МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский технологический университет»
МИРЭА

Физико-технологический институт
Кафедра оптических и биотехнических систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА НА ТЕМУ:

«ИСПОЛЬЗОВАНИЕ МЕТОДОВ РАСПОЗНАВАНИЯ В ЗАДАЧАХ АВТОМАТИЗИРОВАННОГО ОПРЕДЕЛЕНИЯ ЛЕКАРСТВЕННЫХ ПРЕПАРАТОВ НА ОСНОВЕ ДАННЫХ ПАЦИЕНТА»

Студент:
Колосов И.А.

Научный руководитель:
к.т.н., доц. МИРЭА Степанов Д.Ю.

Москва – 2020

Аннотация

Выпускная квалификационная работа посвящена разработке компьютерной программы, автоматизирующей действия врача по подбору лекарственного препарата. Цель работы – использование методов распознавания в задачах автоматизированного определения лекарственных препаратов на основе данных пациента. Использование методов распознавания обусловлено их способностью «обучаться», а также простотой и универсальностью, что подходит под решаемую задачу.

В ходе выполнения работы были изучены действия врача для нахождения процессов для автоматизации и разработана схема процессов. Составлены требования к создаваемой системе. Разработка рабочего прототипа программы велась по итерационной модели, после чего было произведено его тестирование. В качестве ключевых технологий были использованы язык программирования Python и СУБД SQLite.

ОГЛАВЛЕНИЕ

Введение

В настоящее время в работе системы здравоохранения страны закрепился принцип доказательной медицины, позволяющий путем исследований выявить особенности заболеваний и применять эффективные способы лечения. В то же время этот принцип не лишен определенных проблемных моментов и имеет свои слабые места.

Одной из проблем принципа доказательной медицины являются значительные затраты времени, финансовых ресурсов и трудозатрат на проведение исследований. Это заставляет искать компромиссы в виде уменьшения выборки при проведении исследований, увеличения объемов финансирования, проведения исследований с привлечением специальных исследовательских лечебных учреждений и др. Решение данной проблемы – один из путей повышения полноты и достоверности знаний и информации для эффективности лечения заболевания.

Другой путь, дополняющий оптимизацию медико-биологических исследований лечения заболеваний – первичный статистический анализ и использование его данных. По различным причинам анализ не проводится и статистические данные не исследуются. Данными для такого анализа являются физиологические показатели генеральной совокупности пациентов, параметры симптомов новых или редких заболеваний, данные лечения сочетаний заболеваний и другие. Своевременная обработка и использование результатов анализа этих данных дает возможность получать информацию об особенностях и закономерностях заболевания, что должно помочь принимать более верные решения в процессе лечения пациентов отдельному врачу, лечебному учреждению и системе здравоохранения в целом. Лечение – система мероприятий, направленных на восстановление здоровья, предупреждение

осложнений заболевания и устранение тягостных для больного проявлений болезни. Самостоятельным разделом теории и практики лечения заболевания (терапии) является система лечения лекарственными средствами (препаратами).

Актуальность данной работы заключается в применении методов распознавания в решении задачи по поиску лекарственных препаратов при лечении заболевания, при соблюдении условия универсальности используемой методики для большого количества возможных вариантов решаемой задачи. Для того чтобы это было возможным, требуется учесть стандартный метод решения задачи и выявить требования, которые необходимо выполнить для ее решения с использованием методов распознавания. В работе были применены нотации IDEF0 и IDEF3 и выделены требования к разработке программы, решающей задачу для нужд лечебного учреждения (больницы), разработан и протестирован её прототип.

Цель работы – использование (применение) методов распознавания в задачах автоматизированного определения лекарственных препаратов на основе данных пациента. Для достижения цели требуется изучить:

- Итерационную методологию внедрения информационных систем, а также методы распознавания образов.
- Графические нотации проектирования процессов IDEF0, IDEF3 и данных UML CD.
- Структуру описания пользовательских программ и виды тестирования программных продуктов, а также практически их выполнить.
- Идентифицировать требования к процессам обучения и тестирования классификатора по предложению лекарственных средств.

- Спроектировать процессы в IDEF0 и IDEF3 для модели AS-IS и TO-BE до 3-4 уровней детализации; данные – UML CD, включая нормализацию таблиц; структуру приложения, а также блок-схему заданной разрабатываемой функции для каждой итерации.
- Реализовать и количественно оценить программное приложение для автоматизации работы больницы, включая разработку заданной функции, в среде Python, кроме того ознакомиться со средой программирования Python.

Объект исследования – проблема использования ИТ/ИС в медицинских учреждениях. Предмет исследования – работа врача по подбору лекарственного препарата по данным пациента на основе данных предыдущих пациентов. Практическая ценность исследования заключается в описании проблемы и опыта ее решения с использованием исследуемых математических методов в процессе разработки информационной системы. Исследование дает данные о сложности разработки и подходах (и технологиях) ее упрощающих, а также популяризирует их или, напротив, указывает на недостатки.

Раздел 1. Обзор литературных источников

Литература, подобранная для изучения данной темы, касается трёх аспектов работы:

- Проектированию информационных систем.
- Медицинским СУБД. Роль систем управления баз данных (СУБД) очень важна при работе с большим количеством данных. К тому же, предметная (профессиональная) область весьма специфична.
- Графическим нотациям моделирования бизнес-процессов.

Первый основной литературный источник в списке – учебное пособие Пензенского Государственного университета «Проектирование программного обеспечения» 2014 года издания. Цель пособия – помощь в освоении современных методов и средств проектирования программного обеспечения информационных систем в экономике, технике и в других областях. В нем описаны процессы, модели и стадии жизненного цикла программного обеспечения (ПО) экономических информационных систем. Приведены структурный и объектно-ориентированный подходы к проектированию ПО. Отражено применение стандартного языка объектно-ориентированного моделирования и UML. Рассмотрены функции и компоненты CASE-средств и их практическое воплощение в наиболее развитых программных продуктах. Издание подготовлено на кафедре «Высшая прикладная математика» Пензенского государственного университета и предназначено для бакалавров, обучающихся по направлению «Прикладная математика»; представляет интерес для научных исследований.

Второй основной источник – сборник статей «Роль науки в развитии общества». В сборнике опубликованы статьи международной научно-

практической конференции 20 декабря 2015 года в городе Казань. Нас интересует статья «Применение технологии IDEF для моделирования медицинских СУБД» за авторством Д.А. Половодова, Е.А. Половодовой, С.Е. Сергина, магистров Кубанского государственного университета. В этой статье говорится уже сразу о трёх вышеуказанных темах: и графические нотации, и медицинские базы данных, как проектируемые информационные системы.

Касательно баз данных, авторы осветили современное положение дел в этой области программного обеспечения. Отметили важность их использования в медицине в тандеме с применением технологии интернет. Дополнили и развили специфику использования медицинских баз данных, таких как:

- Предоставление пользователю лишь результата поиска, а не самой БД.
- Полнота представления запрошенной информации.
- Высокая скорость обновления, доступа.
- Интерактивность.
- Терминальная (локальная) и удаленная работа пользователя.

Тезисным в работе авторов является следующее. Наиболее удобным языком моделирования бизнес-процессов является IDEF0, где система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации. Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы. Согласованные точки зрения постановщика и исполнителя проектной задачи, дополняющие и не противоречащие нормативным ограничениям.

Стоит отметить, что, несмотря на малое количество сведений о работе медицинских учреждений, разновидностях графических нотаций IDEF и базах данных, работа без лишних слов показывает ключевые и важные моменты, которые должен знать каждый разработчик баз данных медицинского назначения и использованием комплексного проектирования. Тем более, что авторы многое показывают на примерах.

Третьим основным литературным источником является электронный ресурс. Это конспект лекций Дальневосточного государственного университета путей сообщений, подготовленный Анисимовым Владимиром Викторовичем. В отличие от книги и разовой статьи, конспект лекций хорошо подходит для куркового изучения. А тот факт, что это электронный ресурс в виде гипертекста, делает его удобным справочником, позволяющим уточнить интересующую тему. Примерно половину площади страницы занимают иллюстрации, распространяющиеся и на исчерпывающие примеры. Особенно стоит отметить наличие блоков вопросов и рекомендаций. То есть, проходя путь от понятия информационной системы до модели её реализации, читатель будет иметь возможность самоконтроля и валидации приобретенных знаний.

Четвертым и пятым источниками является бакалаврская диссертация Швеца М.Ю. «Монотонные классификаторы для задач медицинской диагностики». В работе описываются особенности методов распознавания, оказывающие влияние на их применение в медицине. Благодаря ей были приняты ключевые алгоритмические решения и ограничена область ответственности пользователя и программы.

