Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 – Программная инженерия
Магистерская программа Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____А.В. Бушманов
«___»_____2017 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Проектирование информационной системы медицинской диагностики на базе нечеткой логики

Исполнитель студент группы 557 ом	_____ (подпись, дата)	Ю.А. Горожанина
Руководитель доцент, канд. физ.-мат. наук	_____ (подпись, дата)	В.В. Ерёмина
Руководитель магистерской программы профессор, доктор техн. наук	_____ (подпись, дата)	Е.Л. Ерёмин
Нормоконтроль доцент, канд. физ.-мат. наук	_____ (подпись, дата)	В.В. Ерёмина
Рецензент доцент, канд. техн. наук	_____ (подпись, дата)	Д.А. Теличенко
Рецензент	_____ (подпись, дата)	А.А. Малынов

Благовещенск 2017

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей диссертации использованы ссылки на следующие стандарты и нормативные документы:

ГОСТ 2.104-68 ЕСПД Основные надписи

ГОСТ 2.105-95 ЕСПД Общие требования к текстовым документам

ГОСТ 2.106-96 ЕСПД Текстовые документы

ГОСТ 2.111-68 ЕСПД Нормоконтроль

ГОСТ 3.1103-83 ЕСПД Основные надписи

ГОСТ 3.1130-93 ЕСПД Основные требования к формам и бланкам документов

ГОСТ 19.101-77 ЕСПД Виды программ и программных документов

ГОСТ 19.106-78 ЕСПД Требования к программным документам выполненным печатным способом

ГОСТ 19.201-93 ЕСПД Техническое задание, требования к содержанию и оформлению

ГОСТ 19.401-78 ЕСПД Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 ЕСПД Описание программы

ГОСТ 19.504-79 ЕСПД Руководство программиста, требования к содержанию и оформлению

ГОСТ 34.602-89 КСАС Техническое задание на создание автоматизированной системы

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных;

БЗ – база знаний;

ЭС – экспертная система;

ЛП – лингвистическая переменная;

ЧП – числовая переменная;

ИНС – искусственные нейронные сети;

МИС – медицинская информационная система;

СУБД – система управления базами данных;

СППР – система поддержки приятия решений.

ВВЕДЕНИЕ

Тема магистерской диссертации – «Проектирование информационной системы медицинской диагностики на базе нечеткой логики».

Если ссылаться на открытые источники информации, то определение «медицинская информативная система (МИС)» – это система автоматизации документооборота для лечебно-профилактических учреждений, в которой связаны система поддержки принятия медицинских решений, электронные медицинские карты о пациентах, сведения медицинских исследований в циф-ровой форме, сведения мониторинга состояния пациента с медицинских при-боров, средства общения между сотрудниками, финансовая и администра-тивная сведения.

Применяемые в настоящий момент в большинстве МИС классические системы поддержки принятия решений, базирующиеся на правилах, никак не способны гарантировать представление «семантики» медицинских данных и клинических назначений и на практике демонстрируют несколько значительных «ограничений».

На решение этих задач ориентировано формирование интеллектуаль-ной экспертной системы с целью персонализации медицинской диагностики, что создается на экспертных клинических рекомендациях и компьютерных средствах представления знаний.

Основной целью проводимого исследования является разработка экспертной медицинской системы диагностики на базе нечеткой логики с целью обеспечение эффективной информационной поддержки процесса постановки диагноза, а также процесса рекомендательного оказания медицинской помо-щи.

Аналогичной ЭС медицинской диагностики, в которой применяют нечеткую логику, является наиболее известная диагностическая система MYCIN, предназначение которой заключается в наблюдении за состоянием больного при менингите и бактериальных инфекциях. Начальная версия MYCIN была

разработана в Стенфордском университете в 70 годах. На се-годняшний день эта система устанавливает заключение на уровне врача-специалиста. Она обладает расширенной базой знаний, вследствие чего спо-собна применяться и в иных областях медицины. Актуальность создания данной информационной системы обусловлена необходимостью снижения вероятности ошибки в постановке диагноза, а также уменьшением времени на постановку диагноза [1, с.35]

Для достижения поставленной цели решаются следующие задачи:

- построение модели предметной области;
- построение математической модели нечеткой логики;
- сбор знаний и накопление базы знаний;
- проектирование информационной системы медицинской диагностики;
- реализация экспертной системы.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Применение информационных систем в медицине

В последние годы под медицинской информационной системой (МИС) обычно понимается автоматизированная информационная система медицинского назначения. При этом определение «медицинского» назначения также отсутствует. В данном случае понятие МИС является специализацией двух понятий: информационная система и автоматизированная система. Ниже приведены краткие описания этих понятий, определения МИС в более узком смысле автоматизированной системы, а также классификации МИС.

Международный стандарт ИСО/МЭК 2382-1:1993 представляет информационную систему в виде системы обработки информации, функционирующей совместно с организационными ресурсами, такими как люди, технические и экономические ресурсы, которые предоставляют и распределяют информацию.

Различные определения МИС и классификации МИС приведены в работах [1] и [2]. Например, в [2] даётся следующее определение: «Совокупность информационных, организационных, программных и технических средств, предназначенных для автоматизации медицинских процессов и(или) организаций.»

В [1] приводится определение, принадлежащее С.А.Гаспаряну:

«Одна из форм организации медицинской деятельности, позволяющая медицинскому персоналу при соответствующей технологической поддержке использовать комплекс математических и технических средств, обеспечивающих сбор, хранение, обработку, анализ и выдачу медицинской информации».

Из выше перечисленного можно сделать вывод, что в узком смысле МИС представляет собой автоматизированную систему, обеспечивающую получение, хранение, обработку, анализ и выдачу медицинской информации, то есть сведений, имеющих отношение к состоянию здоровья конкретного человека [2].

В [1] приводится следующая классификация МИС, принадлежащая С.А.Гаспаряну:

- технологические информационные медицинские системы (ТИМС);
- банки информации медицинских служб (БИМС);
- статистические информационные медицинские системы;
- научно-исследовательские информационные медицинские системы;
- обучающие (образовательные) информационные медицинские системы.

Для технологических информационных систем объектом описания является человек (пациент), пользователем – сотрудники медицинских учреждений (врачи, лаборанты, медицинские сестры), информация объединяется на уровне одного пациента.

Банки данных медицинских служб предоставляют информационную содействие отношений множество пациентов – врачи. Причиной деления банков данных на типы является масштаб охвата обслуживаемого населения.

Статистические информационные медицинские системы оказывают информационную помощь взаимоотношениям популяция (в значении население обслуживаемого региона) – органы, руководящие системой медицинского обслуживания. Разделение статистических информационных систем на типы базируется на различии объектов описания, представленных в статистических отчетах ЛПУ и территориальных органов управления здравоохранением.

Научно-исследовательские информационные медицинские системы делают возможным рассматривать объекты и документы науки. Деление на виды основано на отличительных способностях объектов описания.

Обучающие информационные медицинские системы предоставляют информационную поддержку отношений обучаемые – преподаватели.

Образовательные информационные системы подразделяются на виды в соответствии с педагогическими принципами оценки уровня освоения знаний учащимся.

В [1] приводится следующее определение комплексной автоматизированной информационной системы (АИС) медицинского учреждения:

«Информационная система, автоматизирующая как административные, так и клинические функции, ядром которой является электронная медицинская карта».

Комплексная АИС медицинского учреждения стационарного типа обычно включает в себя:

- систему автоматизации административно-хозяйственной деятельности, включая бухгалтерию, кадры, экономическую службу, канцелярию (делопроизводство), материально-техническое снабжение, инженерное обеспечение;

- медицинскую информационную систему, в том числе систему персонифицированного учёта медицинской помощи и ведения электронной медицинской карты, информационные системы (подсистемы) диагностических и процедурных отделений, больничной аптеки;

- справочно-информационную систему (портал или сайт медицинского учреждения, электронные справочники лекарственных средств и т.д.).

МИС как предмет настоящей методики включает в себя элементы комплексной автоматизированной информационной системы медицинского учреждения, обладающие непосредственное отношение к данных о медицинской помощи, предоставленной определенному больному и обеспечивающие автоматизацию определённых функций.

Оценка МИС проводится по степени широты автоматизируемых функций с учётом эмпирических весовых коэффициентов функций и групп функций, предложенных экспертами. В число МИС не входят локальные системы, не поддерживающие сквозные процессы в ЛПУ, но также используемые для поддержки организационной части лечебно-диагностических процессов.

Такие системы предлагается в отличии от определения «МИС» классифицировать как:

а) диагностические – к ним могут относиться ПАКС, ЛИС, РИС и аналогичные системы, они связаны с получением, передачей, анализов данных, приобретенных в следствии проведения диагностических и лабораторных исследований с внешних специальных систем и приборов. Как правило, такого рода си-

стемы лучше проработаны у тех производителей, кто выделяет подобные системы в отдельные продукты либо полноценные отчуждаемые модули. Неимение подобного перечня возможностей (функционала) внутри МИС никак не способно расцениваться слабым подходом к реализации системы, тем не менее отсутствие возможности (реализованной) интеграции со специальными системами следует рассматривать как технологическую (либо функциональная) недоработку;

б) смежные специальные системы – это специализированные модули для задач конкретного направленности (лечебного или общехозяйственного). К подобным системам могут принадлежать системы ведения бухгалтерского, кадрового и др. учета, полноценные аптечные системы (планирование, закупка, распределение и управление фармацевтическими средствами и инвентарем медицинского назначения), аналитические и статистические системы, а также системы для автоматизации процессов определенных отделений (к примеру, прозектура).

Данный функционал при его наличии в оцениваемой системе или не включается в сравнительную с прочими МИС оценку полностью, или обладает низким весом в интегрированной оценке, потому что он рассматривается по отношению к ключевым функциям МИС как вспомогательный, и его отсутствие в системах, построенных по модульному принципу с реализован-ной интеграцией в данной части с системами других, специализированных производителей никак не должно снижать их интегральную оценку по базовому для всех систем этого класса функционалу.

В данном описании сознательно не приводится более полной собственной классификации информационных систем в здравоохранении, так как для ее полноты и непротиворечивости недостаточно еще определен терминологический ряд более низкого уровня. В данной методике классификация систем не содержит существенного значения, так как предлагается и описывается принципиальный подход, а определенные параметры как в части определения иных осей, так и в части определения удельных весов каждого признака имеют все

шансы устанавливаться в широком диапазоне экспертным методом в зависимости от практических задач при использовании этой технологии.

Область исследований, посвященная формализации способов представления знаний и построению экспертных систем, называют «инженерией знаний». Этот термин введен Е. Фейгенбаумом и в его трактовке означает «привнесение принципов и средств из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов». То есть, экспертные системы применяются для решения нестандартных проблем, к которым относятся задачи, обладающие одной или несколькими характеристиками из следующего списка [8]:

- задачи не могут быть представлены в числовой форме;

- исходные данные и знания о предметной области неоднозначны, неточны, противоречивы;

- цели нельзя выразить с помощью четко определенной целевой функции;

- не существует однозначного алгоритмического решения задачи.

Все эти свойства считаются типичными для медицинских задач, так как в основной массе ситуации они презентованы большим объемом многомерных, запутанных, а иногда и противоречивых клинических данных. СППР дают возможность регулировать задачи диагностики, дифференциальной диагностики, прогнозирования, подбора стратегии и стратегии лечения и др.

1.2 Структура и функционирование экспертных систем

Экспертные системы – это программное средство, использующее знания специалиста для предоставления высокоэффективного решения неформализованных задач в ограниченной предметной области. Основой СППР является база знаний (БЗ) о предметной области, что накапливается в ходе построения и эксплуатации.

Для классификации ЭС [4] используют следующие признаки:

- способ формирования решения;

- способ учета временного признака;

- вид используемых данных;

- число используемых источников решения знаний.

В зависимости от метода формирования решения ЭС делятся на анализирующие и синтезирующие. В системах первого типа осуществляется выбор решения из большого количества распространенных решений на основе анализа знаний, в системах второго типа решение производится с отдельных фрагментов знаний.

В зависимости от метода учета временного признака ЭС различают статические и динамические. Статические ЭС предусмотрены для решения задач с неизменяемыми в ходе решения данными и знаниями, а динамические ЭС дозволяют подобные изменения.

По типам используемых данных и знаний делятся на ЭС с определёнными и неясными познаниями. Под неопределенностью знаний и данных понимаются их недостаточность, ненадежность, неопределенность (нечёткость).

ЭС могут формироваться с применением одного или нескольких источников знаний.

Существует группа прикладных задач, которые решаются с помощью систем, основанных на знаниях, наиболее благополучно, нежели любыми иными средствами. При установлении целесообразности использования подобных систем необходимо придерживаться следующих аспектов:

1) данные и знания надежны и не меняются со временем;

2) пространство возможных решений относительно невелико;

3) в процессе решения задачи должны использоваться формальные рассуждения. Существуют системы, базирующиеся на знаниях, пока ещё не пригодные для решения задач способами проведения аналогий или абстрагирования (человеческий мозг справляется с этим лучше). В свою очередь классические компьютерные программы оказываются эффективнее систем, основанных на знаниях, в тех случаях, когда решение задачи сопряжено с использованием процедурного анализа. Системы, базирующиеся на знаниях, более подходят для решения задач, где требуются формальные рассуждения;

4) должен быть по крайней мере один эксперт, который способен явно

сформулировать свои знания и объяснить свои методы применения этих знаний для решения задач.

Общая структура экспертной системы представлена на рисунке 1. Необходимо учитывать, что реальные ЭС могут обладать наиболее сложной структурой, но блоки, изображенные на рисунке, безусловно присутствуют в любой экспертной системе, так как представляют собой стандарт структуры сегодняшней ЭС.

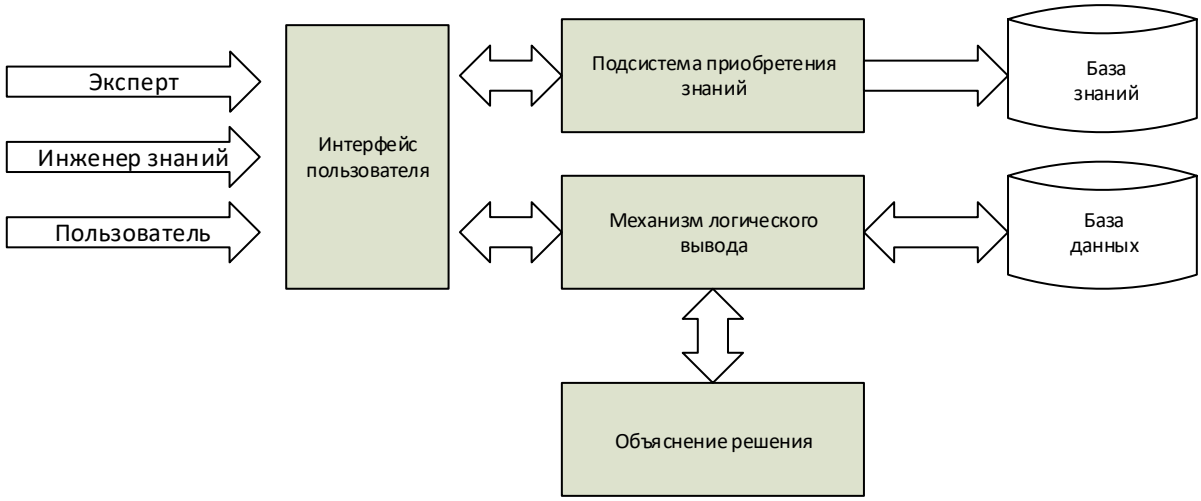

Рисунок 1 - Обобщенная структура экспертной системы

Экспертные системы имеют две категории пользователей и два отдельных «входа», соответствующих различным целям взаимодействия пользователей с ЭС:

1) обычный пользователь (эксперт), которому необходима консультация ЭС – интерактивный режим работы с ней, в ходе которого она решает определенную экспертную задачу. Диалог с ЭС осуществляется посредством диалогового процессора – специальной компоненты ЭС. Существуют две основные формы диалога с ЭС: диалог на ограниченном подмножестве естественного языка (с использованием словаря-меню, при котором на любом шаге диалога система предлагает выбор профессионального лексикона специалистов) и диалог на основе нескольких возможных действий;

2) экспертная группа инженерии знаний, состоящая из специалистов в предметной области и инженеров знаний. В функции данной группы входит за-

полнение базы знаний, исполняемое с помощью специальной диалоговой компоненты ЭС – подсистемы приобретения знаний, что дает возможность отчасти автоматизироваться данный процесс.

Интерфейс пользователя – это совокупность программных и аппаратных средств, которые обеспечивают для конечного пользователя использование ПК для решения задач, возникающих в среде его профессиональной деятельности или без посредников, или с незначительной их помощью. Это комплекс средств интеллектуального интерфейса, обладающих гибкой структурой, что предоставляет возможность синхронизации и восприимчивости в широком спектре интересов конечных пользователей.

Подсистема приобретения знаний специализирована на добавление в базу знаний новых правил и изменение уже имеющихся. В её задачу входит изменение правила к виду, позволяющему подсистеме вывода использовать данное правило в ходе работы. В более сложных системах учтены ещё и средства для контроля вводимых или модифицируемых правил на непротиворечивость с имеющимися правилами.

База знаний – важная компонента экспертной системы, она предназначена для хранения долгосрочных данных, описывающих рассматриваемую предметную область, и правил, описывающих подходящие преобразования данных этой области. В качестве предметной области выбирается узкая (специальная) прикладная область. Затем для создания ЭС в выбранной области собираются факты и правила, которые помещаются в базу знаний совместно с механизмами вывода и упрощения. В отличие от всех остальных компонент ЭС, база знаний – «переменная» часть системы, которая способна пополняться и изменяться инженерами знаний между консультациями (а в некоторых системах и в процессе консультации).

Существует ряд способов представления знаний в ЭС, но общим для всех считается то, что знания представлены в символьной форме (простыми элементами представления знаний являются тексты, списки и прочие символьные структуры). Тем самым, в ЭС реализуется принцип символьной природы рас-

суждений, который состоит в том, что процесс рассуждения представляется как последовательность символьных преобразований. Существуют динамические и статические базы знаний. Динамическая база знаний изменяется со временем. Её содержимое находится в зависимости и от состояния окружающей. Новые факты, добавляемые в базу знаний, считаются результатом вывода, который заключается в применении правил к имеющимся фактам. В системах с монотонным выводом факты, хранимые в базе знаний, статичны, то есть не изменяются в процессе решения задачи. В системах с неоднообразным выводом разрешается перемена либо устранение фактов из базы знаний. В качестве примера системы с немонотонным выводом можно привести ЭС, предназначенную для составления перспективного плана капиталовложения компании. В такой системе по желанию могут быть изменены даже те данные, которые после вывода уже вызвали срабатывание каких-либо правил. Иными словами, имеется возможность модифицировать значения атрибутов в составе фактов, находящихся в рабочей памяти. Изменение фактов в свою очередь приводит к необходимости удаления из базы знаний заключений, полученных с помощью упомянутых правил. Тем самым вывод выполняется повторно для того, чтобы пересмотреть те решения, которые были получены на основе подвергшихся изменению фактов. [3]

База данных (рабочая память) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. [3]

Основой ЭС является подсистема логического заключения [13], которая применяет информацию из базы знаний (БЗ), генерирует рекомендации по решению искомой задачи. Чаще всего с целью представления знаний в ЭС применяются системы продукций и семантические сети. Предположим, БЗ состоит из фактов и правил (если <посылка>, то <заключение>). В случае если ЭС определяет, что посылка верна, то правило признается подходящим для этой консультации и оно запускается в действие. Запуск правила означает принятие заключения данного правила в качестве составляющий части процесса консультации. Целью ЭС является вывод определенного установленного факта, который

называется целевым утверждением (то есть в следствии применения правил добиться того, чтобы данный факт был включен в рабочее множество), либо опровержение этого факта (то есть удостовериться, что его вывести невозможно, следовательно, при данном уровне знаний системы он считается ошибочным). Целевое утверждение может являться или «заложено» предварительно в базу знаний системы, или извлекается системой с диалога с пользователем. Работа системы предполагает собой очередность шагов, на каждом из которых из базы выбирается некоторое правило, которое применяется к текущему содержимому рабочего множества. Цикл завершается, если выведено или опровергнуто целевое утверждение. Цикл работы экспертной системы по-другому именуется логическим выводом. Логический вывод способен осуществляться многими методами, из которых наиболее распространенные – прямой порядок вывода и обратный порядок вывода. Прямой порядок вывода подразумевает переход от фактов, которые находятся в рабочем множестве, к заключению. В случае если подобное заключение удается найти, то оно заносится в рабочее множество. Прямой вывод зачастую называют выводом, управляемым данными.

Объяснительный компонент [3] ЭС разъясняет, как система получила решение задачи (либо вследствие чего она не получила решение), и какие знания она при этом применяла, что облегчает специалисту тестирование и повышает доверие пользователя к полученному результату. Так как системы, базирующиеся на знаниях, реализуются на ПК, то и входная сведения воспринимается в виде, понятном компьютеру, т.е. в битах и байтах. Однако для того, чтобы с системой мог взаимодействовать неподготовленный пользователь, в нее необходимо включить средства общения на естественном языке. Подавляющее большинство систем, основанных на знаниях, владеют довольно простым дизайном на естественном языке - допустимые входные сообщения пользователя ограничены набором понятий, содержащихся в базе знаний.

ЭС работает в двух режимах [3]:

- режиме приобретения знаний;

- режиме решения задачи, называемом также режимом консультации или

режимом использования ЭС.

В режиме приобретения знаний взаимодействие ЭС осуществляет эксперт посредством инженера по знаниям. В данном режиме эксперт, применяя компонент приобретения знаний, заполняет систему знаниями, которые дают возможность ЭС в режиме решения без помощи других (в отсутствии специалиста) решать задачи из проблемной области. Специалист описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы обработки данных, характерные для рассматриваемой области. В традиционном подходе к разработке программного обеспечения режиму получения знаний соответствуют этапы алгоритмизации, программирования и отладки, выполняемые разработчиком программного обеспечения. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ реализовывает не разработчик программного обеспечения, а эксперт (с помощью ЭС), не владеющий навыками программирования.

В режиме консультации взаимодействие с ЭС осуществляет конечный пользователь, которого интересует результат и (либо) способ его получения. Следует выделить, то что в зависимости от назначения ЭС пользователь может не являться специалистом в данной проблемной области (в данном случае он обращается к ЭС за итогом, не умея приобрести его самостоятельно) либо являться экспертом (в данном случае пользователь способен собственноручно получить результат, однако он обращается к ЭС с целью либо ускорить процесс получения результата, или возложить на ЭС эту рутинную работу). В режиме консультации данные о задаче посредством интерфейса поступают в рабочую память (здесь хранятся промежуточные данные решаемой в текущий период задачи). На основе входных данных из рабочей памяти, общих данных о проблемной области и правил базы знаний с помощью механизма логического вывода формируется решение задачи. ЭС при решении задачи не только выполняет предписанную последовательность процедур, но и предварительно формирует её.

1.3 Представление нечетких знаний

Одной из наиболее значимых проблем, характерных для систем, основанных на знаниях, считается проблема представления знаний. Это объясняется тем, что вид представления знаний оказывает значительное воздействие на характеристики и свойства системы. Для этого чтобы манипулировать всевозможными знаниями из реального мира с помощью ПК, следует использовать моделирование. В подобных случаях следует различать знания, предназначенные для обработки компьютером, и знания, используемые человеком.

При проектировании модели представления знаний необходимо принимать во внимание такие факторы, как однородность представления и простота понимания. Однородное представление приводит к упрощению механизма управления логическим выводом и облегчению управления знаниями. Представление знаний обязано являться понятным экспертам и пользователям системы. В противном случае имеет место быть затруднение в приобретении знаний и их оценке. Но осуществить данное условие в равной степени, как для простых, так и для сложных задач достаточно затруднительно. Как правило для несложных задач останавливаются на некотором среднем представлении, однако для решения трудных и больших задач необходимы структуризация и модульное представление.

Типичными моделями представления знаний являются:

- продукционная модель;
- модель, основанная на использовании фреймов;
- модель семантической сети;
- логическая модель;
- модель представления нечетких знаний.

Модели представления нечетких знаний применяются для формализации человеческих знаний, описывающих качественные характеристики (к примеру, большой, сильный, очень сильный, высокий и т.п.) объектов предметной области, которые могут интерпретироваться неопределенно, но содержат немаловажную информацию.

При решении реальных задач зачастую возникают ситуации неопределенности, которые можно разбить на две категории: неполное и недостоверное знание о предметной области и невозможность получить исчерпывающие сведения о конкретном состоянии среды, объекте, ситуации и т.п. В первом случае речь может идти о недостаточно изученных явлениях, противоречивых теориях либо неясно сформулированных концепциях. К примеру, использование в терапии новейших препаратов зачастую дает абсолютно непредсказуемый результат. Существует возможность и противоположной ситуации: предметная область хорошо исследована, однако специалисты предпочитают обращаться к неофициальным, однако наиболее результативным способам, взамен использованию рутинных конкретных методов. К примеру, при поиске неполадки в электрической схеме, как правило, заменяют неисправный блок полностью, за место поиска сгоревших узлов. Существуют и более обыденные источники неопределенности знаний, образцом которых могут являться не внушающие доверия или неточные сведения. Любая информация, поступающая с датчиков приборов: тонометров, глюкометров, аудиометров и иных, обладает некоторой погрешностью. Социологические опросы также предполагают определенный процент погрешностей, предопределенных «человеческим фактором». Информация может быть зашумлена таким образом, что целиком полезный сигнал нереально выделить с общего потока данных. Возможно, также, что приходится использовать информацию, полученную прежде, и которую невозможно не проверить, не дополнить, не получить повторно. Для преодоления трудности неопределенности знаний в области искусственного интеллекта были разработаны различные способы, используемые при построении экспертных систем.

Наиболее неформальный подход – это применение коэффициентов уверенности, выражающих степень достоверности знания. Альтернативный способ заключается в применении теории вероятности. Однако не ясно, как с помощью вероятности представить такие определения равно как «часто», «в некоторых случаях», «молодой», «старый», «небольшой», «умеренный», «высокий» и т.п. Помимо этого, теория вероятности подразумевает существенное количество

вычислений для обновления вероятностных оценок. Обширное распространение приобрели также аппараты нечеткой логики теории и функций доверия. Но в последние годы интерес ученых все больше притягивает теория вероятностей, с её выработанным и строго формализованным аппаратом.

