компьютерном адаптивном предъявлении количество тестовых заданий и их трудность подбираются персонально для каждого экзаменующегося на основе его ответов, а индивидуальная совокупность заданий образует адаптивный тест. Каждый из них в группе испытуемых представляет собой совокупность разных заданий и отличается от другого количеством и трудностью. И чем больше разброс среди испытуемых группы по подготовленности, тем выше эта разница. Добиться одновременного прироста эффективности измерений по всем критериям не представляется возможным, поэтому, как правило, при организации адаптивного тестирования учитывается один или, в лучшем случае, два критерия. Так, в некоторых случаях при быстрой диагностике в адаптивном режиме в приоритете минимизация времени испытания и число

предъявляемых заданий, а вопросы точности оценок становятся чуть менее важны. В других случаях внимание уделяется преимущественно точности измерения и тестирование каждого испытуемого длится, пока не будет достигнута запланированная минимальная ошибка измерения.

На продолжительность адаптивного теста значительно влияет качество структуры знаний учащихся. Обычно, если у них четкая структура знаний, они отвечают на тестовые вопросы нарастающей трудности, уточняя по мере правильного выполнения заданий оценку своей подготовленности. При небольшом количестве заданий адаптивного теста испытуемые быстро доходят до порога своей компетентности. Те, у кого структура знаний нечеткая и чередуются правильные и неправильные ответы, получают колеблющиеся по трудности задания. В этом случае длительность тестирования увеличивается, так как скачкообразное изменение трудности заданий делает невозможным пошаговое нарастание точности измерения, и количество тестовых вопросов, адаптированных по трудности, зачастую возрастает, тогда как в обычном, традиционном тесте всё идёт нормально.

Среди важных преимуществ компьютеризованного адаптивного тестирования можно отметить:

- высокую эффективность;
- высокий уровень секретности;
- персоналиизацию скорости выполнения теста;
- высокий уровень мотивации к тестированию у менее способных обучающихся благодаря исключению из процесса предъявления чрезмерно трудных заданий;
- - осведомление о результате в интервальной шкале тестовых баллов каждого тестируемого сразу же после завершения его работы над индивидуально подобранным набором заданий в адаптивном тесте.

Стратегии предъявления тестовых заданий в адаптивном тестировании делятся на двухшаговые и многошаговые, в соответствии с которыми

применяется различная технология составления адаптивных тестов. Двухшаговая стратегия предполагает состоит из двух этапов. На первом из них всем учащимся выдается одинаковый входной тест, его цель заключается в осуществлении предварительного их распределения вдоль оси переменной измерения. По его результатам на втором этапе включается адаптивный режим и формируются адаптивные тесты.

В результате развития теории IRT, отвечающей за единую интервальную шкалу для оценок параметров испытуемых и трудности заданий теста, стало реальным новое осуществление оптимизации процедуры выбора заданий для моделирования результативных адаптивных тестов. Стало совершенствоваться многошаговые стратегии адаптивного тестирования, которые в процессе выполнения наборов заданий позволяют каждому испытуемому двигаться по своему индивидуальному пути.

Многошаговые стратегии адаптивного тестирования состоят из фиксировано-ветвящихся и варьирующе-ветвящихся. Это зависит от того, как конструируются многошаговые адаптивные тесты. Если для всех испытуемых применяется одинаковый набор тестовых вопросов с их фиксированным расположением на оси трудности, но путь каждого из учащихся в выполнении очередного задания индивидуален, стратегия адаптивного тестирования становится фиксировано-ветвящейся.

Все задания по трудности в наборе, как правило, располагаются на одном и том же расстоянии друг от друга либо же можно выбрать убывающий шаг сообразно нарастанию трудности. Последнее даёт возможность подстроить скорость прохождения тестирования под испытуемого, так как в процессе выполнения заданий он утомляется и теряет мотивацию к переходу к следующему этапу теста.

В рамках варьирующе-ветвящейся стратегии адаптивного тестирования происходит отбор заданий непосредственно из банка по определенным алгоритмам, прогнозирующим оптимальную трудность последующего задания, в зависимости от результатов выполнения испытуемым

предыдущего задания адаптивного теста. Таким образом, постепенно из отдельных заданий складывается адаптивный тест. В нем могут быть разными степени трудности и шаг, определяемый разностью трудностей двух соседних заданий адаптивного теста. Варьирующая-ветвящаяся стратегия адаптивного тестирования отличается главным образом пошаговой переоценкой уровня подготовленности испытуемого, которая происходит сразу же после каждого выполнения очередного задания теста.

Вход и выход из адаптивного тестирования. Выбор начальных оценок для входа в адаптивное тестирование проводится разными способами, в зависимости от вида стратегии и имеющихся технологических возможностей при генерации адаптивных тестов. В основе одного из методов определения начальных оценок лежит выдача испытуемым перед началом адаптивного тестирования входного претеста. Он обычно состоит из 5-10 заданий из различных разделов содержания, включающий по трудности весь спектр предполагаемого расположения тестируемой выборки учащихся на оси переменной измерения.

Чтобы вывести испытуемого из режима тестирования, ограничивают время или число заданий.

1.1 Анализ методов адаптивного тестирования

В [1,2,11] проведён анализ отличительных особенностей автоматизированного тестирования по сравнению с его классическими видами:

- каждый испытуемый получает индивидуальный набор заданий с разными содержанием и длиной теста;
- каждый испытуемый оценивается индивидуально, в зависимости от его уровня.

Главные плюсы адаптивного тестирования, отличающие его от классических форм тестирования:

- вероятность более точной оценки способности тестируемого с наименьшими затратами;
- с точки зрения педагога, время тратится более разумно, а значит, влияние на результаты дополнительных факторов, таких как утомление, волнение, неаккуратность, уменьшается;
- прямая и непосредственная интерактивность между студентом и преподавателем.

Проблемы организации тестирования связаны двумя аспектами – методическими и техническими. К методическим аспектам можно отнести :

- подбор заданий для контроля знаний, умений и навыков тестируемого;
- планирование процесса тестирования;
- определение требований к составлению перечня вопросов и заданий для опроса и другое.

К техническим аспектам относятся:

- формирование набора контрольных заданий, в зависимости от выбранного подхода;
- выбор и применение параметров тестирования;
- выбор алгоритма для оценки знаний испытуемых.

Таким образом, для составления адаптивного тестирования необходимы разные методы его проведения, а также оценки знаний, умений и навыков испытуемого по результатам выполнения им контрольных заданий.

1.1.1. Методы проведения тестирования

Методы проведения тестирования отличаются друг от друга уровнем адаптации к индивидуальным особенностям тестируемых и по способам

выбора параметров процесса тестирования. Существуют методы частичного и полного адаптивного тестирования [5].

В рамках частичного адаптивного тестирования порядок и количество тестовых заданий для сильных, средних и слабых тестируемых различается. Число и сложность заданий выбираются в зависимости от ответов испытуемого, уровня его подготовленности и/или на основе специально разработанных траекторий проведения контроля знаний. Для этого применяются модели обучаемого. Отчасти адаптивные методы реализуются, благодаря использованию случайной выборки, с учетом различных параметров модели обучаемого, на основе ответов испытумого, на основе модели учебного материала [3], с использованием модульно-рейтингового метода [6- 7- 8].

Полное адаптивное тестирование даёт возможность организации индивидуальной проверки знаний каждого тестируемого. Полные адаптивные методы используют модели обучаемого и модели предметной области [9].

Способы построения траектории тестирования у методов проведения тестирования также раазличаются:

- с помощью математической теории педагогических измерений (Item Response Theory, IRT);
- с применением нейросетевых методов;
- заданием переходов между состояниями.

В основе теории IRT лежит теория латентноструктурного анализа. Для формирования сценария тестирования применяется метод максимизации информации, который предполагает подбор каждого последующего задания из банка тестов как наиболее эффективный для оценивания уровня подготовленности данного тестируемого [4].

Для построения траекторий применяются также нейросетевые методы (нейросетевые классификаторы и анализаторы). Так, в работе [10] предлагается использование метода, основанного на итерационном процессе

интерполяции функции трудности заданий. Его идея заключается в том, что на основе анализа ответов тестируемого выявляют их сложности для данного тестируемого. Далее оптимальным способом интерполируют эту сложность на общее количество тестовых заданий. На основе анализа поведения интерполирующей функции тестируемый получает задания, в которых эта функция максимально «нерегулярна».

Формирование траекторий на основе заданных переходов позволяет тестируемому перейти в состояние, наиболее приближенное к состоянию оптимальной образовательной траектории (индивидуальной для каждого тестируемого) с помощью алгоритмов управления. С «технологической» точки зрения в качестве алгоритмов управления применяют следующие методы [11]:

- методы управления на основе сетей Петри (одноуровневые и вложенные сети Петри) [11];
- методы управления на основе вероятностных моделей (байесовские сети [12] и цепи Маркова)
- методы управления на основе конечноавтоматной модели [11].

1.1.2. Методы оценивания результатов тестирования

Методы оценивания итогов тестирования применяют для оценки текущего и конечного результата тестирования. С точки зрения способа вычисления оценки методы оценивания делятся на три группы [5]:

- методы на основе количественных критериев;
- методы на основе вероятностных критериев;
- методы на основе классификационных критериев.

В методах на основе количественных критериев используются количественные шкалы, т. е. оценка в этом случае зависит напрямую от числа. Она может представлять собой сумму баллов, которые тестируемый получил за за верныеответы при прохождении заданий. В более трудных

случаях на оценке сказываются типы и характеристики заданий, а также характеристики выполнения данных заданий испытуемым [13] (время тестирования) [14], количество верно выполненных заданий и попыток их выполнитьи др.).

От методов на основе вероятностных критериев как функций уровня подготовленности тестируемого и параметров задания зависят вероятности правильных ответов. Эти методы базируются на классической теории тестов (Classical Test Theory — CTT модели) [15] и IRT-модели [4].

В рамках методов на основе классификационных критериев осуществляется отнесение тестируемого к одному из нескольких устойчивых классов с учетом суммы признаков, определяющих данного тестируемого. Примерами выступают методы на основе алгоритма вычисления оценок (метод АВО) [16], а также методы на основе нечетких множеств [17].

1.1.3. Методы проверки результатов тестирования

На разницу методов проверки результатов тестирования влияет порядок прохождения заданий в тестировании. С этой точки зрения есть два подхода к способу проверки результатов:

- адаптивное тестирование с постоянной адаптацией;
- адаптивное тестирование с блочной адаптацией.

Адаптивное тестирование с постоянной адаптацией (детерминированно ветвящаяся стратегия) предполагает принятие решения об изменении последовательности заданий на каждом уровне тестирования.

Адаптивное тестирование с блочной адаптацией (варьирующаяся ветвящаяся стратегия) — это тестирование, при котором принятие решения об изменении последовательности заданий происходит после анализа результатов обработки определенного специального блока заданий.

1.1.4. Правила окончания тестирования

Компьютерное адаптивное тестирование предполагает следующие правила его окончания:

- необходимый уровень подготовленности тестируемого;
- число заданий;
- время тестирования.

В качестве критерия требуемого уровня подготовленности испытуемого чаще всего используется правило, основанное на достижении требуемой точности измерений уровня подготовленности тестируемого. Также используются минимальный и максимальный пределы уровня подготовленности тестируемого. Реже применяются такие правила «стоп», как фиксированное время тестирования и фиксированное количество заданий.

В итоге проведенного анализа выделен набор характеристик, влияющий на способ организации автоматизированной системы: цель тестирования, способы построения набора заданий тестирования, методы проведения тестирования, способы проверки тестирования, методы оценивания и правила окончания тестирования.

Цель тестирования является важным фактором для формирования адаптивного тестирования, так как влияет на его содержание. Цели тестирования можно разделить на тестирование знаний по отдельной теме или разделу и по всему курсу.

В зависимости от того, какая поставлена цель, необходимо подготовить тестирующие материалы, т. е. сформировать способ построения перечня заданий для проверки знаний, умений и навыков тестируемых. Существуют два подхода к способу построения набора заданий:

- тематический — учебный курс включает несколько разделов, связанных между собой и содержащих свой набор заданий

- задачный — учебный курс представляет собой набор заданий с разными характеристиками (тип, сложность, и др.)

Методы проведения тестирования различают по способу выбора первого задания и построения траектории тестирования.

От способа выбора первого задания зависит первое задание, способное задаваться по умолчанию (задание среднего трудности) или выбираться индивидуально для каждого тестируемого (или группы тестируемых) на основе известной информации об этом тестируемом.

Метод построения сценария тестирования определяет способ составления перечня заданий на основе обратной связи с тестируемым. Этот способ даёт возможность организации проверки знаний персонально для каждого тестируемого, поддерживая оптимальный для тестируемого уровень сложности заданий и формируя индивидуальные траектории тестирования.

Методы проверки результатов тестирования влияют на порядок прохождения заданий в процессе него

Методы оценивания дают возможность проводить контроль знаний тестируемых и определять уровни знаний, умений и навыков испытуемых по результатам выполнения контрольного задания тестирования, а также по завершении процесса тестирования.

Правила окончания тестирования определяют условия завершения процесса тестирования. В качестве условия окончания тестирования используют фиксированное время тестирования, фиксированное количество заданий и требуемый уровень подготовленности тестируемого. Правила окончания тестирования можно комбинировать и получать, таким образом, новые правила.

Рассмотренные характеристики использованы как классификационные признаки при построении многоаспектной классификации адаптивного тестирования (см. рис.1.1).

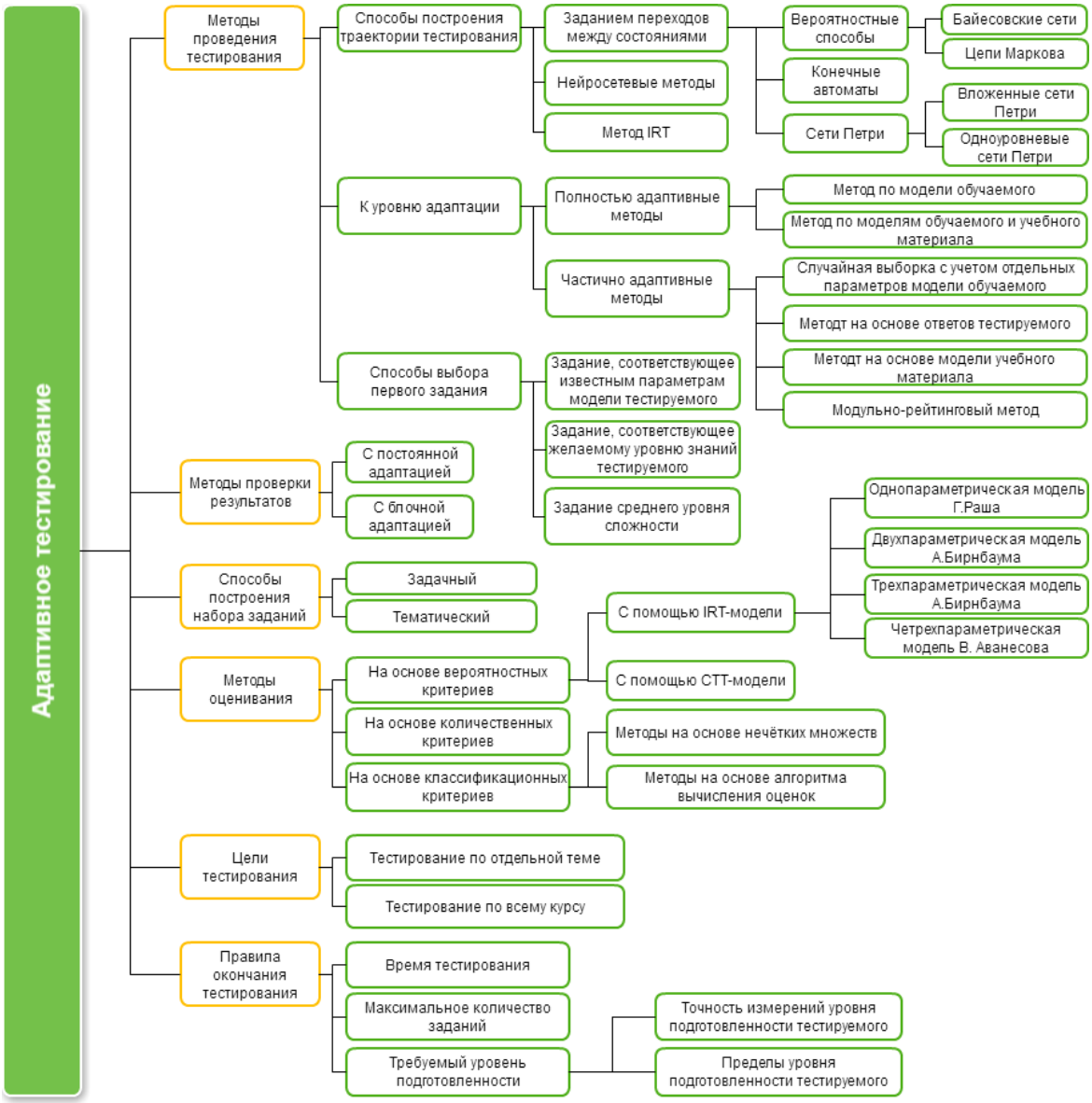

Рис.1.1. Классификация адаптивного тестирования

Существует множество автоматизированных систем обучения и контроля. Наиболее часто встречаются так называемые обучающие программы, разработанные на основе эмпирического подхода, определенный педагогический опыт и здравый смысл (системы "от учебного предмета"). Как правило, у них низкая дидактическая эффективность: по зарубежным оценкам, эффективными являются не более 10% таких программ, а число непригодных – около 90%. Процент автоматизированных систем, которые используют адаптивное тестирование, очень не велик, хотя в большинстве источников считают данный способ наиболее эффективным, поэтому в настоящее время в области исследований, связанных с адаптивным

тестированием, идёт процесс поиска новых методов и технологий или их синтез.