Прочая (не основная) литература состоит из электронных ресурсов в сети интернет, учебных материалов для вузов, ГОСТ и других материалов, в которых были найдены материалы, позволяющие лучше разобраться с темой

работы, подтверждающие или описывающие те тезисы, которые являются основой обоснования проблематики и реализации работы.

Раздел 2. Теоретическая часть

2.1. Анализ методов распознавания образов

Методы распознавания образов – математический аппарат, предназначенный для классификации и верификации объектов реального мира из выборки по их параметрам.

Класс – это множество объектов с общими свойствами (параметрами), существенными для решения конкретной задачи над объектами предметной области. При этом объектам класса дается метка класса. Например, в гражданской авиации для приема пассажиров терминалом аэропорта требуется знать тип самолета, классифицировав его по размеру, требуемому трапу, набору оборудования, к обслуживанию которого надо быть готовым техническому персоналу.

Классификация – процесс причисления объекта к какому-либо классу по его параметрам (приписывание метки класса). Верификация – сопоставление объекта с описанием (моделью) класса.

Объектами класса могут служить как отдельные объекты, которые можно сосчитать, так и дискретные отсчеты непрерывной величины (например, двумерного изображения, дробящегося на пиксели для их классификации). В любом случае, классифицируемый объект представляется вектором $y = (x_0, x_1, \dots, x_{N-1})$, где x – значение признака, а N – количество признаков (N-мерное пространство). Признак может быть любым значением: числом, значением алфавита формального языка, матрицей или иным объектом.

В задачах классификации большую роль играет сама выборка. Во-первых, она должна быть репрезентативна: количество ее членов должно быть достаточно большим, чтобы ее свойства не отличались от свойств генеральной

совокупности. Во-вторых, позиция элемента в выборке может быть одним из параметров объекта. То есть, пусть у нас есть выборка из элементов y_0, y_1, … y_n, тогда y_i = (i,x x_1, x_2, … x_N), где x ∈ (0;N-1). Из этого произрастают такие свойства выборки как монотонность.

Рассмотрим некоторые методы распознания. Их довольно много. Мы сконцентрируемся на используемом нами методе ближайшего соседа и его модификациях. Метод ближайшего соседа. Данный метод является метрическим и причисляет элемент выборки к классу, к которому принадлежит ближайший к нему элемент. Мера близости – величина, вычисляемая из параметров классифицируемого объекта и параметров классифицированных объектов выборки. Например, по теореме Пифагора:

$$M = \sqrt{\sum_{i=0}^{N-1}(x_i - z_i)^2}\,, \qquad (2.1)$$

здесь M – мера расстояния;

x_i – параметр классифицируемого объекта;

z_i – параметр ранее классифицированного объекта;

N – размерность пространства признаков (их количество).

Если была найдена минимальная M, смотрится, к какому классу принадлежит z и x причисляется к этому же классу. Мера ищется разными способами. Примененный нами вариант описан в разделе разработки и тестирования. Этот метод неустойчив к шумовым выбросам и наиболее применим в случае многотонного изменения признака. Для купирования этого недостатка разработаны его дополнения. Без дополнений возникает необходимость ручного отсеивания данных выборки исходя из предположения,

что параметры классифицируемого объекта изменяются монотонно и делят пространство параметров на отчетливые области [1]. В данной работе применяется описанный выше подход.

Модификацией метода ближайшего соседа является метод k ближайших соседей. Объект причисляется к классу, к которому принадлежать большинство из k ближайших к нему элементов классифицированной выборки. Требуется найти не одну минимальную меру близости, а несколько, после чего сосчитать количество мер минимальных мер до каждого элементов каждого класса. Класс, элементов которого, наибольшее количество среди k ближайших элементов, и считается классом классифицируемого объекта.

2.2. Итерационная модель разработки

Программный продукт будем разрабатывать по итерационной модели. Данная модель наилучшим способом подходит задаче, ибо защищает от рисков, характерных для проектов, в ТЗ которых присутствуют: выполняемые функции, условия использования, ведущая технология и прочие подробности касательно жизненного цикла продукта [2-5].

Суть итерационной модели состоит в разбиении процесса разработки продукта на отдельные стадии (итерации). Размер каждой итерации выбирается таким образом, чтобы на каждой полностью завершалась часть работы, без которой дальнейшая разработка затруднена. Итерации не завершаются пока не будут выполнены все требования части продукта. Для этого сами итерации разделяют на этапы: планирование и формулировка требований, реализация, тестирование, принятие решения об успехе итерации (рис. 2.2.1). Распишем и опишем шаги итерации:

- Первый шаг итерации – после начального планирования и в начале каждой итерации проводится бизнес-планирование, результатом

которого являются требования от работы на итерации и разработки в целом.

- Второй шаг итерации состоит из анализа сформулированных на предыдущем этапе требований, проектировании и реализации проектирования с их учетом. В случае если это происходит на последней итерации, полученный продукт выпускается по завершении.
- Третий шаг – тестирование для оценки получившегося прототипа (частей) продукта.
- Четвертый шаг – оценка результатов тестирования для последующего планирования.

Рисунок 2.2.1 – *Этапы одной итерации*

Итерационная модель имеет следующие преимущества:

- Снижение воздействия серьёзных рисков на ранних стадиях проекта, что ведет к минимизации затрат на их устранение.

- Организация эффективной обратной связи проектной команды с потребителем (а также заказчиками, стейкхолдерами) и создание продукта, реально отвечающего его потребностям.
- Акцент усилий на наиболее важные и критичные направления проекта.
- Непрерывное итеративное тестирование, позволяющее оценить успешность всего проекта в целом.
- Раннее обнаружение конфликтов между требованиями, моделями и реализацией проекта.
- Более равномерная загрузка участников проекта.
- Эффективное использование накопленного опыта.
- Реальная оценка текущего состояния проекта и, как следствие, большая уверенность заказчиков и непосредственных участников в его успешном завершении.
- Затраты распределяются по всему проекту, а не группируются в его конце.
- Итерационная модель защищает разработчика от следующих рисков:
  - Дефицит специалистов.
  - Непрекращающийся поток изменений.
  - Нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию.
  - Недостаточная производительность получаемой системы.

- Разрыв между квалификацией специалистов и требованиями проекта.
- Неравномерная загрузка участников проекта.
- Неправильная приоритизация подзадач разработки [6].

Для нашего проекта итерационная модель разработки наиболее подходит по потому, что решаются перечисленные ниже риски. Конкретные проблемы можно ожидать следующие:

- Дефицит специалистов и нехватка квалификации будут вызваны спецификой предметной области. Не все разработчики и тестировщики имеют опыт и знания в области медицины. Купирование этой проблемы обеспечивают этапы левых квадрантов спирали на рисунке. В нашем случае это означает период тестирования и обсуждения всех вопросов с медицинским учреждением и экспертами в этой области. Это убережет также от реализации лишней функциональности и преждевременной неудобной реализации пользовательского интерфейса. Вышеуказанная проблема влечет общие для любой разработки риски, например, риски приватности данных или невозможность внести быстрые изменения в проект при изменении предметной области. Требуется их регулярная оценка, способствующая их выявлению до этапов прототипирования и тестирования. Этап анализа рисков – правый верхний квадрант рисунка [7].
- Такая разработка строго требует графика. Без него потеряется этапность разработки, а за ней и цикличность. Единственный способ этого избежать – оставаться в нужном квадранте цикла, пока всё не будет готово [8].

- На каждой итерации цикла выполняются схожие действия, однако ресурсы они требуют разные и в разном количестве. Одна итерация может быть гораздо более трудоёмкой, чем другая, что влечет отставание от плана и снижение качества из-за стремления разработчика ему следовать. В результате еще больше теряется время во время этапа тестирования и исправления недостатков.
- Проблемы с архитектурой и накладные расходы – при работе с хаотичными требованиями и без проработанного глобального плана архитектура приложения может пострадать, а на её приведение к адекватному виду могут потребоваться дополнительные ресурсы. По сути, за возможность менять требования в ходе создания продукта приходится так или иначе расплачиваться.
- Нет фиксированного бюджета и сроков, а также нужна сильная вовлеченность заказчика в процесс – для некоторых заказчиков это неприемлемые условия сотрудничества с разработчиком, им лучше подойдёт водопадная (каскадная) модель.