1.3.1 Коэффициенты уверенности

Продемонстрируем применение коэффициентов уверенности на примере системы MYCIN. Общий принцип построения правил данной системы можно представить следующим способом:

*Если условие*1 *и…и условие*M, *то прийти со степенью уверенности* x *к заключение*1 *и … и к заключение*N , (1)

Коэффициент уверенности характеризует меру правдоподобия того или другого заключения, первоначально предопределяемого экспертом. Достоверность утверждений, содержащихся в предпосылках правила, может также обладать нечетким характером, к примеру, в силу предыдущих шагов вывода или из-за неточности источника прецедентов. Уже после использования подобных правил к существующим прецедентам формируется более общее правило, содержащее оценку истинности соблюдения условий:

*Если условие*1 *удовлетворяется с истинностью* x1 *и … и условие*M *удовлетворяется с истинностью* xM, *то прийти к заключению*1 *со степенью уверенности* y1 *и … и к заключению*N *со степенью уверенности* yN. , (2)

В системе MYCIN коэффициенты уверенности (степени уверенности, истинность условий) – CF (certainty factor) – могут принимать значения в диапазоне от -1 вплоть до 1. В совокупном случае возможно применять любой диапазон значений. Положительные значения выражают уверенность в прецедентах, заключениях; отрицательные значения, наоборот, выражают ошибочность

утверждений. Итоговый коэффициент уверенности правила в целом рассчитывается как произведение:

$$CF\ (заключение) = CF\ (предпосылки) * CF\ (правила), \qquad (3)$$

В случае если в БЗ найдется ряд правил для данной предпосылки, то в системе MYCIN заключения данных правил объединяются. Допустим, X и Y – коэффициенты уверенности одинаковых заключений, приобретенные при использовании разных правил, в таком случае:

$$CF(X,Y) = \begin{cases} X + Y - X \times Y, \text{ при } X, Y > 0 \\ X + Y + X \times Y, \text{ при } X, Y < 0 \\ \dfrac{X + Y}{1 - \min(|X|, |Y|)}, \text{ при } (X > 0 \text{ и } Y < 0) \text{ или } (X > 0 \text{ и } Y < 0) \end{cases}, \qquad (4)$$

Несомненно, формула обладает свойством коммутативности, и последовательность гипотез не имеет значения. Расчет коэффициентов уверенности обладает модульным характером, то есть при подсчете достаточно той информации, что уже охватывается в правиле. При этом не имеет значимости, как были получены коэффициенты уверенности, которые характеризуют первичные данные. Данная особенность зачастую применяется при построении ЭС – подразумевается, что для всех без исключения правил, имеющих дело с конкретным параметром, предпосылки этих правил закономерно самостоятельны. В случае, если имеет место зависимость между условиями правил, то необходимо перейти к более общему правилу, то есть взамен некоторых правил вида:

*Если условие*1,*то прийти со степенью уверенности x*1 *к заключение.*
...
Если условиеM,*то прийти со степенью уверенности xN к заключение.* ,(5)

следует использовать одно правило вида:

*Если условие*1 *и … и условие*M*, то прийти со степенью уверенности* x *к заключение.* (6)

1.3.2 Правило Байеса

Для представления неопределенности знаний возможно крайне продуктивно использовать тезисы теории вероятностей. Аналогичные представления основываются на понятии условной вероятности. Из определения следует, что условная вероятность события d при данном s – это вероятность того, то что событие d наступит при условии, что наступило событие s. К примеру, условной вероятностью считается вероятность того, что у больного на самом деле существует болезнь d, в случае если у него найден только лишь симптом s. Для расчета условной вероятности применяется следующая формула:

$$P(d|s) = \frac{P(d \wedge s)}{P(s)}, \tag{7}$$

Из данной формулы мы видим, что условная вероятность предопределяется посредством представления совместности или одновременности событий, в таком случае имеется вероятность совпадения событий d и s, разделенное на вероятность события s. Из приведенной формулы очевидно, что вероятность совпадения двух событий равна произведению вероятности одного из них и условной вероятности другого, вычисленную при условии, что первое имело место:

$$P(d \wedge s) = P(s) \times P(d|s), \tag{8}$$

или

$$P(\mathrm{d} \wedge \mathrm{s}) = P(\mathrm{d}) \times \mathrm{P}(\mathrm{s}|d), \tag{9}$$

Подставляя формулы 8 и 9 в определение условной вероятности (формула 7) получим правило Байеса в простом виде:

$$P(d \mid \mathrm{s}) = \frac{P(\mathrm{s} \mid \mathrm{d}) \times \mathrm{P(d)}}{P(s)}, \tag{10}$$

Это правило дает возможность определить вероятность P(d | s) возникновения события d при условии, что состоялось событие s посредством заранее известной условной вероятности P(s | d). В полученном выражении P(d) – предшествующая вероятность наступления события d, а P(d | s) – последующая вероятность, то есть вероятность того, что событие d произойдет, если установлено, что событие s свершилось. Данное правило порой именуют инверсной формулой для условной вероятности, так как она дает возможность определить вероятность P(d | s) посредством P(s | d).

Для систем, сформированных на знаниях, правило Байеса гораздо удобнее формулы нахождения условной вероятности посредством вероятности одновременного наступления событий P(d | s).В этом довольно просто удостовериться. Допустим, у пациента X существует некоторый симптом симптом_Y и следует выяснить, какова вероятность того, что данный признак является источником болезни заболевание_Z. Для того, чтобы напрямую определить P(заболевание_Z | симптом_Y), необходимо дать оценку каким-либо образом, сколько человек в мире страдают данным заболеванием, и какое количество человек одновременно имеют заболевание_Z и симптом_Y. Подобные сведения, как правило, никак не-доступны либо отсутствует вообще. В особенности, это затрагивает подсчёт значения P(заболевание_Z | симптом_Y).

Но, в случае если взглянуть на вероятность не как на объективную частотность событий при довольно длительных независимых испытаниях, а как на индивидуальную оценку совместного наступления событий, то ситуация значительно упрощается. К примеру, врач может не знать либо не иметь возможности установить, какая часть пациентов, обладающих симптом_Y страда-

ют от заболевания_Z. Однако на основе собственного опыта или литературных данных, врач в состоянии дать оценку, у какой части пациентов, имеющих заболевание_Z, встречается симптом_Y. Таким образом, возможно определить P(симптом_Y | заболевание_Z) и использовать инверсную формулу для условной вероятности.

Ситуация существенно усугубляется, если речь пойдет о множестве признаков и множестве заболеваний. В случае если следует определить условную вероятность для одного симптома из определенного множества симптомов, то понадобится m×n + m + n вычислений, где m – число возможных диагнозов, а n – количество различных признаков. Принимая во внимание, что в медицинской диагностике применяются тысячи видов заболеваний и колоссальное число симптомов, данная задача становится нетривиальной.

Ситуация усложнится, в случае если ввести в процесс постановки диагноза сразу несколько симптомов. В общем виде правило Байеса можно представить следующим способом:

$$P(d \mid s_1 \wedge ... \wedge s_k) = \frac{P(s_1 \wedge ... \wedge s_k \mid d) \times P(\mathrm{d})}{P(s_1 \wedge ... \wedge s_k)}, \qquad (11)$$

Вышеприведенная формула запрашивает (m×n)×k + m + n×k вычислений оценок вероятностей, что даже при незначительном k является огромным количеством. Подобное количество оценок необходимо по той причине, что для расчеты P(s1…..sk) в общем случае сперва необходимо определить P(s1|s2…..sk), P(s2|s3…..sk)…..P(sk). Допустим, что симптомы независимы, в таком случае количество вычислений стремительно снижается и становится таким же, как и в случае учета только одного симптома, так как для независимых si и sj P(si . sj) = P(si) . P(sj).

При невозможности установить независимость, возможно допустить наличие так называемой условной независимости. Это предположение может основываться на каких-либо низкоприоритетных знаниях. К примеру, если в

автомобиле отсутствует топливо и не функционирует освещение, то отталкиваясь от общих представлений о конструкции автомобиля можно предположить, что данные симптомы самостоятельны. Однако если автомобиль не заводится и не работает свет, то подобное предположение сделать уже нельзя. В соответствии с этим, в системе, использующей вероятностные оценки достоверности, необходимо предусматривать средства отслеживания зависимости между используемыми данными.

1.4 Математический аппарат нечеткой логики

Характеристикой нечеткого множества выступает функция принадлежности. Обозначим посредством MFc(x) степень принадлежности к нечеткому множеству C, которая представляет собой обобщение понятия характеристической функции обычного множества. Множество упорядоченных пар типа C={MFc(x)/x}, MFc(x)□[0,1] называется нечетким множеством C. Значение MFc(x)=0 означает отсутствие принадлежности к множеству, 1 – абсолютную принадлежность.

Рассмотрим на примере. Формализуем неточное определение «высокая температура (тела пациента)». В качестве X (область рассуждений) будет выступать шкала температуры в градусах Цельсия. Несомненно, она будет изменяться от 35 до 42 градусов. Нечеткое множество для определения «высокая температура (тела пациента)» может выглядеть следующим образом:

$$C = \{\ldots\ 0/36{,}6;\ 0/36{,}7;\ 0{,}1/36{,}8;\ 0{,}3/36{,}9;\ 0{,}5/37;\ 0{,}7/37{,}1;\ 0{,}9/37{,}2;\ 1/37{,}3;\ \ldots\}, \qquad (12)$$

К примеру, температура тела 36,9 °C принадлежит ко множеству «Высокая» со степенью принадлежности 0,3. Для одного человека температура тела 36,9 °C может оказаться высокой, для другого – не слишком высокой. Именно в этом и проявляется нечеткость задания соответствующего множества.

Для нечетких множеств определены логические операции. Основными, необходимыми для расчетов, являются пересечение и объединение.

Пересечение двух нечетких множеств (нечеткое "И"):

$$A \cap B : MF_{AB}(x) = \min(MF_A(x), MF_B(x)), \quad (13)$$

Объединение двух нечетких множеств (нечеткое "ИЛИ"):

$$A \cup B : MF_{AB}(x) = max(MF_A(x), MF_B(x)), \quad (14)$$

В теории нечетких множеств найден общий подход к использованию операторов пересечения, объединения и дополнения, осуществленный в треугольных нормах и конормах. Из рассмотренной выше реализации операций пересечения и объединения – наиболее распространенные случаи t-нормы и t-конормы.

Для представления нечетких множеств даются понятия нечеткой и лингвистической переменных.

Нечеткая переменная отбражается набором (N,X,A), где N – это название переменной, X – универсальное множество (область рассуждений), A – нечеткое множество на X.

Нечёткие переменные могут являться значениями лингвистической переменной, которая находится на более высоком уровне в отличии от нечеткой переменной.

Любая лингвистическая переменная состоит из:

- названия;

- множества своих значений, которое также называется базовым терм-множеством T. Элементы базового терм-множества представляют собой названия нечетких переменных;

- универсального множества X;

- синтаксического правила G, по которому генерируются новые термы с применением слов естественного или формального языка;

- семантического правила P, которое каждому значению лингвистической

переменной ставит в соответствие нечеткое подмножество множества X.

Проанализируем такое нечеткое понятие как «температура тела». Это и есть название лингвистической переменной. Сформируем для нее базовое терм-множество, которое будет состоять из трех нечетких переменных: «низкая», «нормальная» и «высокая» и зададим область рассуждений в виде X= [35;42] (°C). Последнее, что осталось сделать – построить функции принадлежности для каждого лингвистического терма из базового терм-множества T.

Существует свыше десятка стандартных форм кривых для задания функций принадлежности. Наибольшее распространение получили: треугольная, трапецеидальная и гауссова функции принадлежности.

Треугольная функция принадлежности определяется тройкой чисел (a,b,c), и её значение в точке x вычисляется в соответствии с выражением:

$$MF(x)=\begin{cases}1-\dfrac{b-x}{b-a},\ a\le x\le b\\ 1-\dfrac{x-b}{c-b},\ b\le x\le c\\ 0,\ x\notin(a;\mathrm{c})\end{cases}, \tag{15}$$

При (b-a)=(c-b) имеем случай симметричной треугольной функции принадлежности, которая может быть конкретно установлена двумя параметрами из тройки (a,b,c).

Аналогично для задания трапецеидальной функции принадлежности необходима четверка чисел (a,b,c,d):

$$MF(x)=\begin{cases}1-\dfrac{b-x}{b-a},\ a\le x\le b\\ 1,\ b\le x\le c\\ 1-\dfrac{x-c}{d-c},\ c\le x\le d\\ 0,\ x\notin(a;d)\end{cases}, \tag{16}$$

При (b-a)=(d-c) трапецеидальная функция принадлежности принимает симметричный вид. На рисунке 2 изображены типовые кусочно-линейные функции принадлежности.

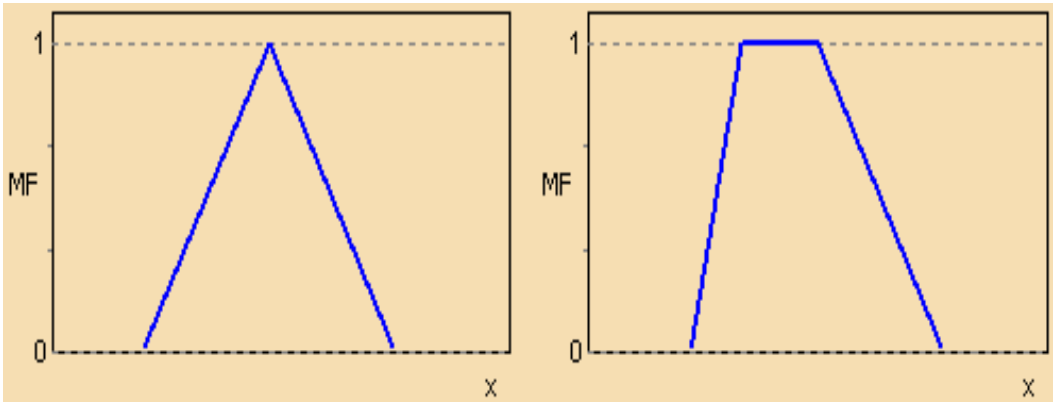

Рисунок 2 – Типовые кусочно-линейные функции принадлежности

Функция принадлежности гауссова типа описывается формулой:

$$MF(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^2\right], \quad (17)$$

Она оперирует двумя параметрами. Параметр c обозначает центр нечеткого множества, а параметр σ отвечает за крутизну функции.

Совокупность функций принадлежности для каждого терма из базового терм-множества T обычно изображаются вместе на одном графике.

На рисунке 3 изображена гауссова функция принадлежности.

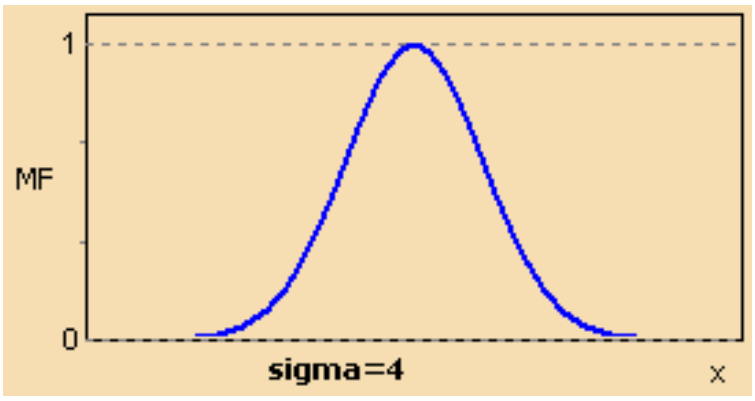

Рисунок 3 – Гауссова функция принадлежности

На рисунке 4 приведен пример описанной выше лингвистической переменной "Температура тела".

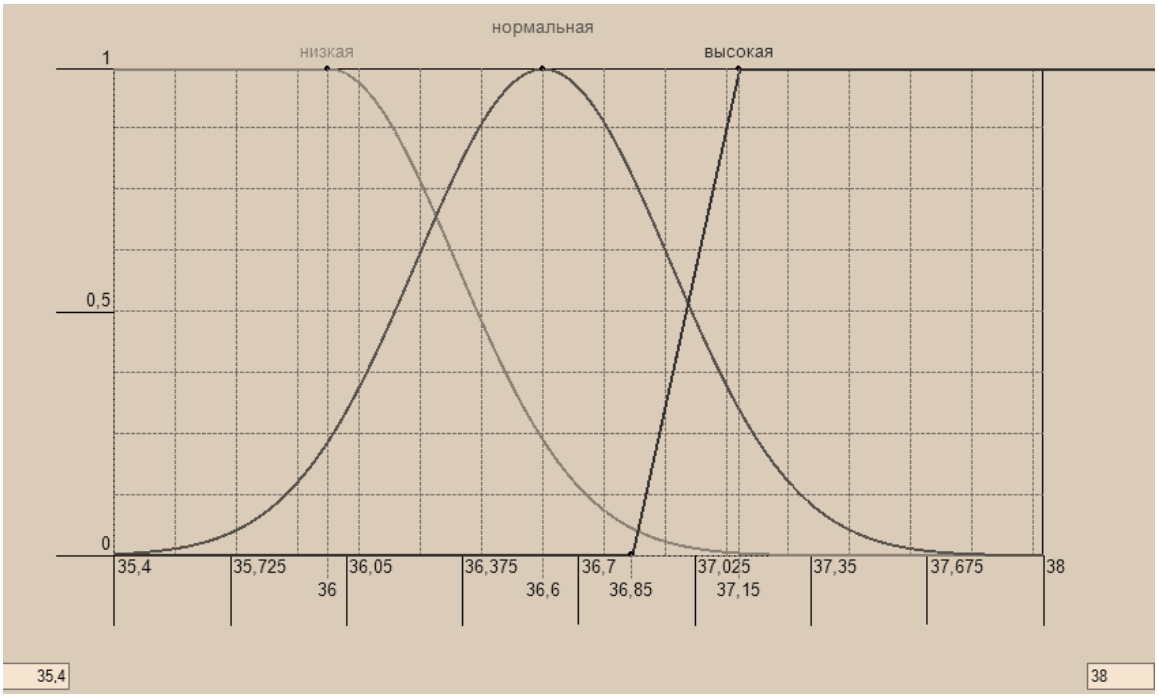

Рисунок 4 – Описание лингвистической переменной «Температура тела»

1.4.1 Нечеткий логический вывод

Основой для выполнения операции нечеткого логического вывода является база правил, включающая в себя нечеткие высказывания в форме «Если-то» и функции принадлежности для соответствующих лингвистических термов. При этом должны соблюдаться следующие условия:

1) существует хотя бы одно правило для каждого лингвистического терма выходной переменной;

2) для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

Пусть в базе правил имеется m правил вида:

$$
\begin{aligned}
&R_1: \textit{ЕСЛИ } x_1 \textit{это } A_{11} \ldots \textit{ И } \ldots x_n \textit{это } A_{1n}, \textit{ ТО } y \textit{ это } B_1 \\
&\ldots \\
&R_i: \textit{ЕСЛИ } x_1 \textit{это } A_{i1} \ldots \textit{ И } \ldots x_n \textit{это } A_{in}, \textit{ ТО } y \textit{ это } B_i \quad , \\
&\ldots \\
&R_m: \textit{ЕСЛИ } x_1 \textit{это } A_{i1} \ldots \textit{ И } \ldots x_n \textit{это } A_{mn}, \textit{ ТО } y \textit{ это } B_m
\end{aligned}
\tag{18}
$$

где x_k , $k=1..n$ – входные переменные;

y – выходная переменная;

A_{ik} – заданные нечеткие множества с функциями принадлежности.

Результатом нечеткого вывода является четкое значение переменной y* на основе заданных четких значений xk , k=1..n.

В общем случае механизм логического вывода включает четыре этапа: введение нечеткости (фазификация), нечеткий вывод, композиция и приведение к четкости, или дефазификация. На рисунке 6 изображена система нечеткого логического вывода.

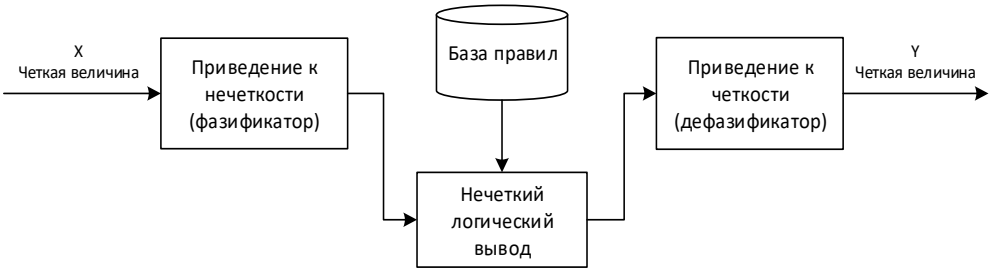

Рисунок 6 – Система нечеткого логического вывода

Алгоритмы нечеткого вывода различаются главным образом видом используемых правил, логических операций и разновидностью метода дефазификации. Разработаны модели нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото.

Рассмотрим нечеткий вывод на примере механизма Мамдани. Это наиболее распространенный способ логического вывода в нечетких системах. В нем используется минимаксная композиция нечетких множеств. Данный механизм включает в себя следующую последовательность действий.

1) Процедура фазификации: определяются степени истинности, т.е. зна-

чения функций принадлежности для левых частей каждого правила (предпосылок). Для базы правил с m правилами обозначим степени истинности как:

$$A_{ik}(x_k),\ i=1..m,\ k=1..n., \tag{19}$$

2) Нечеткий вывод. Сначала определяются уровни "отсечения" для левой части каждого из правил:

$$alfa_i = \min_i(\mathrm{A}_{ik}(x_k)), \tag{20}$$

Далее находятся «усеченные» функции принадлежности:

$$B_i^*(y) = \min_i(alfa_i, \mathrm{B}_i(y)), \tag{21}$$

3) Композиция, или объединение полученных усеченных функций, для чего используется максимальная композиция нечетких множеств:

$$MF(y) = \max_i(B_i^*(y)), \tag{22}$$

где *MF(y)* – функция принадлежности итогового нечеткого множества.

4) Дефазификация, или приведение к четкости. Существует несколько методов дефазификации. Например, метод среднего центра, или центроидный метод:

$$MF(y) = \max_i(B_i^*(y)), \tag{23}$$

Геометрический смысл такого значения – центр тяжести для кривой MF(y). Рисунок 7 графически показывает процесс нечеткого вывода по Мамда-

ни для двух входных переменных и двух нечетких правил R1 и R2.

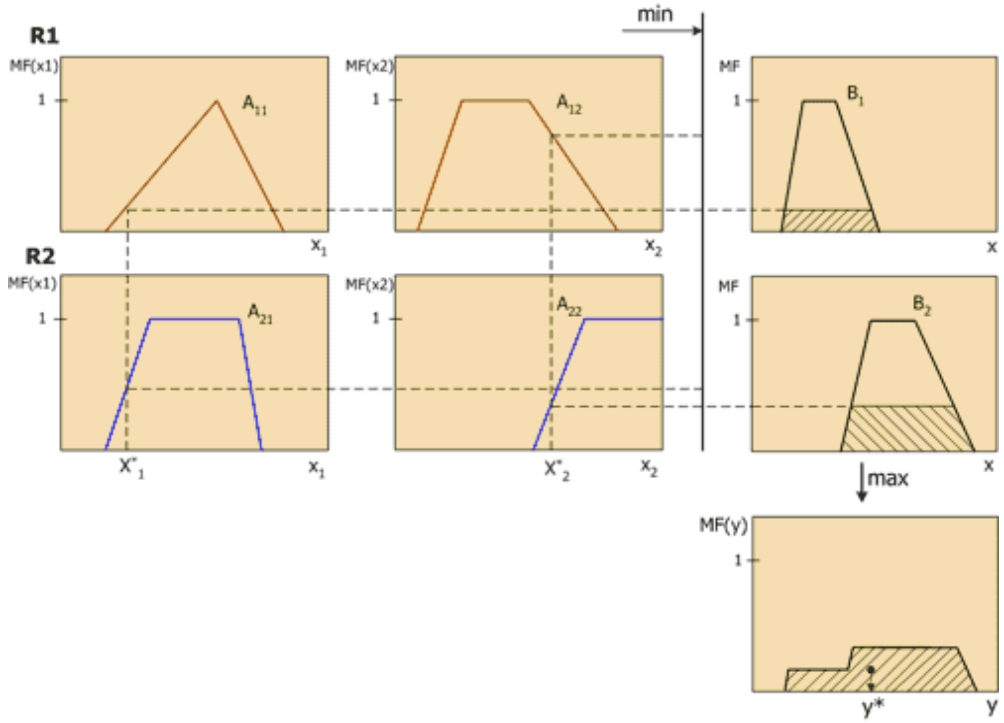

Рисунок 7 – Схема нечеткого вывода по Мамдани

1.4.2 Интеграция с интеллектуальными парадигмами

Гибридизация методов интеллектуальной обработки информации – девиз, под которым прошли 90 годы у западных и американских исследователей. В результате объединения нескольких технологий искусственного интеллекта появился специальный термин – «мягкие вычисления» (soft computing), который ввел Л. Заде в 1994 году. В настоящее время мягкие вычисления объединяют такие области как: нечеткая логика, искусственные нейронные сети, вероятностные рассуждения и эволюционные алгоритмы. Они дополняют друг друга и используются в различных комбинациях для создания гибридных интеллектуальных систем.

Влияние нечеткой логики оказалось, пожалуй, самым значительным. Подобно тому, как нечеткие множества расширили рамки классической математической теории множеств, нечеткая логика «вторглась» практически в большинство методов Data Mining, наделив их новой функциональностью. Ниже приводятся примеры таких объединений:

1) нечеткие нейронные сети (fuzzy-neural networks) осуществляют выводы

на основе аппарата нечеткой логики, однако параметры функций принадлежности настраиваются с применением алгоритмов обучения НС. Поэтому для подбора параметров таких сетей применим метод обратного распространения ошибки, изначально предложенный для обучения многослойного персептрона. Для этого модуль нечеткого управления представляется в форме многослойной сети. Нечеткая нейронная сеть как правило состоит из четырех слоев: слоя фазификации входных переменных, слоя агрегирования значений активации условия, слоя агрегирования нечетких правил и выходного слоя.

Наибольшее распространение в настоящее время получили архитектуры нечеткой НС вида ANFIS и TSK. Доказано, что такие сети являются универсальными аппроксиматорами.

Быстрые алгоритмы обучения и интерпретируемость накопленных знаний сделали сегодня нечеткие нейронные сети одним из самых перспективных и эффективных инструментов мягких вычислений.

2) адаптивные нечеткие системы. Классические нечеткие системы обладают недостатком – для формирования правил и функций принадлежности необходимо мнение экспертов той или иной предметной области, что не всегда представляется возможным. Адаптивные нечеткие системы (adaptive fuzzy systems) решают данную проблему. В таких системах подбор параметров нечеткой системы совершается в процессе обучения на экспериментальных данных. Алгоритмы обучения адаптивных нечетких систем относительно трудоемки и сложны по сравнению с алгоритмами обучения нейронных сетей, и, как правило, состоят из двух стадий: генерации лингвистических правил и корректировки функций принадлежности. Первая задача относится к задаче переборного типа, вторая – к оптимизации в непрерывных пространствах. При этом возникает определенное противоречие: для генерации нечетких правил необходимы функции принадлежности, а для проведения нечеткого вывода – правила. Кроме того, при автоматической генерации нечетких правил необходимо обеспечить их полноту и непротиворечивость.