Рассмотрим некоторые наиболее известные автоматизированные системы:

1. АСТ-Тест – инструментальная среда для разработки педагогических тестов и адаптивного тестирования с использованием OLE-технологии и мультимедиа. Имеет модули "Конструктор тестов", "Система тестирования". В системе осуществляется выбор сложности и его генерация до запуска тестирования, а также жёстко задано количество вопросов в тесте.

2. Гефест – сетевая адаптивная информационно-обучающая система, использующая методы теории автоматов и марковских процессов. В модель адаптивного управления обучением включены объекты "Устройство адаптивного обучения (формирование вопросов и задач, контроль ответов и оценка знаний)", "Модель обучающегося", таким образом данная система также содержит информация, необходимую для начала тестирования, в которую входит и уровень знаний испытуемого.

3. М-Тест – инструментальная среда для поддержки адаптивного тестирования и аттестации сотрудников. Позволяет конструировать мультимедийные задания основных форм, используя технологию связывания объектов OLE, создавать банки таких заданий, визуализировать результаты тестирования (протоколирование), вести статистику.

4. IRT–технология (методология) адаптивного тестирования, получившая название "Тест интеллектуального потенциала" для экспресс-диагностики интеллектуальных способностей людей различных возрастных групп.

Адаптивное тестирование должно быть открытым и постоянно расширяемым. Кроме этого, должен быть реализован адаптивный интеллектуальный выбор следующего тестового задания в зависимости от результатов решения предыдущих. Также, тестовые задания должны быть разделены по категориям трудности и иметь соответствующий коэффициент,

которым можно манипулировать в процессе адаптации. Таким образом, главным преимуществом адаптивного интеллектуального тестирования перед стандартным тестированием является его эффективность. Адаптивность обеспечивает индивидуальный подход к каждому студенту.

1.2. Классификация тестовых заданий и общие требования к ним

В отечественной и зарубежной литературе наиболее часто выделяют классификацию тестовых заданий по типу ответа:

1) традиционное задание с множеством ответов, в котором обучающийся должен выбрать единственный правильный;

2) задание с множеством ответов, в котором обучающийся должен выбрать ряд правильных ответов;

3) задание, требующее от обучающегося самостоятельное получение ответа; задание с ответом, который нужно сформировать;

4) задание, требующее от обучающегося установить соответствие среди ряда элементов взаимосвязанных множеств.

Обозначенные формы тестовых заданий наиболее популярны и являются базовыми. В зависимости от поставленных задач контролируемого предмета периодически возникает потребность в использование новых форм, которые отвечают специфике разработки теста. В основе таких форм лежит сочетание простых элементов типовых заданий, и , как правило, представляются такие инновации в виде комбинаторики общепринятых четырёх типов заданий.

Для дифференциации и отбора заданий вводится понятие предтестового задания. По М.Б. Челышковой предтестовое задание – это единица контрольного материала, содержание, логическая структура и форма представления которого удовлетворяют ряду специфических требований и обеспечивают однозначность оценок результатов испытуемых в выбранной шкале.

Предтестовые задания проходят испытания опытным путём, опирающимся на непосредственное наблюдение, по итогом которого принимается решение о переходе предтестового задания в тестовое. Не прошедшие эмпирическую проверку должны быть удалены из изначального множества заданий теста.

Вне зависимости от формы предтестовые задания должны отвечать следующим общим требованиям к процессу своего применения для решения поставленных задач:

- предтестовое задание должно иметь порядковый номер, который зависит от выбора политики предъявления заданий теста. Необходимо учесть возможность изменения оценки трудности задания после статистической апробации;
- предтестовое задание имеет стандарт правильного ответа (стандарт оценивания для упражнений с ответом, который нужно сформировать);
- все элементы предтестового упражнения должны быть расположены на строго определенных местах, которые должны быть зафиксированы в пределах выбранной формы;
- для предтестовых заданий должна быть разработана типовая инструкция по выполнению, которая в пределах каждой формы остаётся неизменной и определяет формулировку заданий, которые будут включены в тест;
- для заданий должен быть разработан принцип выставления оценки, общий для любой формы;
- наряду с заданием разрабатывается инструкция, в которой определены правила и условия подсчета первичных баллов и проверки.

В целях приведения к единому стандарту процедуры тестовых измерений выдерживаются ряд требований:

- обучающиеся находятся в равнозначных условиях по отношению друг к другу;
- система подсчета оценок без приоритетна без исключения к любому обучающемуся;
- тестирование испытуемых проводится в одинаковых условиях в одно и тоже время;
- мотивации испытуемых выравнена;
- одинаковые задания для всех обучающихся;
- задания подобраны с оптимальными весовыми коэффициентами;
- значения коэффициентов определяются статистическим путем.

Рассмотрим предтестовое задание 1 типа с множеством ответов, в котором обучающийся должен выбрать единственный правильный или ряд ответов

Задания этого типа называют закрытыми. В них выделяют основу, определяющую содержание проблемы, и множество вариантов ответов, подготовленных ранее преподавателем. Обучающимся необходимо выбрать единственный или ряд правильных.

На этапом этапе появляется необходимость введения понятия дистрактор, который является неправильным ответом, но правдоподобным. При подборе оптимального числа дистракторов учитывается фактор угадывания правильного ответа. Чаще всего количество дистракторов к правильным ответам соотносится как три к одному.

Преимущества данного типа:

- быстрота выполнения;
- элементарный подсчет результирующего балла;
- возможность автоматизации процесса проверки;
- применение для любой области любого контролируемого предмета.

Основным недостатком данного типа является эффект угадывания. Для исключения эффекта вводят повторяющиеся, сходные задания с

ранжированным коэффициентом, также проводятся инструктажи, разъяснительные работы перед проверкой, ориентированные на пропуск задания в случае отсутствия правильного ответа.

Перейдём к исследованию заданий 2 типа с ответом, который нужно сформировать. Задания этого типа называют открытыми и бывают двух видов. В них обучающемуся готовые ответы не представлены, их необходимо получить или создать.

При решении заданий первого вида испытуемый должен получить ответ, строго упорядоченный по содержанию и форме представления. Это может быть число, символ, слово, словосочетание, формула и т.д. Фактор краткости сужает сферу применения этого типа упражнений. Как правило, с их помощью контролируются навыки воспроизводить и применять познанения в типовой знакомой ситуации, а также выявляются уровень понимания контролируемого предмета.

При решении заданий второго вида испытуемому необходимо дать развёрнутый, свободно конструируемый ответ.

Второй — задания со свободно конструируемыми ответами, в которых учащиеся составляют развернутые ответы, произвольные по длине и форме представления и содержащие полное решения задачи с пояснениями, микросочинения (эссе) и т.д. [4].

Для разработки заданий с конструируемым регламентированным ответом необходимо мысленно сформулировать вопрос, затем записать четкий и краткий ответ, в котором на месте ключевого слова, символа или числа ставится прочерк. В силу однозначности правильного ответа проверка результатов выполнения заданий с конструируемым регламентированным ответом носит довольно объективный характер, ее осуществляют в компьютерной форме с последующей перепроверкой всех неправильных ответов учащихся экспертным путем. Ответы на задания приводятся на месте прочерка или заносятся учащимся в специальный бланк.

Проведём анализ 3 типа заданий, требующее от обучающегося установить соответствие среди ряда элементов взаимосвязанных множеств.

Общая характеристика. Задания на соответствие имеют специфический вид: под инструкцией располагаются элементы двух множеств, соответствие между которыми предлагается установить учащемуся [4]; слева обычно приводятся элементы задающего множества, содержащего постановку проблемы; справа — элементы, подлежащие выбору.

Соответствие между элементами двух столбцов может быть взаимно однозначным, когда каждому элементу слева соответствует только один элемент справа. Если число элементов в двух столбцах одинаковое, то для последнего элемента задающего множества выбора не произойдет, поэтому в множество для выбора стараются включить несколько дистракторов.

Дискриминативность тестового задания

Дискриминативность (дифференцирующая способность, различающая способность) задания - это способность задания дифференцировать испытуемых по уровню достижений, на сильных и слабых. Если задание одинаково выполняется и слабыми, и сильными, то можно говорить о низкой дискриминативности задания. Если задание выполняется сильными испытуемыми, а слабые дают отрицательный результат, то мы имеем высокую дискриминативность. Высокая дискриминативность тестовых заданий важна для нормативно-ориентированных тестов, основная цель которых - ранжирование учащихся по уровню достижений.

Один из способов вычисления дискриминативности - вычисление с применением метода крайних групп, где для расчета берутся показатели самых слабых и самых сильных испытуемых. Чаще всего это 27 (30) % худших и 27 (30) % лучших по результатам выполнения тестового задания.

Индекс дискриминативности определяется как разность долей правильных ответов сильной и слабой групп.

$(r_{дис})_j = (p_1)_j - (p_0)_j$,

или

$(r_{дис})_j = ((P_1)_j - (P_0)_j)/100\%$, если трудность задана в процентах

где r - индекс дискриминативности, p_1 - доля правильных ответов в сильной подгруппе (27 % от всего количества), p_0 - доля правильных ответов в слабой группе (27 %). Значение индекса дискриминативности располагается в интервале [- 1; 1]. Если индекс дискриминативности выше нуля (больше 0,3 считается удовлетворительным), а еще лучше стремится к 1, то это свидетельствует о том, что задание обладает хорошим (максимальным) дифференцирующим эффектом. Если r = 0, то это значит, что и слабые, и сильные испытуемые выполняют задание одинаково. Отрицательный показатель дискриминативности, появляется в том случае, когда слабые учащиеся выполняют задание правильно, а сильные - неправильно, что свидетельствует о некачественном (невалидном) задании.

Трудность задания в классической теории тестов определяется через соотношение количества испытуемых, справившихся с данным заданием, и общего количества испытуемых, т.е. трудность задания - это доля учащихся, которые справились с заданием.

$p_j = \frac{Y_j}{N}$ - трудность задания вычисляется по формуле

где p_j - *доля правильных ответов на j-ое задание;* Y_j - *количество испытуемых, выполнивших j - ое задание верно,* N - *число испытуемых в группе, j - номер задания. Или в процентах,* P_j *– трудность j – ого задания в процентах:* $P_j = \frac{Y_j}{N} \cdot 100\%$

Из формулы видно, что чем выше показатель трудности, тем задание легче, и соответственно, чем меньше показатель трудности задания, тем задание сложнее. Например, если p = 30 %, то это значит, что только 30% испытуемых справились с этим заданием, а если p = 70 %, то 70 % справилось с заданием, и получается, что первое задание сложнее, чем второе.

Иногда вводится доля неправильных ответов - *q*, которая определяется по формуле *q = 1 – p.*

Но по сложившейся традиции в рамках классической теории тестов трудность задания определяется как доля правильных ответов (p).

Показатель трудности очень важен для определения характеристики тестового задания и помогает проранжировать задания, входящие в тест по степени сложности. Благодаря этому можно определить место задания в тесте. Напомним, что в правильно сконструированном тесте задания должны располагаться по нарастанию сложности, т.е. сначала даются самые легкие, далее все сложнее и сложнее. В хорошо сбалансированном по трудности тесте есть несколько самых трудных заданий со значением $p \to 0$. Есть несколько самых легких с $p \to 1$. Остальные задания по значениям *p* занимают промежуточное положение между крайними ситуациями и имеют в основном трудность 60 - 70 % в критериально-ориентированном тесте и 40-60 % в нормативно-ориентированном.

В рамках нормативно-ориентированного подхода наиболее удачными считаются задания средней трудности p=q=0,5, которые обеспечивают максимальную дисперсию теста дисперсия $\sigma = pq$.

Это произведение достигает максимального значения (0,5 x 0,5 = 0,25) при p = 0,5.

1.3. Постановка задачи

В рамках выполнения магистерской работы стоит задача изучить принципы компьютерного тестирования, изучить различные автоматизированные системы для обучения и тестирования, оценить преимущества и возможные недостатки адаптивного тестирования перед традиционным, рассмотреть методы индивидуально-ориентированного тестирования, осуществить подбор оптимального математического аппарата

для решения задач адаптивного тестирования и подбор методов, которые будут использоваться при разработке автоматизированной системы.

После изучения имеющихся на настоящий момент разработок, проектов, систем по теме материала были акцентированы следующие направления:

- разработка метода оценки сложности тестовых заданий при априорно заданных субъективных оценках сложности вопросов, основанного на теории нечетких множеств;
- разработка способ построения траектории тестирования с помощью теории конечных автоматов;
- разработка правила окончания тестирования и выставление оценки испытуемому.

Исходя из имеющихся навыков, знания и опыта в разработке программ, проанализировав востребованность на рынке, наиболее подходящим программным продуктом для реализации системы тестирования является C++Builder.

Borland C++Builder - это мощная и надежная среда быстрой разработки высокоэффективных web-служб и приложений для электронного бизнеса. Кросс-платформенная библиотека компонентов CLX предоставляет обширные возможности для разработки высокопроизводительных Windows-приложений, переносимых на платформу Linux с минимальными изменениями [4].

Borland C++Builder был выбран в качестве среды для разработки приложения по следующим преимуществам:

- Во-первых, позволяет быстро решать поставленные задачи с помощью набора стандартных классов,
- Во-вторых, позволяет генерировать «безопасный код» и избежать большую часть ошибок, возникающих при использовании динамической памяти.

- В-третьих, используемая технология ADO позволяет достаточно просто подключиться к базе данных, что крайне необходимо в процессе выполнения магистерской работы.
- В-четвертых, немало важным фактором, является наиболее большой опыт работы в данной среде.

Конечной целью является разработка проекта и программная реализация системы адаптивного тестирования.

Подводя итог первой главы можно сделать следующий вывод, что к настоящему времени разработано достаточно большое количество методов и алгоритмов компьютерного тестирования, но так как по статистике эффективность их приравнивается к 10 %, то разработаем собственную методику оценивания сложности тестовых заданий и выработаем индивидуальный алгоритм построения траектории тестируемого.

2. МАТЕМАТИЧЕСКИЙ АППАРАТ ЗАДАЧИ АДАПТИВНОГО ТЕСТИРОВАНИЯ

2.1 Настройка сложности тестового вопроса

В настоящее время дистанционное образование охватывает все более широкие массы учащихся. В этих условиях особую актуальность приобретает применение информационных технологий в сфере образования и, в частности, средств автоматизации контроля знаний учащихся.

Одной из важнейших задач в области контроля знаний является оценки сложности тестовых заданий. Обычно при составлении тестовых заданий заготавливается список вопросов, имеющих отношение к определенной области знаний. Далее методист выбирает из списка вопросы так, чтобы получившиеся тесты имели приблизительно одинаковый уровень сложности и были ориентированы на среднего студента.

Создание такой системы требует привлечения математического аппарата, позволяющего учесть субъективность оценок. Наиболее подходящим математическим аппаратом для решения локальной задачи магистерской работы является теория нечетких множеств [18].

Анализ принципов и основ теории для учёта субъективности оценивания уровня сложности тестового задания привел к выводу о перспективности и эффективности градации вопросов в расширенном диапазоне относительно классической системы оценивания.

Создание системы следует начинать с определения формальной постановки задачи:

1. Шкала сложности $Comp = \{1, 2, \ldots, 10\}$, содержащая десять оценок от 1 до 10. Значение оценки, равное 1, соответствует минимальной сложности, а значение, равное 10 – максимальной.

2. Конечное множество вопросов $Q = \{q_1, q_2, \ldots, q_m\}$.