2.3. Идентификация требований

Перед разработкой программы требуется собрать требования заказчика [9] чтобы понять, способна ли фирма разработчик создать продукт. Ведь несмотря на наличие задания, в заказе остается много неизвестного для исполнителя [10]. Простой список требований без выставления приоритетов не подходит и требует доработки [11]. По технике приоритезации MuSCoW требования можно разделить на типы по важности:

- Must have (обязательные) – требования с наибольшим приоритетом, без выполнения которых релиз продукта невозможен.

- Should have (желательные) – высокоприоритетные требования, которые критичны для функционала;
- Could have (возможные) – требования, которые можно включить в текущий релиз, но которые не влияют существенно на его успех;
- Won't have (отсутствующие) – требования, которые не являются необходимыми в текущем релизе, но которые можно включить в следующие.

Таблица 2.3.1 – *Таблица требований*

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования согласно MuSCoW
1	Хранение параметров пациентов	Таблица параметров пациентов	База данных SQLite с таблицей данных	Must have (должно быть)
2	Хранение названий параметров пациентов	Таблица названий параметров пациентов	База данных SQLite с полем названий параметров	Must have (должно быть)
3	Хранение названий применимых лекарств	Таблица названий применимых лекарств	База данных SQLite с полем лекарств	Must have (должно быть)

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования согласно MuSCoW
4	Обучение классификатора на основе метода распознавания образов при помощи обучающей выборки	Алгоритм расчета параметров (весов) классификатора	Программный код на ЯП Python, выполняющий расчет параметров (весов) классификатора	Should have (желательно)
5	Применение классификатора классификации записей классифицируемой выборки	Классификатор, использующий метод Распознавания образов и вычисленные раннее параметры (веса)	Программный код классификатора на ЯП Python, использующий метод Распознавания образов и вычисленные раннее параметры (веса)	Should have (желательно)
6	Создание выборки для новой медицинской задачи с произвольны ми параметрами и препаратами	Создание таблицы с произвольны ми атрибутами параметров списком лекарств	Окно «Создать новую базу данных» одноименное окно	Could have (возможные)
7	Ввод данных в классифицируемую выборку (таблицу)	Возможность создания записей замеров параметров пациентов	Режим главного окна для ввода данных и запускающий его пункт меню	Could have (возможные)

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования согласно MuSCoW
8	Редактирование данных обучающей выборки (таблицы)	Возможность вносить изменения в поля таблицы обучающей выборки	Режим главного окна для редактирования таблицы обучающей БД и пункт меню для его запуска	Could have (возможные)
9	Классификация записей пациентов их таблицы с целью выяснения подходящего для лечения препарата	Возможность применения классификатора к записям	Режим главного окна для классификации записей таблицы классифицируемой БД и пункт меню для его запуска	Could have (возможные)
10	Подсказки для пользователя	Возможность получить информацию о порядке пользования программой	Подсказка в панели статуса главного окна	Won’t have (отсутствующие)

2.4. Моделирование бизнес-процессов в нотациях IDEF0 и IDEF3

Для описания процессов воспользуемся двумя нотациями проектирования: IDEF0 для высокоуровневого проектирования и IDEF3 для низкоуровневого. Эти нотации проектирования выбраны из-за своей пригодности к описанию бизнес-процессов в области медицины [12]. Но для начала разберемся, что это за нотации, для чего используются и чем отличаются друг от друга.

Эта нотация используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции. Элементы, используемые в процессе моделирования с помощью нотации IDEF0, приведены на рис. 2.4.1.

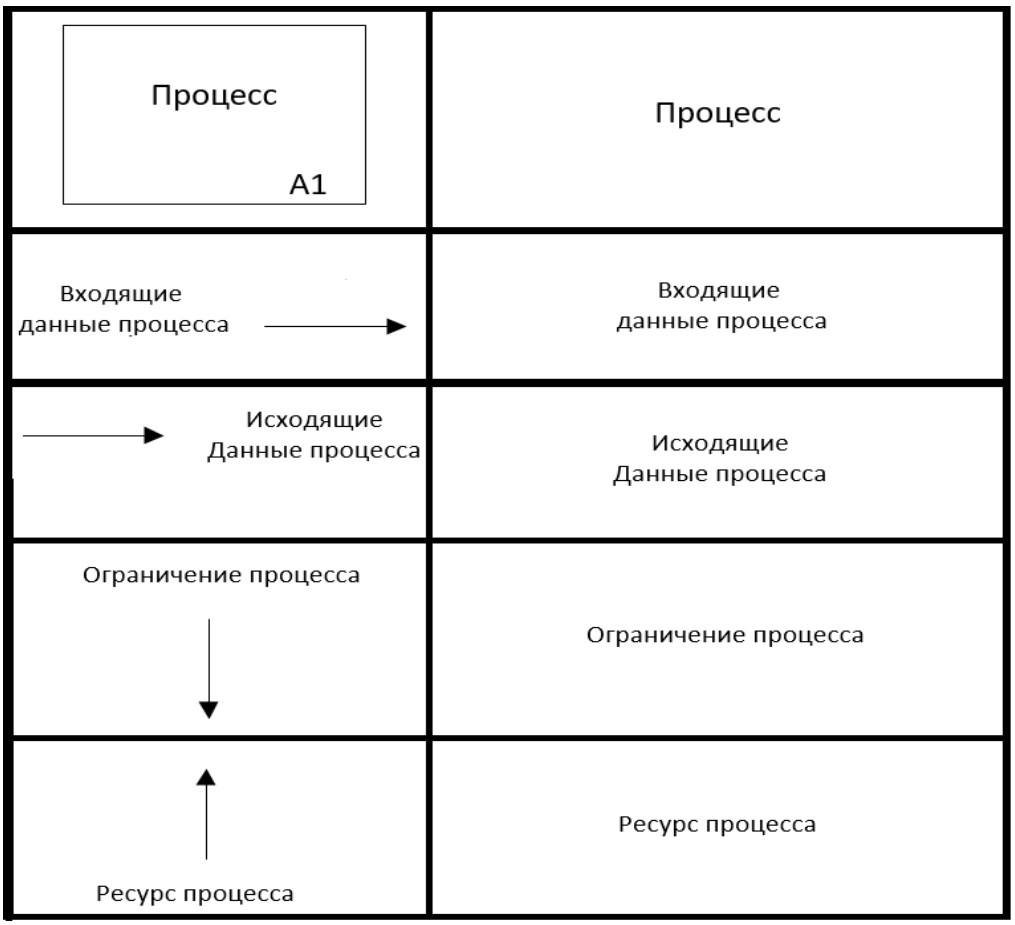


Рисунок 2.4.1 – *Обозначения нотации IDEF0*

Второй нотацией моделирования является IDEF3. Она предназначена для низкоуровневого моделирования. Элементы, используемые в процессе моделирования, представлены на рисунке 2.4.2.