Значительная часть методов обучения нечетких систем использует гене-

тические алгоритмы. В англоязычной литературе этому соответствует специальный термин – Genetic Fuzzy Systems.

Значительный вклад в развитие теории и практики нечетких систем с эволюционной адаптацией внесла группа испанских исследователей во главе с Ф. Херрера (F. Herrera).

3) нечеткие запросы к базам данных (fuzzy queries) – перспективное направление в современных системах обработки информации. Данный инструмент дает возможность формулировать запросы на естественном языке, к примеру: "Вывести список недорогих предложений о съеме жилья близко к центру города", что невозможно при использовании стандартного механизма запросов. Для этой цели разработана нечеткая реляционная алгебра и специальные расширения языков SQL для нечетких запросов. Большая часть исследований в этой области принадлежит западноевропейским ученым Д. Дюбуа и Г. Праде.

4) нечеткие ассоциативные правила (fuzzy associative rules) – инструмент для извлечения из баз данных закономерностей, которые формулируются в виде лингвистических высказываний, для этого введены специальные понятия нечеткой транзакции, поддержки и достоверности нечеткого ассоциативного правила.

5) нечеткие когнитивные карты (fuzzy cognitive maps) были предложены Б. Коско в 1986 г. и используются для моделирования причинных взаимосвязей, выявленных между концептами некоторой области. В отличие от простых когнитивных карт, нечеткие когнитивные карты представляют собой нечеткий ориентированный граф, узлы которого являются нечеткими множествами. Направленные ребра графа не только отражают причинно-следственные связи между концептами, но и устанавливают степень влияния (вес) связываемых концептов. Активное использование нечетких когнитивных карт в качестве средства моделирования систем обусловлено возможностью наглядного представления анализируемой системы и легкостью интерпретации причинно-следственных связей между концептами. Основные проблемы связаны с процессом построения когнитивной карты, который не поддается формализации.

Кроме того, необходимо доказать, что построенная когнитивная карта адекватна реальной моделируемой системе. Для решения данных проблем разработаны алгоритмы автоматического построения когнитивных карт на основе выборки данных.

6) нечеткие методы кластеризации, в отличие от четких методов (например, нейронные сети Кохонена), позволяют одному и тому же объекту принадлежать одновременно нескольким кластерам, но с различной степенью. Нечеткая кластеризация во многих ситуациях более "естественна", чем четкая, например, для объектов, расположенных на границе кластеров. Наиболее распространены: алгоритм нечеткой самоорганизации c-means и его обобщение в виде алгоритма Густафсона-Кесселя.

Список можно продолжить и дальше: нечеткие деревья решений, нечеткие сети Петри, нечеткая ассоциативная память, нечеткие самоорганизующиеся карты и другие гибридные методы.

2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Обоснование выбора среды разработки

Выбор среды разработки ЭС для полноценного приложения, которое в дальнейшем сможет расти и развиваться – задача, требующая тщательного анализа всех достоинств и недостатков той или иной среды. Так же и выбор системы управления базами данных (СУБД) представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений баз данных. Выбранный программный продукт должен удовлетворять как текущим, так и будущим потребностям конечных пользователей разрабатываемого продукта, при этом следует учитывать финансовые затраты на приобретение необходимого оборудования, самой системы, разработку необходимого программного обеспечения на ее основе, а также обучение персонала.

В конечном итоге средой разработки была выбрана IDE Microsoft Visual Studio 2017 Community, так как обладает огромными возможностями разработки ПО с бесплатной лицензией и гибкой библиотекой встроенный средств, инструментов и фреймоворков.

Языком программирования был выбран C# с использованием .Net Framework v.4.5.2 из-за своей гибкости и кроссплатформенности. Возможно, в дальнейшем появиться необходимость портирования разрабатываемого программного продукта на мобильные устройства, что в перспективе C# дает реализовать без особых проблем.

В качестве СУБД была выбрана MS SQL Server, так как уже встроена в инструментарий Visual Studio 2017 и позволяет проектировать программные продукты с поддержкой большого числа пользователей и создавать централизованное хранилище данных и знаний. Для работы с локальной копией БД используется SQLite.

2.2 Функциональное проектирование

Прежде чем приступить к функциональному проектированию медицин-

ской информационной системе, согласно определению, МИС – это совокупность информационных, организационных, программных и технических средств, предназначенных для автоматизации медицинских процессов и(или) организаций [2], необходимо определить какой именно медицинский процесс она будет автоматизировать.

В качестве примера функциональной модели медицинской организации была выбрана ГАУЗ АО «АОКБ».

ГАУЗ АО «Амурская областная клиническая больница» – это ведущее многопрофильное лечебно-профилактическое учреждение Амурской области, одно из крупнейших специализированных медицинских учреждений Дальневосточного Федерального округа, оказывающее плановую и экстренную медицинскую помощь жителям Амурской области практически по всем медицинским профилям в круглосуточном режиме.

Стационар больницы рассчитан на 1060 коек: 29 специализированных отделений, из них - 11 единственные в области. Лечебно-диагностические отделения оснащены самым современным медицинским оборудованием: МРТ, компьютерными томографами, ангиографом, современными рентгенохирургическими и ультразвуковыми установками. Постоянно укрепляется и совершенствуется материально-техническая база, применяются высокотехнологические методы диагностики и лечения, проводятся различного рода реконструктивно-восстановительные операции. Широко внедряются лапароскопическая и лазерная хирургия, дистанционная литотрипсия, эндовидеохирургия, косметологическая хирургия, ультразвуковое и лазерное облучение крови, плазмоферез (расширена служба амбулаторного диализа, работают 17 аппаратов нового поколения «Искусственная почка» и milti Filtrate) гипербарическая оксигенация. Каждое отделение больницы имеет доступ в Интернет.

В составе больницы работают следующие подразделения:

1) амбулаторно-поликлиническая служба с детским отделением - включает взрослое отделение на 800 посещений в день, где ведется прием по 31 врачебным специальностям. Более 220 детей ежедневно получают лечебно-

консультативную помощь у 20 специалистов педиатрического отделения. Выездная бригада врачей областной поликлиники ежемесячно оказывает консультативную помощь сельским жителям Амурской области;

2) стационар - мощность лечебного учреждения 1125 коек;

3) роддом – 120 коек.

На базе Амурской областной клинической больницы работают 6 центров:

1) территориальный центр медицины катастроф;

2) центр планирования семьи и репродукции;

3) паталогоанатомический центр;

4) колопроктологический центр;

5) реабилитационный Астма-центр;

6) томографический (МРТ и КТ) и УЗ технологий.

Подробная функциональная модель деятельности ГАУЗ АО «АОКБ» представлена в приложении А.

Проектируемая медицинская система будет автоматизировать процесс постановки и уточнения диагноза и должна обеспечивать реализацию следующих функций:

1) информационная:

Система должна предоставлять группе пользователей доступ к информации:

- личные данные пациентов;

- дата и результат анализов пациентов;

- результаты обследования пациентов;

- результаты осмотра пациентов.

2) учёт:

- выявление, измерение, сбор, интерпретации, обобщение, подготовка и предоставление важной для диагностирования информации;

- контроль состояния пациента для постановки диагноза и назначения лечения.

Выходными данными являются данные, получаемые в результате обра-

ботки входной информации.

Так же МИС должна включать в себя модуль экспертной системы, способной в режиме консультации в форме диалога с пользователем поставить предварительный диагноз, назначить лечение и в случае необходимости, с целью обучения персонала иметь поэтапную систему вывода логики «размышлений» при постановке диагноза. Функциональная модель МИС представлена в приложении Б.

2.3 Проектирование БД

2.3.1 Инфологическое проектирование

Цель инфологического проектирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных [7, с.15].

Из анализа задачи проектирования, а также требований, предъявляемых к системе, выделим информацию, необходимую при постановке диагноза. Для этого выделим сущности (таблица 1).

Таблица 1 – Формирование сущностей

Название сущности	Описание сущности	Количество экземпляров
1	2	3
Карта пациента	Краткие сведения о пациенте, о перенесенных заболеваниях, а также результаты первичного осмотра	>100
История болезни	История болезни пациента, в которой хранятся данные о первом обращении, жалобах, диагнозах	>10
Жалобы	Сведения о жалобах пациента, представленные в виде антецедентов с использованием переменных нечеткой логики	>5
Антецедент	Сведения, являющиеся предпосылкой или причиной к заключению в нечеткой логике. В данном случае это симптом с указанием нечеткой переменной и квантификатором. Сущность представляет собой набор структур данных в виде односвязных списков.	>10
Диагноз	Сведения о заболевании или диагнозе, включающие в себя описание, симптомы и рекомендуемое лечение.	>100
База знаний	Хранит в себе набор правил нечеткой логики, заранее составленных экспертами по средствам инженера по знаниям. Включает в себя набор из условий	>100

Продолжение таблицы 1

1	2	3
База знаний	в виде односвязного списка антецедентов, объединённых нечетким оператором «И» и заключением в виде диагноза	>100
Квантификатор	Сведения об используемых квантификаторах, их семантическое представление и математическое правило для изменения функции принадлежности	5
Лингвистическая переменная	Сведения о симптомах в представлении нечеткой логики. Включает в себя название симптома, множество своих значений, которое также называется базовым терм-множеством T (элементы базового терм-множества представляют собой названия нечетких переменных), универсального множества X, синтаксического правила G, по которому генерируются новые термы с применением слов естественного или формального языка, семантического правила P, которое каждому значению лингвистической переменной ставит в соответствие нечеткое подмножество множества X.	>1000
Тип функции принадлежности	Содержатся сведения о типах функций принадлежности.	3
Пограничный тип функции	Сведения о граничных типах функции, таких как левая (убывающая), средняя, правая (возрастающая)	3
Гауссова функция принадлежности	Содержит параметры, характеризующие гауссову функцию принадлежности для каждого терма	>5000
Треугольная функция принадлежности	Содержит параметры, характеризующие линейную треугольную функцию принадлежности для каждого терма	>5000
Трапецеидальная функция принадлежности	Содержит параметры, характеризующие линейную трапецеидальную функцию принадлежности для каждого терма	>5000
Нечеткая переменная (Терм)	Содержит сведения о нечетких переменный, таких как «высоко», «низко» и т.п. и конкретные значения параметров для выбранного типа функции принадлежности	>5000
Пользователь	Содержит сведения о пользователях: имя пользователя, пароль, группа пользователей	>10
Группа пользователей	Содержит описание групп пользователей, а также права и политики для доступа к различным функциям приложения	4

Назначим приведенным выше сущностям атрибуты в форме таблиц 2-17.

Атрибуты сущности «Карта пациента» описаны в таблице 2.

Таблица 2 – Спецификация атрибутов сущности «Карта пациента»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер карты	Номер карты пациентки, присваиваемый карте в регистратуре	1,2…	-	422
Фамилия	Фамилия пациента	-	-	Иванов
Имя	Имя пациента	-	-	Иван
Отчество	Отчество пациента	-	-	Иванович
Пол	Пол пациента	М, Ж	-	М
Дата рождения	Дата рождения пациента	-	-	13.08.1987
Адрес	Адрес регистрации пациента	-	-	г. Благовещенск, ул. Зеленая 21 к.2 кв. 79
Номер телефона	Номер телефона пациента	-	-	+79153816925

Атрибуты сущности «История болезни» описаны в таблице 3.

Таблица 3 – Спецификация атрибутов сущности «История болезни»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер истории	Номер истории болезни	1,2…	-	422
Дата обращения	Дата обращения в больницу	-	-	29.05.2017
Дата окончания лечения	Дата окончания лечения	-	-	25.06.2017
Итоги лечения	Результат лечения, описанный лечащим врачом	-	-	-
Заключение	Врачебное заключение	-	-	-

Атрибуты сущности «Жалобы» описаны в таблице 4.

Таблица 4 – Спецификация атрибутов сущности «Жалобы»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
1	2	3	4	5
Номер жалобы	Порядковый номер описываемой жалобы пациента	1,2…	-	422
Дата первичного проявления	Дата первичного проявления жалобы (день, когда появился симптом)	-	-	25.05.2017
Описание жалобы	Описание жалобы в виде антецедента с использование термов НЛ	-	-	-

Атрибуты сущности «Антецедент» описаны в таблице 5.

Таблица 5 – Спецификация атрибутов сущности «Антецедент»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер антецедента	Порядковый номер описываемого антецедента	1,2…	-	422
Описание	Описание антецедента со ссылками на лингвистическую переменную (симптом), на нечеткую переменную, на квантификатор	-	-	-
Номер следующего антецедента	Поле для реализации алгоритма односвязного списка. Здесь указывается номер следующего антецедента с использованием логической «И». Если это последний антецедент, то указывается 0	0,1,2…	-	0

Атрибуты сущности «Диагноз» описаны в таблице 6.

Таблица 6 – Спецификация атрибутов сущности «Диагноз»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер диагноза	Порядковый номер диагноза, хранящегося в БД	1,2…	-	422
Наименование диагноза	Наименование диагноза или название болезни	-	-	Простуда
Описание	Описание заболевания с указанием возможных осложнений	-	-	-
Симптомы	Перечисление симптомов данного заболевания на естественном языке, понятном для пользователя	-	-	-
Лечение	Описание рекомендуемого курса лечения.	-	-	-

Атрибуты сущности «База знаний» описаны в таблице 7.

Таблица 7 – Спецификация атрибутов сущности «База знаний»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер правила	Номер правила, хранящегося в базе знаний	1,2…	-	422
Описание антецедентов	Описание антецедентов, участвующих в правиле, в виде односвязного списка	-	-	-
Заключение	Заключение правила	-	-	-

Атрибуты сущности «Квантификатор» описаны в таблице 8.

Таблица 8 – Спецификация атрибутов сущности «Квантификатор»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер квантификатора	Номер квантификатора, хранящегося в БД	1,2…	-	422
Наименование	Наименование на естественном языке	Не, Очень, Более-менее	-	не
Правило воздействия на функцию принадлежности	Правило воздействия на функцию принадлежности включает в себя математическую формулу, которая изменяет вид графика функции принадлежности при использовании этого квантификатора	-	-	1 – MF(X)

Атрибуты сущности «Лингвистическая переменная» описаны в таблице 9.

Таблица 9 – Спецификация атрибутов сущности «Лингвистическая переменная»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
1	2	3	4	5
Номер лингвистической переменной	Номер лингвистической переменной, хранящейся в БД	1,2…	-	422
Имя симптома	В конкретном случае проектирования МИС, в виде лингвистических переменных представлены симптомы заболеваний	-	-	температура
Нижняя граница области рассуждений	Численный показатель нижней границы области рассуждения для этого симптома	-	-	35
Верхняя граница области рассуждения	Численный показатель верхней границы области рассуждения	-	-	42
Единица измерения области рассуждения	В каких единицах измерений отображается область рассуждения	-	-	ºС

Атрибуты сущности «Тип функции принадлежности» описаны в таблице 10.

Таблица 10 – Атрибуты сущности «Тип функции принадлежности»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
1	2	3	4	5
Номер типа функции принадлежности	Номер типа функции принадлежности	1,2…	-	422
Название	Название типа функции принадлежности	Гауссова, Треугольная, трапецеидальная	-	Гауссова
Описание	Описание типа функции принадлежности	-	-	-

Атрибуты сущности «Пограничный тип функции» описаны в таблице 11.

Таблица 11 – Спецификация атрибутов сущности «Пограничный тип функции»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер пограничного типа	Номер пограничного типа функции принадлежности, который идентифицирует этот тип в БД	1,2…	-	422
Название	Название пограничного типа функции принадлежности	Левая, средняя, правая	-	средняя
Описание	Описание пограничного типа функции принадлежности	-	-	-

Атрибуты сущности «Гауссова функция принадлежности» описаны в таблице 12.

Таблица 12 – Спецификация атрибутов сущности «Гауссова функция принадлежности»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер гауссовой функции	Номер гауссовой функции принадлежности	1,2…	-	422
C	Параметр *c* в гауссовой функции принадлежности в формуле (17)	-	-	35.5
Sigma	Параметр σ в гауссовой функции принадлежности в формуле (17)	-	-	1.5

Атрибуты сущности «Треугольная функция принадлежности» описаны в таблице 13.

Таблица 13 – Спецификация атрибутов сущности «Треугольная функция принадлежности»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер треугольной функции	Номер треугольной функции принадлежности	1,2…	-	422
A	Параметр a в треугольной функции принадлежности в формуле (15)	-	-	10.1
B	Параметр b в треугольной функции принадлежности в формуле (15)	-	-	11.5
C	Параметр c в треугольной функции принадлежности в формуле (15)	-	-	13.5

Атрибуты сущности «Трапецеидальная функция принадлежности» описаны в таблице 14.

Таблица 14 – Спецификация атрибутов сущности «Трапецеидальная функция принадлежности»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
1	2	3	4	5
Номер трапецеидальной функции	Номер трапецеидальной функции принадлежности	1,2…	-	422
A	Параметр a в треугольной функции принадлежности в формуле (16)	-	-	10.1
B	Параметр b в треугольной функции принадлежности в формуле (16)	-	-	11.5
C	Параметр c в треугольной функции принадлежности в формуле (16)	-	-	12.5
D	Параметр d в треугольной функции принадлежности в формуле (16)	-	-	14.2

Атрибуты сущности «Нечеткая переменная» описаны в таблице 15.

Таблица 15 – Спецификация атрибутов сущности «Нечеткая переменная»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
1	2	3	4	5
Номер терма	Номер НП в БД	1,2…	-	422

Продолжение таблицы 15

1	2	3	4	5
Имя терма	Наименование нечеткой переменной является названием терма	-	-	Высокий
Тип функции принадлежности	Тип функции принадлежности, которая описывает данный терм и его нечеткое множество	Гауссова, Треугольная, трапецеидальная	-	гауссова
Цвете на графике	Цвет RGB линий на графике для отображения на интерфейсе пользователя	(0…255; 0…255; 0…255)	-	(255; 127; 0)

Атрибуты сущности «Пользователь» описаны в таблице 16.

Таблица 16 – Спецификация атрибутов сущности «Пользователь»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер пользователя	Номер пользователя в БД	1,2…	-	422
Имя пользователя	Имя пользователя (login) для входа в систему	-	-	admin
Пароль	Пароль пользователя для входа в систему	-	-	гауссова
Группа пользователей	Группа пользователя, которая определяет права и политики доступа к различным функциям МИС	Администраторы, Пользователи, Эксперты, Мед. персонал	-	Администратор

Атрибуты сущности «Группа пользователей» описаны в таблице 17.

Таблица 17 – Спецификация атрибутов сущности «Группа пользователей»

Название атрибута	Описание атрибута	Диапазон значений	Единицы измерений	Пример
Номер группы	Номер группы пользователей	1,2…	-	422
Наименование	Наименование группы пользователей	-	-	Администратор
Описание	Описание прав и политик доступа пользователей к различным модулям системы	-	-	

Следующим этапом инфологического проектирования является выбор

идентифицирующего атрибута (первичного ключа) для сущностей.

Идентифицирующим атрибутом сущности «Карта пациента» является атрибут – «Номер карты», т.к. этот атрибут однозначно идентифицирует каждого пациента.

Для сущности «История болезни» атрибут «Номер истории» – является однозначно идентифицирующим.

Идентифицирующим атрибутом сущности «Жалоба» является атрибут – «Номер жалобы», т.к. этот атрибут однозначно идентифицирует каждую описанную пациентом жалобу.

Идентифицирующим атрибутом сущности «Антецедент» является атрибут – «Номер антецедента», т.к. этот атрибут однозначно идентифицирует каждый описанный антецедент для правил базы знаний.

Идентифицирующим атрибутом сущности «Диагноз» является атрибут – «Номер диагноза», т.к. этот атрибут однозначно идентифицирует каждый описанный диагноз базы знаний, который будет являться заключением правила.

Идентифицирующим атрибутом сущности «База знаний» является атрибут – «Номер правила», т.к. этот атрибут однозначно идентифицирует каждое, представленное в базе знаний, правило.

Идентифицирующим атрибутом сущности «Квантификатор» является атрибут – «Номер квантификатора», т.к. этот атрибут однозначно идентифицирует каждый квантификатор.

Идентифицирующим атрибутом сущности «Лингвистическая переменная» является атрибут – «Номер лингвистической переменной», т.к. этот атрибут однозначно идентифицирует каждую лингвистическую переменную или, в частном случае, каждый симптом в представлении нечеткой логики.

Идентифицирующим атрибутом сущности «Тип функции принадлежности» является атрибут – «Номер типа функции принадлежности», т.к. этот атрибут однозначно идентифицирует каждый тип функции принадлежности.

Идентифицирующим атрибутом сущности «Пограничный тип функции» является атрибут – «Номер пограничного типа», т.к. этот атрибут однозначно

идентифицирует каждый пограничный тип функции принадлежности.

Идентифицирующим атрибутом сущности «Гауссова функция принадлежности» является атрибут – «Номер гауссовой функции», т.к. этот атрибут однозначно идентифицирует каждую заданную гауссовою функцию принадлежности.

Идентифицирующим атрибутом сущности «Треугольная функция принадлежности» является атрибут – «Номер треугольной функции», т.к. этот атрибут однозначно идентифицирует каждую заданную треугольную функцию принадлежности.

Идентифицирующим атрибутом сущности «Трапецеидальная функция принадлежности» является атрибут – «Номер трапецеидальной функции», т.к. этот атрибут однозначно идентифицирует каждую заданную трапецеидальную функцию принадлежности.

Идентифицирующим атрибутом сущности «Нечеткая переменная (Терм)» является атрибут – «Номер терма», т.к. этот атрибут однозначно идентифицирует каждый заданный терм или имя нечеткой переменной.

Идентифицирующим атрибутом сущности «Пользователь» является атрибут – «Номер Пользователя», т.к. этот атрибут однозначно идентифицирует каждого пользователя системы.

Идентифицирующим атрибутом сущности «Группа пользователей» является атрибут – «Номер группы пользователей», т.к. этот атрибут однозначно идентифицирует каждую группу пользователей системы.

Для получения концептуальной инфологической модели, которая позволяет моделировать объекты предметной области и связи между ними, необходимо установить связи между сущностями на основе модели предметной области «сущность-связь».

Модель «сущность-связь» предполагает несколько типов связи: «один-к–одному», «один-ко-многим», «многие-ко-многим». Связь «один-к-одному» означает, что в каждый момент времени каждому экземпляру сущности A соответствует 1 и только 1 экземпляр сущности B и наоборот. Связь «один-ко-

многим» обозначает, что одному представителю сущности А соответствуют 0, 1 или несколько представителей сущности В, но каждому экземпляру сущности В соответствует только 1 экземпляр сущности А. Связь «многие-ко-многим» показывает, что одному представителю сущности А соответствуют 0, 1 или несколько представителей сущности В и наоборот [8, с.21].

Установим связи между выделенными сущностями. В таблице 18 представлены все связи между сущностями проектируемой системы.

Таблица 18 – Связи между сущностями

Название первой сущности, участвующей в связи	Название второй сущности, участвующей в связи	Название связи	Тип связи	Обоснование выбора типа связи
1	2	3	4	5
Карта пациента	История болезни	содержит	Один ко многим	Каждой записи сущности «Карта пациента» соответствует много записей сущности «История болезни», каждой записи сущности «История болезни» соответствует только одна запись сущности «Карта пациента». Номер карты уникален для каждого объекта
История болезни	Жалобы	содержит	Один ко многим	Каждой записи сущности «История болезни» соответствует много записей сущности «Жалобы», каждой записи сущности «Жалобы» соответствует только одна запись сущности «История болезни»
История болезни	Диагноз	содержит	Один ко многим	Каждой записи сущности «Диагноз» соответствует много записей сущности «История болезни», каждой записи сущности «История болезни» соответствует только одна запись сущности «Диагноз», в истории болезни рассматривается один страховой случай ОМС

Продолжение таблицы 18

1	2	3	4	5
Жалобы	Антецедент	содержит	Один ко многим	Каждой записи сущности «Антецедент» соответствует много записей сущности «Жалобы», каждой записи сущности «Жалобы» соответствует только одна запись сущности «Антецедент»
База знаний	Антецедент	содержит	Один ко многим	Каждой записи сущности «Антецедент» соответствует много записей сущности «База знаний», каждой записи сущности «База знаний» соответствует только одна запись сущности «Антецедент»
База знаний	Диагноз	содержит	Один ко многим	Каждой записи сущности «Диагноз» соответствует много записей сущности «База знаний», каждой записи сущности «База знаний» соответствует одна запись сущности «Диагноз»
Антецедент	Антецедент	относится	Один к одному	Каждой записи сущности «Антецедент» соответствует одна записей сущности «Антецедент», тем самым реализуя структуру данных односвязный список.
Антецедент	Квантификатор	содержит	Один ко многим	Каждой записи сущности «Квантификатор» соответствует много записей сущности «Антецедент», каждой записи сущности «Антецедент» соответствует только одна запись сущности «Квантификатор»
Лингвистическая переменная	Нечеткая переменная (Терм)	содержит	Один ко многим	Каждой записи сущности «ЛП» соответствует много записей сущности «Нечеткая переменная», каждой записи сущности «Нечеткая переменная» соответствует только одна запись сущности «ЛН»

Продолжение таблицы 18

1	2	3	4	5
Антецедент	Нечеткая переменная (Терм)	содержит	Один ко многим	Каждой записи сущности «Нечеткая переменная» соответствует много записей сущности «Антецедент», каждой записи сущности «Антецедент» соответствует только одна запись сущности «Нечеткая переменная»
Тип функции принадлежности	Нечеткая переменная (Терм)	содержит	Один ко многим	Каждой записи сущности «Тип функции принадлежности» соответствует много записей сущности «Нечеткая переменная», каждой записи сущности «Нечеткая переменная» соответствует только одна запись сущности «Тип функции принадлежности»
Пограничный тип функции	Нечеткая переменная (Терм)	содержит	Один ко многим	Каждой записи сущности «Пограничный тип функции» соответствует много записей сущности «Нечеткая переменная», каждой записи сущности «Нечеткая переменная» соответствует только одна запись сущности «Пограничный тип функции»
Гауссова функция принадлежности	Нечеткая переменная (Терм)	относится	Один к одному	Каждой записи сущности «Гауссова функция принадлежности» соответствует только одна запись сущности «Нечеткая переменная», каждой записи сущности «Нечеткая переменная» соответствует только одна запись сущности «Гауссова функция принадлежности»
Треугольная функция принадлежности	Нечеткая переменная (Терм)	относится	Один к одному	Каждой записи сущности «Треугольная функция принадлежности» соответствует только одна запись сущности «Нечеткая переменная», каждой записи сущности «Нечеткая

Продолжение таблицы 18

1	2	3	4	5
Треугольная функция принадлежности	Нечеткая переменная (Терм)	относится	Один к одному	переменная» соответствует только одна запись сущности «Треугольная функция принадлежности»
Трапецеидальная функция принадлежности	Нечеткая переменная (Терм)	относится	Один к одному	Каждой записи сущности «Трапецеидальная функция принадлежности» соответствует только одна запись сущности «Нечеткая переменная (Терм)», каждой записи сущности «Нечеткая переменная (Терм)» соответствует только одна запись сущности «Трапецеидальная функция принадлежности»
Группа пользователей	Пользователь	содержит	Один ко многим	Каждой записи сущности «Группа пользователей» соответствует много записей сущности «Пользователь», каждой записи сущности «Пользователь» соответствует только одна запись сущности «Группа пользователей»

2.3.2 Логическое проектирование

С целью создания совокупности нормализованных отношений, в которых реализованы связи между объектами предметной области и выполнены все преобразования, необходимые для эффективной реализации в среде конкретной СУБД, необходимо провести этап логического проектирования, который выполняется в три этапа:

- установление дополнительных логических связей;

- отображение полученной концептуально-инфологической модели на реляционную модель путем совместного представления в ее отношениях ключевых элементов взаимосвязанных записей;

- анализ полученных отношений на соответствие трем нормальным фор-

мам.