3. Субъективная оценка сложности каждого вопроса $q_i \in Q,\ i = 1 \ldots m$, которая задана в виде нечеткого числа на шкале сложности.

Согласно [19, 20], нечетким числом называется нормальное и выпуклое нечеткое множество во множестве действительных чисел. В нашем случае, когда мы имеем дело со шкалой сложности, представляющей собой первые десять натуральных чисел, уместно назвать субъективную оценку сложности нечетким натуральным числом. Обозначим множество натуральных нечетких чисел на шкале сложности символом $FUZZY(Comp)$. Каждое нечеткое натуральное число, принадлежащее $FUZZY(Comp)$, является нечетким множеством и полностью определяется своей функцией принадлежности [18] $\mu : Comp \to [0, 1]$, удовлетворяющей следующим свойствам:

* $\exists\, c \in Comp,\ \mu(c) = 1$ (нормальность);

* $\forall\, c_1, c_2, c_3 \in Comp$, если $c_1 \le c_2 \le c_3$, то $\mu(c_2) \ge \min(\mu(c_1), \mu(c_3))$ (выпуклость).

В теории нечетких множеств принято рассматривать нечеткое множество, как совокупность пар вида: $< \mu(u), u >$, первая компонента которых – значение принадлежности элемента *u* данному нечеткому множеству, а вторая – сам этот элемент. Для удобства такие пары записываются следующим образом: $\mu(u) / u$. Таким образом, имеем:

$FUZZY(Comp) = \{\{\mu(c) / c \mid c \in Comp\} \mid \mu : Comp \to [0, 1]\}$.

Субъективную оценку сложности (далее просто сложность) вопросов формально будем рассматривать как отображение *C* из *Q* в *FUZZY(Comp)*:

$C : Q \to FUZZY(Comp)$.

Сложность конкретного вопроса *q* будем обозначать *C(q).*

4. Три особых оценки в *FUZZY(Comp)*:

EASY = {1/1, 0,9/2, 0,6/3, 0,3/4, 0/5, 0/6, 0/7, 0/8, 0/9, 0/10};

MIDDLE = {0/1, 0,3/2, 0,6/3, 0,9/4, 1/5, 1/6, 0,9/7, 0,6/8, 0,3/9, 0/10};

COMPLICATED = {0/1, 0/2, 0/3, 0/4, 0/5, 0/6, 0,3/7, 0,6/8 0,9/9, 1/10},

соответствующих таким субъективным понятиям, как "простой", "средний" и "сложный".

После формирования общих параметров задачи следует найти её решение.

Общая стратегия реализации первой части задачи заключается в том, чтобы сформировать разбиение $\aleph$ множества Q на простые, средние и сложные вопросы и, затем, выбрать из каждого элемента разбиения по заданному соответствующими параметрам теста количеству вопросов.

Для построения разбиения воспользуемся понятием степени включения нечетких множеств [21]:

$$\upsilon(A,B) = \mathop{\&}_{u \in U} (\mu_A(u) \to \mu_B(u)),$$

где A и B – нечеткие множества в универсуме U; $\mu_A(u)$ и $\mu_B(u)$ – значения принадлежности элемента $u \in U$ соответствующим нечетким множествам, которые в данном случае рассматриваются как нечеткие высказывательные переменные; "$\to$" – операция импликации нечетких высказываний $(a \to b = \max(1-a,b))$; "&" – операция конъюнкции ($a \,\&\, b = \min(a,b)$), которая берется по всем $u \in U$. Степень включения $\upsilon(A,B) \in [0,1]$ показывает насколько нечеткое множество A содержится в нечетком множестве B.

С помощью этой оценки определим степени включения сложности каждого вопроса в заданные понятия *EASY*, *MIDDLE* и *COMPLICATED*, причем будем их рассматривать как значения принадлежности соответствующих вопросов нечетким множествам простых, средних и сложных вопросов соответственно:

$$\forall q \in Q, \mu_{Q_E}(q) = \upsilon(C(q), EASY)$$

$$\forall q \in Q, \mu_{Q_M}(q) = \upsilon(C(q), MIDDLE)$$

$$\forall q \in Q, \mu_{Q_C}(q) = \upsilon(C(q), COMPLICATED)$$

В итоге получим три нечетких множества в универсуме Q: Q_E, – простые вопросы, Q_M – средние вопросы и Q_C – сложные вопросы, которые и представляют собой искомое разбиение: $\aleph = \{Q_E, Q_M, Q_C\}$.

Для проверки работоспособности обозначенной выше методики необходимо реализовать в рамках конкретного примера.

Пусть множество Q состоит из 10 вопросов: $Q = \{q_1, q_2, \ldots, q_{10}\}$. Сложность этих вопросов следующая:

$C(q_1)$={1/1, 1/2, 0,5/3, 0/4, 0/5, 0/6, 0/7, 0/8, 0/9, 0/10}

$C(q_2)$={1/1, 0/2, 0/3, 0/4, 0/5, 0/6, 0/7, 0/8, 0/9, 0/10}

$C(q_3)$={0/1, 0/2, 0,1/3, 0,5/4, 1/5, 0,5/6, 0,1/7, 0/8, 0/9, 0/10}

$C(q_4)$={0/1, 0/2, 0/3, 0,8/4, 1/5, 1/6, 0,8/7, 0/8, 0/9, 0/10}

$C(q_5)$={0/1, 0/2, 0/3, 0/4, 0/5, 1/6, 0/7, 0/8, 0/9, 0/10}

$C(q_6)$={0/1, 0/2, 0,5/3, 1/4, 0,5/5, 0/6, 0/7, 0/8, 0/9 0/10}

$C(q_7)$={0/1, 0/2, 0/3, 0/4, 0/5, 0,5/6, 1/7, 0,5/8, 0/9, 0/10}

$C(q_8)$={0/1, 0/2, 0/3, 0/4, 0/5, 0/6, 0/7, 0/8, 0/9, 1/10}

$C(q_9)$={0/1, 0/2, 0/3, 0/4, 0/5, 0/6, 0,3/7, 0,6/8, 0,9/9, 1/10}

$C(q_{10})$={0/1, 0/2, 0/3, 0/4 0/5, 0/6, 0/7, 0,5/8, 1/9, 0,5/10}

Для синтеза теста определим степени включения сложностей вопросов $q_1, q_2, \ldots, q_{10}$ в нечеткие понятия *EASY*, *MIDDLE* и *COMPLICATED*:

$\upsilon(C(q_1), EASY) = \min($ max*(1–1, 1)*, max*(1–1, 0,9)*, max*(1–0,5, 0,6)*, max*(1–0, 0,3)*, max*(1–0, 0)*, max*(1–0, 0)*, max*(1–0, 0)*, max*(1–0, 0)*, max*(1–0, 0)) = 0,6;*

$\upsilon(C(q_1), MIDDLE) = 0;$ $\upsilon(C(q_1), COMPLICATED) = 0;$

$\upsilon(C(q_2), EASY) = 1;$ $\upsilon(C(q_2), MIDDLE) = 0;$ $\upsilon(C(q_2), COMPLICATED) = 0;$

$\upsilon(C(q_3), EASY) = 0;$ $\upsilon(C(q_3), MIDDLE) = 0{,}9;$ $\upsilon(C(q_3), COMPLICATED) = 0;$

$\upsilon(C(q_4), EASY) = 0;$ $\upsilon(C(q_4), MIDDLE) = 0{,}9;$ $\upsilon(C(q_4), COMPLICATED) = 0;$

$\upsilon(C(q_5), EASY) = 0;$ $\upsilon(C(q_5), MIDDLE) = 1;$ $\upsilon(C(q_5), COMPLICATED) = 0;$

$\upsilon(C(q_6), EASY) = 0{,}3;$ $\upsilon(C(q_6), MIDDLE) = 0{,}6;$ $\upsilon(C(q_6), COMPLICATED) = 0;$

$\upsilon(C(q_7), EASY) = 0;$ $\upsilon(C(q_7), MIDDLE) = 0{,}6;$ $\upsilon(C(q_6), COMPLICATED) = 0{,}3;$

$\upsilon(C(q_8), EASY) = 0;$ $\upsilon(C(q_8), MIDDLE) = 0;$ $\upsilon(C(q_8), COMPLICATED) = 1;$

$\upsilon(C(q_9), EASY) = 0;$ $\upsilon(C(q_9), MIDDLE) = 0;$ $\upsilon(C(q_9), COMPLICATED) = 0{,}6;$

$\upsilon(C(q_{10}), EASY) = 0;$ $\upsilon(C(q_{10}), MIDDLE) = 0{,}3;$ $\upsilon(C(q_{10}), COMPLICATED) = 0{,}6;$

Тем самым мы сформируем нечеткие множества простых, средних и сложных вопросов:

$Q_E = \{0{,}6/q_1, 1/q_2, 0/q_3, 0/q_4, 0/q_5, 0{,}3/q_6, 0/q_7, 0/q_8, 0/q_9, 0/q_{10}\};$

$Q_M = \{0/q_1, 0/q_2, 0{,}9/q_3, 0{,}9/q_4, 1/q_5, 0{,}6/q_6, 0{,}6/q_7, 0/q_8, 0/q_9, 0{,}3/q_{10}\};$

$Q_C = \{0/q_1, 0/q_2, 0/q_3, 0/q_4, 0/q_5, 0/q_6, 0{,}3/q_7, 1/q_8, 0{,}6/q_9, 0{,}6/q_{10}\}.$

2.2 Организация адаптивного тестирования

Автоматизированный синтез тестовых заданий, даже при заранее заготовленных вопросах, не столь тривиальная задача, как кажется. Дело в том, что вопросы имеют различную степень сложности, которая, как правило, оценивается субъективно, да и оценка сложности полученного теста так же субъективна. Следовательно, система автоматизированного синтеза и оценки сложности тестовых заданий должна осуществлять выбор в условиях неопределенности, и ее можно отнести к классу интеллектуальных систем.

Для решения аспекта задач адаптивного тестирования, связанного с проблемой построения траектории тестирования, был выбран такой математический аппарат как теория конечных автоматов. Способ построения схемы тестирования, использующей данную теорию, позволяет оптимизировать количество заданий каждого тестируемого от уровня знаний.

Согласно [] автоматом называется дискретный преобразователь информации, способный принимать различные состояния, переходить под воздействием входных сигналов из одного состояния в другое и вырабатывать выходные сигналы. Если множество состояний автомата, а так

же множества входных и выходных сигналов конечны, то автомат называется конечным автоматом.

Для задания конечного автомата S необходимо описать все компоненты его кортежа $S(A, Z, W, \delta, \gamma)$, где $A = |a_0,\ a_1, .. a_n\ |$ - множество внутренних состояний автомата; $Z = |z_1,\ z_2, .. a_m\ |$ - множество входных сигналов (входной алфавит); $W = |w_1,\ w_2, \dots w_p\ |$ - множество выходных сигналов (выходной алфавит), δ – функция переходов, определяющая состояние автомата, а γ – функция выходов. Для описания конечных автоматов используют таблицы переходов-выходов и графы.

Рассмотри данную теорию на примере задачи магистерской диссертации. Пусть заданы следующие параметры:

$A = |a_0,\ a_1,\ a_3\ |$ - *множество состояний прохождения тестирования,*

$Z = |+, -\ |$ - *множество ответов тестируемых,*

$W = |\text{простой}, \text{средний}, \text{сложный}|$ - *множество уровней сложности,*

$\delta: A \times Z \to A$ *– функция переходов,*

$\gamma: A \times Z \to W$ *– функция выходов.*

Представим данный конечный автомат в виде графа, вершины которого являются состояниями, а ребра – переходами между ними (см.рис.2.1).

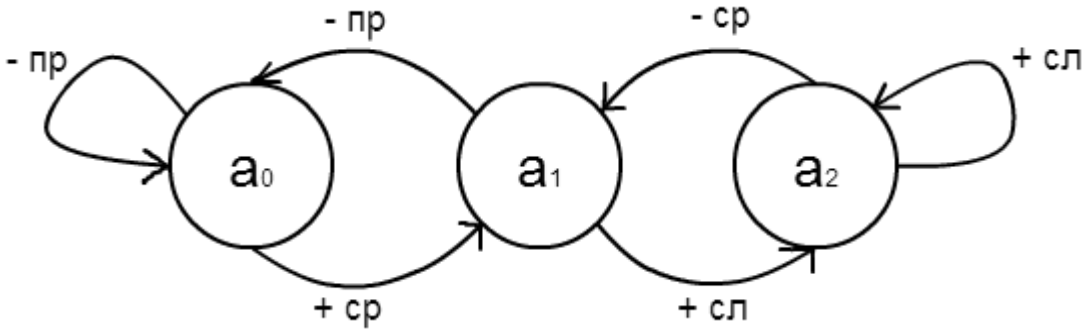

Рис.2.1. Граф автомата, описывающий адаптивное тестирование, при W=3

На данной диаграмме выходное значение зависит лишь от текущего состояния данного автомата и не зависит напрямую от входных значений,

поэтому данный автомат можно считать конечным автоматом Мура. В связи с этим таблицы переходов и выходов задаются вместе. Соответствующая этому автомату таблица представлена в табл.2.1.

Таблица 2.1

Таблица переходов-выходов автомата Мура

γ	**простой**	**средний**	**сложный**
δ	**a_0**	**a_1**	**a_2**
+	a_1	a_2	a_2
-	a_0	a_0	a_1

Данный вариант автомата является не единственным решением. В зависимости от уровней сложностей можно сделать и больше количество возможных состояний прохождения тестирования, тем самым большее количество переходов и выходов. Например, с 6-уровневой шкалой сложности траектория прохождения тестируемого будет выглядеть следующим образом, как представлено на рис. 2.1.

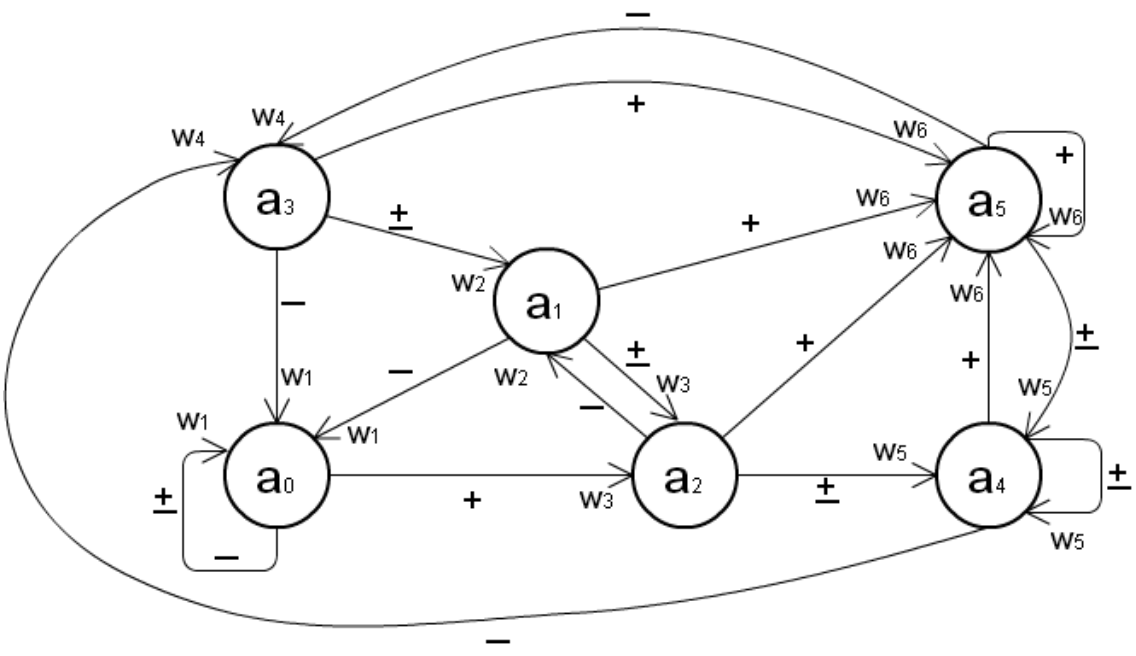

Рис.2.2. Граф автомата, описывающий сценарий тестирования при W=6

Использования данного математического аппарата дает возможность применить процедуру выбора первого задания среднего уровня для наибольшей адекватности построения дальнейшей траектории.

2.3. Вычисление функции сложности теста и выставление оценки

Определение уровня сложности получившегося теста позволит определить итоговую оценку тестирования. Наиболее подходящим математическим аппаратом для решения данной локальной задачи магистерской работы является также теория нечетких множеств.