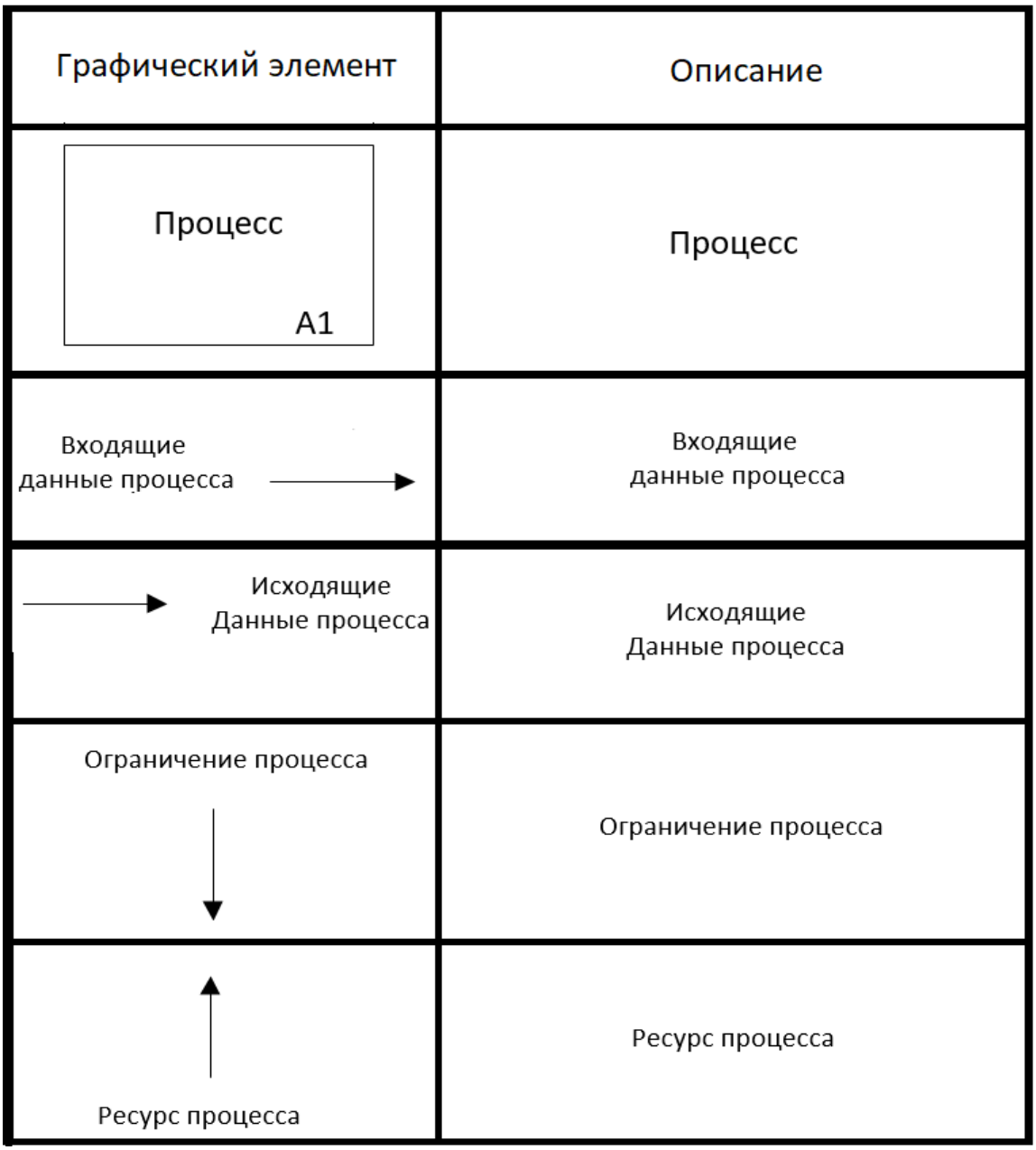


Рисунок 2.4.2 – *Обозначения нотации IDEF3*

Низкоуровневое проектирование в отличие от высокоуровневого не рассматривает элементы процесса с точки зрения наличия самого факта воздействия одного элемента на другой. В нотациях нижнего уровня указываются логические связи между процессами, от состояния которых ведется манипуляция объектами [13]. Произведем декомпозицию процессов,

происходящих в больнице при выборе лекарственного препарата или методики лечения таких, какие обычно проходят без нашей разрабатываемой технологии (AS-IS) и с ней (TO-BE) [14].