В установление дополнительных логических связей нет необходимости, т.к. все необходимые связи уже установлены.

Далее рассматривается каждая связь между сущностями. Сущность «Гауссова функция принадлежности» и сущность «Нечеткая переменная (Терм)» имеют связь «один – к – одному», выбор исходной сущности производится произвольным образом. В данном случае исходной выберем сущность «Нечеткая переменная (Терм)», а сущность «Гауссова функция принадлежности» будет порожденной.

Сущность «Треугольная функция принадлежности» и сущность «Нечеткая переменная (Терм)» имеют связь «один – к – одному», выбор исходной сущности производится произвольным образом. В данном случае исходной выберем сущность «Нечеткая переменная (Терм)», а сущность «Треугольная функция принадлежности» будет порожденной.

Сущность «Трапецеидальная функция принадлежности» и сущность «Нечеткая переменная (Терм)» имеют связь «один – к – одному», выбор исходной сущности производится произвольным образом. В данном случае исходной выберем сущность «Нечеткая переменная (Терм)», а сущность «Трапецеидальная функция принадлежности» будет порожденной.

Сущность «Карта пациента» и сущность «История болезни» имеют связь «один-ко-многим», а значит исходной сущностью является «Карта пациента», а порожденной сущность «История болезни». К атрибутам сущности «История болезни» добавиться внешний ключ «Номер карты» из сущности «Карта пациента».

Сущность «История болезни» и сущность «Жалобы» имеют связь «один-ко-многим». К атрибутам сущности «Жалобы» добавиться внешний ключ «Номер истории» из сущности «История болезни».

Сущность «Антецедент» и сущность «Жалобы» имеют связь «один-ко-многим». К атрибутам сущности «Жалобы» добавиться внешний ключ «Номер антецедента» из сущности «Антецедент».

Сущность «Диагноз» и сущность «История болезни» имеют связь «один-ко-многим». К атрибутам сущности «История болезни» добавиться внешний ключ «Номер диагноза» из сущности «Диагноз».

Сущность «Диагноз» и сущность «База знаний» имеют связь «один-ко-многим». К атрибутам сущности «База знаний» добавиться внешний ключ «Номер диагноза» из сущности «Диагноз».

Сущность «Антецедент» и сущность «База знаний» имеют связь «один-ко-многим. К атрибутам сущности «База знаний» добавиться внешний ключ «Номер антецедента» из сущности «Диагноз».

Сущность «Квантификатор» и сущность «Антецедент» имеют связь «один-ко-многим». К атрибутам сущности «Антецедент» добавиться внешний ключ «Номер квантификатора» из сущности «Квантификатор».

Сущность «Лингвистическая переменная» и сущность «Нечеткая переменная (Терм)» имеют связь «один-ко-многим», а значит исходной сущностью является «Лингвистическая переменная», а порожденной сущность «Нечеткая переменная (Терм)». К атрибутам сущности «Нечеткая переменная (Терм)» добавиться внешний ключ «Номер лингвистической переменной» из сущности «Лингвистическая переменная».

Сущность «Тип функции принадлежности» и сущность «Нечеткая переменная (Терм)» имеют связь «один-ко-многим». К атрибутам сущности «Нечеткая переменная (Терм)» добавиться внешний ключ «Номер типа функции» из сущности «Тип функции принадлежности».

Сущность «Пограничный тип функции» и сущность «Нечеткая переменная (Терм)» имеют связь «один-ко-многим». К атрибутам сущности «Нечеткая переменная (Терм)» добавиться внешний ключ «Номер пограничного типа» из сущности «Пограничный тип функции».

Сущность «Нечеткая переменная (Терм)» и сущность «Антецедент» имеют связь «один-ко-многим». К атрибутам сущности «Антецедент» добавиться внешний ключ «Номер терма» из сущности «Нечеткая переменная (Терм)».

Сущность «Группа пользователей» и сущность «Пользователь» имеют

связь «один-ко-многим», а значит исходной сущностью является «Группа пользователей», а порожденной сущность «Пользователь». К атрибутам сущности «Пользователь» добавиться внешний ключ «Номер группы пользователя» из сущности «Группа пользователей».

Следующий этап логического проектирования сводится к нормализации отношений, которая представляет собой формальный аппарат ограничений на формирование отношений, позволяющий устранить дублирование, обеспечивает непротиворечивость хранимых данных, и уменьшает трудозатраты на ведение базы данных.

Все полученные отношения находятся в первой нормальной форме, так как не имеют в своём составе повторяющихся групп атрибутов или сложных атрибутов.

Отношения находятся во второй нормальной форме, если они являются отношениями в первой нормальной форме, и каждый ее не ключевой атрибут функционально полно зависит от ключа.

Все отношения находятся во второй нормальной форме, т.к. они находятся в соответствии с первой нормальной формой и не имеют составного ключа отношения.

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и все атрибуты, которые не являются ключевыми, не имеют транзитивной зависимости от ключевых атрибутов. Проанализировав все отношения, можно сделать вывод, что они находятся в третьей нормальной форме.

Далее представим логическую модель базы данных, полученную с помощью CASE-средства ERwin Model Navigator r7.3. Модель отображена в приложении В.

2.3.3 Физическое проектирование

На этапе физического проектирования составляются проекты таблиц, которые будут реализованы в СУБД.

Каждому отношению поставим в соответствие физическую таблицу (таб-

лицы 19-34).

Таблица 19 – Карта пациента

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер карты	числовой	длинное целое	>0	-	нет	да
Фамилия	текстовый	50	-	-	да	нет
Имя	текстовый	50	-	-	да	нет
Отчество	текстовый	50	-	-	да	нет
Пол	текстовый	10	In(‘мужской’,’женский’)	-	да	нет
Дата рождения	дата/время	-	<= Date()	-	да	нет
Адрес	текстовый	200	-	-	да	нет
Номер телефона	текстовый	200	-	-	да	нет

Таблица 20 – История болезни

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер истории	числовой	длинное целое	>0	-	нет	да
Номер карты	числовой	длинное целое	>0	-	нет	да
Дата обращения	дата/время	-	-	-	да	нет
Дата окончания лечения	дата/время	-	-	-	да	нет
Номер диагноза	числовой	длинное целое	>0	-	да	нет
Заключение	текстовый	-	-	-	да	нет

Таблица 21 – Жалобы

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
1	2	3	4	5	6	7
Номер жалобы	числовой	длинное целое	>0	-	нет	да
Дата первичного проявления	дата/время	-	<= Date()	-	да	нет
Номер антецедента	числовой	длинное целое	>0	-	нет	нет

Продолжение таблицы 21

1	2	3	4	5	6	7
Номер истории болезни	числовой	длинное целое	>0	-	нет	нет

Таблица 22 – Антецедент

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер антецедента	числовой	длинное целое	>0	-	нет	да
Номер следующего антецедента	числовой	длинное целое	>0	-	да	нет
Номер лингвистической переменной	числовой	длинное целое	>0	-	нет	нет
Номер нечеткой переменной	числовой	длинное целое	>0	-	нет	нет
Номер квантификатора	числовой	длинное целое	>0	-	да	нет

Таблица 23 – Диагноз

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер диагноза	числовой	длинное целое	>0	-	нет	да
Наименование диагноза	текстовый	100	-	-	нет	нет
Описание	текстовый	-	-	-	да	нет
Симптомы	текстовый	-	-	-	да	нет
Лечение	текстовый	-	-	-	да	нет

Таблица 24 – База знаний

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер правила	числовой	длинное целое	>0	-	нет	да
Номер антецедента	числовой	длинное целое	>0	-	нет	да
Номер диагноза	числовой	длинное целое	>0	-	нет	да

Таблица 25 – Квантификатор

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер квантификатора	числовой	длинное целое	>0	-	нет	да
Наименование	текстовый	20	-	-	да	нет
Правило воздействия на функцию принадлежности	текстовый	100	-	-	да	нет

Таблица 26 – Лингвистическая переменная

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер лингвистической переменной	числовой	длинное целое	>0	-	нет	да
Имя симптома	текстовый	100	-	-	да	нет
Нижняя граница области рассуждений	числовой	с плавающей точкой	-	-	да	нет
Верхняя граница области рассуждения	числовой	с плавающей точкой	-	-	да	нет
Единица измерения области рассуждения	текстовый	15	-	-	да	нет

Таблица 27 – Тип функции принадлежности

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер типа функции принадлежности	числовой	длинное целое	>0	-	нет	да
Название	текстовый	50	-	-	да	нет
Описание	текстовый	-	-	-	да	нет

Таблица 28 – Пограничный тип функции

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
1	2	3	4	5	6	7

Продолжение таблицы 28

1	2	3	4	5	6	7
Номер пограничного типа функции	числовой	длинное целое	>0	-	нет	да
Название	текстовый	50	-	-	да	нет
Описание	текстовый	-	-	-	да	нет

Таблица 29 – Гауссова функция принадлежности

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер гауссовой функции	числовой	длинное целое	>0	-	нет	да
C	числовой	с плавающей точкой	-	-	нет	нет
Sigma	числовой	с плавающей точкой	-	-	нет	нет

Таблица 30 – Треугольная функция принадлежности

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер треугольной функции	числовой	длинное целое	>0	-	нет	да
A	числовой	с плавающей точкой	-	-	нет	нет
B	числовой	с плавающей точкой	-	-	нет	нет
C	числовой	с плавающей точкой	-	-	нет	нет

Таблица 31 – Трапецеидальная функция принадлежности

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
1	2	3	4	5	6	7
Номер трапецеидальной функции	числовой	длинное целое	>0	-	нет	да
A	числовой	с плавающей точкой	-	-	нет	нет

Продолжение таблицы 31

1	2	3	4	5	6	7
B	числовой	с плавающей точкой	-	-	нет	нет
C	числовой	с плавающей точкой	-	-	нет	нет
D	числовой	с плавающей точкой	-	-	нет	нет

Таблица 32 – Нечеткая переменная (Терм)

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер терма	числовой	длинное целое	>0	-	нет	да
Номер лингвистической переменной	числовой	длинное целое	>0	-	да	нет
Наименование терма	текстовый	50	-	-	да	нет
Номер типа функции принадлежности	числовой	длинное целое	>0	-	да	нет
Номер пограничного типа	числовой	длинное целое	>0	-	да	нет
Номер гауссовой функции	числовой	длинное целое	>0	-	да	нет
Номер треугольной функции	числовой	длинное целое	>0	-	да	нет
Номер трапецеидальной функции	числовой	длинное целое	>0	-	да	нет
Red	числовой	целое	>=0, <=255	255	нет	нет
Green	числовой	целое	>=0, <=255	127	нет	нет
Blue	числовой	целое	>=0, <=255	0	нет	нет

Таблица 33 – Пользователь

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
1	2	3	4	5	6	7

Продолжение таблицы 33

1	2	3	4	5	6	7
Номер пользователя	числовой	длинное целое	>0	-	нет	да
Номер группы пользователей	числовой	длинное целое	>0	-	нет	да
Имя пользователя	текстовый	50	-	-	нет	нет
Пароль	текстовый	14	-	-	да	нет

Таблица 34 – Группа пользователей

Название поля	Тип данных	Длина	Ограничение	Значение по умолчанию	Допустимость Null	Индексация
Номер группы пользователей	числовой	длинное целое	>0	-	нет	да
Наименование	текстовый	100	-	-	да	нет
Описание	текстовый	-	-	-	да	нет

Требования ссылочной целостности одинаковы для всех таблиц и представлены в таблице 35.

Таблица 35 – Правила ссылочной целостности

Название таблицы	Внешний ключ	Требование ссылочной целостности
1	2	3
История болезни	Номер карты	Каскадное обновление, если в таблице «Карта пациента» изменяется Номер карты пациента, то в таблице автоматически во всех записях, относящихся к данному пациенту, изменяется Номер карты. Каскадное удаление, т.е. при удалении записи о пациенте из таблицы, записи в таблице «История болезни» с таким же номером карты - удаляются. Каскадное добавление, т.е. при добавлении записи в любую таблицу необходимо наличие записи с таким же номером карты в таблице «Карта пациента»
История болезни	Номер диагноза	Каскадное обновление, если в таблице «Диагноз» изменяется номер диагноза, то в таблице «История болезни» автомати-

Продолжение таблицы 35

1	2	3
История болезни	Номер диагноза	чески во всех записях изменяется номер диагноза.
Жалобы	Номер истории	Каскадное обновление, если в таблице «История болезни» изменяется номер истории, то в таблице «Жалобы» автоматически во всех записях, относящихся к данной истории, изменяется Номер истории. Каскадное удаление, т.е. при удалении записи об истории болезни из таблицы «История болезни», записи в таблице «Жалобы» с таким же номером истории - удаляются.
Жалобы	Номер антецедента	Каскадное обновление, если в таблице «Антецедент» изменяется номер антецедента, то в таблице «Жалобы» автоматически во всех записях, относящихся к данному антецеденту, изменяется номер антецедента. Каскадное удаление, т.е. при удалении записи об антецеденте из таблицы «Антецедент», записи в таблице «Жалобы» с таким же номером истории - удаляются.
База знаний	Номер диагноза	Каскадное обновление, если в таблице «Диагноз» изменяется Номер диагноза, то в таблице «База знаний» автоматически во всех записях, относящихся к данному диагнозу, изменяется Номер диагноза. Каскадное удаление, т.е. при удалении записи о диагнозе из таблицы «Диагноз», записи в таблице «База знаний» с таким же номером диагноза - удаляются.
База знаний	Антецедент	Каскадное обновление, если в таблице «Антецедент» изменяется Номер антецедента, то в таблице «База знаний» автоматически во всех записях, относящихся к данному антецеден-

Продолжение таблицы 35

1	2	3
База знаний	Антецедент	ту изменяется Номер антецедента. Каскадное удаление, т.е. при удалении записи о антецеденте из таблицы «Антецедент», записи в таблице «База знаний» с таким же номером диагноза - удаляются.
Антецедент	Номер лингвистической переменной	Каскадное обновление, если в таблице «Лингвистическая переменная» изменяется Номер переменной, то в таблице «Антецедент» автоматически во всех записях, относящихся к данной лингвистической переменной, изменяется Номер данной лингвистической переменной, изменяется Номер переменной. Каскадное удаление, т.е. при удалении записи о лингвистической переменной из таблицы «Лингвистическая переменная», записи в таблице «Антецедент» с таким же номером переменной - удаляются.
Антецедент	Номер терма	Каскадное обновление, если в таблице «Нечеткая переменная (Терм)» изменяется Номер терма, то в таблице «Антецедент» автоматически во всех записях, относящихся к данной переменной, изменяется Номер терма. Каскадное удаление, т.е. при удалении записи о переменной из таблицы «Нечеткая перемен ная (Терм)», записи в таблице «Антецедент» с таким же номером переменной - удаляются.
Нечеткая переменная (Терм)	Номер лингвистической переменной	Каскадное обновление, если в таблице «Лингвистическая переменная» изменяется Номер переменной, то в таблице автоматически во всех записях, относящихся к данной лингвистической переменной, изменяется Номер переменной.

Продолжение таблицы 35

1	2	3
Нечеткая переменная (Терм)	Номер лингвистической переменной	Каскадное удаление, т.е. при удалении записи о лингвистической переменной из таблицы «Лингвистическая переменная», записи в таблице «Нечеткая переменная (Терм)» с таким же номером переменной - удаляются.
Пользователь	Номер группы пользователя	Каскадное обновление, если в таблице «Группы пользователей» изменяется Номер группы, то в таблице «Пользователь» автоматически во всех записях, относящихся к данной группе пользователей, изменяется Номер группы.

Далее представим физическую модель базы данных, полученную с помощью CASE-средства ERwin Model Navigator r7.3. Модель отображена в приложении Г.

2.4 Программная реализация

Программная реализация МИС «Диагноз» на базе нечеткой логики была реализована с помощью языка программирования C# с использованием .Net Framework. Среда разработки Microsoft Visual Studio 2017 поддерживает возможность контроля версий. Так же встроенные средства IDE включают в себя СУБД MS SQL server.

SQL - язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Листинг реализации таблиц БД приведен в приложении Д.

Структура проекта приложения приведена на рисунке 8.

Диаграмма классов приведена на рисунке Е.1 в приложении Е. Программная реализация взаимодействия с БД на языке программирования C# представлена в листинге в приложении Ж в виде класса DatabaseManager.cs.

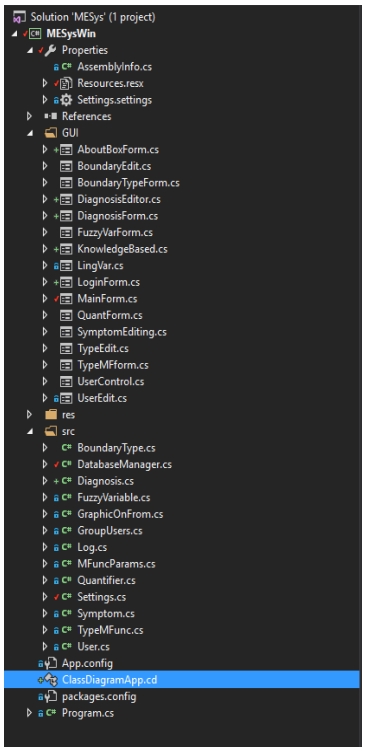

Рисунок 8 – Структура проекта МИС

3 ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ПРОДУКТА

3.1 Характеристика структуры программы

Программные продукты отличаются от привычных программ, которые не имеют определенного набора качественных характеристик, предопределяемых при формировании проектов, эти характеристики невозможно предварительно конкретизировать, т.к. функции обработки, обеспечиваемые программным средством, могут обладать разной глубиной проработки. Время и затраты на разработку программных продуктов не могут быть заранее точно спланированы.

Разработанный программный продукт МИС «Диагноз» на базе нечеткой логики можно отнести к категории алгоритмически сложных программных продуктов, так как основные вычислительные функции МИС имеют достаточно объемную и сложную математическую модель. Несмотря на сложную структуру, приложение обладает следующими качествами:

1) Надежность. Надежность функционирования программного продукта определяется работой без сбоев и стабильностью в работе программ, правильностью выполнения предписанных команд при обработке данных, возможностью диагностики возникающих ошибок в процессе работы приложения. Это качество позволило реализовать гибкая система обработки исключений, включенная в состав .Net Framework. Каждая непредвиденная программная ситуация сразу оповестит пользователя всплывающим сообщением о возможной ошибке.

2) Эффективность. Эффективность программного продукта рассматривается с позиций прямого его назначения - требований пользователя и с точки зрения расхода вычислительных ресурсов, необходимых для его эксплуатации. Несмотря на сложность решаемых задач приложением, оно потребляет относительно малое количество оперативной памяти и, за счет использования двойной буферизации прорисовки элементов управления, переносит часть вычислительных ресурсов на GPU.

3) Учет человеческого фактора означает обеспечение интуитивно понятного интерфейса для работы фактического пользователя, наличие вспомогательного действия при допущении ошибки или обучающего модуля в составе программного средства, хорошей технической документации для продуктивного применения пользователем заложенных в программном средстве функциональных возможностей, анализ и диагностику возникших ошибок и др. МИС разработана для диалогового взаимодействия с пользователем, интуитивно понятна и не отягощена различными элементами управления для настройки сложных взаимосвязей. Весь сложных функционал разбит на более мелкие и понятные и вынесен в отдельные диалоговые формы.

4) Модифицируемость программных продуктов представляет собой совокупность свойств, позволяющих вносить требуемые изменения: расширение функций обработки, переход на другую техническую базу обработки и др. МИС реализована таким образом, что с помощью инженера по знаниям, эксперты могут изменять, пополнять и удалять любые данные, включая те, которые расширяют функционал приложения. МИС обладает удобным редактором базы знаний, нечетких и лингвистических переменных, а также других данных, влияющих на диалоговые формы.

Системные требование для работы МИС «Диагноз» следующие:

- Процессор 300 MHz или выше;
- Оперативная память – 128 Мб RAM или выше;
- Видеоадаптер и монитор – SuperVGA (1024x720) или выше;
- Свободное место на HDD – 20 Мб или выше;
- клавиатура и мышь;
- Операционная система Windows XP / Vista / 7 / 8 / 8.1 / 10.

3.2 Руководство пользователя

При запуске МИС «Диагноз» открывается стартовое окно приложения. Экранная форма приведена на рисунке 9.

Приложение по умолчанию открывается из-под пользователя «Гость». Имя пользователя и режим, в котором работает приложение в данный момент

отображаются в правом верхнем углу. Функционал приложения, учитывая права неавторизованного пользователя, ограничивает доступ к функциям редактирования данных и просмотра карт пациентов. Как видно из контекстного меню, доступны следующие пункты: Файл, консультация, сервис и помощь.

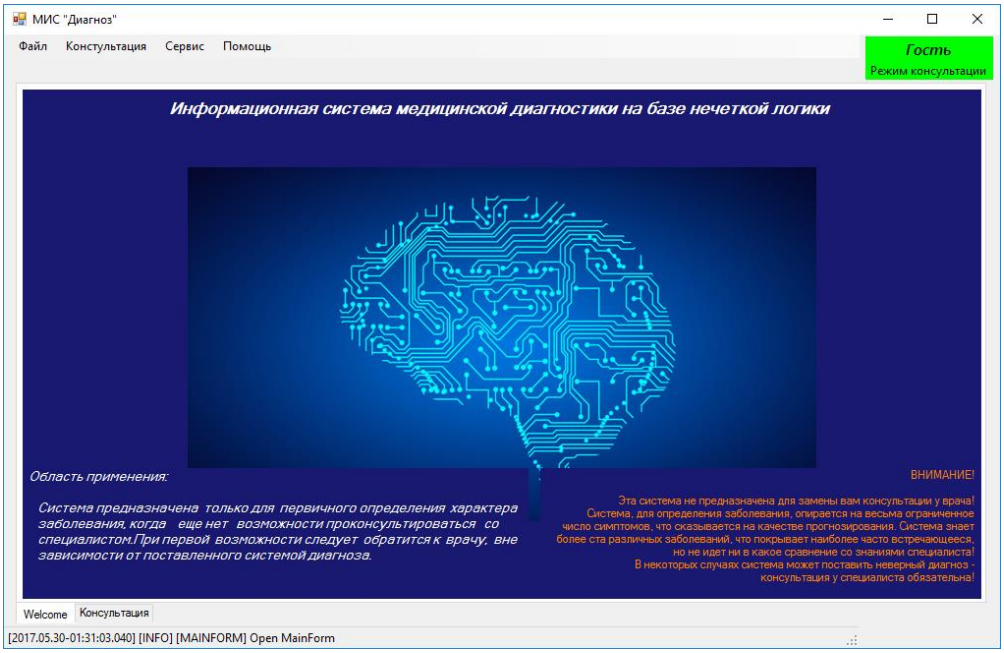

Рисунок 9 – Главная форма приложения МИС «Диагноз»

В пункте меню «Помощь» если выбрать подпункт «О программе…», то открывается экранная форма, которая изображена на рисунке 10.

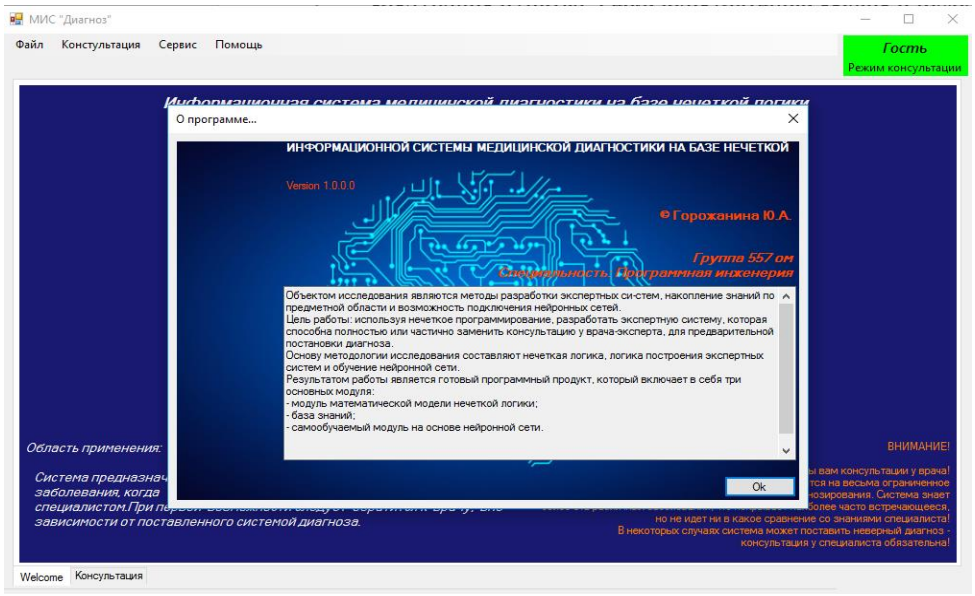

Рисунок 10 – О программе

Пункт меню «Сервис» включает в себя подпункты «Настройка приложения» и «Настройка пользователя». Причем для пользователя «Гость», второй пункт меню недоступен.

В пункте меню «Консультация» есть следующие подпункты: «Карта пациента» и «Диф. диагностика методом диалога с ЭС», причем «Карта пациента» так же недоступна из-за недостатка прав пользователя.

Если выбрать пункт меню «Диф. диагностика методом диалога» то на главной форме откроется вкладка «Консультация». Внешний вид экранной формы представлен на рисунке 11.