Решение поставленной задачи состоит в вычислении функции сложности теста, которая определятся как $\tilde{C}: 2^Q \to FUZZY(Comp)$, где 2^Q – множество всех подмножеств Q. Для заданного теста T будем вычислять ее как среднее арифметическое сложности вопросов, составляющих тест:

$$\tilde{C}(T) = \frac{\sum_{q \in T} C(q)}{n}.$$

Поскольку в данном случае мы имеем дело с нечеткими натуральными числами требуется обобщение на нечеткий случай операций сложения и целочисленного деления. Применяя принцип обобщения обычных операций над числами на нечеткие числа [5] получим расширенные операции сложения и целочисленного деления. В результате их применения будут получаться нечеткие натуральные числа, имеющие следующие функции принадлежности:

$$\forall x, y, z \in \mathbf{N},\ \mu_{A+B}(z) = \sup\{\min(\mu_A(x), \mu_B(y)) \mid x + y = z\},$$

$$\forall x, y, n \in \mathbf{N},\ \mu_{A\ div\ n}(y) = \sup\{\mu_A(x) \mid x\ div\ n = y\},$$

где $\mathbf{N}$ – множество натуральных чисел; A и B – нечеткие натуральные числа; sup – наименьшая верхняя грань множества; *div* – операция целочисленного деления; n – заданное натуральное число.

Получившаяся в итоге сложность теста $\tilde{C}(T)$ позволяет дать ей лингвистическую оценку. Для этого необходимо вычислить степени включения $\tilde{C}(T)$ в заданные оценки *EASY, MIDDLE* и *COMPLICATED*. Наибольшая из этих степеней включения и даст лингвистическую оценку сложности теста: если $\upsilon(\tilde{C}(T), EASY)$ – наибольшая степень включения, то тест простой, если наибольшая степень включения $\upsilon(\tilde{C}(T), MIDDLE)$, то тест средний, если же наибольшая $\upsilon(\tilde{C}(T), COMPLICATED)$, то тест сложный.

Допустим после ответов учащего был сформирован тес следующим образом: один вопрос, имеющий наибольшую принадлежность к Q_E – это q_2; три вопроса, имеющих наибольшую принадлежность к Q_M – это q_3 , q_4 и q_5; один вопрос, имеющий наибольшую принадлежность к Q_C – это q_8 . В итоге получим:

$$T=\{q_2, q_3, q_4, q_5, q_8\}.$$

Вычислим сложность полученного теста. Для этого с помощью расширенной операции сложения получим сумму сложности вопросов, входящих в тест. Из–за громоздкости не будем приводить все пять примеров сложения, а рассмотрим лишь сложение $C(q_3)$ и $C(q_4)$, а затем приведем окончательный результат.

Для уменьшения объема выкладок, в нечетких натуральных числах $C(q_3)$ и $C(q_4)$, оставим только элементы, имеющие принадлежность отличную от нуля:

$$C(q_3)=\{0,1/3,0,5/4, 1/5, \quad 0,5/6, 0,1/7\};$$
$$C(q_4)=\{0,8/4,1/5, \quad 1/6, \quad 0,8/7\}.$$

Элементами их суммы, имеющими принадлежность отличную от нуля, будут натуральные числа от 7 до 14. Согласно приведенной формулы для расчета значений функции принадлежности суммы нечетких натуральных чисел получим:

$$\mu_{C(q3)+C(q4)}(7) = \sup\{ \min(\mu_{C(q3)}(3),\mu_{C(q4)}(4)) \} = \min(0,1, 0,8) = 0,1;$$

$\mu_{C(q3)+C(q4)}(8) = \sup\{ \min(\mu_{C(q3)}(3),\mu_{C(q4)}(5)), \min(\mu_{C(q3)}(4),\mu_{C(q4)}(4)) \} =$

$= \max[\min(0,1, 1), \min(0,5, 1)] = 0,5;$

$\mu_{C(q3)+C(q4)}(9) = \sup\{ \min(\mu_{C(q3)}(3),\mu_{C(q4)}(6)), \min(\mu_{C(q3)}(4),\mu_{C(q4)}(5)),$

$\min(\mu_{C(q3)}(5),\mu_{C(q4)}(4)) \} = \max[\min(0,1, 1), \min(0,5, 1), \min(1, 0,8)] = 0,8;$

$\mu_{C(q3)+C(q4)}(10) = \sup\{ \min(\mu_{C(q3)}(3),\mu_{C(q4)}(7)), \min(\mu_{C(q3)}(4),\mu_{C(q4)}(6)),$

$\min(\mu_{C(q3)}(5),\mu_{C(q4)}(5)), \min(\mu_{C(q3)}(6),\mu_{C(q4)}(4)) \} = \max[\min(0,1, 0,8), \min(0,5, 1),$

$\min(1, 1), \min(0,5, 0,8)] = 1;$

$\mu_{C(q3)+C(q4)}(11) = \sup\{ \min(\mu_{C(q3)}(4),\mu_{C(q4)}(7)), \min(\mu_{C(q3)}(5),\mu_{C(q4)}(6)),$

$\min(\mu_{C(q3)}(6),\mu_{C(q4)}(5)), \min(\mu_{C(q3)}(7),\mu_{C(q4)}(4)) \} = \max[\min(0,5, 0,8), \min(1, 1),$

$\min(0,5, 1), \min(0,1, 0,8)] = 1;$

$\mu_{C(q3)+C(q4)}(12) = \sup\{ \min(\mu_{C(q3)}(5),\mu_{C(q4)}(7)), \min(\mu_{C(q3)}(6),\mu_{C(q4)}(6)),$

$\min(\mu_{C(q3)}(7),\mu_{C(q4)}(5))\} = \max[\min(1, 0,8), \min(0,5, 1), \min(0,1, 1)] = 0,5;$

$\mu_{C(q3)+C(q4)}(13) = \sup\{ \min(\mu_{C(q3)}(6),\mu_{C(q4)}(7)), \min(\mu_{C(q3)}(7),\mu_{C(q4)}(6)) \} =$

$= \max[\min(0,5, 0,8), \min(0,1, 1)] = 0,5;$

$\mu_{C(q3)+C(q4)}(14) = \sup\{ \min(\mu_{C(q3)}(7),\mu_{C(q4)}(7)) \} = \min(0,1, 0,8) = 0,1.$

Таким образом: $C(q_3)+C(q_4) = \{0,1/7, 0,5/8, 0,8/9, 1/10, 1/11, 0,8/12, 0,5/13, 0,1/14\}.$

Итоговая сумма сложностей вопросов, входящих в тест:

$\sum_{q \in T} C(q) = \{0/5, 0/6, 0/7, 0/8, 0/9, 0/10, 0/11, 0/12, 0/13, 0/14, 0/15, 0/16,$

$0/17, 0/18, 0/19, 0/20, 0/21, 0/22, 0/23, 0,1/24, 0,5/25, 0,8/26, 1/27, 1/28, 0,8/29,$

$0,5/30, 0,1/31, 0/32, 0/33, 0/34, 0/35, 0/36, 0/37, 0/38, 0/39, 0/40, 0/41, 0/42, 0/43,$

$0/44, 0/45, 0/46, 0/47, 0/48, 0/49, 0/50\}.$

Теперь, применяя аналогичным образом формулу для вычисления значений функции принадлежности результата целочисленного деления нечеткого натурального числа на заданное число $n=5$, вычислим сложность теста:

$$\tilde{C}(T) = \frac{\sum_{q \in T} C(q)}{n} = \{0/1, 0/2, 0/3, 0,1/4, 1/5, 0,5/6, 0/7, 0/8, 0/9, 0/10\}.$$

Определим степени включения сложности теста в заданные понятия *EASY*, *MIDDLE* и *COMPLICATED*:

$$\upsilon(\tilde{C}(T), EASY) = 0;$$

$$\upsilon(\tilde{C}(T), MIDDLE) = 0{,}9;$$

$$\upsilon(\tilde{C}(T), COMPLICATED) = 0.$$

Сложность теста имеет наибольшую степень включения в понятие *MIDDLE*, то есть данный тест можно оценивать как средний по сложности и следовательно оценка приравнивается к отметке «4».

Подводя итог второй главы можно сделать следующий вывод, что предложенный в магистерской работе подход позволяет создать систему автоматизированного синтеза и оценки сложности тестовых заданий, для которой реально реализовать программное обеспечение. Применение рассмотренного метода подходит для интерактивного тестирования, при котором тестовая система выбирает очередной вопрос тестового задания непосредственно в процессе тестирования. Адаптивное тестирование может быть организовано так, что система будет как бы "вести" студента, предлагая ему в зависимости от правильности ответа на текущий или более простой или более сложный следующий вопрос, а при выводе итоговой оценки тестирования, учитывать не только количество верных ответов, но и сложность выполненного теста.

3. РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

3.1. Разработка базы данных

Для разработки автоматизированной системы тестирования необходимо спроектировать и создать базу данных, которая будет хранить необходимую информацию. Для этого была создана база данных «Тестирование» при помощи СУБД Interbase, которая содержит 11 таблиц.

Таблица «Вопросы», содержащая текст вопроса и субъективные оценки сложности по 10-бальной шкале, таблица «Типы вопросов», отражающая различные типы вопросов и соответствующие 3 таблицы с ответами: таблица «Ответы1», которая отражает ответы на тип задания с выбором одного или нескольких правильных ответов, таблица «Ответы2», которая отражает тип задания с конструируемым ответом, где учащийся вводит правильный ответ и таблица «Ответы3», отражающая тип заданий на установление соответствия. А также были разработаны 4 таблицы, которые идентифицируют личность тестируемого: «Студенты», «Группы», «Специальности», «Институты». Для ведения статистики ответов учащихся были созданы 2 таблицы: «Ответы студентов» и «Результаты».

Физическую модель базы данных, отражающую связи между таблицами, можно увидеть на рис.3.1.

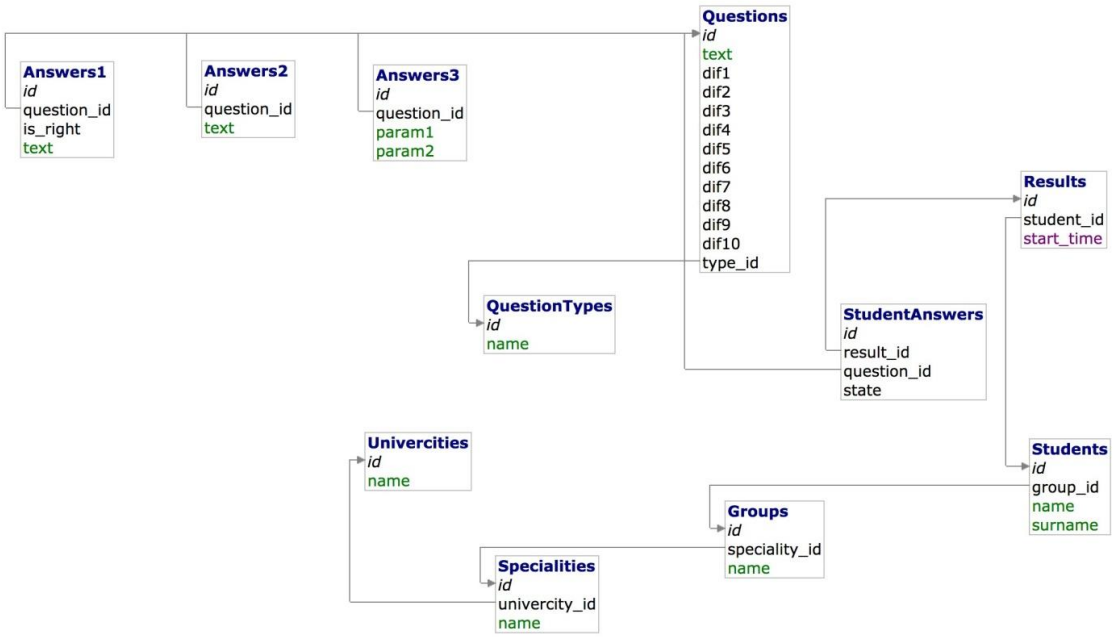

Рис.3.1. Физическая модель базы данных «Автоматизированная система адаптивного тестирования»

Опираясь на повторяющиеся значения типов полей, были созданы домены, с помощью которых достигается унификация типов данных (см. рис.3.2).

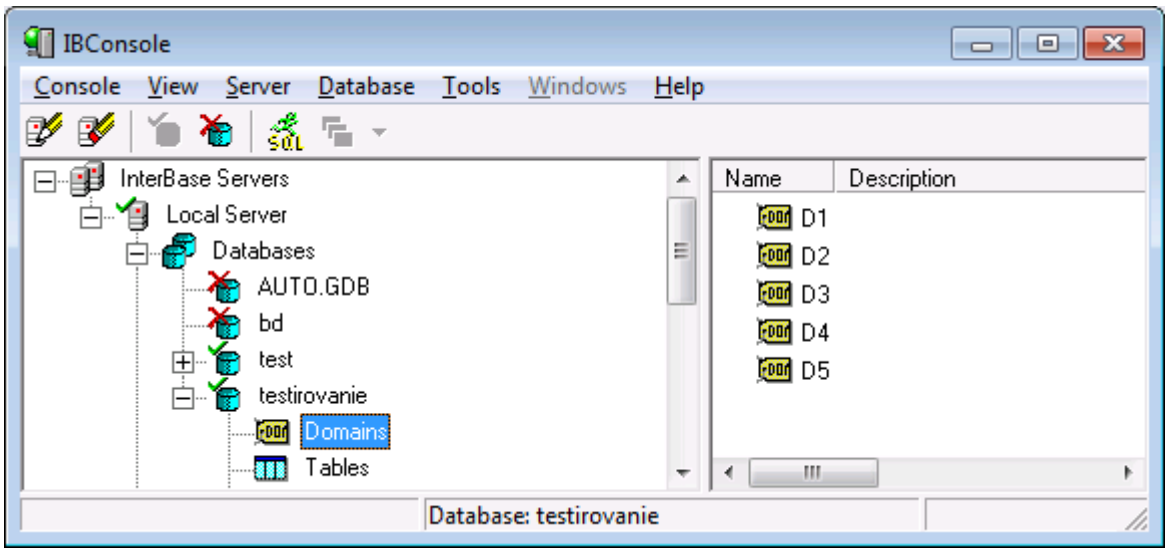

Рис.3.2. Созданные домены системы

Пример создания домена «D1»:

CREATE DOMAIN "D_1" AS INTEGER NOT NULL check (value>0);

Теперь можно приступить к созданию таблиц с использованием физической модели базы данных. Ниже приведен пример создания таблицы «Вопросы», которая имеет связь одни ко многим с таблицей «Типы вопросов»:

```
CREATE TABLE "QUESTIONS"
(
 "ID_QUESTIONS"      "D1",
 "TEXT"     "D2",
 "ID_QUESTIONTYPES"      "D1",
 "DIF1"     "D3",
 "DIF2"     "D3",
 "DIF3"     "D3",
 "DIF4"     "D3",
 "DIF5"     "D3",
 "DIF6"     "D3",
 "DIF7"     "D3",
 "DIF8"     "D3",
 "DIF9"     "D3",
 "DIF10"    "D3",
 PRIMARY KEY ("ID_QUESTIONS")
);
ALTER TABLE "QUESTIONS" ADD CONSTRAINT "RQUESTIONTYPES_QUESTIONS" FOREIGN KEY ("ID_QUESTIONTYPES") REFERENCES QUESTIONTYPES ("ID_QUESTIONTYPES") ON UPDATE CASCADE ON DELETE CASCADE
```

На рис.3.3 представлены аналогичным способом все созданные таблицы.