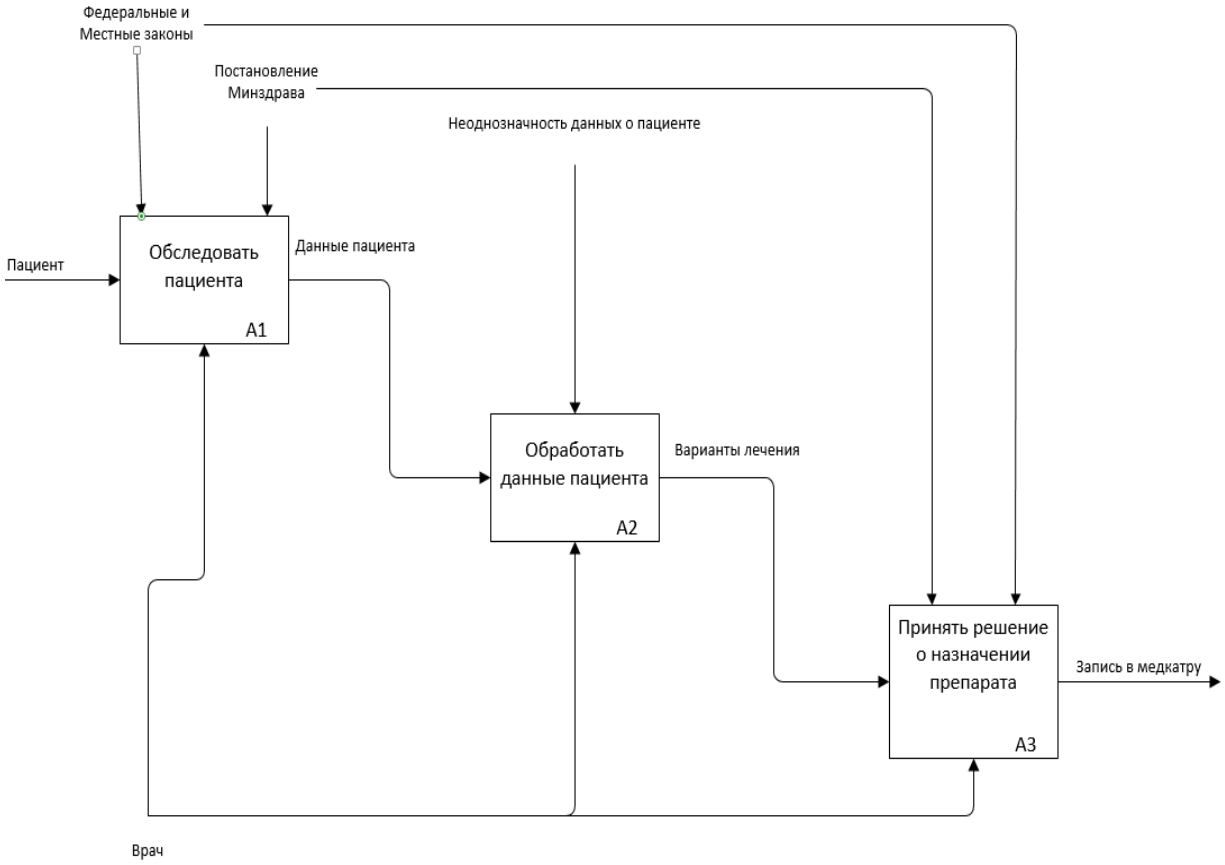


Рисунок 2.4.3 – *Начальный уровень процессов в AS-IS*

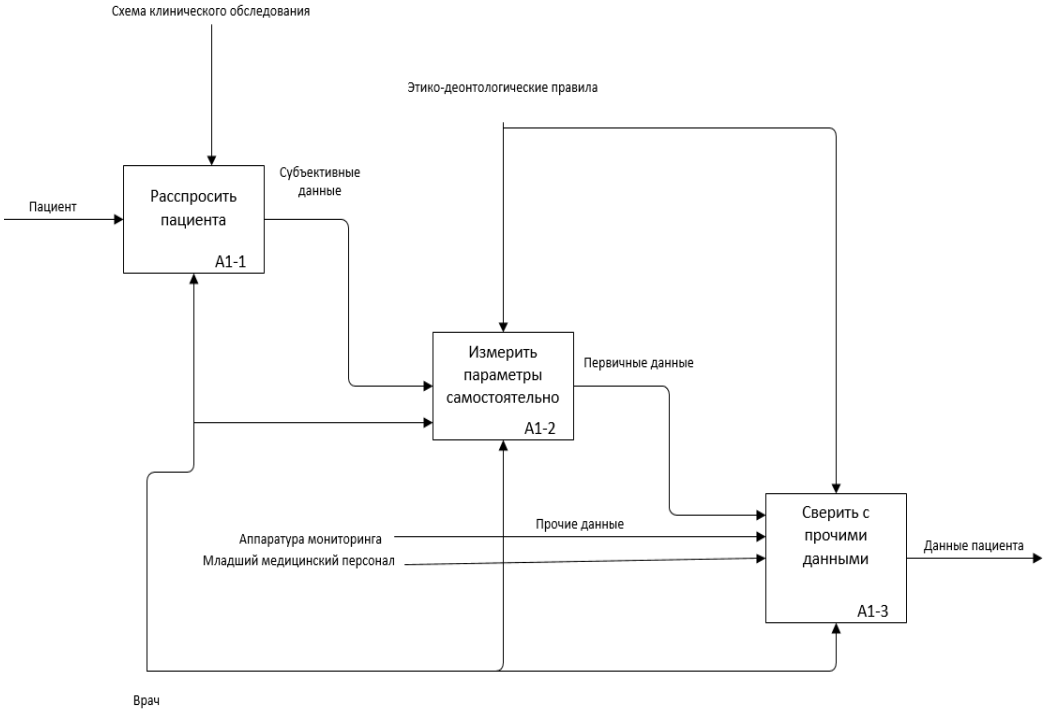


Рисунок 2.4.4 – *Первый уровень подпроцесса A1 в AS-IS*

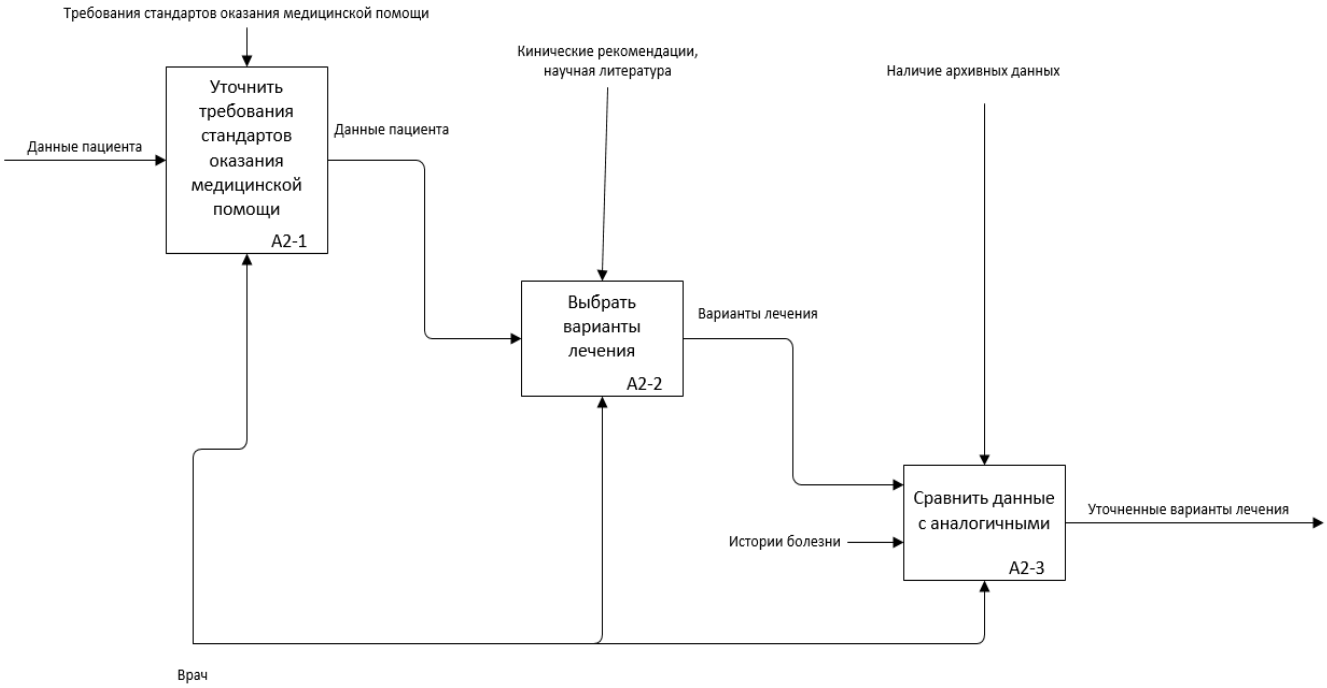


Рисунок 2.4.5 – *Первый уровень подпроцесса A2 в AS-IS*

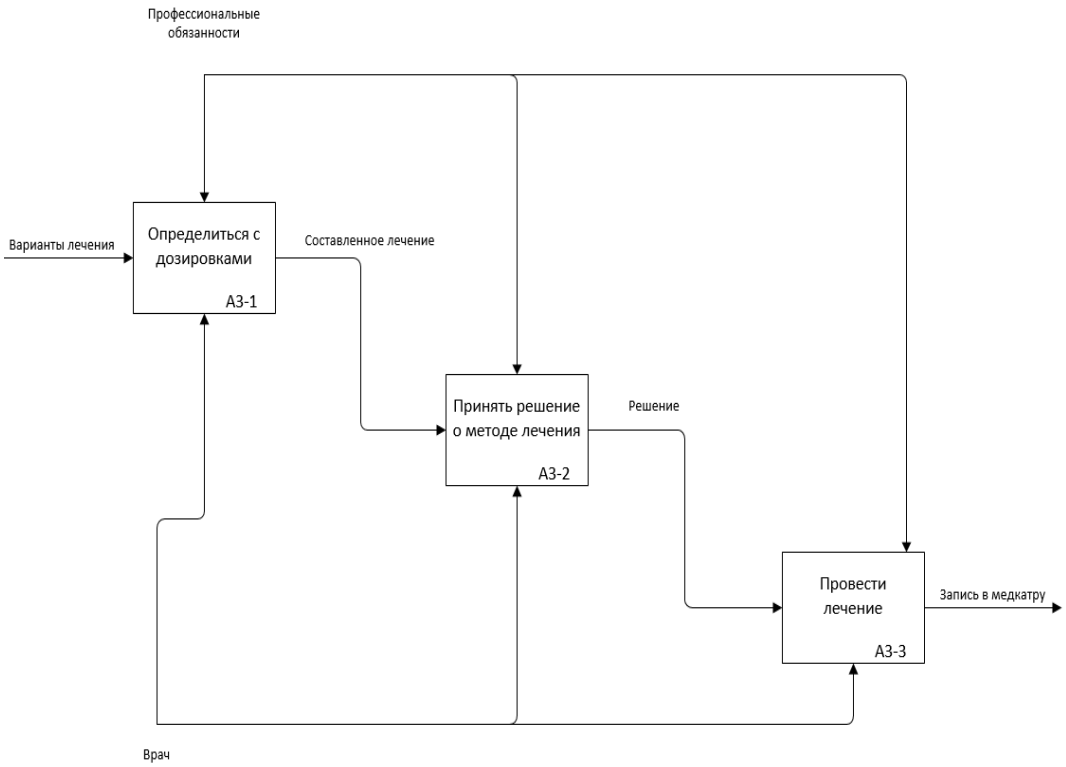


Рисунок 2.4.6 – *Декомпозиция первого подпроцесса А3 в AS-IS*

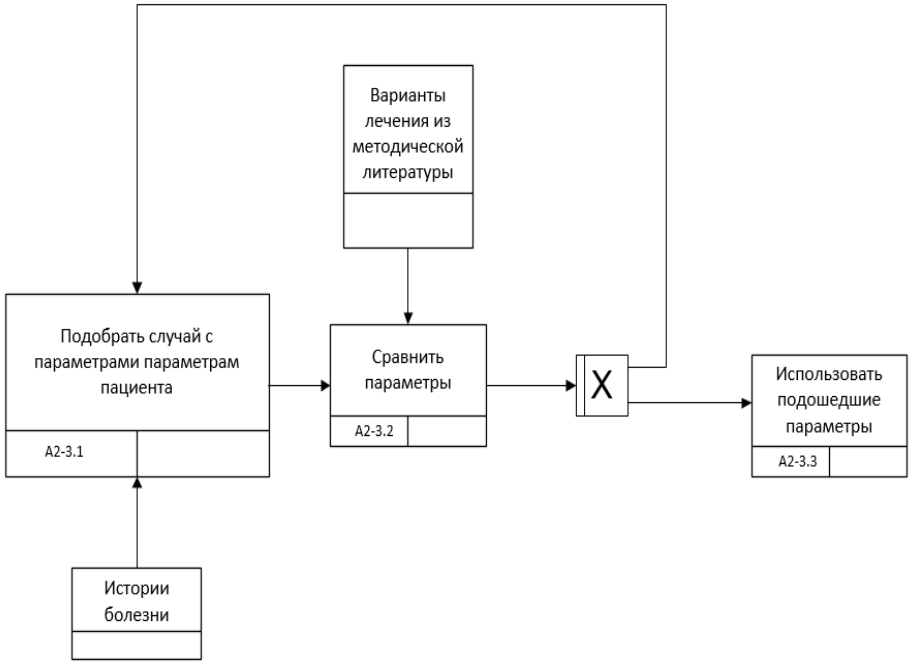


Рисунок 2.4.7 – *Декомпозиция первого подпроцесса А2-3 в нотации IDEF3 в AS-IS*