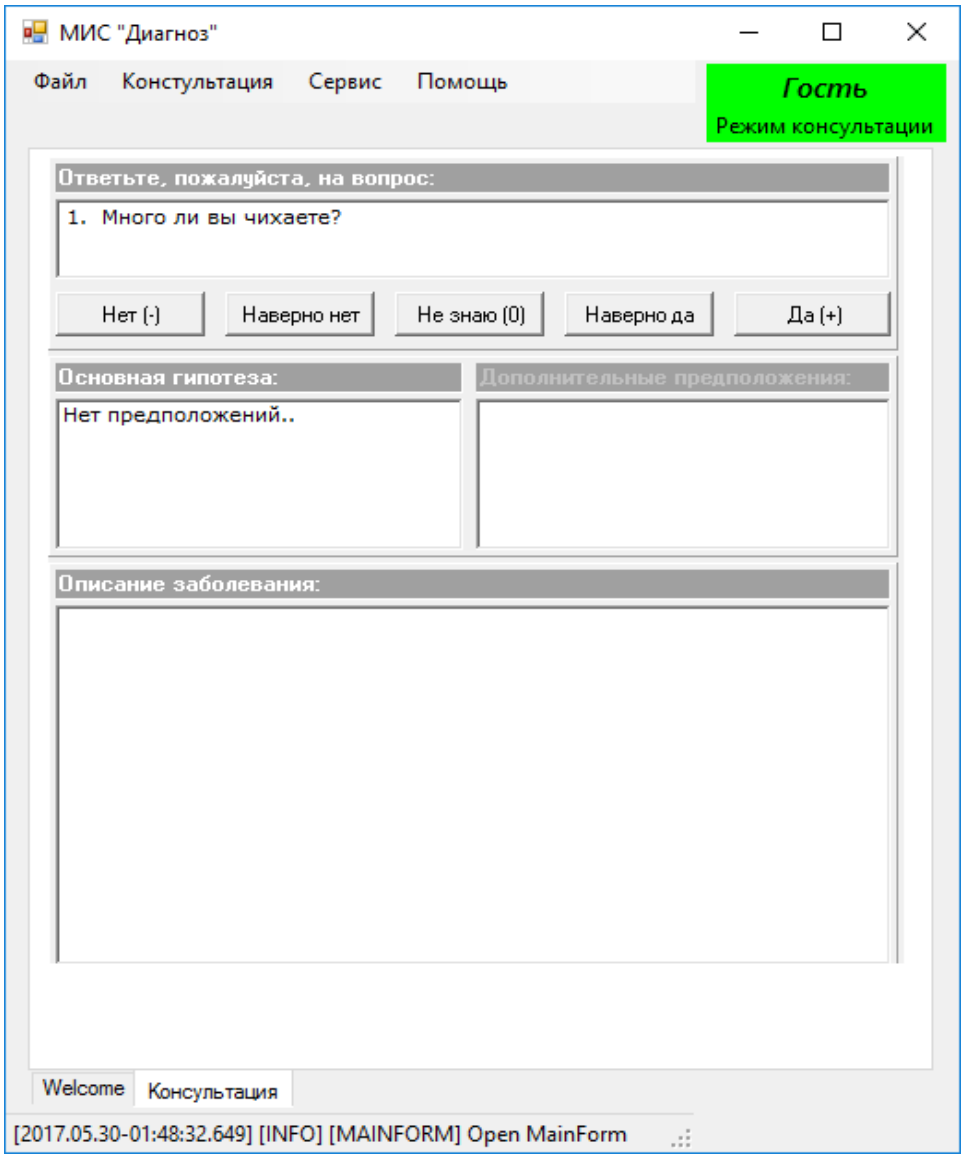

Рисунок 11 – Вкладка «Консультация»

Если начать отвечать на вопросы экспертной системы, то начнут форми-

ровать предположения о возможных диагнозах, которые МИС попробует подтвердить или опровергнуть, задавая дополнительные вопросы, опираясь на знания, занесенные экспертом и/или инженером по знаниям в базу знаний. Пример экранной формы в процессе консультации приведен на рисунке 12.

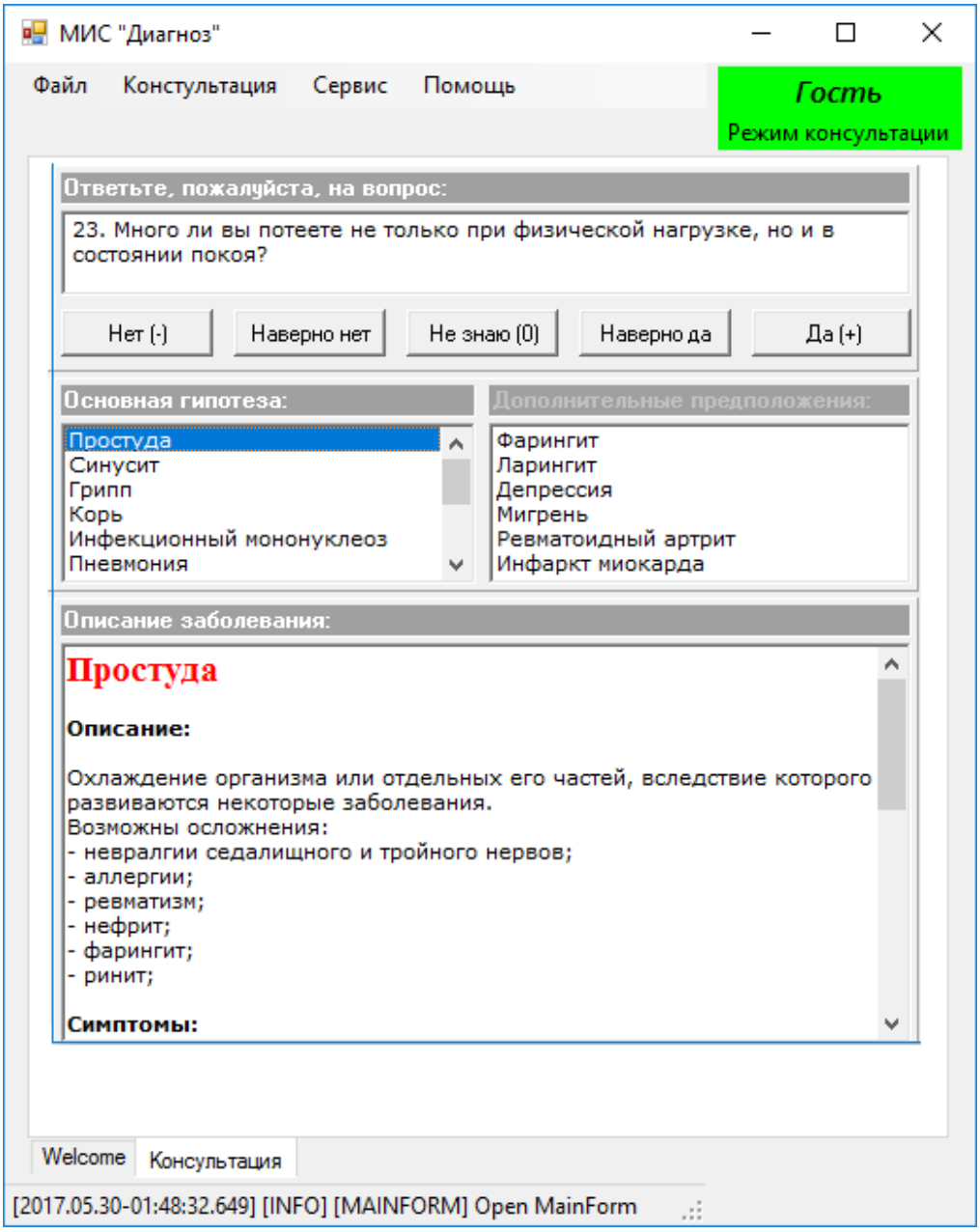

Рисунок 12 – Процесс консультации с ЭС

Чтобы сменить пользователя, необходима кликнуть на имени пользователя в правом верхнем углу формы и ввести логин и пароль. Процесс авторизации пользователя admin с паролем admin можно увидеть на рисунке 13.

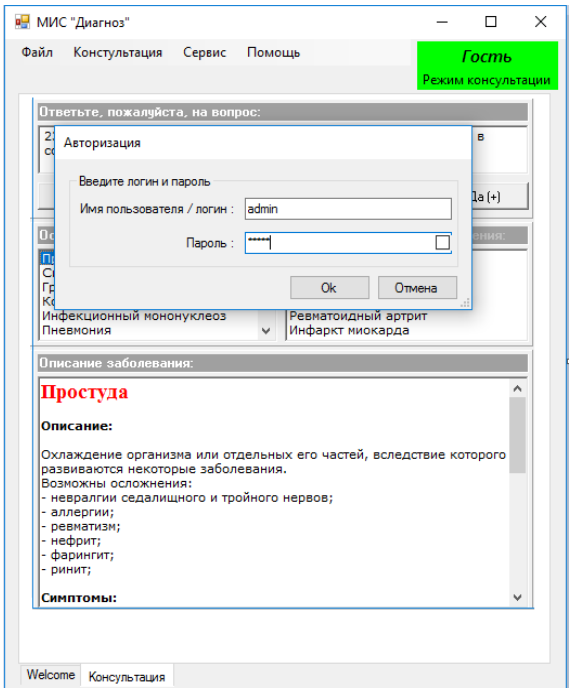

Рисунок 13 – Процесс авторизации администратора

После авторизации пользователя admin, в правом верхнем углу появиться надпись с именем пользователя и подписью «Полный доступ», а также измениться цвет кнопки на красный, что означает, что текущий пользователь имеет полный набор прав и имеет доступ ко всем функциям приложения. Например пользователи из группы администраторов могут добавлять, изменять и удалять других пользователей системы. На рисунке 14 изображено окно редактирования пользователя.

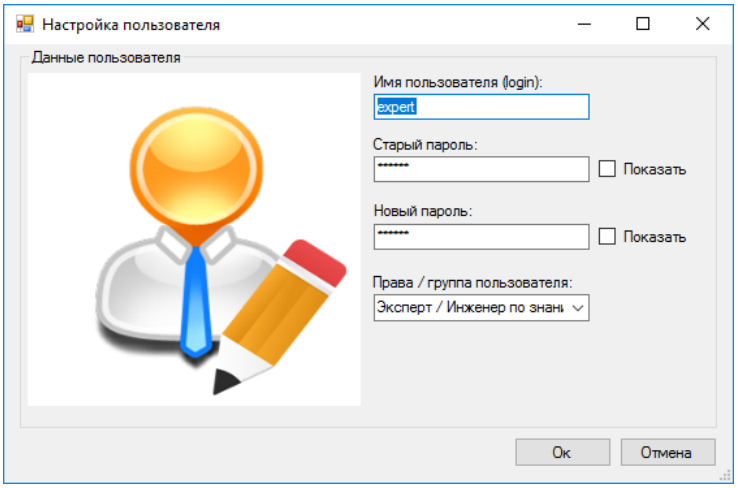

Рисунок 14 – Редактирование пользователя Expert

На рисунке 15 показан раздел администрирования – управление пользова-

телями.

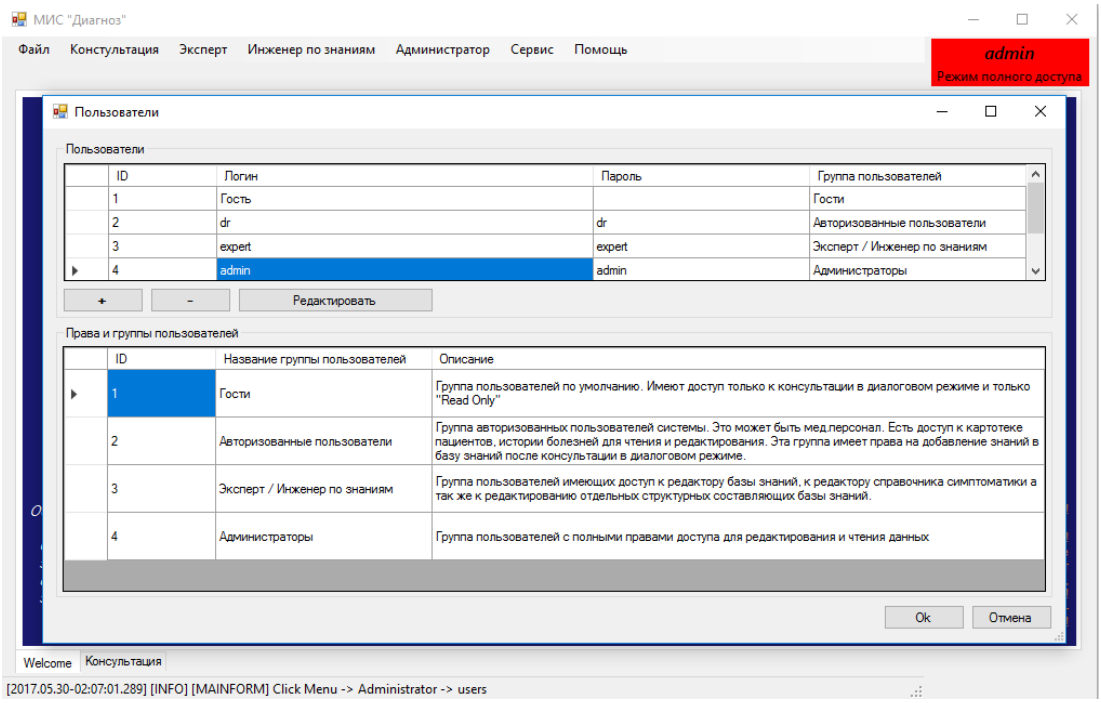

Рисунок 15 – Администрирование и управление пользователями

Различные редакторы для добавления ключевых данных представлены на рисунке 16.

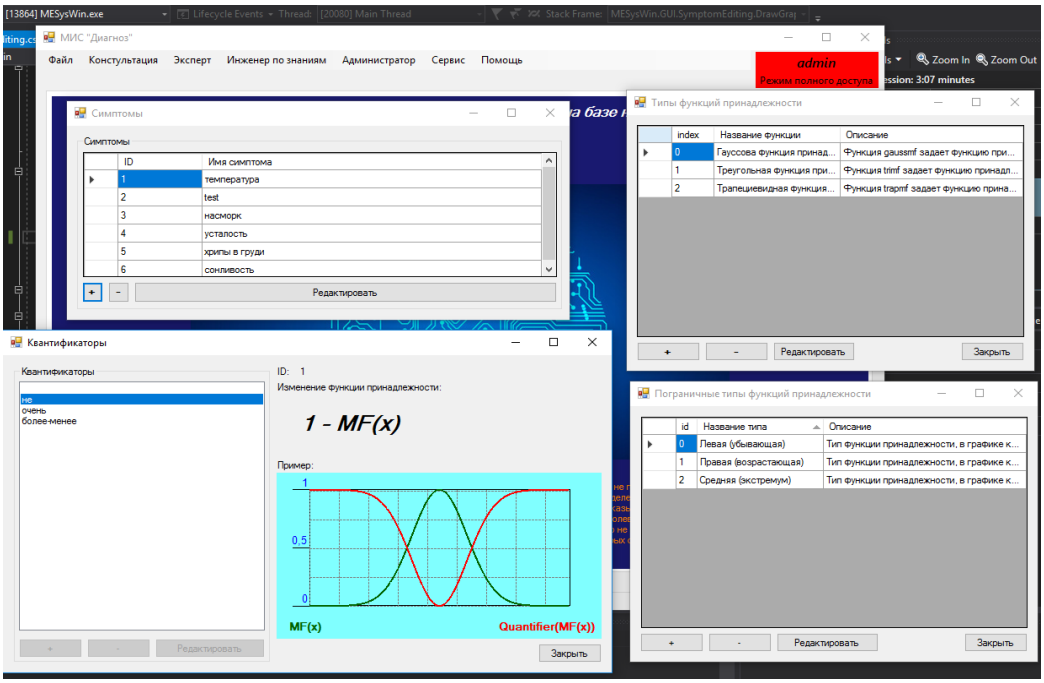

Рисунок 16 – Различные редакторы данных информационной системы

Редактор симптомов с отображением термов на графике представлен на

рисунке 17.

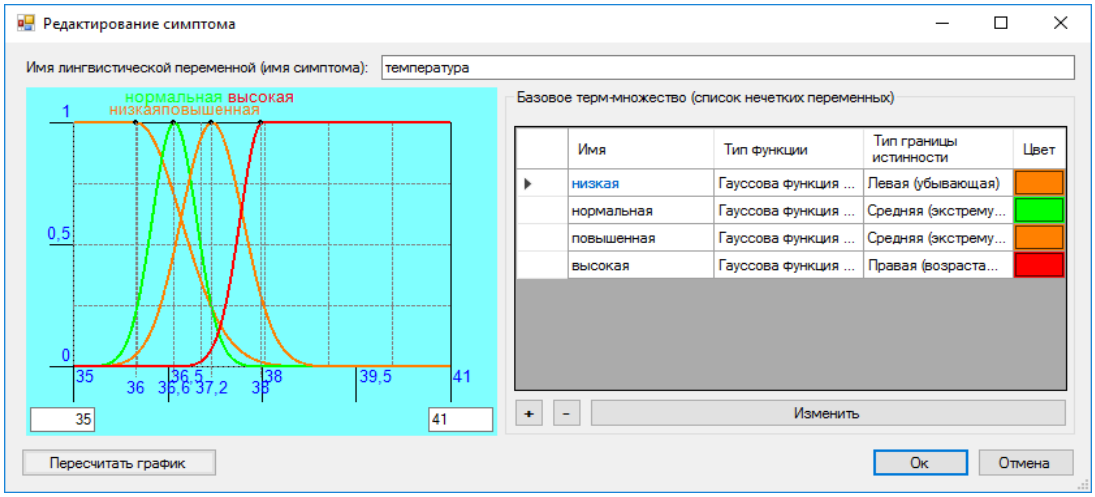

Рисунок 17 – Редактор лингвистических переменных нечеткой логики

ЗАКЛЮЧЕНИЕ

В процессе работы были собраны и изучены материалы согласно вопросам, связанным с проведением диагностики различных заболеваний, а также были изучены способы проектирования и реализации экспертных систем на базе нечеткой логики.

Среди методов организации экспертных систем были рассмотрены ключевые направления с позиции достоинств и недостатков, и наиболее оптимальной была подобрана структура с использованием нечетких множеств и правил на их основе. Благодаря тому, что на сегодняшний день структура логических рассуждений медицинских работников-диагностов аналогична структуре систем с нечеткой логикой, система дает возможность отвлечься от конкретных значений замеров, и тем самым исключить повторения правил. Это позволило существенно сократить объём базы знаний и повысить темп работы программного продукта.

После консультации с медиками-диагностами было также принято решение о внедрении в пользовательский интерфейс справочной системы. Это руководство ориентировано на упрощение принятия окончательного решения лично врачом-диагностом уже после получения списка предполагаемых диагнозов.

Также было реализовано объединение с интеллектуальными парадигмами для обучения экспертной системы способом машинного обучения с применением нейронных сетей Хопфилда.

Результатом работы является медицинская информационная система «Диагноз», которая дает возможность посодействовать медицинским работникам разных направлений и квалификаций в постановке диагноза пациентам.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Колесова, Ю.Д. Проектирование системы поддержки принятия решений при гинекологическом обследовании во время беременности / Ю.Д. Колесова, А.В. Ростова, Ю.В. Григорьева // Инновации в современном мире: сб. статей Международной научно-практической конференции, 20 февраля 2015 г. – М.: ООО «ЭФИР», 2015. – С. 35–37.

2 Безруков, Н.С. Автоматизированная система «Medical toolbox» для диагностики бронхиальной астмы по показателям реонцэфалографии / Н.С. Безруков, Е.Л. Ерёмин, Е.В. Ермакова, В.П. Колосов, Ю.М. Перельман //Медицинская информатика (2006) №1 (11).

3 Ерёмина, В.В. Разработка базы данных для поддержки принятия диагностических решений при гинекологическом обследовании во время беременности / В.В. Ерёмина, Ю.В. Григорьева, Ю.Д. Колесова, А.В. Ростова // Материалы VIII международной научной конференции «Системный анализ в медицине». – Благовещенск: ФГБУ «ДНЦ ФПД» СО РАМН, 2014. – С. 40–42.

4 Бутов, М.А. Применение нечеткой логики в видеоэндоскопических системах с расширенными функциональными возможностями / М.А. Бутов, В.Н. Локтюхин, О.А. Маслова, А.А. Черепнин // Вестник Рязанского ГРТУ. – 2008. – Вып. 23. – 6 с.

7 Грекул, В.И. Введение в реляционные базы данных / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина. – М. : ИнтУИТ, 2005. – 304 с.

8 Джарратано, Дж. Экспертные системы: принципы разработки и программирование / Джозеф Джарратано, Гари Райли. : Пер. с англ. – М.: Вильямс. – 2006. – 1152 с.

9 Чепак, Л. В. Методические указания к выполнению курсовой работы по дисциплине: «Базы данных»; Практикум / Л.В. Чепак, А.Г. Масловская. – Благовещенск : Изд–во Амур. гос. ун-та, 2010. – 58 с.

10 Баула, В.Г. Основы программирования и алгоритмические языки :

Учебное пособие / В.Г. Баула [и др.] – М. : Энергоатомиздат, 1991. – 400 с.

11 Бадриев, И.Б. Разработка графического пользовательского интерфейса в среде MATLAB : учебное пособие / И.Б. Бадриев, В.В. Бандеров, О.А. Задворнов. – Казань: Казанский государственный университет, 2010. – 113 с.

12 Леоненков, А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH / А.В. Леоненков. – СПб. : БХВ-Петербург, 2005. – 736 с.

13 Штовба, С.Д. Введение в теорию нечетких множеств и нечеткую логику / С.Д. Штовба. – Винница : УНИВЕРСУМ-Винница, 2001. – 756 с.

14 Осовский, С. Нейронные сети для обработки информации / С. Осовский. – М., 2002. – 344 с.

15 Еремина, В.В. Проектирование информационной системы медицинской диагностики на базе нечёткой логики / В.В. Еремина, Ю.А. Горожанина. Наука, образование и инновации: сборник статей Международной научно-практической конференции (28 декабря 2015 г., г. Челябинск). В 5 ч. – Уфа : РИО МЦИИ ОМЕГА САЙНС, 2015. – Ч. 5. – С. 143-144.

16 Чернуха, Е.А. Анатомически и клинически узкий таз / Е.А. Чернуха, А.И. Волобуев, Т.К. Пучко. – М. : Триада-Х, 2005. – 256 с.

17 Дейт, К. Введение в системы баз данных / К. Дейт – Киев : Диалектика, 2003. – 784 с.

18 Медицинская экспертная система дифференциальной диагностики [Электронный ресурс] – Электронные данные – Режим доступа: http://uber-tek.ru/Project/medexpert

19 Васильев, Н. Расчет эффективности внедрения ИС [Электронный ресурс] / Экономика и управление.Ру : офиц.сайт – 10.03.2009 . – Режим доступа : http://www.economica-upravlenie.ru/content/view/286/206/ – 15.05.2017.

20 Ерёмина, В.В. Проектирование экспертной системы диагностики на базе нечеткой логики // Ю.А. Горожанина, В.В. Ерёмина Современные научные исследования и инновации. 2017. № 6 [Электронный ресурс]. – Режим доступа : http://web.snauka.ru/issues/2017/06/83588. – 13.06.2017.

21 Гаврилова, И.В. Разработка приложений: учеб. пособие / И.В. Гаврило-

ва. – 2-е изд., стер. – М : ФЛИНТА, 2012. – 241с.

22 Рассел, Ст. Искусственный интеллект: современный подход / Стюарт Рассел, Питер Норвиг. – 2-е изд..: Пер. с англ. – М.: Вильямс, 2006. – 1408с.

23 Ерёмина, В.В. Проектирование информационной системы медицинской диагностики на базе нечеткой логики [Электронный ресурс]. / В.В. Ерёмина, Ю.В. Григорьева, Ю.Д. Колесова, А.В. Ростова // Электронное научное издание «Учёные заметки ТОГУ», 2015. – Т. 6., №1 – С. 310 – 313. – Режим доступа : http://pnu.edu.ru/media/ejournal/articles-2015/TGU_6_53.pdf. – 10.05.2017.

24 Колесова, Ю.Д. Создание экспертной системы диагностики и лечения гинекологических заболеваний у беременных женщин / Ю.Д. Колесова, А.В. Ростова // Сборник трудов участников международной научно-практической конференции «Медицинская наука: достижения и перспективы», 15 июля 2014 г. – М. : ООО «МИА-МЕД», 2014. – С. 191–194.

25 Колесова, Ю.Д. Модели, алгоритмы и средства для поддержки принятия диагностических решений при гинекологическом обследовании во время беременности на основе технологии нечеткой логики / Ю.Д. Колесова, А.В. Ростова // Молодежь XXI века: шаг в будущее: материалы XV региональной научно – практической конференции (22 мая 2014 г., Благовещенск): в 7 томах. – Благовещенск : типография АмГУ, 2014. – Т. 5. – С. 53–54.

26 Экспертные системы медицинской диагностики. Достоинства и опыт реализации, обоснование экономической эффективности [Электронный ресурс]. – Электронные данные. – Режим доступа: http://diagnos.ru.

27 Лешек, А.М. Анализ и проектирование информационных систем // А.М. Лешек, Мацашек. – М. : Вильямс, 2008. – 816 с.

28 Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем. / С.В. Маклаков. - М. : ДИАЛОГ-МИФИ, 2000. – 311 с.

29 Тарасян, В.С. Пакет Fuzzy Logic Toolbox for Matlab : учеб. пособие / В.С. Тарасян. – Екатеринбург : Изд-во УрГУПС, 2013. – 112 с.

30 Шураков, В.В. Надежность программного обеспечения систем обработки данных: Учебник. / В.В. Шураков. – 2-е изд., перераб. и доп. – М. : Фи-

нансы и статистика, 2008. – 272 с.

31 Fuzzy Logic Toolbox. For Use with MATLAB: User's Guide. – Natick: The MathWorks, Inc., 1998. – 235 с.

32 Кобринский Б.А., Зарубина Т.В. Медицинская информатика: Учебник. / Б.А. Кобринский. – М.: Изд. Центр «Академия», 2009. – 192с.

33 Джарратано Дж. Экспертные системы: принципы разработки и программирование. / Джозеф Джарратано, Гари Райли. : Пер. с англ. – М.: Вильямс, 2006. – 1152 с.

34 Андрейчиков А.В., Андрейчикова О.Н. Интеллектуальные информационные системы: М. Наука, 2004.