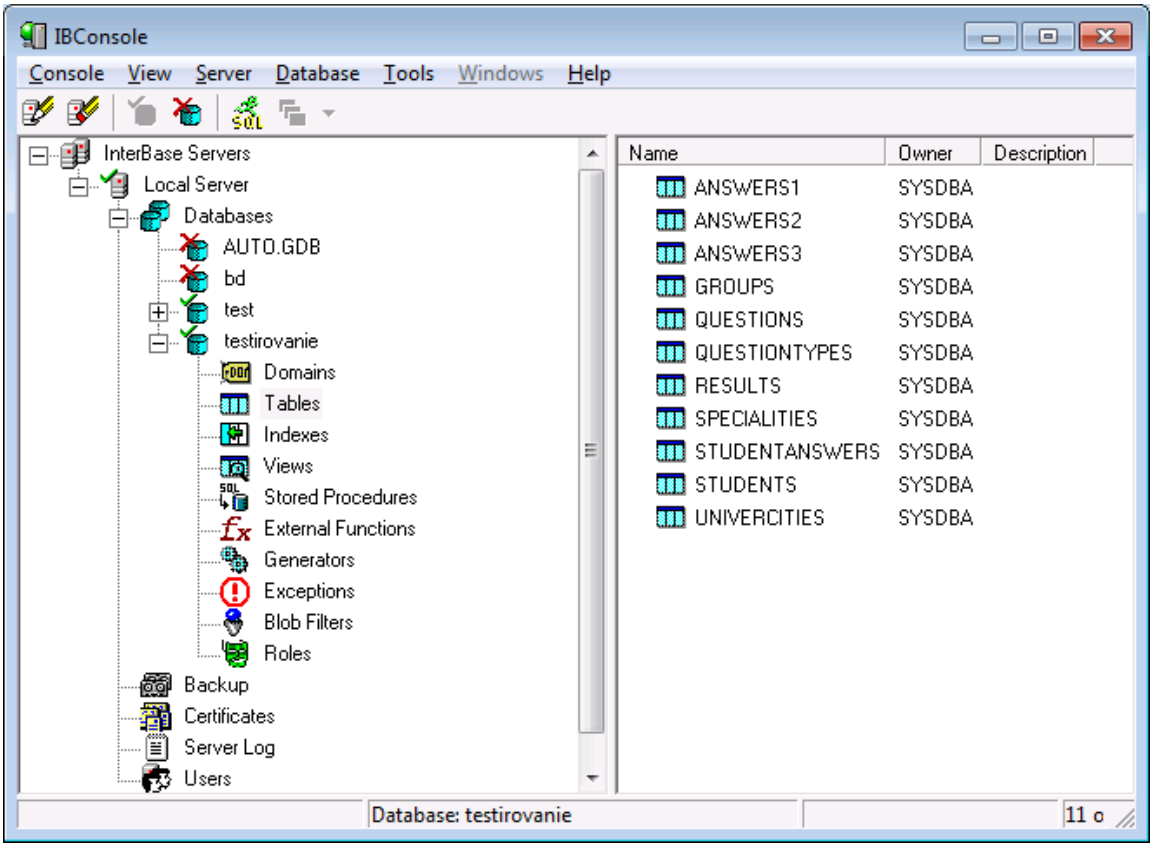

Рис.3.3. Таблицы базы данных автоматизированной системы

Следующим этапом программирования на стороне сервера является создание генераторов и триггеров. Триггер представляет собой процедуру, которая находится на сервере БД и вызывается автоматически при модификации записей БД, т. е. при изменении столбцов или при их удалении и добавлении. В отличие от хранимых процедур, триггеры нельзя вызывать из приложения клиента, а также передавать им параметры и получать от них результаты. Триггер по своей сути похож на обработчики событий BeforeEdit, AfterEdit, Beforelnsert, Afterlnsert, BeforeDelete И AfterDelete, связанных с модификацией таблиц. Триггер может вызываться при редактировании, добавлении или удалении записей до и/или после этих событий.

В связи с тем, что InterBase не поддерживает автоинкрементные поля, при создании ключевого столбца, требующего уникальности значений, рекомендуется поступать следующим образом:

1. При создании таблицы задать ключевой столбец целочисленного типа.

2. Создать генератор, который при обращении к нему возвращает уникальное целочисленное значение.

3. Создать триггер, который при добавлении к таблице новой записи обращается к генератору и заносит возвращаемое им значение в ключевое поле.

Код создания генератора для одной из таблиц продемонстрирован на рис.3.4. При добавлении в таблицу новой записи этот генератор увеличит свое значение на единицу и поможет создать уникальный ключ для новой записи.

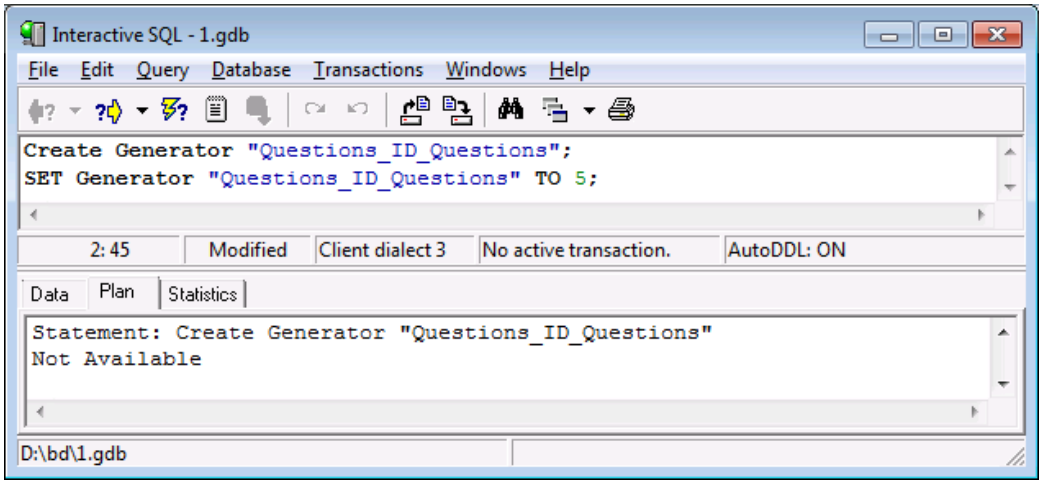

Рис3.4. Создание генератора

Код создания триггера для автоматического инкрементирования поля после вставки новой записи в таблицу показан на рис.3.5.

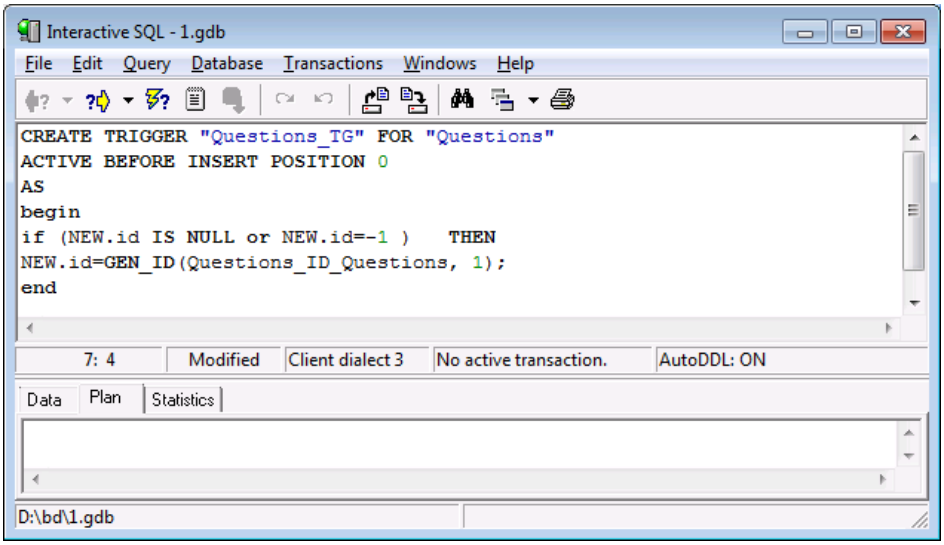

Рис.3.5. Создание триггера для таблицы «Questions»

Представление— виртуальная (логическая) таблица, представляющая собой поименованный запрос (алиас к запросу), который будет подставлен как подзапрос при использовании представления.

В отличие от обычных таблиц реляционной БД, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Изменение данных в реальной таблице БД немедленно отражается в содержимом всех представлений, построенных на основании этой таблицы.

В базе данных для магистерской работы были созданы следующие представления: «Представление регистрации», «Представление вопросов по функциям принадлежности» и «Представление с ответами тестируемых». Представление, отображающие вопросы по функциями принадлежности уровням сложности EASY, MEDIUM, HARD соответственно. Для создания такого представления был создан следующий сложный sql-запрос:

```
CREATE VIEW "VIEW_QUESTIONS"
AS SELECT
  "Questions"."id" AS "id",
  "Questions"."text" AS "text",
  "Questions"."type_id" AS "type_id",
```

```
least(
greatest((1 - "Questions"."dif1"),1),
greatest((1 - "Questions"."dif2"),0.9),
greatest((1 - "Questions"."dif3"),0.6),
greatest((1 - "Questions"."dif4"),0.3),
greatest((1 - "Questions"."dif5"),0),
greatest((1 - "Questions"."dif6"),0),
greatest((1 - "Questions"."dif7"),0),
greatest((1 - "Questions"."dif8"),0),
greatest((1 - "Questions"."dif9"),0),
greatest((1 - "Questions"."dif10"),0)) AS "easy",
least(
greatest((1 - "Questions"."dif1"),0),
greatest((1 - "Questions"."dif2"),0.3),
greatest((1 - "Questions"."dif3"),0.6),
greatest((1 - "Questions"."dif4"),0.9),
greatest((1 - "Questions"."dif5"),1),
greatest((1 - "Questions"."dif6"),1),
greatest((1 - "Questions"."dif7"),0.9),
greatest((1 - "Questions"."dif8"),0.6),
greatest((1 - "Questions"."dif9"),0.3),
greatest((1 - "Questions"."dif10"),0)) AS "medium",
least(
greatest((1 - "Questions"."dif1"),0),
greatest((1 - "Questions"."dif2"),0),
greatest((1 - "Questions"."dif3"),0),
greatest((1 - "Questions"."dif4"),0),
greatest((1 - "Questions"."dif5"),0),
greatest((1 - "Questions"."dif6"),0),
greatest((1 - "Questions"."dif7"),0.3),
greatest((1 - "Questions"."dif8"),0.6),
greatest((1 - "Questions"."dif9"),0.9),
greatest((1 - "Questions"."dif10"),1)) AS "hard"
FROM "Questions";
```

Данный запрос осуществляет подсчёт параметров функции принадлежности путём максимального сравнения поочерёдно имеющихся у каждого вопроса субъективных оценок по 10-бальной шкале с заданными тремя особыми оценками в *FUZZY(Comp)*, которые описаны во второй главе (см. пункт 2.1.).

Результат работы sql-запроса продемонстрирован на рис.3.6 и 3.7.

```
SELECT *
FROM Questions
```

id	text	dif1	dif2	dif3	dif4	dif5	dif6	dif7	dif8	dif9	dif10	type_id
33	Текст вопроса 1	1	0.8	0.6	0.5	0	0	0	0	0	0	1
65	Текст вопроса 2	0	0.2	0.8	1	1	0.3	0.2	0	0	0	1
97	Текст вопроса 3	1	1	0.5	0	0	0	0	0	0	0	1

Рис.3.6. Таблица вопросов с субъективными оценками сложностями

```
SELECT *
FROM VIEW_QUESTIONS
```

id	text	type_id	easy	medium	hard
33	Текст вопроса 1	1	0.5	0	0
65	Текст вопроса 2	1	0	0.6	0
97	Текст вопроса 3	1	0.6	0	0

Рис.3.7. Результат отображения данных в представлении

Заключительным этапом работы с базой данной является создание хранимых процедур. В данной работы были разработаны следующие процедуры:

– Хранимые процедуры, позволяющие добавлять записи в соответствующие таблицы;

– Хранимые процедуры, позволяющие удалять записи из соответствующих таблиц;

– Хранимые процедуры, позволяющие обновлять записи из соответствующих таблиц.

Создание описанных выше пунктов, вынесено в приложение (см. Приложение).

3.2 Разработка windows-приложения

В клиентском приложении будем использовать технологию доступа к данным InterBase Express. В качестве инструментального средства для

разработки клиентских приложений будем использовать Builder C++ фирмы Borland.

Первым шагом создания клиентского приложения служит разработка структуры самого Windows – приложения. При запуске выводится главное окно работы с приложением, которое содержит 4 режима запуска(см. рис.3.8):

1. Режим регистрации. Перед тем, как начать тестирование, каждый учащийся должен пройти авторизацию, где необходимо последовательно выбрать подходящие данные: институт/факультет, направление, группу и ФИО тестируемого. Система выстроена таким образом, что данные внесены в базу данных заранее.

2. Режим тестирования. Данный режим является главным, собственно здесь и осуществляется разработанный алгоритм прохождения адаптивного тестирования.

3. Режим просмотра результатов. Автоматизированная система сохраняет в базу данных все данные о прохождении когда-либо учащимся теста.

4. Режим редактора тестов. В данном режиме педагог пополняет банк вопросов системы.

Рассмотрим создание и разработку каждого из режимов тестирования.

Рис.3.8. Главное меню автоматизированной системы тестирования

Компоненты, предназначенные для работы по технологии InterBase Express, расположены на странице InterBase палитры компонентов.

За установление соединения с сервером БД отвечает компонент IBDatabase. Компонент DataModule отвечает за хранение не визуальных компонентов, его подключаем к форме (см. рис.3.9).

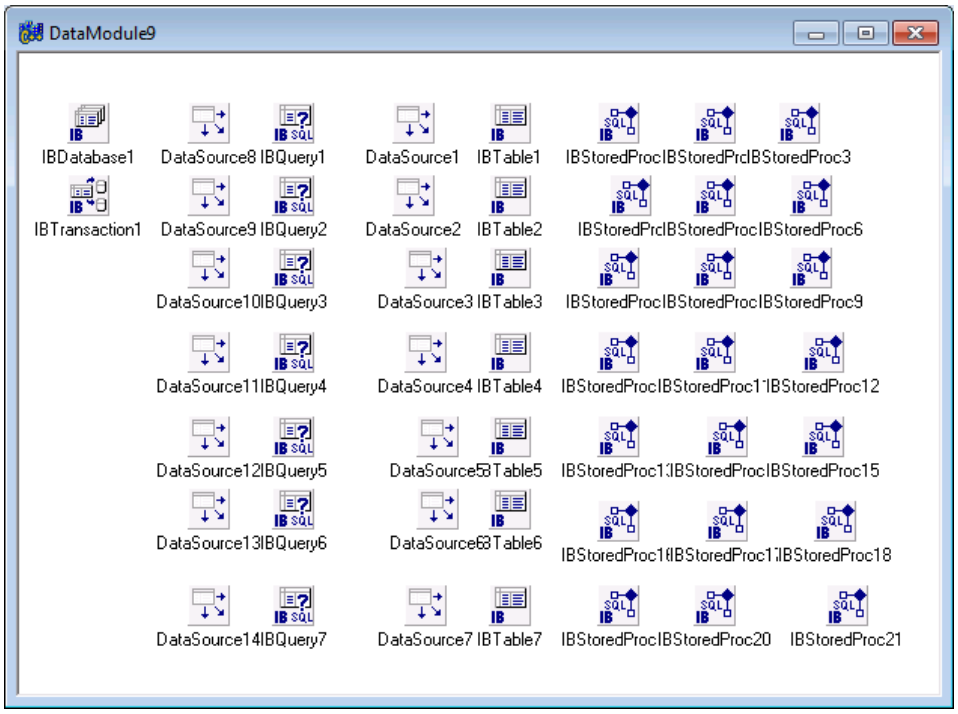

Рис.3.9. Окно компонента DataModule

Для управления транзакциями при работе с сервером InterBase служит компонент IBTransaction. Этот компонент необходимо соединить с IBDatabase при помощи свойства Databases. Для управления хранимыми процедурами используется компонент IBStoredProc. Для доступа к таблицам и представлениям по технологии InterBase Express будем использовать компоненты IBTable и IBQuery. Для получения данных из БД в свойстве SQL необходимо сформировать нужный запрос.

Далее необходимо настроить компонент DataSource для каждого IBTable. Для этого компонента необходимо задать свойство DataSet.

Далее на форме размещаем необходимые компоненты визуализации, компонент DBGrid через свойство DataSource соединяем с DataSource. В итоге должна отобразится таблица в компоненте DBGrid.

Окно ввода идентификационных данных тестируемого продемонстрировано на рис.3.10.