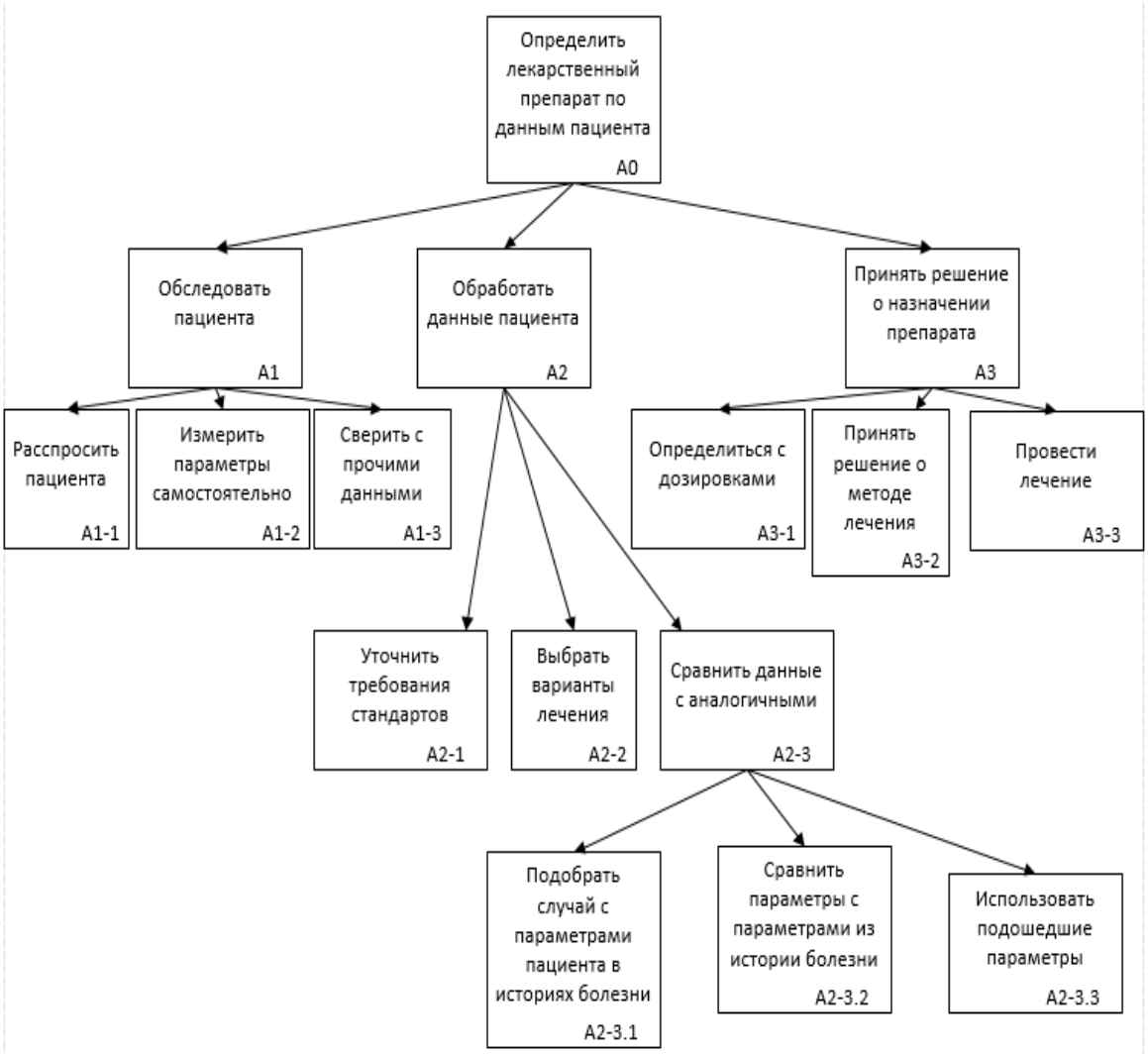


Рисунок 2.4.8 – *Карта процессов в AS-IS*

Подпроцесс принятия решения требует рассмотрения, участвующих в нем источников информации (и данных) в виде эмпирических данных в историях болезни и научных данных из клинических рекомендаций, и прочих источников. Поэтому для его описания на низком уровне лучшим образом подходит нотация IDEF3. Внедряя систему выбора методики лечения, мы возлагаем работу по обработке данных пациента на нее. И меняем нотацию бизнес-процесса [15]. Процесс A2-3 опишем с изменениями в нотации IDEF3.