ПРИЛОЖЕНИЕ А

Функциональная модель ГАУЗ АО «АОКБ»

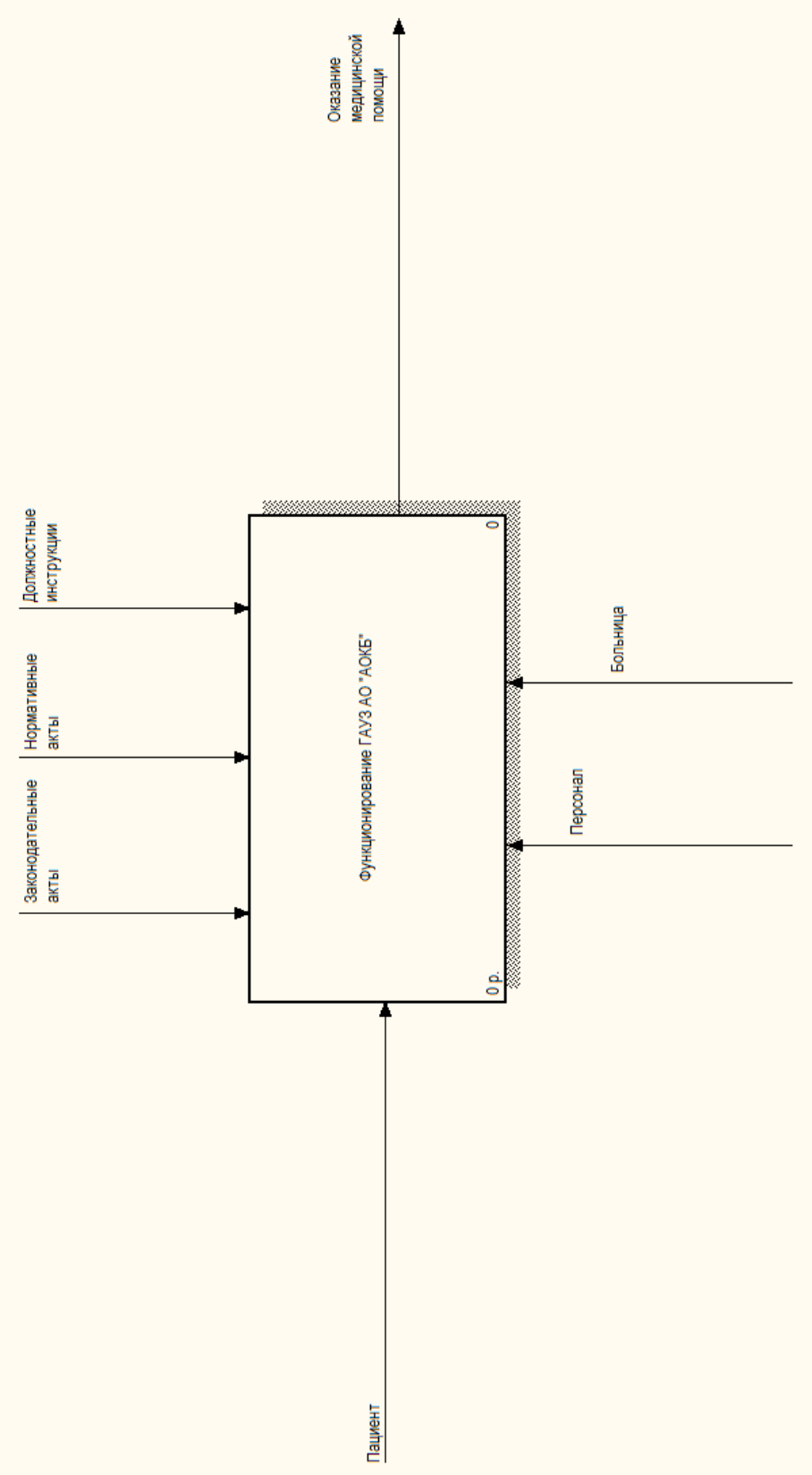

Рисунок А.1 – Функциональная модель организации

Продолжение ПРИЛОЖЕНИЯ А

Функциональная модель ГАУЗ АО «АОКБ»

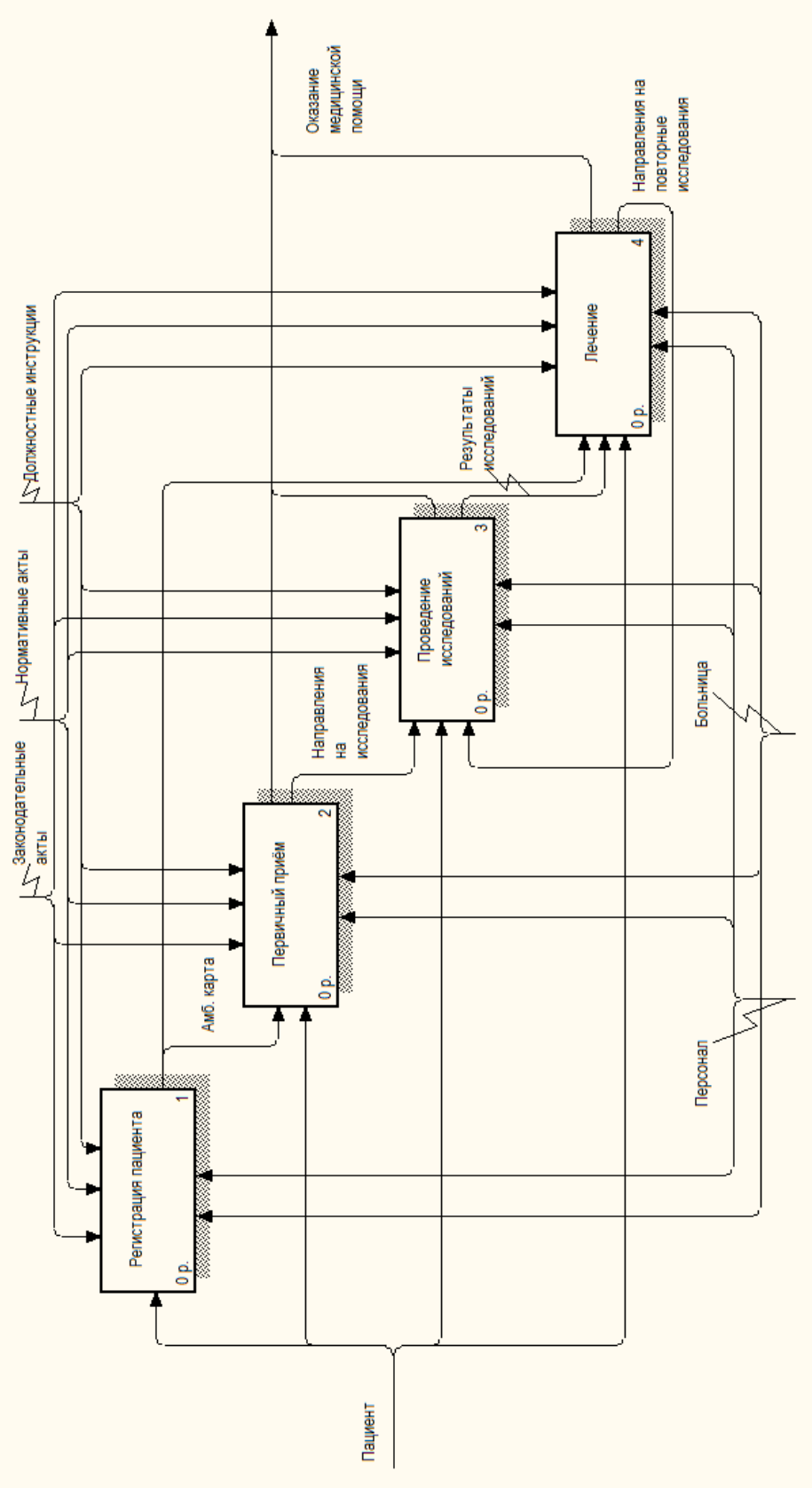

Рисунок А.2 – Декомпозиция функциональной модели организации

Продолжение ПРИЛОЖЕНИЯ А

Функциональная модель ГАУЗ АО «АОКБ»

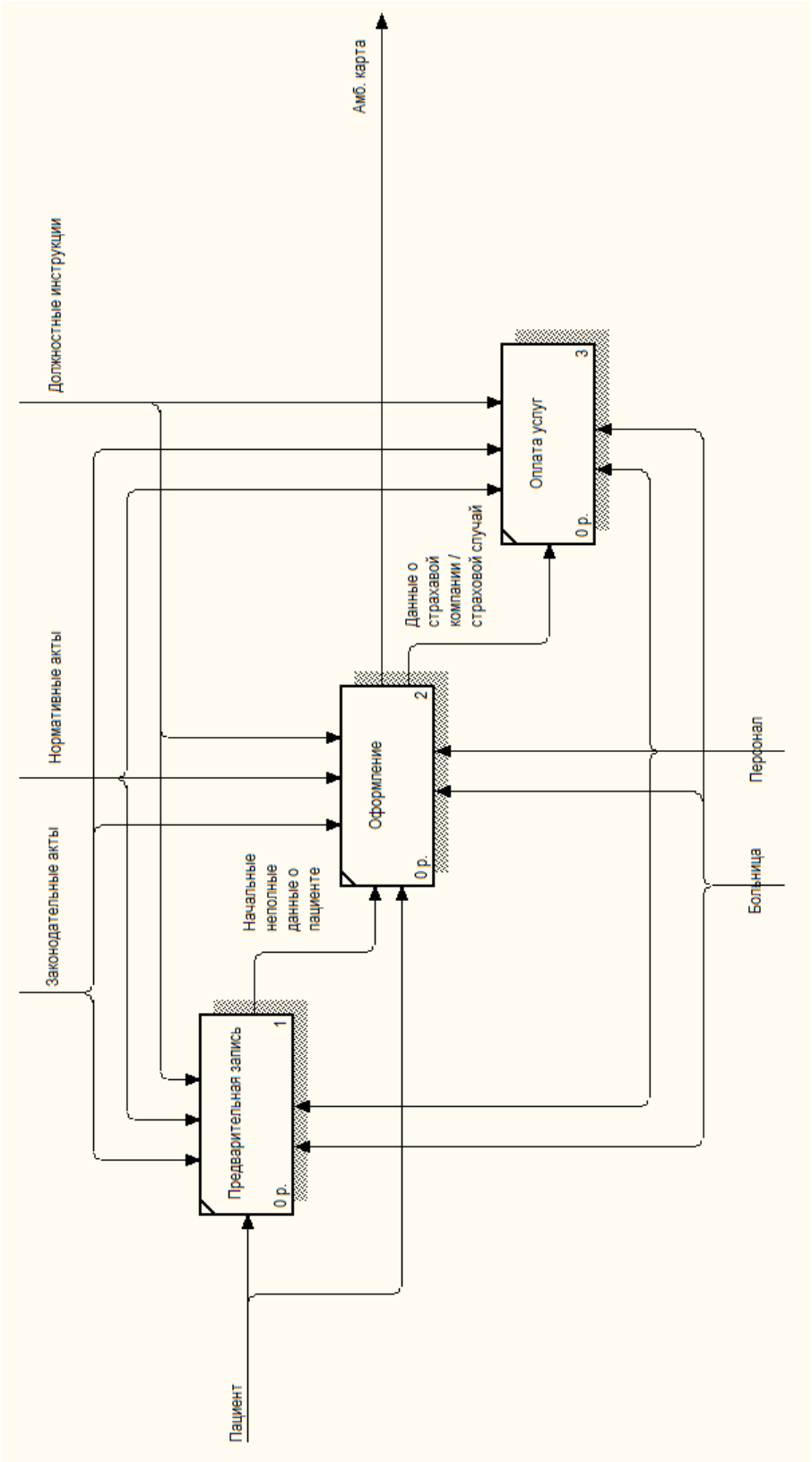

Рисунок А.3 – Декомпозиция «Регистрация пациентов»

Продолжение ПРИЛОЖЕНИЯ А
Функциональная модель ГАУЗ АО «АОКБ»

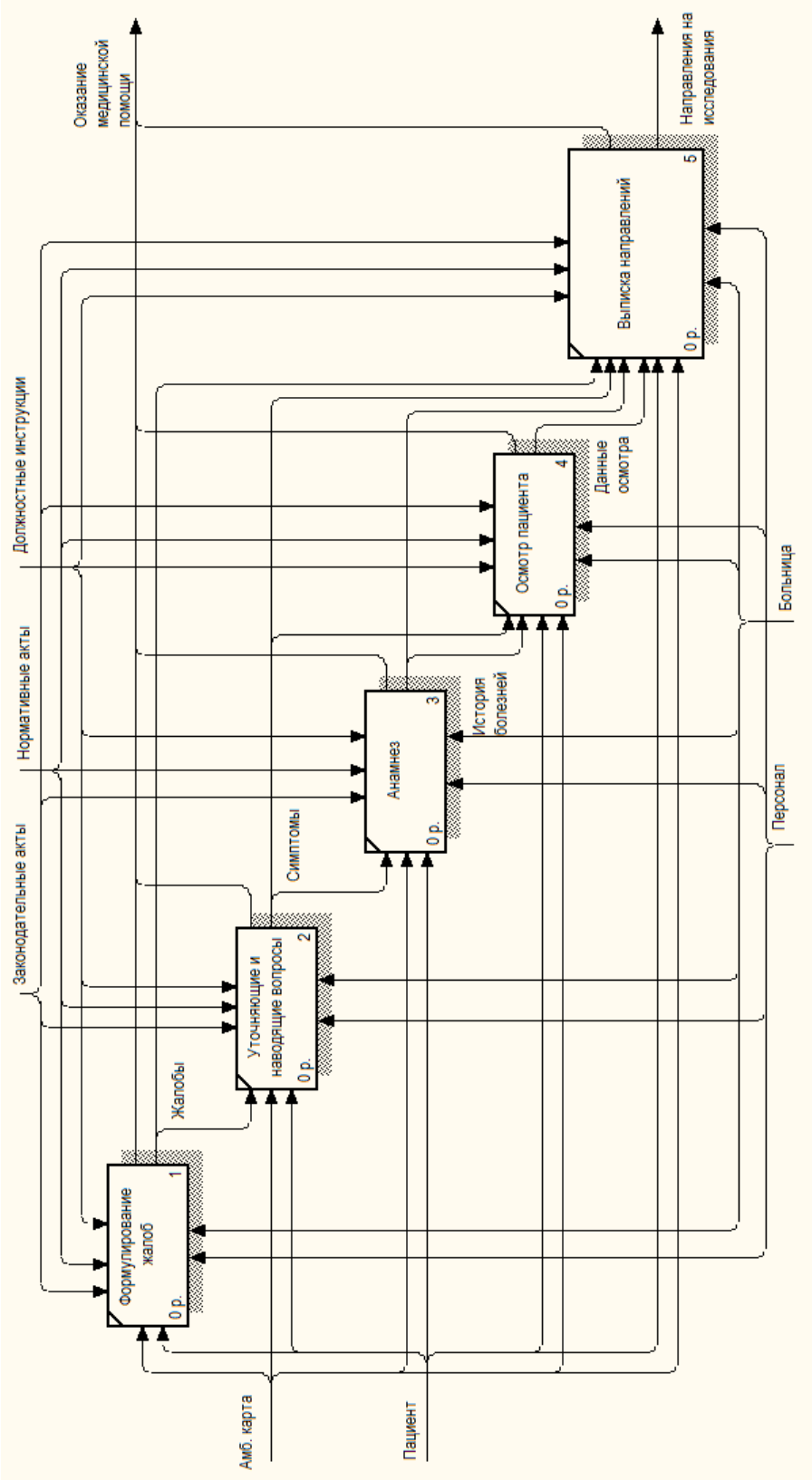

Рисунок А.4 – Декомпозиция «Первичный прием»

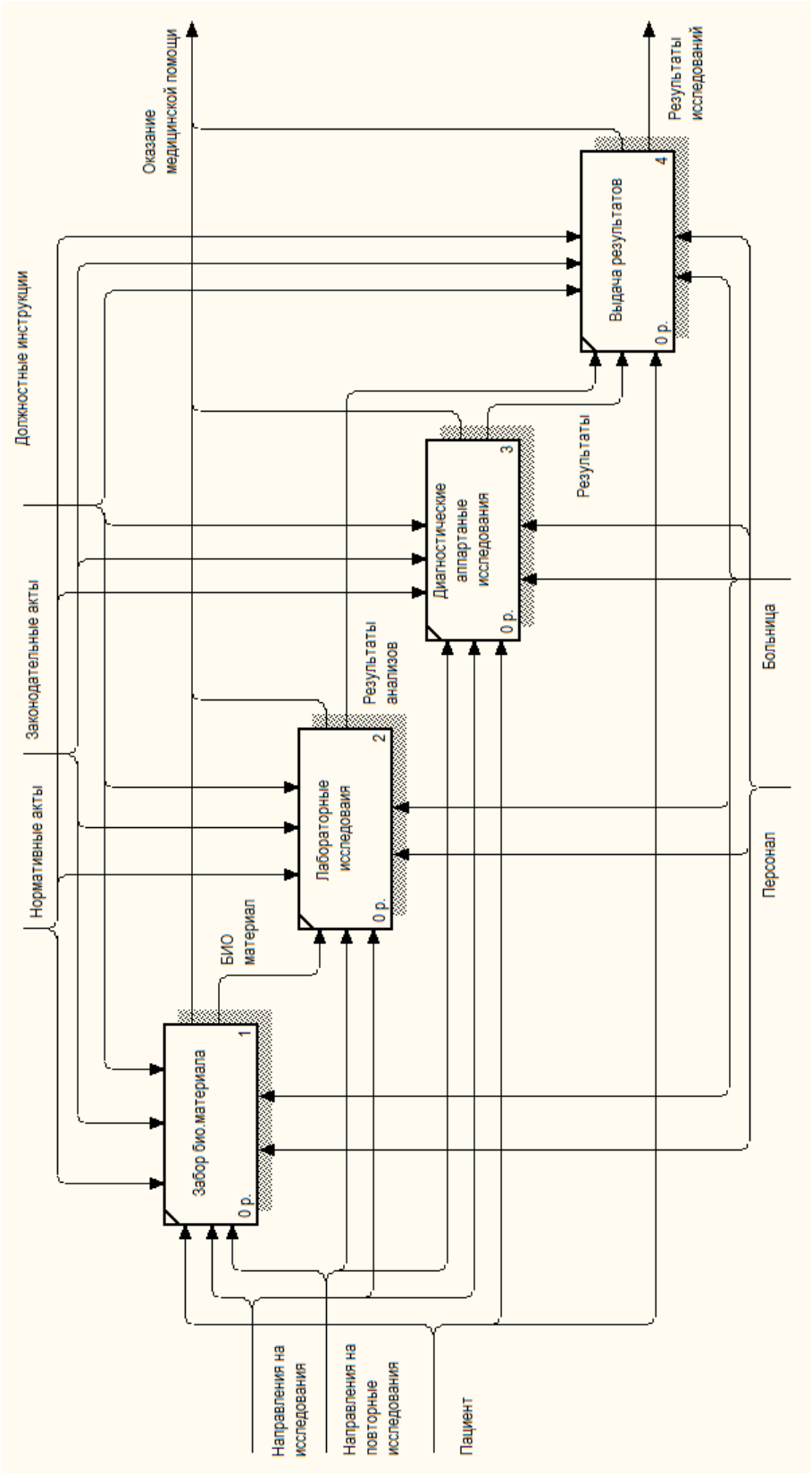

Рисунок А.5 – Декомпозиция «Проведение исследований»

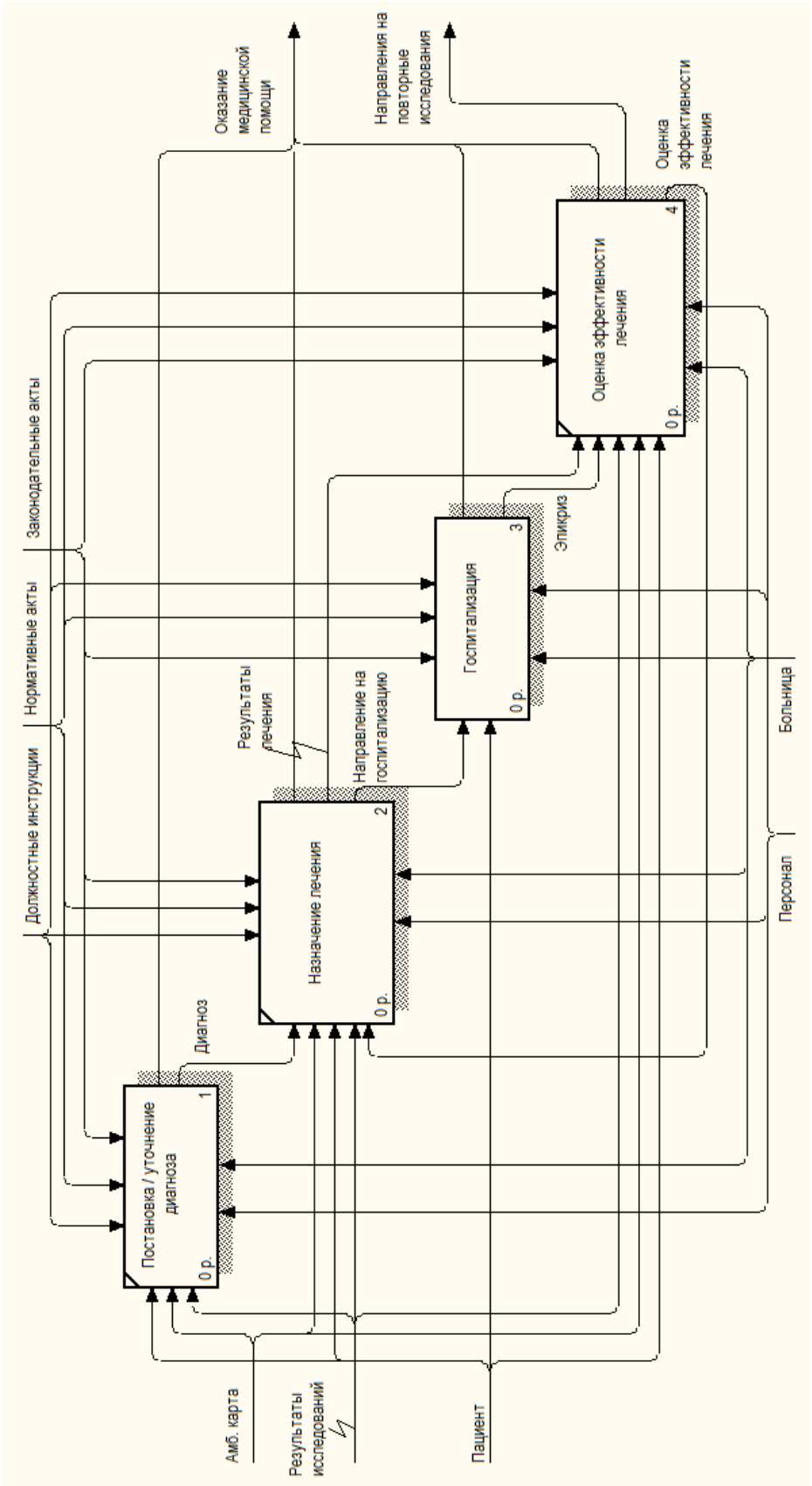

Рисунок А.6 – Декомпозиция «Лечение»

ПРИЛОЖЕНИЕ Б

Функциональная модель МИС «Диагноз»

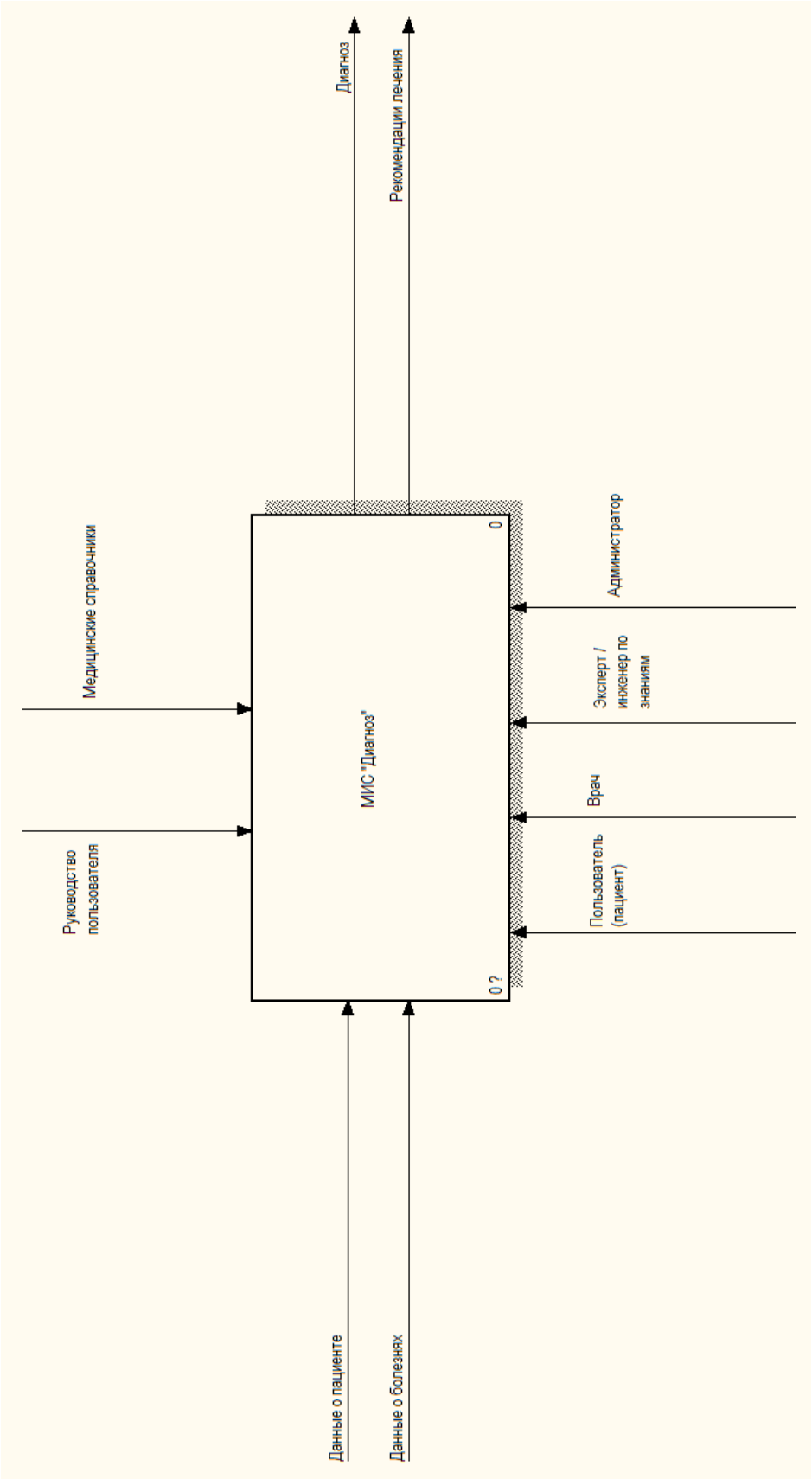

Рисунок Б.1 – Функциональная модель МИС «Диагноз»

Продолжение ПРИЛОЖЕНИЯ Б

Функциональная модель МИС «Диагноз»