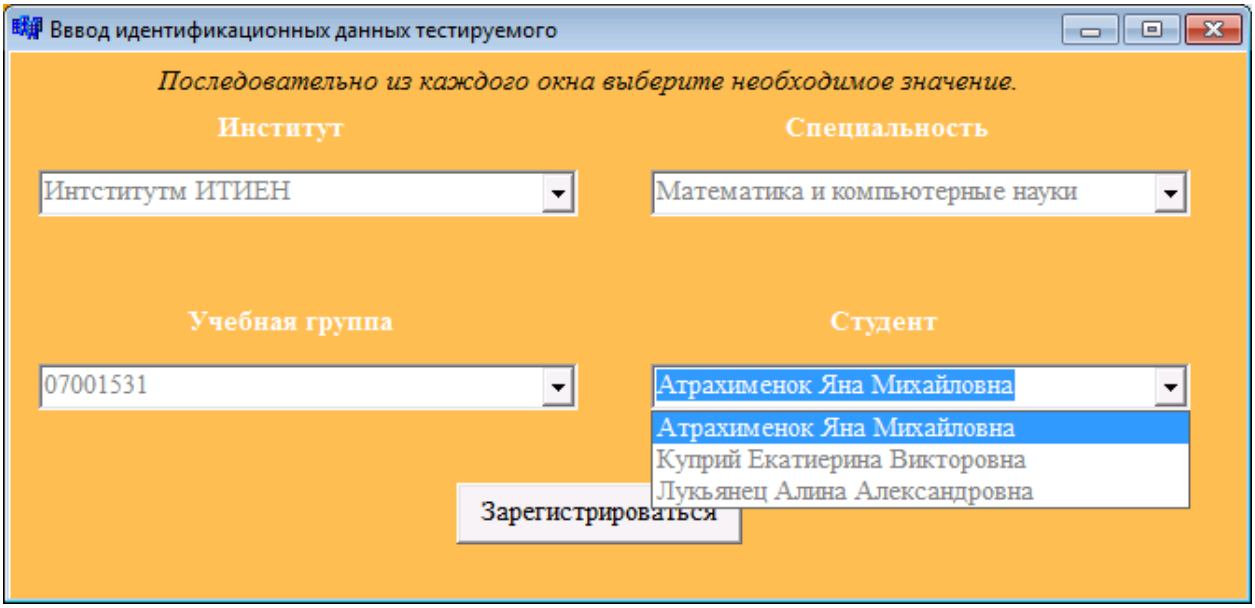

Рис.3.10. Окно регистрации тестируемого

После регистрации учащийся попадает режим в тестирования. Главная форма автоматизированной системы, отвечающая за адаптивное

тестирование. Как только тестируемый попал в данный режим, тестирование началось и испытуемому подаётся по правилам автомата, созданного во второй главе, вопрос средний сложности. Далее в зависимости от правильности ответа тестируемого система переключает сложность вопросов, либо оставляет его на том же уровне. Реализация программного кода выглядит следующим образом:

```
if (difficultly == 1 && right>0)
difficultly=2;
else if (difficultly == 3 && right==0)
difficultly = 2;
else difficultly = right ? 3 : 1;
```

Для загрузки банка тестов из базы данных была написана функция loadNextQuestion(). Листинг данной функции вынесен в приложение.

В системе предусмотрено тестирование по 3 типа вопросов. Первый тип - это задания с выбором, которые разбивается на 2 подтипа: с 1 правильным ответом или несколькими правильными ответам. В зависимости от разных вопросов система выдаёт разные формы как представлено на рис.3.11 и 3.12.

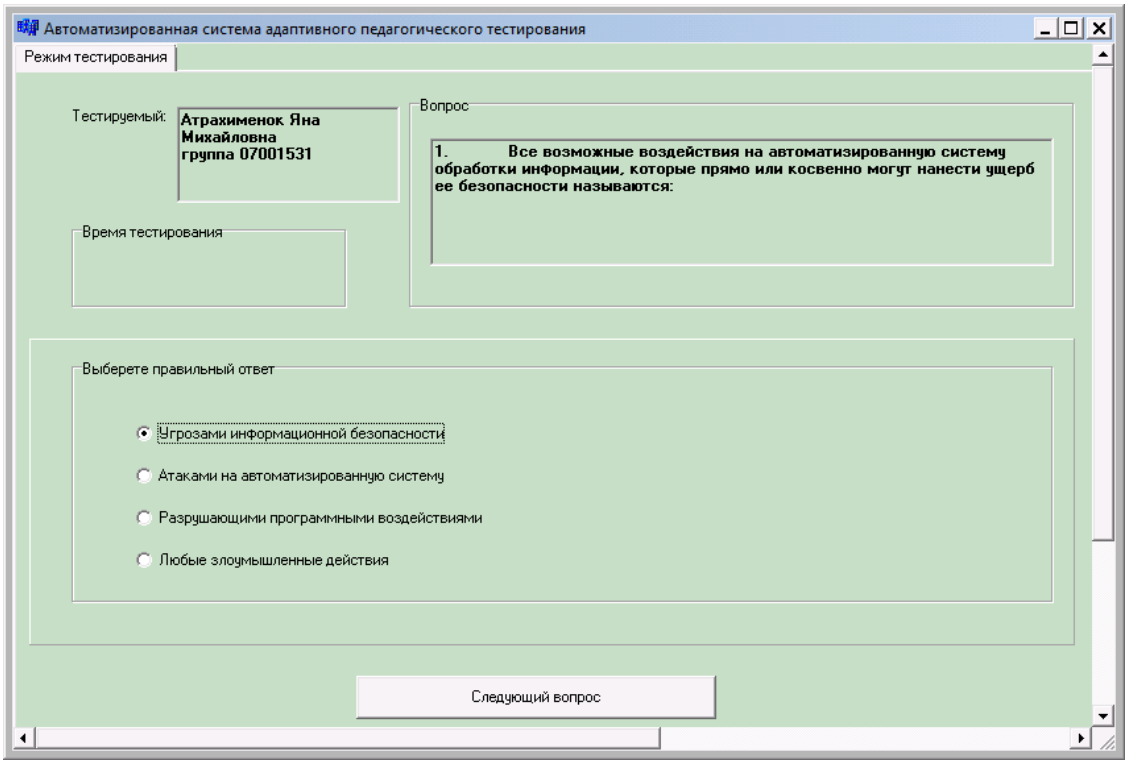

Рис.3.11 Форма тестирования при выборе 1 правильного ответа

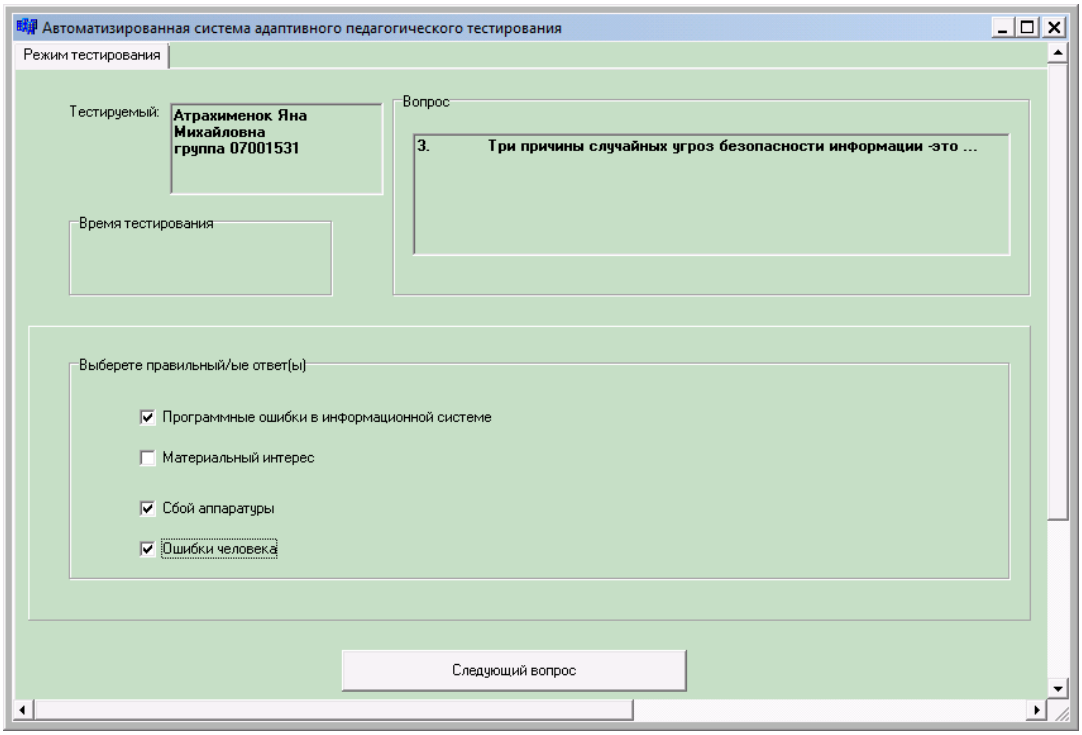

Рис.3.12. Форма тестирования при выборе нескольких правильных ответов

Второй вид задания называется с конструируемым ответом. Здесь тестируемому необходимо самому вписать правильный ответ(см.рис.3.13)

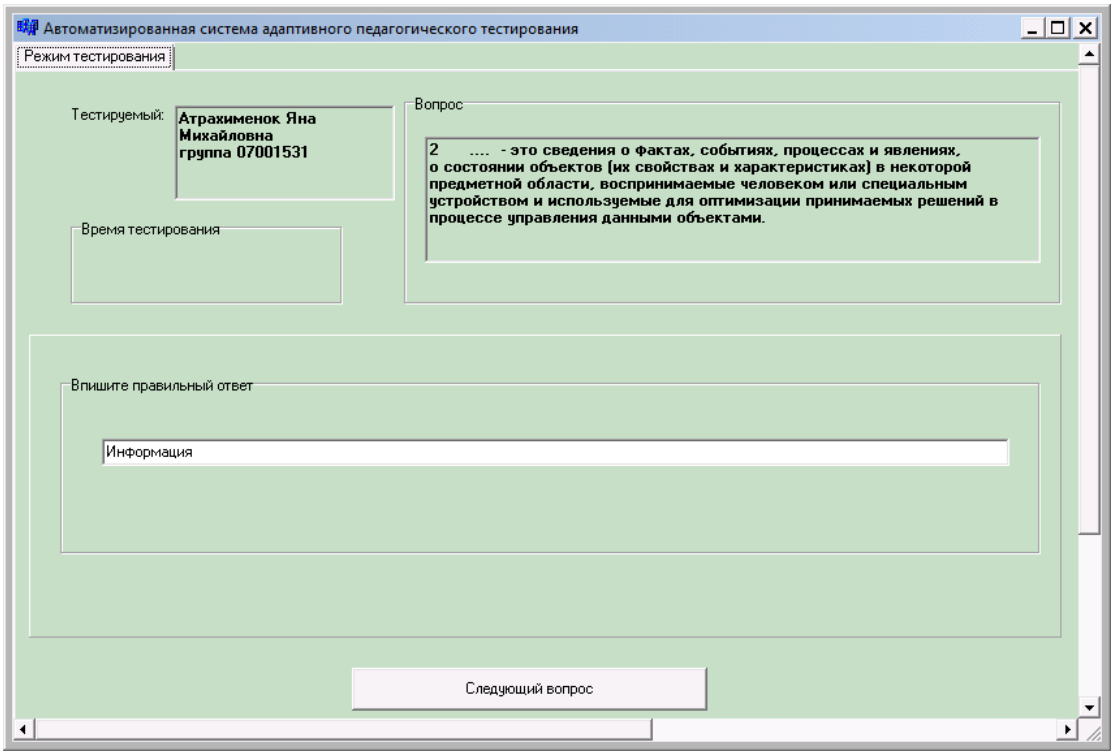

Рис.3.13. Форма тестирования конструированного типа задания

Последний тип заданий является вопросом на установление соответствия. Форма предоставления информации данного типа продемонстрирована на рис.3.14.

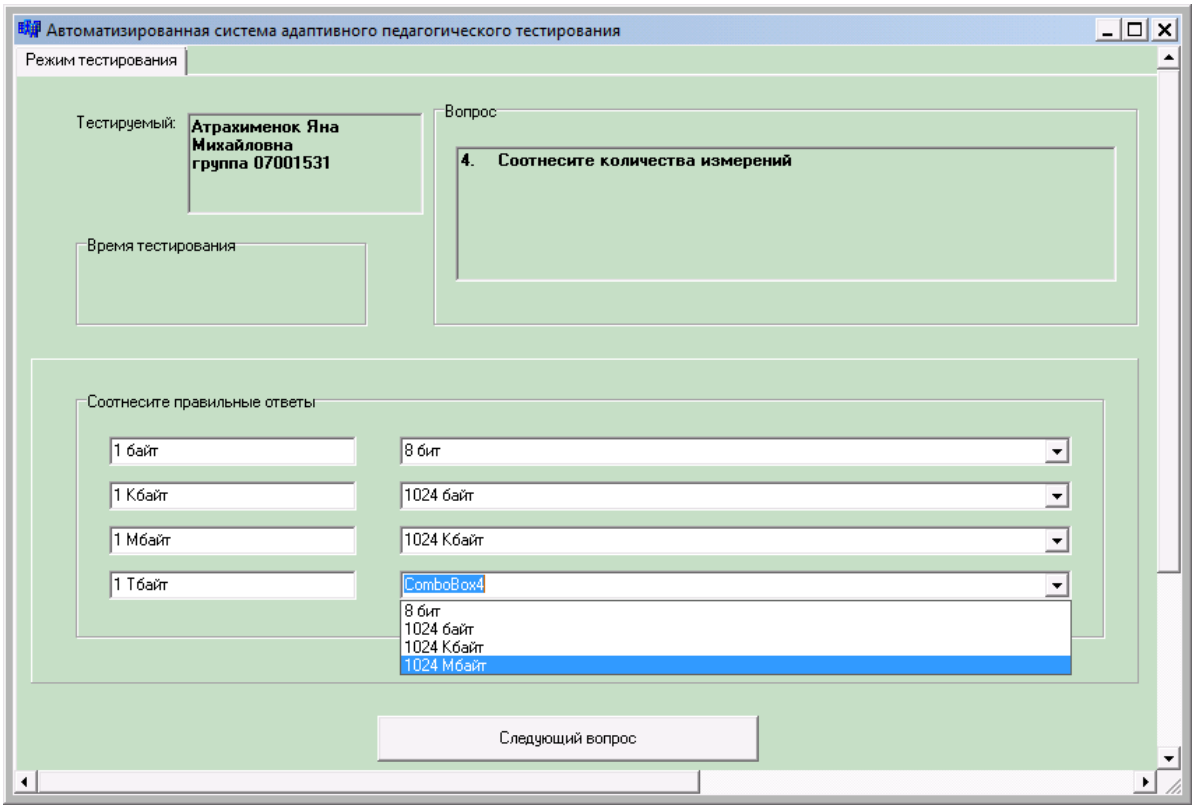

Рис3.14 Форма тестирования на установление соответствия

Выход из тестирования осуществляется, если хоть какой-нибудь счётчик количества вопросов, равняется 15, вне зависимости от правильности ответа. Тем самым у каждого тестируемого будет разное количество баллов и индивидуальная траектория ответов.

Выход из тестирования заканчивается окном, в котором отображаются количество вопросов, правильных ответов и сложность самого теста, которая определяется по функциям принадлежности.

Окно результатов продемонстрировано на рис.3.15.

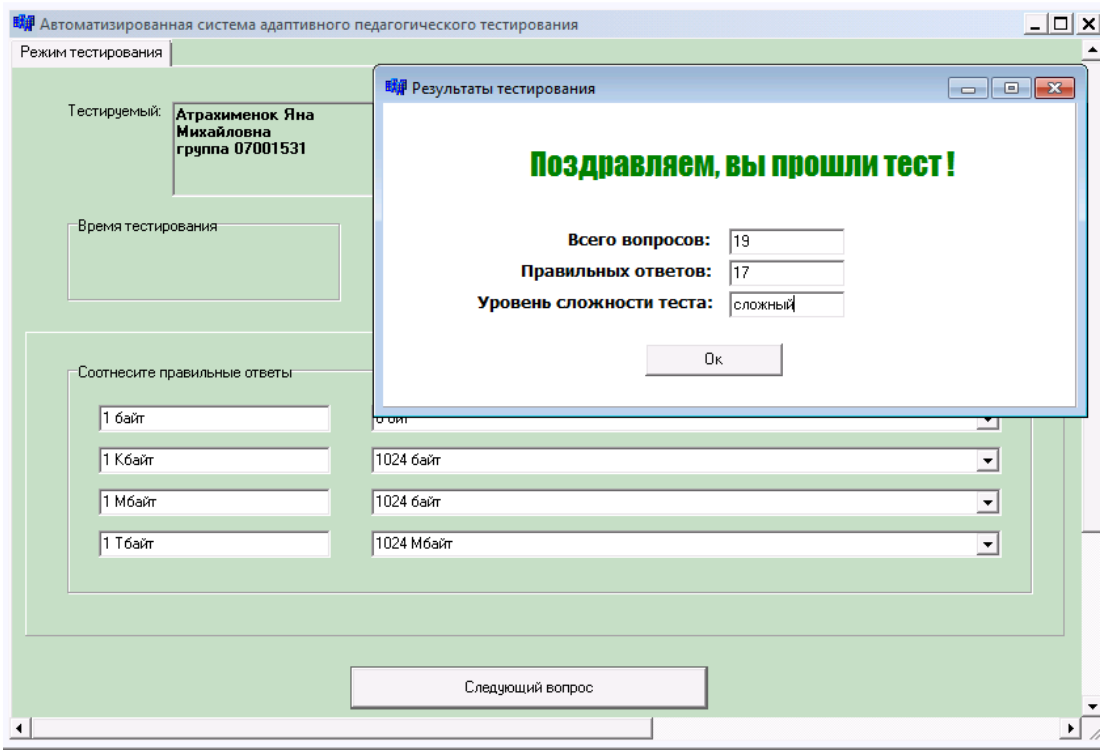

Рис.3.15. Окно окончания тестирования

Режим просмотра результатов возможен только для педагогов, поэтому перед тем как попасть в данный режим, необходимо пройти авторизацию. Окно авторизации представлено на рис.3.16.