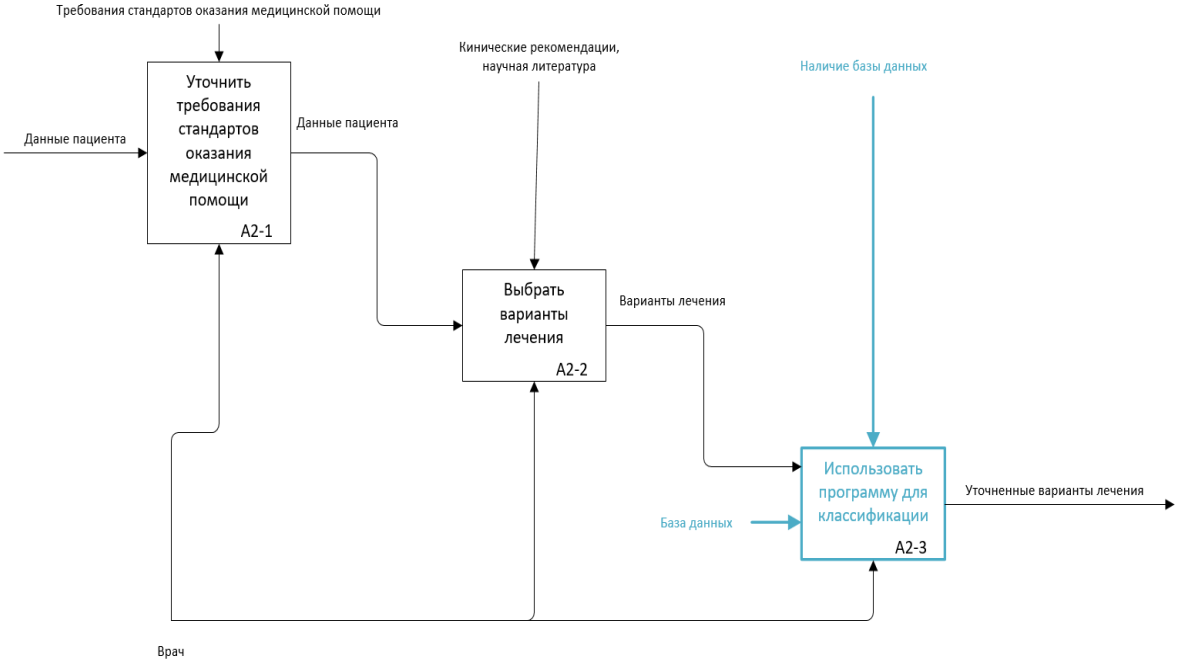


Рисунок 2.4.9 – *Декомпозиция подпроцесса A2 в TO-BE*

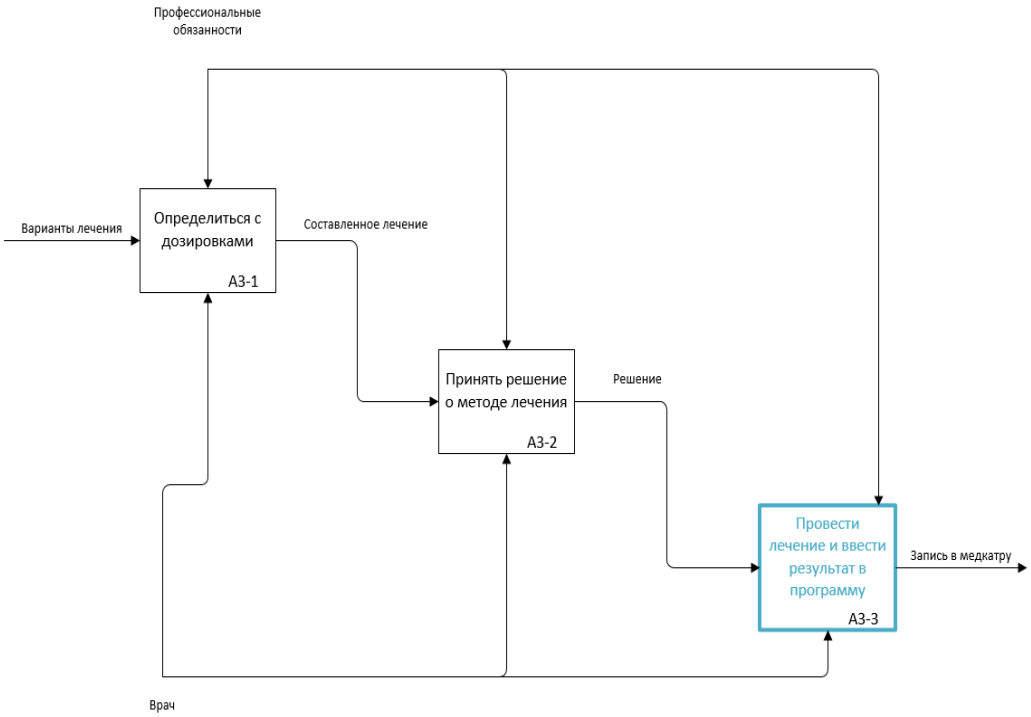


Рисунок 2.4.10 – *Декомпозиция подпроцесса A3 в TO-BE*

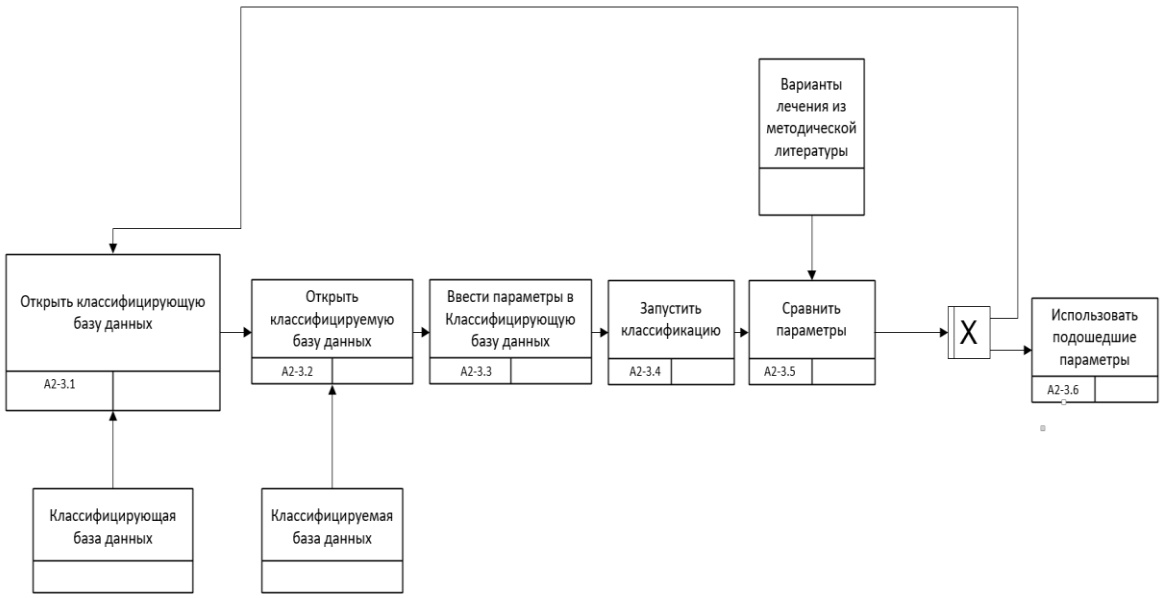


Рисунок 2.4.11 – *Процесс A2-3 в нотации IDEF3 в TO-BE*

Автоматизация подбора метода лечения в данной новации основана на методах распознавания, одним из главных свойств которых является возможность обучаться на выборках. Для того чтобы задействовать эту возможность надо внести изменения в подпроцесс A3 и провести дополнительную декомпозицию для описания логики работы с программой в нотации IDEF3.

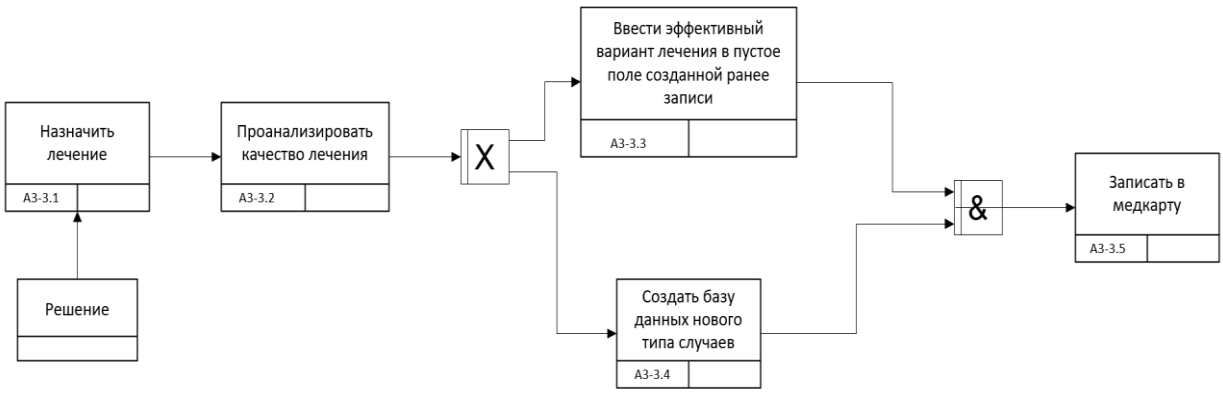


Рисунок 2.4.12 – *Процесс A3-3 в нотации IDEF3 в TO-BE*

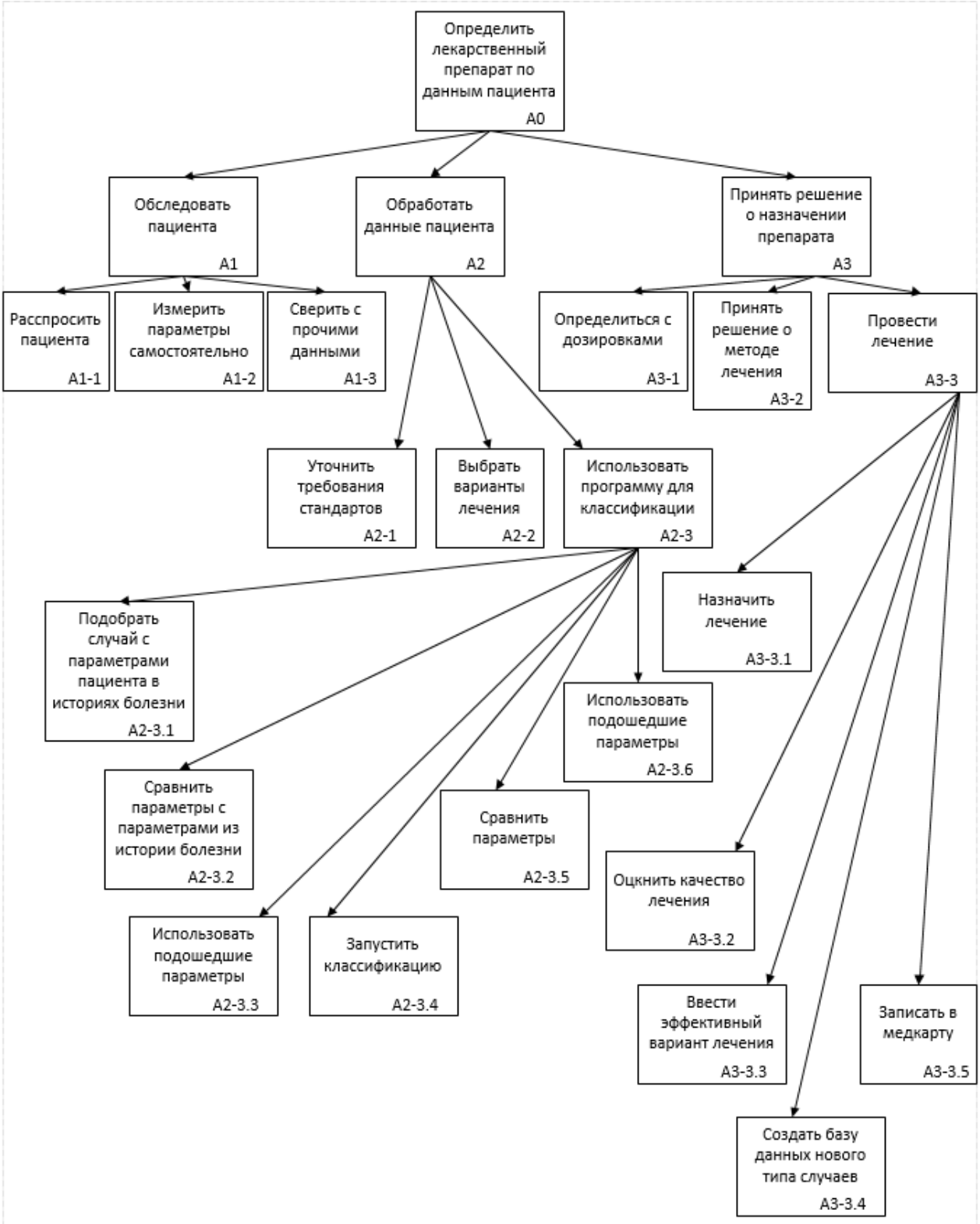


Рисунок 2.4.13 – *Карта процессов в TO-BE*

Раздел 3. Программно-алгоритмическая часть

3.1. Первый прототип программы (первая итерация)

На первой итерации проектируется архитектура данных. При планировании обсуждаются базы данных и таблицы, которые нужно реализовать [16]. Проектирование данных предметной области до алгоритмов является важным аспектом разработки продукта [17]. Имеем список пациентов (обучающую выборку) с указанием достоверно правильной методики лечения исходя из историй болезни. Номер методики – класс, к которому надо причислить пациентов из второй, изучаемой выборки, для которых методика не известна или требует уточнения. Данные о разрешенных для ввода лекарствах (классах) и названиях параметров хранятся в двух текстовых полях отдельно сознанной таблицы метаданных. Такой нестандартный род призван инкапсулировать метаданные, не затрагивая данные. С развитием программы может потребоваться новый род, чтобы хранить больше метаданных для ее внутренних нужд. Названия параметров препаратов предполагается использовать для контроля вводимых данных. Для разработки типовой базы данных была применена открытая СУБД SQLiteStudio.