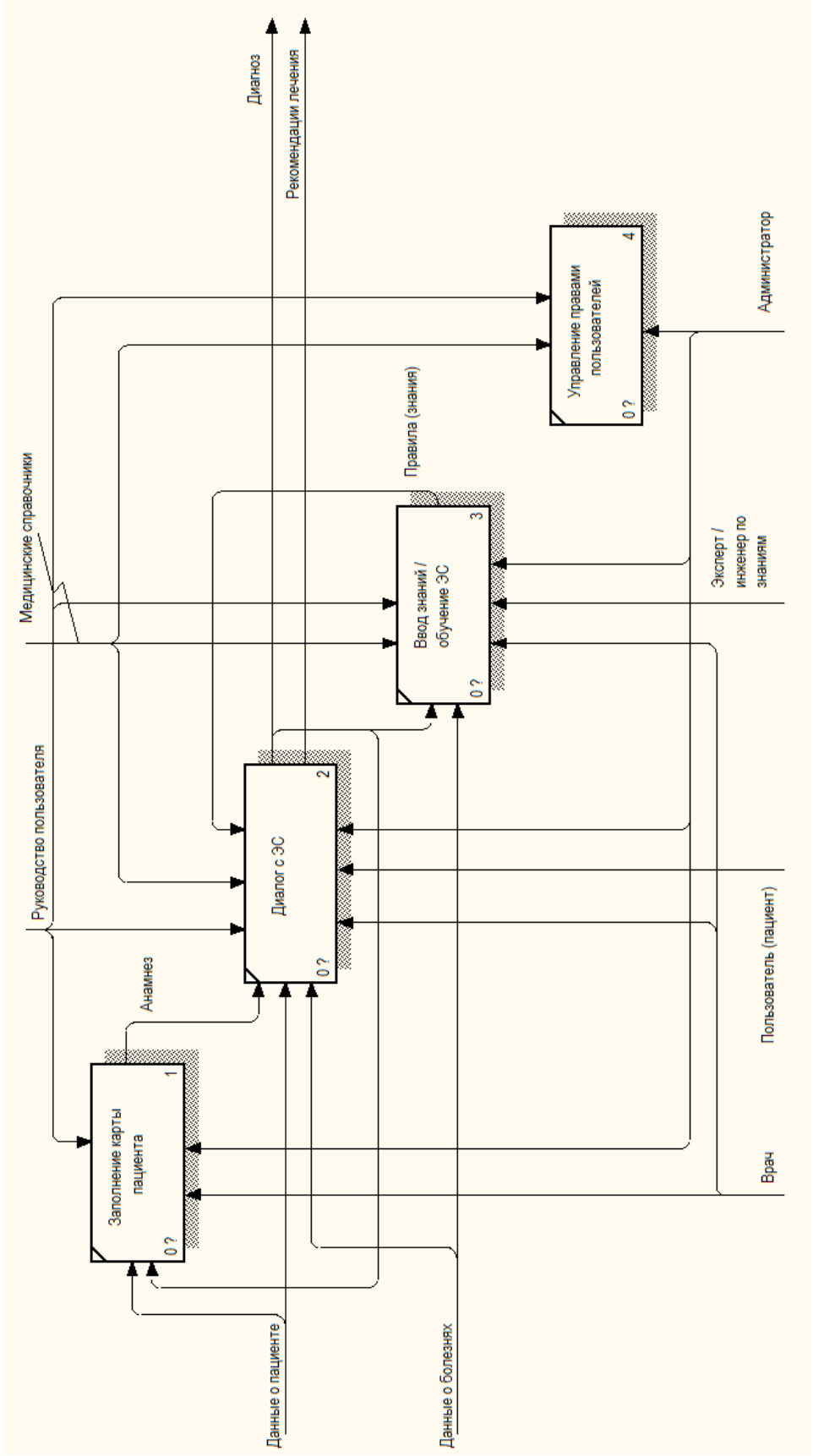

Рисунок Б.2 – Декомпозиция функциональной модели

ПРИЛОЖЕНИЕ В

Логическая модель БД

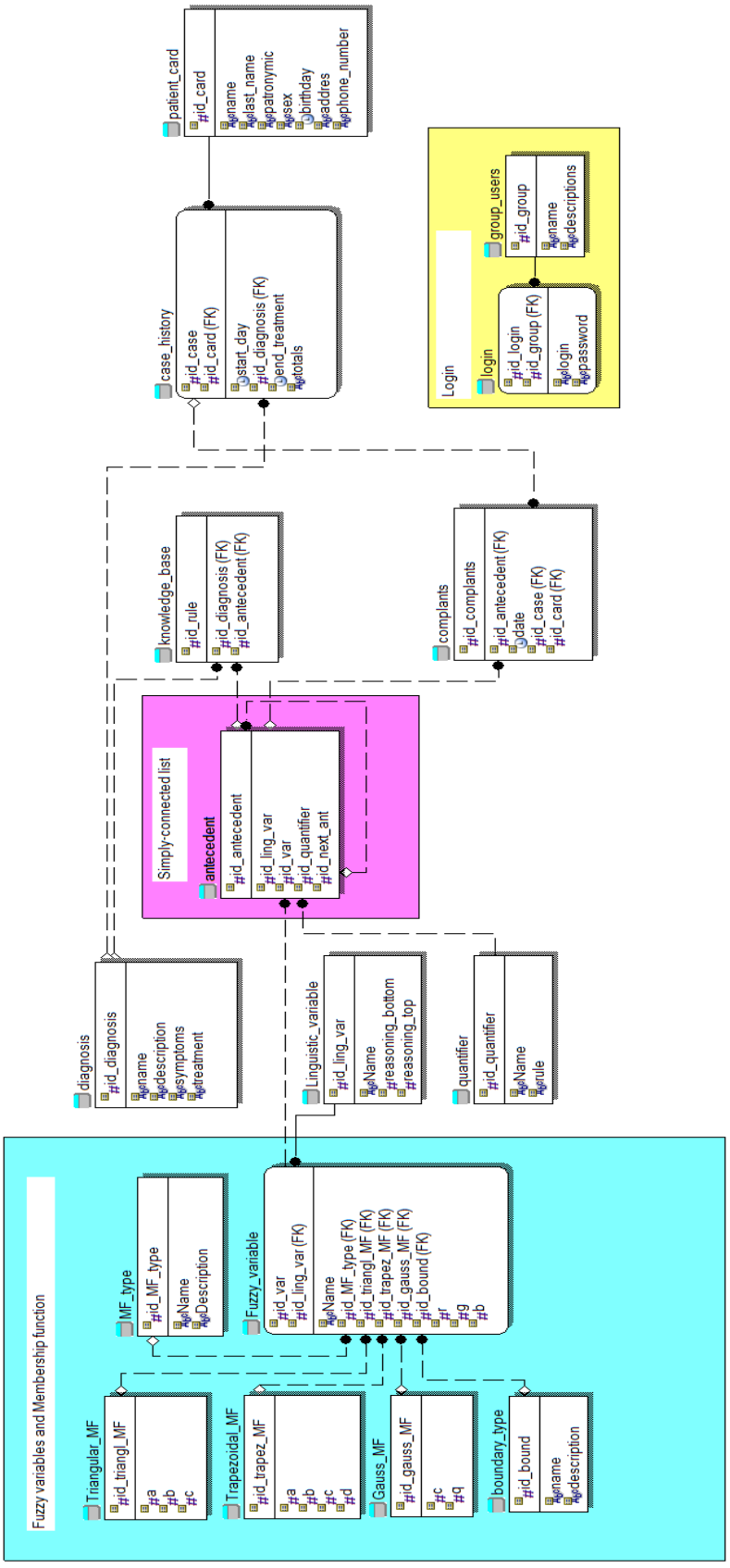

Рисунок В.1 – Логическая модель БД приложения

ПРИЛОЖЕНИЕ Г

Физическая модель БД

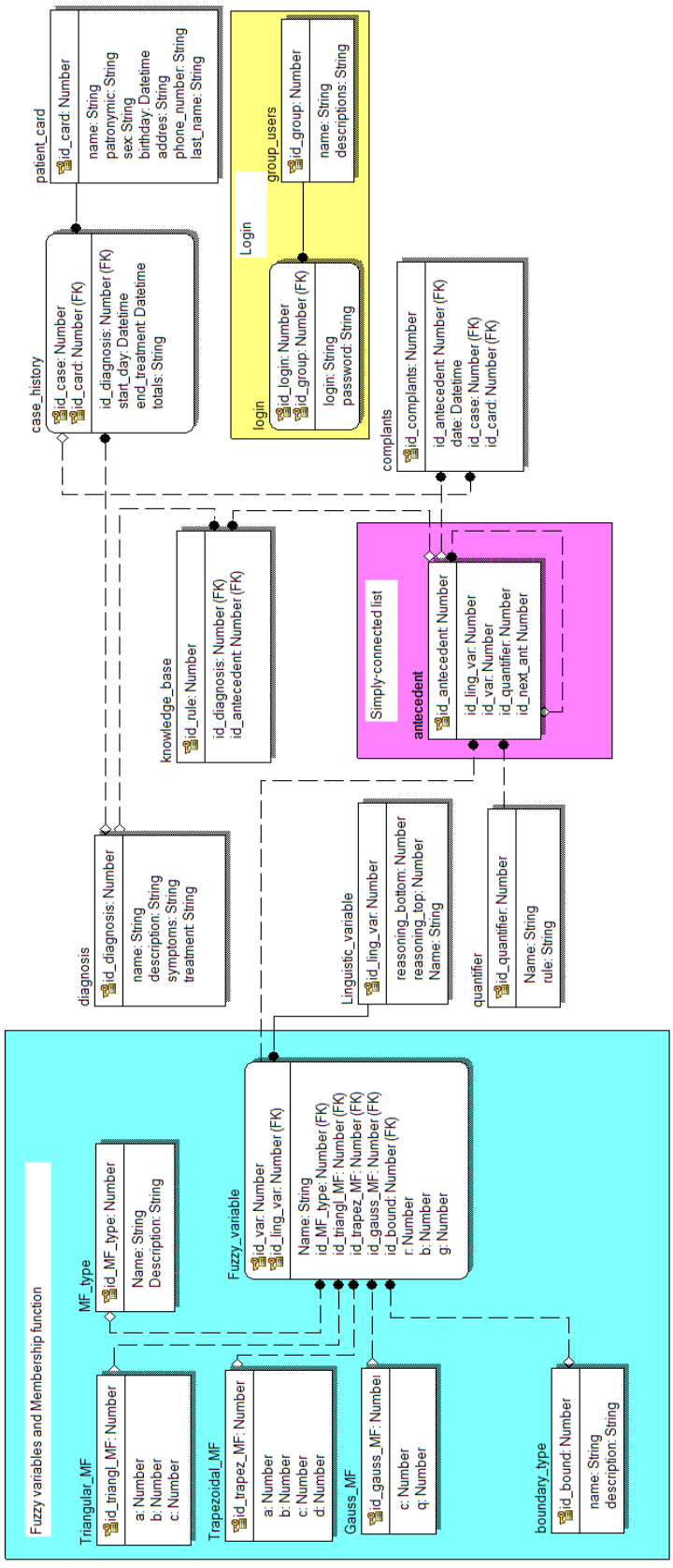

Рисунок Г.1 – Физическая модель БД информационной системы

ПРИЛОЖЕНИЕ Д

Листинг реализации БД на языке SQL

```
CREATE TABLE antecedent
(
            id_antecedent INTEGER PRIMARY KEY autoincrement,
            id_ling_var   INTEGER,
            id_var        INTEGER,
            id_quantifier INTEGER,
            id_next_ant   INTEGER,
            FOREIGN KEY (id_ling_var) REFERENCES linguistic_variable(id_ling_var),
            FOREIGN KEY (id_var) REFERENCES fuzzy_variable(id_var),
            FOREIGN KEY (id_quantifier) REFERENCES quantifier(id_quantifier)
)

CREATE TABLE boundary_type
(
              id_bound    INTEGER PRIMARY KEY,
              name        TEXT NOT NULL,
              description TEXT
)

CREATE TABLE diagnosis
(
            id_diagnosis INTEGER PRIMARY KEY autoincrement,
            name text,
            description text,
            symptoms text,
            treatment text
)

CREATE TABLE fuzzy_variable
(
            id_var      INTEGER PRIMARY KEY autoincrement,
            id_ling_var INTEGER NOT NULL,
            name text NOT NULL,
            id_mf_type    INTEGER,
            id_bound      INTEGER,
            id_triangl_mf INTEGER,
            id_trapez_mf  INTEGER,
            id_gauss_mf   INTEGER,
            r             INTEGER,
            g             INTEGER,
            b             INTEGER,
            FOREIGN KEY (id_ling_var) REFERENCES linguistic_variable(id_ling_var),
            FOREIGN KEY (id_mf_type) REFERENCES mf_type(id_mf_type),
            FOREIGN KEY (id_bound) REFERENCES boundary_type(id_bound),
            FOREIGN KEY (id_triangl_mf) REFERENCES triangular_mf(id_triangl_mf),
            FOREIGN KEY (id_trapez_mf) REFERENCES trapezoidal_mf(id_trapez_mf),
            FOREIGN KEY (id_gauss_mf) REFERENCES gauss_mf(id_gauss_mf)
)
```

Продолжение ПРИЛОЖЕНИЯ Д

Листинг реализации БД на языке SQL

```
CREATE TABLE gauss_mf
(
            id_gauss_mf INTEGER PRIMARY KEY autoincrement,
            c           REAL NOT NULL,
            q           REAL NOT NULL
)

CREATE TABLE knowledge_base
(
            id_rule       INTEGER PRIMARY KEY autoincrement,
            id_diagnosis  INTEGER,
            id_antecedent INTEGER,
            FOREIGN KEY (id_diagnosis) REFERENCES diagnosis(id_diagnosis),
            FOREIGN KEY (id_antecedent) REFERENCES antecedent(id_antecedent)
)

CREATE TABLE linguistic_variable
(
            id_ling_var INTEGER PRIMARY KEY autoincrement,
            name text NOT NULL,
            reasoning_bottom REAL NOT NULL,
            reasoning_top    REAL NOT NULL
)

CREATE TABLE login
(
            id_login INTEGER PRIMARY KEY autoincrement,
            id_group INTEGER,
            login text,
            password text,
            FOREIGN KEY(id_group) REFERENCES user_group(id_group)
)

CREATE TABLE mf_type
(
  id_mf_type  INTEGER PRIMARY KEY,
  name        TEXT NOT NULL,
  description TEXT
)

CREATE TABLE quantifier
(
            id_quantifier INTEGER PRIMARY KEY autoincrement,
            name text,
            rule text
)
```

Продолжение ПРИЛОЖЕНИЯ Д

Листинг реализации БД на языке SQL

```
CREATE TABLE trapezoidal_mf
(
            id_trapez_mf INTEGER PRIMARY KEY autoincrement,
            a            REAL NOT NULL,
            b            REAL NOT NULL,
            c            REAL NOT NULL,
            d            REAL NOT NULL
)

CREATE TABLE triangular_mf
(
            id_triangl_mf INTEGER PRIMARY KEY autoincrement,
            a             REAL NOT NULL,
            b             REAL NOT NULL,
            c             REAL NOT NULL
)

CREATE TABLE user_group
(
            id_group INTEGER PRIMARY KEY autoincrement,
            name text,
            description text
)
```

ПРИЛОЖЕНИЕ Е

Диаграмма классов в проекте приложения

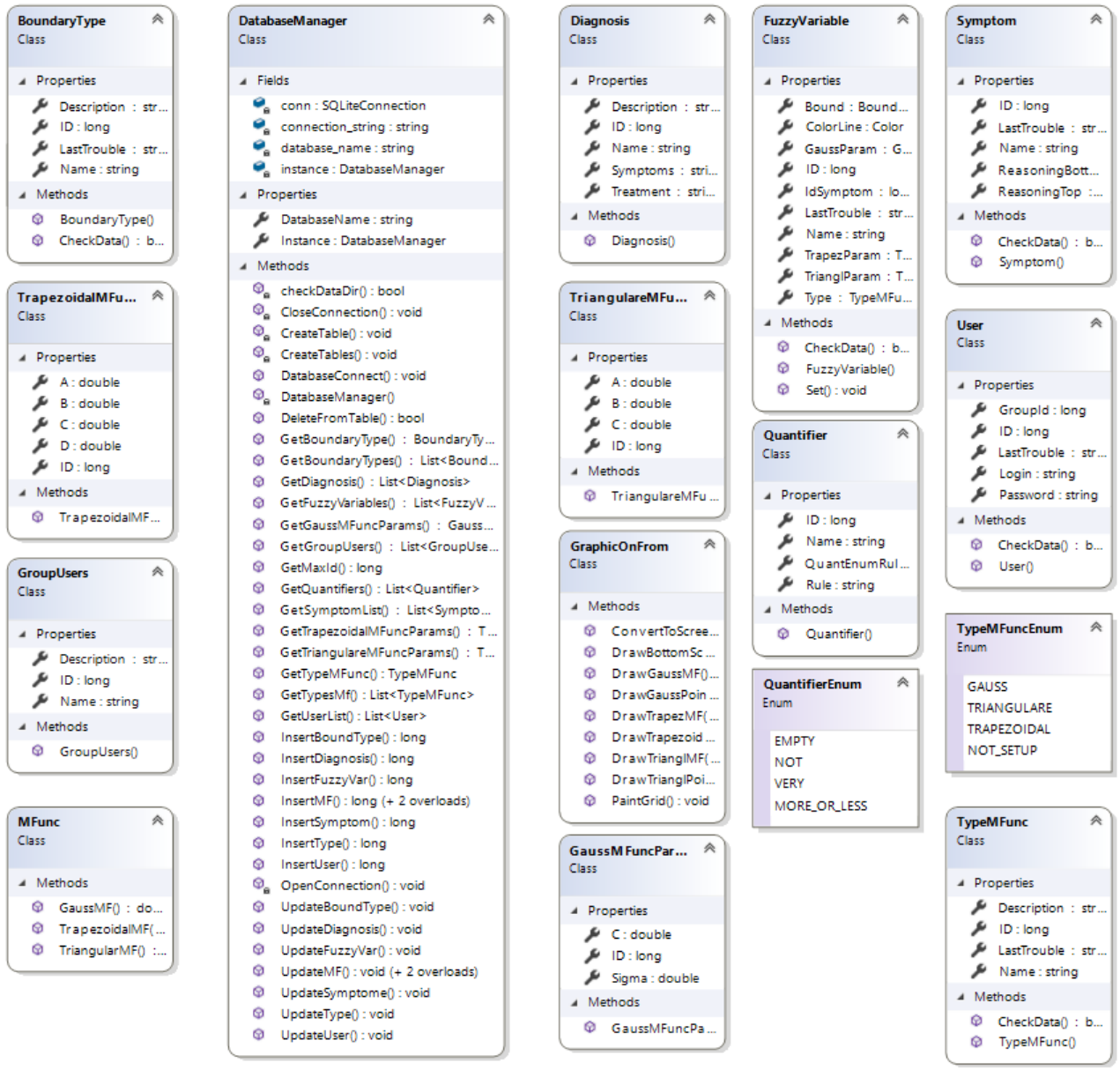

Рисунок Е.1 - Диаграмма классов в проекте приложения

ПРИЛОЖЕНИЕ Ж

Листинг класса DatabaseManager

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Data.SQLite;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MESysWin.src {
 class DatabaseManager {
  // Реализация паттерна Singletone
  private static DatabaseManager instance;

  private DatabaseManager() {
   // Проверим папку DATA
   checkDataDir();
   // Проверим файл базы данных
   if (!File.Exists(DatabaseName)) {
    // Если не существует, то сообщим, что создадим новую.
    Log.Print("Knowledge base and database do not exist. A new database file is created in \"" + DatabaseName +
"\" the path.", "DatabaseManager", Log.type.WARNING);
    MessageBox.Show("Knowledge base and database do not exist. A new database file is created in \"" + Data-
baseName + "\" the path.", "Database not exist", MessageBoxButtons.OK, MessageBoxIcon.Warning);
   }

   // Строка подключения к базе данных
   connection_string = "Data Source=" + DatabaseName + ";";

   //conn = new SQLiteConnection(connection_string);
  }

  public static DatabaseManager Instance {
   get {
    if (instance == null) {
     instance = new DatabaseManager();
     Log.Print("Create DatabaseManager instance singletone", "DatabaseManager", Log.type.INFO);
    }
    return instance;
   }
  }

  // Имя базы данных
  // KBofMD - Knowledge base of medical diagnosis
  private string database_name = "data\\KBofMD.db";
  public string DatabaseName {
   get {
    return database_name;
   }
  }

  // Проверка наличия папки DATA
  private bool checkDataDir() {
   if (!Directory.Exists("data")) {
    try {
     Directory.CreateDirectory("data");
     Log.Print("Create directory DATA", "DatabaseManager", Log.type.INFO);

     return true;
    } catch (Exception ex) {
     Log.Print(ex.Message, ex.Source, Log.type.ERROR);
     Console.WriteLine(ex.Message);
     MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
     return false;
    }
   } else {
    return true;
   }
  }
```

Продолжение ПРИЛОЖЕНИЯ Ж

Листинг класса DatabaseManager

```
private string connection_string;
private SQLiteConnection conn;

private void OpenConnection() {
 conn = new SQLiteConnection(connection_string);

 try {
  // Подключаемся
  conn.Open();
  //Log.Print("Open connection database", "DatabaseManager", Log.type.INFO);
 } catch (SQLiteException ex) {
  // Сообщаем об ошибке подключения
  Log.Print(ex.Message, ex.Source, Log.type.ERROR);
  Console.WriteLine(ex.Message);
  MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
}

private void CreateTables() {
 // Если подключение есть
 if (conn.State == ConnectionState.Open) {
  // Создадим таблицу Linguistic_variable, если она не существует
  CreateTable(conn, "linguistic_variable",
   "id_ling_var INTEGER PRIMARY KEY AUTOINCREMENT",
   "name TEXT NOT NULL",
   "reasoning_bottom REAL NOT NULL",
   "reasoning_top REAL NOT NULL");

  // Создадим таблицу MF_type, если она не существует
  CreateTable(conn, "mf_type",
   "id_mf_type INTEGER PRIMARY KEY",
   "name TEXT NOT NULL",
   "description TEXT");

  // Создадим таблицу boundary_type, если она не существует
  CreateTable(conn, "boundary_type",
   "id_bound INTEGER PRIMARY KEY",
   "name TEXT NOT NULL",
   "description TEXT");

  // Создадим таблицу Triangular_MF, если она не существует
  CreateTable(conn, "triangular_mf",
   "id_triangl_mf INTEGER PRIMARY KEY AUTOINCREMENT",
   //"id_mf_type INTEGER NOT NULL",
   "a REAL NOT NULL",
   "b REAL NOT NULL",
   "c REAL NOT NULL");

  // Создадим таблицу Trapezoidal_MF, если она не существует
  CreateTable(conn, "trapezoidal_mf",
   "id_trapez_mf INTEGER PRIMARY KEY AUTOINCREMENT",
   //"id_mf_type INTEGER NOT NULL",
   "a REAL NOT NULL",
   "b REAL NOT NULL",
   "c REAL NOT NULL",
   "d REAL NOT NULL");

  // Создадим таблицу Gauss_MF, если она не существует
  CreateTable(conn, "gauss_mf",
   "id_gauss_mf INTEGER PRIMARY KEY AUTOINCREMENT",
   //"id_mf_type INTEGER NOT NULL",
   "c REAL NOT NULL",
   "q REAL NOT NULL");

  // Создадим таблицу Fuzzy_variable, если она не существует
  CreateTable(conn, "fuzzy_variable",
   "id_var INTEGER PRIMARY KEY AUTOINCREMENT",
   "id_ling_var INTEGER NOT NULL",
   "name TEXT NOT NULL",
   "id_mf_type INTEGER",
   "id_bound INTEGER",
   "id_triangl_mf INTEGER",
   "id_trapez_mf INTEGER",
```

Листинг класса DatabaseManager

```
    "id_gauss_mf INTEGER",
    "r INTEGER",
    "g INTEGER",
    "b INTEGER",
    "FOREIGN KEY (id_ling_var) REFERENCES linguistic_variable(id_ling_var)",
    "FOREIGN KEY (id_mf_type) REFERENCES mf_type(id_mf_type)",
    "FOREIGN KEY (id_bound) REFERENCES boundary_type(id_bound)",
    "FOREIGN KEY (id_triangl_mf) REFERENCES triangular_mf(id_triangl_mf)",
    "FOREIGN KEY (id_trapez_mf) REFERENCES trapezoidal_mf(id_trapez_mf)",
    "FOREIGN KEY (id_gauss_mf) REFERENCES gauss_mf(id_gauss_mf)");

   // Создадим таблицу Quantifier, если она не создана
   CreateTable(conn, "quantifier",
    "id_quantifier INTEGER PRIMARY KEY AUTOINCREMENT",
    "name TEXT",
    "rule TEXT");

   // Создадим таблицу Diagnosis, если она не создана
   CreateTable(conn, "diagnosis",
    "id_diagnosis INTEGER PRIMARY KEY AUTOINCREMENT",
    "name TEXT",
    "description TEXT",
    "symptoms TEXT",
    "treatment TEXT");

   // Создадим таблицу Antecedent, если она не создана
   CreateTable(conn, "antecedent",
    "id_antecedent INTEGER PRIMARY KEY AUTOINCREMENT",
    "id_ling_var INTEGER",
    "id_var INTEGER",
    "id_quantifier INTEGER",
    "id_next_ant INTEGER",
    "FOREIGN KEY (id_ling_var) REFERENCES linguistic_variable(id_ling_var)",
    "FOREIGN KEY (id_var) REFERENCES fuzzy_variable(id_var)",
    "FOREIGN KEY (id_quantifier) REFERENCES quantifier(id_quantifier)");

   // Создадим таблицу knowledge_base, если она не создана
   CreateTable(conn, "knowledge_base",
    "id_rule INTEGER PRIMARY KEY AUTOINCREMENT",
    "id_diagnosis INTEGER",
    "id_antecedent INTEGER",
    "FOREIGN KEY (id_diagnosis) REFERENCES diagnosis(id_diagnosis)",
    "FOREIGN KEY (id_antecedent) REFERENCES antecedent(id_antecedent)");

   // Создадим таблицу user_grop, если она не создана
   CreateTable(conn, "user_group",
    "id_group INTEGER PRIMARY KEY AUTOINCREMENT",
    "name TEXT",
    "description TEXT");

   // Создаем таблицу login, если она не создана
   CreateTable(conn, "login",
    "id_login INTEGER PRIMARY KEY AUTOINCREMENT",
    "id_group INTEGER",
    "login TEXT",
    "password TEXT",
    "FOREIGN KEY(id_group) REFERENCES user_group(id_group)");
  }
 }

 private void CloseConnection() {
  try {
   // Закрываем подключение
   conn.Dispose();
   //conn.Close();
   conn = null;
  } catch (SQLiteException ex) {
   // Сообщаем об ошибке отключения
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
   MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
  }
```