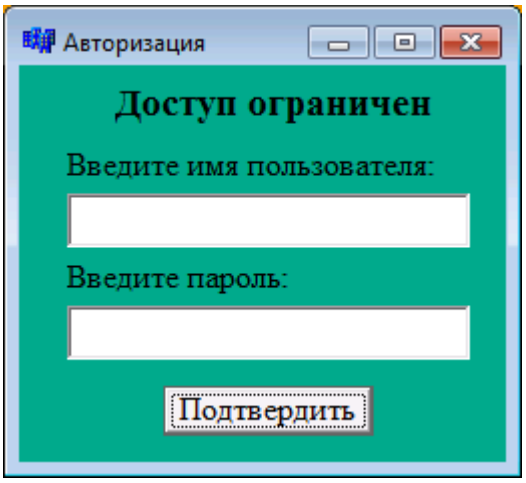

Рис.3.16. Окно авторизации

После авторизации педагогу становится доступно 2 режима: режим проверки результатов тестирования и режим редактирования самих тестов.

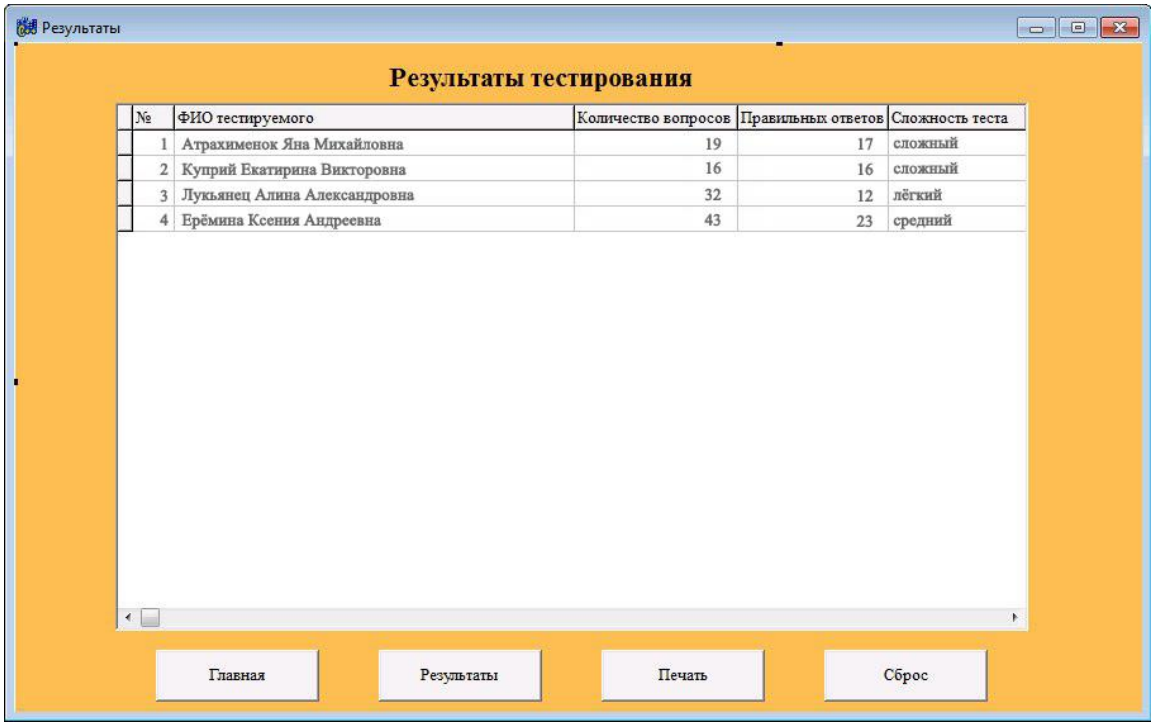

Рис.3.17. Режим результатов тестирования

Окно редактор тестов будет описано чуть позже, но будет.

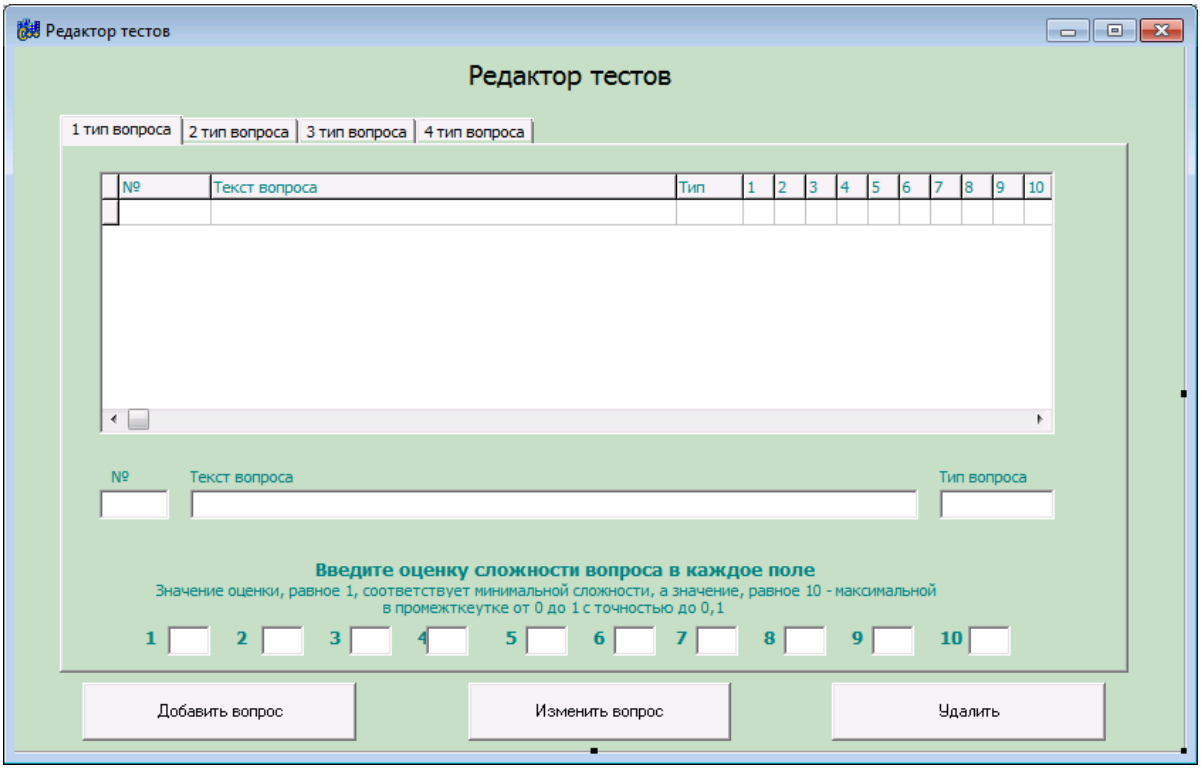

Рис.3.18. Режим «Редактор тестов»

Подводя итог третей главы можно сделать вывод, что создание автоматизированной системы адаптивного педагогического тестирования успешно выполнено.

ЗАКЛЮЧЕНИЕ

Основной целью написания данной выпускной квалификационной работы является разработка автоматизированной системы адаптивного педагогического тестирования. В результате проделанной работе можно сказать, что цель полностью достигнута, а поставленные задачи решены:

- изучена специфика адаптивного тестирования;
- проанализированные методы автоматизации индивидуального тестирования;
- разработана собственная методика для решения задач автоматизированного тестирования;
- реализована модель автоматизированной системы адаптивного педагогического тестирования.

Результатом данной магистерской работы является автоматизированная система адаптивного тестирования, разработанная для автоматизации построения индивидуального сценария тестируемого в зависимости от его ответов.

В первой главе был проведен анализ проблемы автоматизации педагогического тестирования, а также существующих систем электронного обучения, в ходе которого была разработана многопараметренная классификация различных адаптивных технологий и методик, использующиеся в них.

Во второй главе диплома на основе теории нечетких множеств и теории автоматов была разработана методика решения задач адаптивного тестирования Также одной из основных наработок является создание методики построения траектории тестируемого.

В третьей главе выпускной квалификационной работы рассматривается реализация автоматизированной системы адаптивного педагогического тестирования, основанной на созданных методиках магистерской работы.

Внедрение разработанной автоматизированной системы адаптивного тестирования в каком-либо учебном заведении позволит наиболее эффективно вести контроль знаний и даст обучающимся возможность показать свои знания беспристрастно с удаче. Данная система позволяет оперативно просматривать результаты тестирования и формировать отчеты по ним, что приведет к сокращению времени.

В будущем возможна модернизация и расширение информационной системы. Например, разбиение банка тестов тематически. Ещё одним вариантов модернизации, получившейся методики и в дальнейшем системы, является представление оценок сложности динамически, т.е. в зависимости от правильного ответа тестируемого оценка по 10-бальной шкале может смещаться в сторону лёгкости и наоборот. При апробации и выработки конкретного алгоритма с высоким процентом отказоустойчивости планируется внедрение данной методики в рамках систем обучения университета.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Истоки экспериментальной психологии [Электронный ресурс]. - Режим доступа : http://www.effecton.ru/ 199.html.

2. Elena C. Papanastasiou Computer-adaptive testing in science education / [Электронный ресурс]. - Режим доступа : http://cblis.utc.sk/cblis-cd-old/2003/3.PartB/Papers/ Sci-ence_Ed/Testing-Assessment/Papanastasiou.pdf.

3. Катаев, А. В. Открытая модель игрока для оценки знаний и навыков в компьютерных обучающих играх /

А. В. Катаев, О. А. Шабалина // Известия Волгоградского государственного технического университета : межвуз. сб. науч. ст. № 9 (82) / ВолгГТУ. - Волгоград : ИУНЛ ВолгГТУ, 2011. - (Сер. Актуальные проблемы управления, вычислительной техники и информатики в технических системах. Вып. 8). - С. 79-85.

4. Lord F.M. Application of Item Response Theory to Practical Testing Problems. Hillsdale N-J. Lawrence Erlbaum Ass., Publ. 1980. - 266 pp.

5. Зайцева, Л. В. Модели и методы адаптивного контроля знаний / Л. В. Зайцева, Н. О. Прокофьева // Educa

tional Technology & Society. - Nr.7(4), 2004 ISSN 14364522 (Международный электронный журнал) [Электронный ресурс]. - Режим доступа : http://ifets.ieee.org/russian/ depository/v7_i4/html/1.htm. - С. 265-277.

6. Андреев, А. Б. Экспертная система анализа знаний «Эксперт-ТС» / А. Б. Андреев, А. В. Акимов, Ю. Е. Усачев // Proceedings. IEEE International Conference on Advanced Learning Technologies (ICALT 2002). 9-12 September 2002. Kazan, Tatrstan, Russia, 2002. - С. 97-101.

7. Артемов, А. Модульно-рейтинговая система / А. Артемов, Н. Павлова, Т. Сидорова // Высшее образование в России. - 1999. - № 4. - С. 121-125.

8. Galeev I., Sosnovsky S., Chepegin V. MONAP-II: the analysis of quality of the learning process model / I. Galeev, S. Sosnovsky, V. Chepegin //

Proceedings. IEEE International Conference on Advanced Learning Technologies (ICALT 2002). 9-12 September 2002. Kazan, Tatrstan, Russia, 2002. -P. 116-120.

9. Грушецкий, С. В. Построение модели адаптивного тестирования с использованием элементов теории графов / С. В. Грушецкий, И. Д. Рудинский // Труды XIV Международной конференции-выставки ИТ0-2004 / [Электронный ресурс]. - Режим доступа : http://sputnik.mto.ru/ Docs_41/Mat_edu_conf/doc/4617.html.

10. Комлев, В. В. Экономико-математические модели, структурно-параметрическая оптимизация и управление качеством технологий обучения: автореф. дис. на соиск. уч. степ. канд. экон. наук. Специальность 08.00.13 / В. В. Комлев. - Иваново, 2006. - 19 с.

11. Карпенко, А. П. Модельное обеспечение автоматизированных обучающих систем. Обзор / А. П. Карпенко // Наука и Образование. - 2011. - № 7.

12. Шабалина, О. А. Модель пользователя для изучения языков программирования в адаптивной обучающей системе / О. А. Шабалина // Вестник компьютерных и информационных технологий. - 2005. - № 2. - С. 36-39.

13. Шкиль, А. С. Методика оценивания в компьютерной системе тестирования знаний / А. С. Шкиль, С. В. Чу-маченко, С. В. Напрасник // Образование и виртуальность, 2002 : сб. науч. тр. 5-й Междунар. конф. - Харьков - Ялта : УАДО, 2003. - С. 340-345.

14. Лаптев, В. В. Учет времени при оценивании результатов автоматизированного контроля / В. В. Лаптев,

В. И. Сербин // Известия Волгоградского государственного технического университета : межвуз. сб. науч. ст. № 11 (71) / ВолгГТУ. - Волгоград : ИУНЛ ВолгГТУ, 2010. - (Сер. Актуальные проблемы управления, вычислительной техники и информатики в технических системах. Вып. 9). -С. 102-105.

15. Лаврухина, Н. А. Методы оценки качества тестов по результатам тестирования / Н. А. Лаврухина, Н. И. Абасова // Информационные технологии и проблемы математического моделирования сложных систем. - Иркутск : ИИТМ ИрГУПС, 2010. - Вып. 8. - С. 124-134.

16. Зайцева, Л. В. Разработка и применение автоматизированных обучающих систем на базе ЭВМ / Л. В. Зайцева, Л. П. Новицкий, В. А. Грибков ; под ред. Л. В. Ни-цецкого. - Рига : Зинатне, 1989. - 174 с.

17. Попов, Д. И. Способ оценки знаний в дистанционном обучении на основе нечетких отношений / Д. И. Попов // Дистанционное образование. - 2000. - № 6.

18. Zadeh L.A. Fuzzy set // Information and Control. – 1965. – Vol. 8. – P. 338 – 353.

19. Дюбуа Д., Прад А. Теория возможностей. Приложения к представлению знаний в информатике. – М.: Радио и связь, 1990. – 288 с.

20. Модели принятия решений на основе лингвистической переменной / А.Н. Борисов, А.В. Алексеев, О.А. Крумберг и др. – Рига: Зинатне, 1982. – 256 с.

21. Мелихов А.Н., Бернштейн Л.С., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. – М.: Наука, 1990. – 272 с.

22. Нечеткие множества в моделях управления и искусственного интеллекта / А.Н. Аверкин, И.З. Батыршин, А.Ф. Блишун, и др.; Под. ред. Д.А. Поспелова. – М.: Наука, 1986. – 312 с.