```
}

// Первое подключение к базе данных
public void DatabaseConnect() {
 OpenConnection();
 CreateTables();
 CloseConnection();
}

private void CreateTable(SQLiteConnection SQLiteCon, string name_table, params string[] fields) {
 SQLiteCommand cmd = SQLiteCon.CreateCommand();

 string sql_command = "CREATE table IF NOT EXISTS " + name_table + " (";
 for (int i = 0; i < fields.Length; i++) {
  sql_command += fields[i];
  if ((i + 1) < fields.Length) {
   sql_command += ", ";
  } else {
   sql_command += ");";
  }
 }

 cmd.CommandText = sql_command;
 try {
  cmd.ExecuteNonQuery();
  //Log.Print("CREATE table IF NOT EXISTS " + name_table, "DatabaseManager", Log.type.INFO);
 } catch (SQLiteException ex) {
  Log.Print(ex.Message, ex.Source, Log.type.ERROR);
  Console.WriteLine(ex.Message);
  //MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
}

// Вернуть список всех симптомов и их ID
public List < Symptom > GetSymptomList() {
 List < Symptom > res = new List < Symptom > ();

 OpenConnection();

 SQLiteCommand cmd = conn.CreateCommand();
 cmd.CommandText = "SELECT id_ling_var, name, reasoning_bottom, reasoning_top " + "FROM linguistic_variable";

 try {
  SQLiteDataReader r = cmd.ExecuteReader();

  while (r.Read()) {
   res.Add(new Symptom(r.GetInt64(0), r.GetString(1), r.GetDouble(2), r.GetDouble(3)));
  }
  r.Close();
 } catch (SQLiteException ex) {
  Log.Print(ex.Message, ex.Source, Log.type.ERROR);
  Console.WriteLine(ex.Message);
 }

 CloseConnection();

 return res;
}

public List < TypeMFunc > GetTypesMf() {
 List < TypeMFunc > list = new List < TypeMFunc > ();

 OpenConnection();

 SQLiteCommand cmd = conn.CreateCommand();

 cmd.CommandText = "SELECT id_mf_type, name, description FROM mf_type";

 try {
  SQLiteDataReader r = cmd.ExecuteReader();

  while (r.Read()) {
```

Листинг класса DatabaseManager

```
            list.Add(new TypeMFunc(r.GetInt64(0), r.GetString(1), r.GetString(2)));
     }
     r.Close();
    } catch (SQLiteException ex) {
     Log.Print(ex.Message, ex.Source, Log.type.ERROR);
     Console.WriteLine(ex.Message);
    }

    CloseConnection();

    return list;
   }

   public List < FuzzyVariable > GetFuzzyVariables(long id_symptom) {
    List < FuzzyVariable > reslist = new List < FuzzyVariable > ();
    OpenConnection();

    SQLiteCommand cmd = conn.CreateCommand();

    cmd.CommandText = "SELECT " + "id_var, " + "id_ling_var, " + "name, " + "id_mf_type, " + "id_bound, " +
"id_triangl_mf, " + "id_trapez_mf, " + "id_gauss_mf, " + "r, g, b " + "FROM fuzzy_variable " + "WHERE
id_ling_var = @id_ling_var";

    cmd.Parameters.AddWithValue("@id_ling_var", id_symptom);

    try {
     SQLiteDataReader r = cmd.ExecuteReader();

     while (r.Read()) {
      System.Drawing.Color clr = new System.Drawing.Color();
      try {
       clr = System.Drawing.Color.FromArgb(r.GetInt32(8), r.GetInt32(9), r.GetInt32(10));
      } catch {
       clr = System.Drawing.Color.Yellow;
      }

      FuzzyVariable fv;
      try {
       fv = new FuzzyVariable(r.GetInt64(0), r.GetInt64(1), r.GetString(2), clr);
      } catch {
       fv = new FuzzyVariable(r.GetInt64(0), r.GetInt64(1), "", clr);
      }

      switch (r.GetInt64(3)) {
       case 0:
        fv.Type = TypeMFuncEnum.GAUSS;
        break;
       case 1:
        fv.Type = TypeMFuncEnum.TRIANGULARE;
        break;
       case 2:
        fv.Type = TypeMFuncEnum.TRAPEZOIDAL;
        break;
       default:
        fv.Type = TypeMFuncEnum.NOT_SETUP;
        break;
      }

      switch (r.GetInt64(4)) {
       case 0:
        fv.Bound = BoundaryTypeEnum.LEFT;
        break;
       case 1:
        fv.Bound = BoundaryTypeEnum.RIGHT;
        break;
       default:
        fv.Bound = BoundaryTypeEnum.MIDDLE;
        break;
      }

      fv.ColorLine = clr;
```

Листинг класса DatabaseManager

```
        try {
     fv.TrianglParam.ID = r.GetInt64(5);
    } catch {
     fv.TrianglParam.ID = -1;
    }
    try {
     fv.TrapezParam.ID = r.GetInt64(6);
    } catch {
     fv.TrapezParam.ID = -1;
    }
    try {
     fv.GaussParam.ID = r.GetInt64(7);
    } catch {
     fv.GaussParam.ID = -1;
    }

    reslist.Add(fv);
   }
   r.Close();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
  }

  CloseConnection();
  return reslist;
 }

 public List < BoundaryType > GetBoundaryTypes() {
  var list = new List < BoundaryType > ();

  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = "SELECT id_bound, name, description FROM boundary_type";

  try {
   SQLiteDataReader r = cmd.ExecuteReader();

   while (r.Read()) {
    list.Add(new BoundaryType(r.GetInt64(0), r.GetString(1), r.GetString(2)));
   }
   r.Close();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
  }

  CloseConnection();

  return list;
 }

 public bool DeleteFromTable(long id, string tableName, string columnName) {
  bool res = false;
  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = String.Format("DELETE FROM {0} WHERE {1} = '{2}'",
   tableName, columnName, id);

  try {
   cmd.ExecuteNonQuery();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
   MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
  }

  CloseConnection();
```

Листинг класса DatabaseManager

```
        return res;
    }

    public long GetMaxId(string table, string columnId) {
     long res = -1;

     OpenConnection();
     SQLiteCommand cmd = conn.CreateCommand();

     cmd.CommandText = String.Format("SELECT max({0}) FROM {1}", columnId, table);

     try {
      var obj = cmd.ExecuteScalar();
      res = Convert.ToInt32(obj);
     } catch (SQLiteException ex) {
      Log.Print(ex.Message, ex.Source, Log.type.ERROR);
      Console.WriteLine(ex.Message);
      MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
     } catch (System.InvalidCastException) {
      res = -1;
     }

     CloseConnection();
     return res;
    }

    public long InsertType(TypeMFunc type) {
     long inserted_id = -1;

     OpenConnection();

     SQLiteCommand cmd = conn.CreateCommand();

     cmd.CommandText = "INSERT INTO mf_type " + "(id_mf_type, name, description) " + "VALUES (@id_mf_type, @name,
@description)";

     cmd.Parameters.AddWithValue("@id_mf_type", type.ID);
     cmd.Parameters.AddWithValue("@name", type.Name);
     cmd.Parameters.AddWithValue("@description", type.Description);

     try {
      cmd.ExecuteNonQuery();
      type.ID = inserted_id = conn.LastInsertRowId;
     } catch (SQLiteException ex) {
      Log.Print(ex.Message, ex.Source, Log.type.ERROR);
      Console.WriteLine(ex.Message);
      MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
     }

     CloseConnection();
     return inserted_id;
    }

    public void UpdateType(TypeMFunc type) {
     OpenConnection();

     SQLiteCommand cmd = conn.CreateCommand();

     cmd.CommandText = "UPDATE mf_type " + "SET name = @name, description = @description " + "WHERE id_mf_type =
@id_mf_type";

     cmd.Parameters.AddWithValue("@id_mf_type", type.ID);
     cmd.Parameters.AddWithValue("@name", type.Name);
     cmd.Parameters.AddWithValue("@description", type.Description);

     try {
      cmd.ExecuteNonQuery();
     } catch (SQLiteException ex) {
      Log.Print(ex.Message, ex.Source, Log.type.ERROR);
      Console.WriteLine(ex.Message);
      MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
     }
```

Листинг класса DatabaseManager

```
   CloseConnection();
  }

  public long InsertSymptom(Symptom smp) {
   long inserted_id = -1;

   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "INSERT INTO linguistic_variable " + "(name, reasoning_bottom, reasoning_top) " + "VALUES
(@name, @reasoning_bottom, @reasoning_top)";

   cmd.Parameters.AddWithValue("@name", smp.Name);
   cmd.Parameters.AddWithValue("@reasoning_bottom", smp.ReasoningBottom);
   cmd.Parameters.AddWithValue("@reasoning_top", smp.ReasoningTop);

   try {
    cmd.ExecuteNonQuery();

    smp.ID = inserted_id = conn.LastInsertRowId;
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();

   return inserted_id;
  }

  public void UpdateSymptome(Symptom smp) {
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "UPDATE linguistic_variable " + "SET name = @name, " + "reasoning_bottom = @reasoning_bottom,
" + "reasoning_top = @reasoning_top " + "WHERE id_ling_var = @id_ling_var";

   cmd.Parameters.AddWithValue("@id_ling_var", smp.ID);
   cmd.Parameters.AddWithValue("@name", smp.Name);
   cmd.Parameters.AddWithValue("@reasoning_bottom", smp.ReasoningBottom);
   cmd.Parameters.AddWithValue("@reasoning_top", smp.ReasoningTop);

   try {
    cmd.ExecuteNonQuery();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();
  }

  //public long InsertMF(double c, double sigma)
  public long InsertMF(GaussMFuncParams param) {
   long inserted_id = -1;

   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "INSERT INTO gauss_mf " + "(c, q) " + "VALUES (@c, @q)";

   cmd.Parameters.AddWithValue("@c", param.C);
   cmd.Parameters.AddWithValue("@q", param.Sigma);

   try {
    cmd.ExecuteNonQuery();
```

Листинг класса DatabaseManager

```
   param.ID = inserted_id = conn.LastInsertRowId;
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
   MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
  }

  CloseConnection();

  return inserted_id;
 }

 public GaussMFuncParams GetGaussMFuncParams(long id) {
  GaussMFuncParams res = new GaussMFuncParams(0, 0);
  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = "SELECT id_gauss_mf, c, q " + "FROM gauss_mf " + "WHERE id_gauss_mf = @id_gauss_mf";

  cmd.Parameters.AddWithValue("@id_gauss_mf", id);

  try {
   SQLiteDataReader r = cmd.ExecuteReader();

   while (r.Read()) {
    res = new GaussMFuncParams( /*r.GetInt64(0),*/ r.GetDouble(1), r.GetDouble(2));
    res.ID = r.GetInt64(0);
   }
   r.Close();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
  }

  CloseConnection();
  return res;
 }

 public void UpdateMF(GaussMFuncParams param) {
  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = "UPDATE gauss_mf " + "SET c = @c, q = @q " + "WHERE id_gauss_mf = @id_gauss_mf";

  cmd.Parameters.AddWithValue("@id_gauss_mf", param.ID);
  cmd.Parameters.AddWithValue("@c", param.C);
  cmd.Parameters.AddWithValue("@q", param.Sigma);

  try {
   cmd.ExecuteNonQuery();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
   MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
  }

  CloseConnection();
 }

 //public long InsertMF(double a, double b, double c)
 public long InsertMF(TriangulareMFuncParams param) {
  long inserted_id = -1;

  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = "INSERT INTO triangular_mf " + "(a, b, c) " + "VALUES (@a, @b, @c)";

  cmd.Parameters.AddWithValue("@a", param.A);
```

Листинг класса DatabaseManager

```
   cmd.Parameters.AddWithValue("@b", param.B);
   cmd.Parameters.AddWithValue("@c", param.C);

   try {
    cmd.ExecuteNonQuery();

    param.ID = inserted_id = conn.LastInsertRowId;
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();

   return inserted_id;
  }

  public TriangulareMFuncParams GetTriangulareMFuncParams(long id) {
   TriangulareMFuncParams res = new TriangulareMFuncParams(0, 0, 0);
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "SELECT id_triangl_mf, a, b, c " + "FROM triangular_mf " + "WHERE id_triangl_mf =
@id_triangl_mf";

   cmd.Parameters.AddWithValue("@id_triangl_mf", id);

   try {
    SQLiteDataReader r = cmd.ExecuteReader();

    while (r.Read()) {
     res = new TriangulareMFuncParams( /*r.GetInt64(0),*/ r.GetDouble(1), r.GetDouble(2), r.GetDouble(3));
     res.ID = r.GetInt64(0);
    }
    r.Close();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
   }

   CloseConnection();
   return res;
  }

  public void UpdateMF(TriangulareMFuncParams param) {
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "UPDATE triangular_mf " + "SET a = @a, b = @b, c = @c " + "WHERE id_triangl_mf =
@id_triangl_mf";

   cmd.Parameters.AddWithValue("@id_triangl_mf", param.ID);
   cmd.Parameters.AddWithValue("@a", param.A);
   cmd.Parameters.AddWithValue("@b", param.B);
   cmd.Parameters.AddWithValue("@c", param.C);

   try {
    cmd.ExecuteNonQuery();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();
  }

  //public long InsertMF(double a, double b, double c, double d)
  public long InsertMF(TrapezoidalMFuncParams param) {
```

Листинг класса DatabaseManager

```
   long inserted_id = -1;

   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "INSERT INTO trapezoidal_mf " + "(a, b, c, d) " + "VALUES (@a, @b, @c, @d)";

   cmd.Parameters.AddWithValue("@a", param.A);
   cmd.Parameters.AddWithValue("@b", param.B);
   cmd.Parameters.AddWithValue("@c", param.C);
   cmd.Parameters.AddWithValue("@d", param.D);

   try {
    cmd.ExecuteNonQuery();

    param.ID = inserted_id = conn.LastInsertRowId;
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();

   return inserted_id;
  }

  public TrapezoidalMFuncParams GetTrapezoidalMFuncParams(long id) {
   TrapezoidalMFuncParams res = new TrapezoidalMFuncParams(0, 0, 0, 0);
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "SELECT id_trapez_mf, a, b, c, d " + "FROM trapezoidal_mf " + "WHERE id_trapez_mf =
@id_trapez_mf";

   cmd.Parameters.AddWithValue("@id_trapez_mf", id);

   try {
    SQLiteDataReader r = cmd.ExecuteReader();

    while (r.Read()) {
     res = new TrapezoidalMFuncParams( /*r.GetInt64(0),*/ r.GetDouble(1), r.GetDouble(2), r.GetDouble(3),
r.GetDouble(4));
     res.ID = r.GetInt64(0);
    }
    r.Close();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
   }

   CloseConnection();
   return res;
  }

  public void UpdateMF(TrapezoidalMFuncParams param) {
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "UPDATE trapezoidal_mf " + "SET a = @a, b = @b, c = @c, d = @d " + "WHERE id_trapez_mf =
@id_trapez_mf";

   cmd.Parameters.AddWithValue("@id_trapez_mf", param.ID);
   cmd.Parameters.AddWithValue("@a", param.A);
   cmd.Parameters.AddWithValue("@b", param.B);
   cmd.Parameters.AddWithValue("@c", param.C);
   cmd.Parameters.AddWithValue("@c", param.D);
```

Листинг класса DatabaseManager

```
   try {
    cmd.ExecuteNonQuery();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();
  }

  public long InsertFuzzyVar(FuzzyVariable fv) {
   long inserted_id = -1;

   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "INSERT INTO fuzzy_variable " + "(id_ling_var, name, id_mf_type, id_bound, id_triangl_mf,
id_trapez_mf, " + "id_gauss_mf, r, g, b) " + "VALUES (@id_ling_var, @name, @id_mf_type, @id_bound, @id_triangl_mf,
@id_trapez_mf, " + "@id_gauss_mf, @r, @g, @b)";

   cmd.Parameters.AddWithValue("@id_ling_var", fv.IdSymptom);
   cmd.Parameters.AddWithValue("@name", fv.Name);
   cmd.Parameters.AddWithValue("@id_mf_type", (int) fv.Type);
   cmd.Parameters.AddWithValue("@id_bound", (int) fv.Bound);
   cmd.Parameters.AddWithValue("@id_triangl_mf", fv.TrianglParam.ID);
   cmd.Parameters.AddWithValue("@id_trapez_mf", fv.TrapezParam.ID);
   cmd.Parameters.AddWithValue("@id_gauss_mf", fv.GaussParam.ID);
   cmd.Parameters.AddWithValue("@r", fv.ColorLine.R);
   cmd.Parameters.AddWithValue("@g", fv.ColorLine.G);
   cmd.Parameters.AddWithValue("@b", fv.ColorLine.B);

   try {
    cmd.ExecuteNonQuery();

    fv.ID = inserted_id = conn.LastInsertRowId;
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();

   return inserted_id;
  }

  public void UpdateFuzzyVar(FuzzyVariable fv) {
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "UPDATE fuzzy_variable " + "SET id_ling_var = @id_ling_var, name = @name, id_mf_type =
@id_mf_type, id_bound = @id_bound, id_triangl_mf = @id_triangl_mf, " + "id_trapez_mf = @id_trapez_mf, id_gauss_mf
= @id_gauss_mf, r = @r, g = @g, b = @b " + "WHERE id_var = @id_var";

   cmd.Parameters.AddWithValue("@id_var", fv.ID);
   cmd.Parameters.AddWithValue("@id_ling_var", fv.IdSymptom);
   cmd.Parameters.AddWithValue("@name", fv.Name);
   cmd.Parameters.AddWithValue("@id_mf_type", (int) fv.Type);
   cmd.Parameters.AddWithValue("@id_bound", (int) fv.Bound);
   cmd.Parameters.AddWithValue("@id_triangl_mf", fv.TrianglParam.ID);
   cmd.Parameters.AddWithValue("@id_trapez_mf", fv.TrapezParam.ID);
   cmd.Parameters.AddWithValue("@id_gauss_mf", fv.GaussParam.ID);
   cmd.Parameters.AddWithValue("@r", fv.ColorLine.R);
   cmd.Parameters.AddWithValue("@g", fv.ColorLine.G);
   cmd.Parameters.AddWithValue("@b", fv.ColorLine.B);

   try {
    cmd.ExecuteNonQuery();
   } catch (SQLiteException ex) {
```

Листинг класса DatabaseManager

```
            Log.Print(ex.Message, ex.Source, Log.type.ERROR);
     Console.WriteLine(ex.Message);
     MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    CloseConnection();
   }

   public long InsertBoundType(BoundaryType bound) {
    long inserted_id = -1;

    OpenConnection();

    SQLiteCommand cmd = conn.CreateCommand();

    cmd.CommandText = "INSERT INTO boundary_type " + "(id_bound, name, description) " + "VALUES (@id_bound, @name,
@description)";

    cmd.Parameters.AddWithValue("@id_bound", bound.ID);
    cmd.Parameters.AddWithValue("@name", bound.Name);
    cmd.Parameters.AddWithValue("@description", bound.Description);

    try {
     cmd.ExecuteNonQuery();
     bound.ID = inserted_id = conn.LastInsertRowId;
    } catch (SQLiteException ex) {
     Log.Print(ex.Message, ex.Source, Log.type.ERROR);
     Console.WriteLine(ex.Message);
     MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    CloseConnection();

    return inserted_id;
   }

   public void UpdateBoundType(BoundaryType bound) {
    OpenConnection();

    SQLiteCommand cmd = conn.CreateCommand();

    cmd.CommandText = "UPDATE boundary_type " + "SET name = @name, description = @description " + "WHERE id_bound =
@id_bound";

    cmd.Parameters.AddWithValue("@id_bound", bound.ID);
    cmd.Parameters.AddWithValue("@name", bound.Name);
    cmd.Parameters.AddWithValue("@description", bound.Description);

    try {
     cmd.ExecuteNonQuery();
    } catch (SQLiteException ex) {
     Log.Print(ex.Message, ex.Source, Log.type.ERROR);
     Console.WriteLine(ex.Message);
     MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    CloseConnection();
   }

   public TypeMFunc GetTypeMFunc(long id) {
    TypeMFunc res = new TypeMFunc(-1, "", "");

    OpenConnection();

    SQLiteCommand cmd = conn.CreateCommand();

    cmd.CommandText = "SELECT id_mf_type, name, description FROM mf_type " + "WHERE id_mf_type = @id_mf_type";

    cmd.Parameters.AddWithValue("@id_mf_type", id);

    try {
     SQLiteDataReader r = cmd.ExecuteReader();
```

Листинг класса DatabaseManager

```
   while (r.Read()) {
    res = new TypeMFunc(r.GetInt64(0), r.GetString(1), r.GetString(2));
   }
   r.Close();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
  }

  CloseConnection();

  return res;
 }

 public BoundaryType GetBoundaryType(long id) {
  BoundaryType res = new BoundaryType(-1, "", "");
  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = "SELECT id_bound, name, description FROM boundary_type " + "WHERE id_bound = @id_bound";

  cmd.Parameters.AddWithValue("@id_bound", id);

  try {
   SQLiteDataReader r = cmd.ExecuteReader();

   while (r.Read()) {
    res = new BoundaryType(r.GetInt64(0), r.GetString(1), r.GetString(2));
   }
   r.Close();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
  }

  CloseConnection();
  return res;
 }

 public List < Quantifier > GetQuantifiers() {
  var resList = new List < Quantifier > ();

  OpenConnection();

  SQLiteCommand cmd = conn.CreateCommand();

  cmd.CommandText = "SELECT id_quantifier, name, rule FROM quantifier";

  try {
   SQLiteDataReader r = cmd.ExecuteReader();

   while (r.Read()) {
    var curr = new Quantifier(r.GetString(1), (QuantifierEnum) r.GetInt64(0));
    curr.ID = r.GetInt64(0);
    curr.Rule = r.GetString(2);

    resList.Add(curr);
   }
   r.Close();
  } catch (SQLiteException ex) {
   Log.Print(ex.Message, ex.Source, Log.type.ERROR);
   Console.WriteLine(ex.Message);
  }

  CloseConnection();

  return resList;
 }

 public List < GroupUsers > GetGroupUsers() {
```

Листинг класса DatabaseManager

```
        List < GroupUsers > list = new List < GroupUsers > ();

 OpenConnection();

 SQLiteCommand cmd = conn.CreateCommand();

 cmd.CommandText = "SELECT id_group, name, description FROM user_group";

 try {
  SQLiteDataReader r = cmd.ExecuteReader();

  while (r.Read()) {
   var curr = new GroupUsers();
   curr.ID = r.GetInt64(0);
   curr.Name = r.GetString(1);
   curr.Description = r.GetString(2);

   list.Add(curr);
  }
  r.Close();
 } catch (SQLiteException ex) {
  Log.Print(ex.Message, ex.Source, Log.type.ERROR);
  Console.WriteLine(ex.Message);
 }

 CloseConnection();

 return list;
}

public List < User > GetUserList() {
 List < User > list = new List < User > ();

 OpenConnection();

 SQLiteCommand cmd = conn.CreateCommand();

 cmd.CommandText = "SELECT id_login, id_group, login, password FROM login";

 try {
  SQLiteDataReader r = cmd.ExecuteReader();

  while (r.Read()) {
   var curr = new User();
   curr.ID = r.GetInt64(0);
   curr.GroupId = r.GetInt64(1);
   curr.Login = r.GetString(2);
   try {
    curr.Password = r.GetString(3);
   } catch {
    curr.Password = String.Empty;
   }


   list.Add(curr);
  }
  r.Close();
 } catch (SQLiteException ex) {
  Log.Print(ex.Message, ex.Source, Log.type.ERROR);
  Console.WriteLine(ex.Message);
 }

 CloseConnection();

 return list;
}

public long InsertUser(User usr) {
 long inserted_id = -1;

 OpenConnection();
```

Листинг класса DatabaseManager

```
   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "INSERT INTO login " + "(id_group, login, password) " + "VALUES (@id_group, @login,
@password)";

   cmd.Parameters.AddWithValue("@id_group", usr.GroupId);
   cmd.Parameters.AddWithValue("@login", usr.Login);
   cmd.Parameters.AddWithValue("@password", usr.Password);

   try {
    cmd.ExecuteNonQuery();
    usr.ID = inserted_id = conn.LastInsertRowId;
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();

   return inserted_id;
  }

  public void UpdateUser(User usr) {
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "UPDATE login " + "SET id_group = @id_group, login = @login, password = @password " + "WHERE
id_login = @id_login";

   cmd.Parameters.AddWithValue("@id_login", usr.ID);
   cmd.Parameters.AddWithValue("@id_group", usr.GroupId);
   cmd.Parameters.AddWithValue("@login", usr.Login);
   cmd.Parameters.AddWithValue("@password", usr.Password);

   try {
    cmd.ExecuteNonQuery();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();
  }

  public List < Diagnosis > GetDiagnosis() {
   var list = new List < Diagnosis > ();

   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "SELECT id_diagnosis, name, description, symptoms, treatment FROM diagnosis";

   try {
    SQLiteDataReader r = cmd.ExecuteReader();

    while (r.Read()) {
     var curr = new Diagnosis(r.GetString(1));
     curr.ID = r.GetInt64(0);
     //curr.Name = r.GetString(1);
     curr.Description = r.GetString(2);
     curr.Symptoms = r.GetString(3);
     curr.Treatment = r.GetString(4);

     list.Add(curr);
    }
    r.Close();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
```

Листинг класса DatabaseManager

```
    Console.WriteLine(ex.Message);
   }

   CloseConnection();

   return list;
  }

  public long InsertDiagnosis(Diagnosis diagnosis) {
   long res = -1;

   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "INSERT INTO diagnosis " + "(name, description, symptoms, treatment) " + "VALUES (@name,
@description, @symptoms, @treatment)";

   cmd.Parameters.AddWithValue("@name", diagnosis.Name);
   cmd.Parameters.AddWithValue("@description", diagnosis.Description);
   cmd.Parameters.AddWithValue("@symptoms", diagnosis.Symptoms);
   cmd.Parameters.AddWithValue("@treatment", diagnosis.Treatment);

   try {
    cmd.ExecuteNonQuery();
    diagnosis.ID = res = conn.LastInsertRowId;
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();

   return res;
  }

  public void UpdateDiagnosis(Diagnosis diagnosis) {
   OpenConnection();

   SQLiteCommand cmd = conn.CreateCommand();

   cmd.CommandText = "UPDATE diagnosis " + "SET name = @name, description = @description, symptoms = @symptoms,
treatment = @treatment " + "WHERE id_diagnosis = @id_diagnosis";

   cmd.Parameters.AddWithValue("@id_diagnosis", diagnosis.ID);
   cmd.Parameters.AddWithValue("@name", diagnosis.Name);
   cmd.Parameters.AddWithValue("@description", diagnosis.Description);
   cmd.Parameters.AddWithValue("@symptoms", diagnosis.Symptoms);
   cmd.Parameters.AddWithValue("@treatment", diagnosis.Treatment);

   try {
    cmd.ExecuteNonQuery();
   } catch (SQLiteException ex) {
    Log.Print(ex.Message, ex.Source, Log.type.ERROR);
    Console.WriteLine(ex.Message);
    MessageBox.Show(ex.Message, ex.Source, MessageBoxButtons.OK, MessageBoxIcon.Error);
   }

   CloseConnection();
  }
 }
}
```