ПРИЛОЖЕНИЕ

SQL – код программы на стороне сервера InterBase

```
CREATE DOMAIN "D1" AS CHAR(511) CHARACTER SET WIN1251 NOT NULL;
CREATE DOMAIN "D2" AS INTEGER NOT NULL;
CREATE DOMAIN "D3" AS VARCHAR(250) CHARACTER SET WIN1251 NOT NULL;
CREATE DOMAIN "D4" AS DATE;
CREATE DOMAIN "D5" AS FLOAT;

CREATE TABLE "Answers1"
(
  "id"    "D2",
  "text" "D3",
  "is_right"      "D2",
  "question_id"           "D2",
 PRIMARY KEY ("id")
);

CREATE TABLE "Answers2"
(
  "id"    "D2",
  "text" "D3",
  "question_id"           "D2",
 PRIMARY KEY ("id")
);

CREATE TABLE "Answers3"
(
  "id"    "D2",
  "param1"      "D3",
  "param2"      "D3",
  "question_id"           "D2",
 PRIMARY KEY ("id")
);

CREATE TABLE "Groups"
(
  "id"    "D2",
  "name"         "D3",
  "speciality_id"         "D2",
 PRIMARY KEY ("id")
);


CREATE TABLE "QuestionTypes"
(
  "id"    "D2",
```

```
  "name"         "D3",
 PRIMARY KEY ("id")
);
CREATE TABLE "Questions"
(
  "id"    "D2",
  "text" "D3",
  "dif1""D5",
  "dif2""D5",
  "dif3""D5",
  "dif4""D5",
  "dif5""D5",
  "dif6""D5",
  "dif7""D5",
  "dif8""D5",
  "dif9""D5",
  "dif10"         "D5",
  "QuestionTypes_id" "D2",
 PRIMARY KEY ("id")
);

CREATE TABLE "Results"
(
  "id"    "D2",
  "start_date"   "D4",
  "start_time"   TIME,
  "student_id"  "D2",
 PRIMARY KEY ("id")
);
CREATE TABLE "Specialities"
(
  "id"    "D2",
  "name"         "D3",
  "univercity_id"         "D2",
 PRIMARY KEY ("id")
);
CREATE TABLE "StudentAnswers"
(
  "id"    "D2",
  "state"          "D2",
  "result_id"     "D2",
  "question_id"           "D2",
 PRIMARY KEY ("id")
);
CREATE TABLE "Students"
(
  "id"    "D2",
  "name"         "D3",
  "surname"     "D3",
  "group_id"    "D2",
 PRIMARY KEY ("id")
);
```

```
CREATE TABLE "Univercities"
(
  "id"    "D2",
  "name"         "D3",
 PRIMARY KEY ("id")
);



DROP TABLE IF EXISTS `Answers1`;
CREATE TABLE `Answers1`
 ( `id` int(11) NOT NULL AUTO_INCREMENT, `question_id` int(11) NOT NULL, `is_right`
tinyint(4) NOT NULL, `text` varchar(250) NOT NULL, PRIMARY KEY (`id`), KEY
`question_id` (`question_id`), CONSTRAINT `Answers1_ibfk_1` FOREIGN KEY
(`question_id`) REFERENCES `Questions` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Answers2`;
CREATE TABLE `Answers2`
( `id` int(11) NOT NULL AUTO_INCREMENT, `question_id` int(11) NOT NULL, `text`
varchar(250) NOT NULL, PRIMARY KEY (`id`), KEY `question_id` (`question_id`),
CONSTRAINT `Answers2_ibfk_1` FOREIGN KEY (`question_id`) REFERENCES `Questions`
(`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Answers3`;
CREATE TABLE `Answers3` ( `id` int(11) NOT NULL AUTO_INCREMENT, `question_id`
int(11) NOT NULL, `param1` varchar(250) NOT NULL, `param2` varchar(250) NOT NULL,
PRIMARY KEY (`id`), KEY `question_id` (`question_id`), CONSTRAINT `Answers3_ibfk_1`
FOREIGN KEY (`question_id`) REFERENCES `Questions` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Groups`;
CREATE TABLE `Groups` ( `id` int(11) NOT NULL AUTO_INCREMENT, `speciality_id`
int(11) NOT NULL, `name` varchar(250) NOT NULL, PRIMARY KEY (`id`), KEY
`speciality_id` (`speciality_id`), CONSTRAINT `Groups_ibfk_1` FOREIGN KEY
(`speciality_id`) REFERENCES `Specialities` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Questions`;
CREATE TABLE `Questions` ( `id` int(11) NOT NULL AUTO_INCREMENT, `text`
varchar(250) NOT NULL, `dif1` int(11) NOT NULL, `dif2` int(11) NOT NULL, `dif3` int(11)
NOT NULL, `dif4` int(11) NOT NULL, `dif5` int(11) NOT NULL, `dif6` int(11) NOT NULL,
`dif7` int(11) NOT NULL, `dif8` int(11) NOT NULL, `dif9` int(11) NOT NULL, `dif10` int(11)
NOT NULL, `type_id` int(11) NOT NULL, PRIMARY KEY (`id`), KEY `type_id` (`type_id`),
CONSTRAINT `Questions_ibfk_1` FOREIGN KEY (`type_id`) REFERENCES
`QuestionTypes` (`id`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `QuestionTypes`;
CREATE TABLE `QuestionTypes` ( `id` int(11) NOT NULL AUTO_INCREMENT, `name`
varchar(200) NOT NULL, PRIMARY KEY (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
DROP TABLE IF EXISTS `Results`;
CREATE TABLE `Results` ( `id` int(11) NOT NULL AUTO_INCREMENT, `student_id`
int(11) NOT NULL, `start_time` datetime NOT NULL, PRIMARY KEY (`id`), KEY
`student_id` (`student_id`), CONSTRAINT `Results_ibfk_1` FOREIGN KEY (`student_id`)
REFERENCES `Students` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Specialities`;
CREATE TABLE `Specialities` ( `id` int(11) NOT NULL AUTO_INCREMENT,
`univercity_id` int(11) NOT NULL, `name` varchar(250) NOT NULL, PRIMARY KEY (`id`),
KEY `univercity_id` (`univercity_id`), CONSTRAINT `Specialities_ibfk_1` FOREIGN KEY
(`univercity_id`) REFERENCES `Univercities` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `StudentAnswers`;
CREATE TABLE `StudentAnswers` ( `id` int(11) NOT NULL AUTO_INCREMENT,
`result_id` int(11) NOT NULL, `question_id` int(11) NOT NULL, `state` tinyint(4) NOT NULL,
PRIMARY KEY (`id`), KEY `result_id` (`result_id`), KEY `question_id` (`question_id`),
CONSTRAINT `StudentAnswers_ibfk_1` FOREIGN KEY (`result_id`) REFERENCES
`Results` (`id`), CONSTRAINT `StudentAnswers_ibfk_2` FOREIGN KEY (`question_id`)
REFERENCES `Questions` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Students`;
CREATE TABLE `Students` ( `id` int(11) NOT NULL AUTO_INCREMENT, `group_id`
int(11) NOT NULL, `name` varchar(250) NOT NULL, `surname` varchar(250) NOT NULL,
PRIMARY KEY (`id`), KEY `group_id` (`group_id`), CONSTRAINT `Students_ibfk_1`
FOREIGN KEY (`group_id`) REFERENCES `Groups` (`id`) )
ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `Univercities`;
CREATE TABLE `Univercities` ( `id` int(11) NOT NULL AUTO_INCREMENT, `name`
varchar(250) NOT NULL, PRIMARY KEY (`id`) ) ENGINE=InnoDB DEFAULT
CHARSET=latin1;


CREATE VIEW "VIEW_QUESTIONS"
AS SELECT

  "Questions"."id" AS "id",

  "Questions"."text" AS "text",

  "Questions"."type_id" AS "type_id",

  least(
  greatest((1 - "Questions"."dif1"),1),
  greatest((1 - "Questions"."dif2"),0.9),
  greatest((1 - "Questions"."dif3"),0.6),
  greatest((1 - "Questions"."dif4"),0.3),
  greatest((1 - "Questions"."dif5"),0),
```

```
  greatest((1 - "Questions"."dif6"),0),
  greatest((1 - "Questions"."dif7"),0),
  greatest((1 - "Questions"."dif8"),0),
  greatest((1 - "Questions"."dif9"),0),
  greatest((1 - "Questions"."dif10"),0)) AS "easy",

  least(
  greatest((1 - "Questions"."dif1"),0),
  greatest((1 - "Questions"."dif2"),0.3),
  greatest((1 - "Questions"."dif3"),0.6),
  greatest((1 - "Questions"."dif4"),0.9),
  greatest((1 - "Questions"."dif5"),1),
  greatest((1 - "Questions"."dif6"),1),
  greatest((1 - "Questions"."dif7"),0.9),
  greatest((1 - "Questions"."dif8"),0.6),
  greatest((1 - "Questions"."dif9"),0.3),
  greatest((1 - "Questions"."dif10"),0)) AS "medium",

  least(
  greatest((1 - "Questions"."dif1"),0),
  greatest((1 - "Questions"."dif2"),0),
  greatest((1 - "Questions"."dif3"),0),
  greatest((1 - "Questions"."dif4"),0),
  greatest((1 - "Questions"."dif5"),0),
  greatest((1 - "Questions"."dif6"),0),
  greatest((1 - "Questions"."dif7"),0.3),
  greatest((1 - "Questions"."dif8"),0.6),
  greatest((1 - "Questions"."dif9"),0.9),
  greatest((1 - "Questions"."dif10"),1)) AS "hard"

FROM "Questions";
```

Листинг Windows-приложения

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
// сначала средняя сложность. Грузим вопрос
difficultly = 2;
         this->loadNextQuestion();
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1:loadNextQuestion()
{
            IBQuery1->SQL->Clear();
           IBQuery1->Active=false;
           AnsiString myquery = "";

           // вставляем новый результат
           myquery+="INSERT INTO Results VALUES
(null,"+IntToStr(student_id)+",TIMESTAMP);";
           myquery->Active=true;
           IBQuery1->SQL->Add(myquery);
           IBQuery1->Active=true;
           IBQuery1->Exec();

           // и берем его ид из базы
           myquery = "SELECT MAX(id) FROM Results;";
                    myquery->Active=true;
           IBQuery1->SQL->Add(myquery);
           IBQuery1->Active=true;
           IBQuery1->Exec();
           result_id = DBGrid1->Columns->Items[0]->Field->AsInt;

           // выбираем вопрос из вьюшки
           myquery+="SELECT id, text, type_id, easy,medium,hard FROM
VIEW_QUESTIONS WHERE id NOT IN ";
           myquery+="(SELECT id FROM StudentAnswers WHERE result_id =
"+IntToStr(result_id)+") order by ";
           // в зависимости от сложности сортируем по разному параметру
           if (difficultly == 1)
              myquery+="easy DESC "  ;
           else if (difficultly == 2)
             myquery+="medium DESC "  ;
           else
           myquery+="hard DESC "  ;
           myquery+="ROWS 1");
           IBQuery1->SQL->Add(myquery);
           IBQuery1->Active=true;
           IBQuery1->Open();

           // прячем панели с ответами  (затем покажем только нужную в зависимости от
типа вопроса)
           panel1->Visible = false;
           panel2->Visible = false;
           panel3->Visible = false;
           panel4->Visible = false;

           // сброс радио, комбобоксов
           radioButton1->Checked = false;
           radioButton2->Checked = false ;
```

```
radioButton3->Checked = false  ;
radioButton4->Checked = false   ;

ComboBox1->Items->Clear();
ComboBox2->Items->Clear();
ComboBox3->Items->Clear();
ComboBox4->Items->Clear();

LabelAnswer->Caption  = "";

// опредеяем тип вопроса чтобы знать откуда брать ответы
int question_type = DBGrid1->Columns->Items[2]->Field->AsInt;
question_id =  DBGrid1->Columns->Items[0]->Field->AsInt;
QuestionLabel->Caption =  DBGrid1->Columns->Items[1]->Field->AsString;

// проверяем тип и показываем вопрос
if (question_type == 1)
{
  panel1->Visible = true;
                IBQuery1->SQL->Clear();
IBQuery1->Active=false;
AnsiString myquery = "";
myquery+="SELECT * from answers1 where question_id = ";
myquery+=DBGrid1->Columns->Items[0]->Field->AsString ;
IBQuery1->SQL->Add(myquery);
IBQuery1->Active=true;
IBQuery1->Open();
// показываем варианты ответов
Ans1Label1->Caption = DBGrid1->Columns->Items[0]->Field->AsString;
Ans1Label2->Caption = DBGrid1->Columns->Items[1]->Field->AsString;
Ans1Label3->Caption = DBGrid1->Columns->Items[2]->Field->AsString;
Ans1Label4->Caption = DBGrid1->Columns->Items[3]->Field->AsString;
}
else if (question_type == 2)
{
panel2->Visible = true;
// вопрос в котором можно выбрать несколько вариантов
    IBQuery1->SQL->Clear();
IBQuery1->Active=false;
AnsiString myquery = "";
myquery+="SELECT * from answers1 where question_id = ";
myquery+=DBGrid1->Columns->Items[0]->Field->AsString ;
IBQuery1->SQL->Add(myquery);
IBQuery1->Active=true;
IBQuery1->Open();
          Ans1Label1->Caption = DBGrid1->Columns->Items[0]->Field->AsString;
Ans1Label2->Caption = DBGrid1->Columns->Items[1]->Field->AsString;
Ans1Label3->Caption = DBGrid1->Columns->Items[2]->Field->AsString;
Ans1Label4->Caption = DBGrid1->Columns->Items[3]->Field->AsString;
}
}
else if (question_type == 3)
```

```
            {
            panel3->Visible = true;
             IBQuery1->SQL->Clear();
            IBQuery1->Active=false;
            AnsiString myquery = "";
            myquery+="SELECT * from answers2 where question_id = ";
            myquery+=DBGrid1->Columns->Items[0]->Field->AsString ;
            IBQuery1->SQL->Add(myquery);
            IBQuery1->Active=true;
            IBQuery1->Open();
            // вопрос где нужно соотнести варианты в комбобоксах
            for (int i =0; i<4; i++)  {
               ComboBox1->Items->Add(DBGrid1->Columns->Items[i]->Field->AsString);
               ComboBox2->Items->Add(DBGrid1->Columns->Items[i]->Field->AsString);
               ComboBox3->Items->Add(DBGrid1->Columns->Items[i]->Field->AsString);
               ComboBox4->Items->Add(DBGrid1->Columns->Items[i]->Field->AsString);     }
            }
                        else if (question_type == 4)
            {
            panel4->Visible = true;
            // вопрос с единственным правильным ответом
             IBQuery1->SQL->Clear();
            IBQuery1->Active=false;
            AnsiString myquery = "";
            myquery+="SELECT * from answers3 where question_id = ";
            myquery+=DBGrid1->Columns->Items[0]->Field->AsString ;
            IBQuery1->SQL->Add(myquery);
            IBQuery1->Active=true;
            IBQuery1->Open();
            LabelAnswer->Caption =   DBGrid1->Columns->Items[i]->Field->AsString;

            }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
// проверка ответа по нажатию на кнопки
      int right = 0;
      if (question_type == 1)
      {
      // в первом типе вопросов надо выбрать один правильный ответ
           if (RadioButton1->Checked && DBGrid1->Columns->Items[0]->Field->AsInt)
            right = 1;
            if (RadioButton2->Checked && DBGrid1->Columns->Items[1]->Field->AsInt)
            right = 1;
            if (RadioButton3->Checked && DBGrid1->Columns->Items[2]->Field->AsInt)
            right = 1;
            if (RadioButton4->Checked && DBGrid1->Columns->Items[3]->Field->AsInt)
            right = 1;
      }
      else if (question_type == 2)
      {        // во втором варианте надо выбрать несколько правильных ответов
```

```
            if (question_type == 1)
        {
            if ((CheckBox1->Checked == DBGrid1->Columns->Items[0]->Field->AsInt)&&
            (CheckBox2->Checked == DBGrid1->Columns->Items[1]->Field->AsInt) &&
             (CheckBox3->Checked == DBGrid1->Columns->Items[2]->Field->AsInt) &&
              (CheckBox4->Checked == DBGrid1->Columns->Items[3]->Field->AsInt)  )
              right = 1;
        }
        }
        else if (question_type == 4)
        {
              // 4 тип вопроса - надо чтобы ответ совпал с текстом из базы
           right = Edit1->Caption == LabelAnswer->Caption;
        }
        else
        {
// 3 тип вопроса на соотвествие, проверяем что в комбобоксах выбраны правильные
варианты
            if (ComboBox1->Items[Combobox->ItemIndex] == DBGrid1->Columns->Items[0]-
>Field->AsString &&
            ComboBox2->Items[Combobox->ItemIndex] == DBGrid1->Columns->Items[1]-
>Field->AsString &&
            ComboBox3->Items[Combobox->ItemIndex] == DBGrid1->Columns->Items[2]-
>Field->AsString &&)
            right = 1;
        }
          // добавляем запись про ответ студента в базу
       IBUpdateSQL1->SQL = "insert into StudentResults VALUES
(null,"+IntToStr(result_id)+","+IntToStr(question_id)+","+IntToStr(right)+");";
        IBUpdateSQL1->ExecSQL();

        // обновляем счетчики ответов
        if (difficultly == 1)
             easyAnswers += 1;
        if (difficultly == 2)
          mediumAnswers += 1;
        if (difficultly == 3)
        hardAnswers += 1;

                // переключаем сложность в зависимости от того как ответил
        if (difficultly == 1 && right>0)
        difficultly=2;
        else if (difficultly == 3 && right==0)
          difficultly = 2;
          else difficultly = right ? 3 : 1;



          // если ответов больше 15 то показываем результат
          if (easyAnswers==15 || mediumAnswers==15 || hardAnswers==15)
           this->showResults;
           else
```

```
            this->loadNextQuestion();
}
//---------------------------------------------------------------------------

  void __fastcall TForm1:showResults()
  {
      // берем из базы результаты и смотрим количество правильных
      IBQuery1->SQL->Clear();
            IBQuery1->Active=false;
            AnsiString myquery = "";
            myquery+="SELECT summ(is_right) from StudentsResults where result_id =
"+IntToStr(result_id)+";";
                        IBQuery1->SQL->Add(myquery);
            IBQuery1->Active=true;
            IBQuery1->Open();

            Label1->Caption = "Правильных ответов: "+DBGrid1->Columns->Items[0]->Field-
>AsString;
  }
```