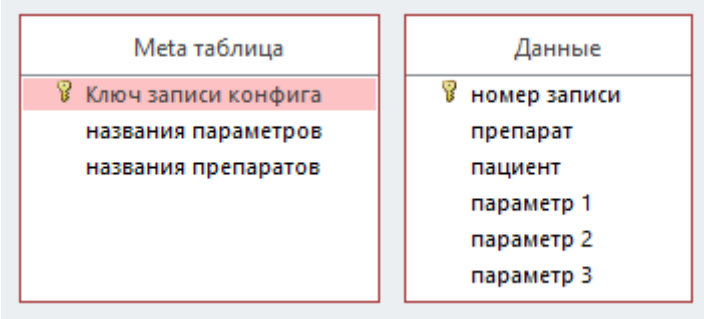


Рисунок 3.1.1 – *Схема данных*

sampleid	patientid	temperature	pain	cirrhosis
1	0	36.6	94.0	2.6
2	1	36.3	96.0	2.1
3	2	36.1	96.0	2.6
4	3	36.4	91.0	2.5
5	4	36.6	90.0	2.3
6	5	36.0	93.0	2.3
7	6	36.6	96.0	2.6
8	7	36.6	94.0	2.5
9	8	36.1	90.0	2.6
10	9	36.5	95.0	2.4
11	10	36.6	92.0	2.2
12	11	36.4	91.0	2.3
13	12	36.4	90.0	2.4
14	13	36.1	95.0	2.4
15	14	36.6	94.0	2.3
16	15	36.3	96.0	2.2
17	16	36.6	91.0	2.6
18	17	36.0	90.0	2.4
19	18	36.3	94.0	2.3
20	19	36.1	93.0	2.6

Рисунок 3.1.2 – *Таблица данных классифицируемой базы данных (выборки)*

В обучающей выборке указан номер пациента, его класс и параметры, определяющие этот класс, например, для купирования боли и снижения температуры больных, страдающих циррозом печени значимыми параметрами, являются температура, субъективное ощущение боли по шкале от 1 до 100 и степень печеночной недостаточности по шкале Чайльда-Пью (от 1 до 15 баллов).

sampleid	classname	patientid	temperature	pain	cirrhosis
1	Aspirin	0	36.3	96.0	2.3
2	Aspirin	1	36.1	90.0	2.6
3	Aspirin	2	36.4	95.0	2.0
4	Aspirin	3	36.6	92.0	2.4
5	Aspirin	4	36.3	91.0	2.0
6	Aspirin	5	36.5	93.0	2.0
7	Aspirin	6	36.4	96.0	2.5
8	Aspirin	7	36.5	91.0	2.1
9	Aspirin	8	36.1	95.0	2.3
10	Aspirin	9	36.1	92.0	2.2
11	Aspirin	10	36.4	90.0	2.1
12	Aspirin	11	36.1	96.0	2.0
13	Aspirin	12	36.5	96.0	2.3
14	Aspirin	13	36.2	93.0	2.0
15	Aspirin	14	36.6	92.0	2.1
16	Aspirin	15	36.6	93.0	2.2
17	Aspirin	16	36.6	90.0	2.5
18	Aspirin	17	36.0	95.0	2.5
19	Aspirin	18	36.0	93.0	2.5
20	Aspirin	19	36.4	93.0	2.3

Рисунок 3.1.3 – *Таблица данных классифицируемой базы данных (базы данных)*

	singlekey	paramnames	classesnames
1	1	temperature, pain, cirrhosis	Aspirin, Diflunisal, Ibuprofen, Dexibuprofen, Naproxen, Fenoprofen, Ketoprofen, Flurbiprofen, Dexket

Рисунок 3.1.4 – *Поля таблицы метаданных*

Выводы по итерации. На этой итерации были решены механизмы хранения и поддержания внутренней логики базы данных. Метаданные не позволят пользователю ввести неразрешенное создателем базы данных лекарство или параметр. Метаданные были представлены в виде полей с текстом, что при двухтабличной реляционной БД не нарушает согласованность данных.

3.2. Второй прототип программы (вторая итерация)

На второй итерации ведется решение используемых математических методов и их реализация в виде алгоритмов программы. Первая задача из списка требований – расчет весов классификатора. Обладая по n параметров p на каждого пациента, измеряем различия (расстояния) в каждом параметре (от параметра с номером 0 до параметра с номером n) по формуле:

$$r = |p_1 - p_2|/\max(p_1, p_2) \quad (3.2.1)$$

Реализуется классификатор, рассчитывающий веса (численно выраженные коэффициенты значимости физиологических параметров пациентов). Создается программный код, выполняющий вычисление меры расстояния, между элементами обучающей и классифицируемой выборки с использованием вычисленных весов. Создаем таблицы для каждого из параметров, где по кромкам будут записываться класс и номер пациента, а на пересечениях столбцов и строк – расстояния.

температура

	sn1cl0	sn2cl0	sn3cl0
sn1cl0	0.0	0.0	0.002762
sn2cl0	0.0	0.0	0.002762
sn3cl0	0.002762	0.002762	0.0

Рисунок 3.2.1 – *Таблица расстояний по параметру «температура»*

Узнаем, с какой частотой (ошибкой) для каждого параметра ближайшее значение этого параметра является ближайшим до пациента другого класса. Исходя из этого (ошибки) считаем вес каждого параметра W, в классификаторе так, что сумма весов равна единице. Веса – основные параметры классификатора. Для их вычисления вычтем ошибку по каждому из единицы и каждое из полученных значений поделим на их сумму. Вычислим веса по формуле:

$$W = (1 - E_i)/\sum_{j=1}^{k} E_j \quad (3.2.2)$$

Вычислим финальную ошибку классификации по формуле:

$$E = \sum_{i=1}^{k} W_i r_i \quad (3.2.3)$$

Построим таблицу расстояний.

финальная ошибка

	sn1cl0	sn2cl0	sn3cl0
sn1cl0	0.0	0.069786	0.020978
sn2cl0	0.069786	0.0	0.059777
sn3cl0	0.020978	0.059777	0.0

Рисунок 3.2.4 – *Таблица расстояний по финальной ошибке классификатора*

```
Ошибки для каждого типа параметров
[0.35000000000000003, 0.26499999999999996, 0.22]
Вес для каждого типа параметров
0.300230946882217
0.33949191685912244
0.36027713625866054
Процент ошибки классификатора = 7%
```

Рисунок 3.2.5 – *Вывод программой весов и ошибок*

В программе производится классификация всей исследуемой выборки целиком, для каждой записи соблюдается условие:

$$R = \sum_{i=1}^{k} \frac{r_i}{W_i} = min \quad (3.2.6)$$

Вес находится в знаменателе для того, чтобы расстояния по параметру с определенном значением веса было обратно пропорционально степени достоверности параметра обучающей выборки. Производится тестирование на двух сгенерированных выборках вместе с тестированием всех сопутствующих интерфейсом. Генерируемые выборки генерируются с незначительными отличиями, бороться с проявлениями которых призваны рассчитанные классификаторы.

Выводы по итерации. Алгоритмы доказали свою работоспособность, однако их производительность может потребовать доработок в случае больших нагрузок.

sampleid	patientid	temperature	pain	cirrhosis
1	0	36.4	92.0	2.1
2	1	36.6	93.0	2.0
3	2	36.0	95.0	2.4
4	3	36.2	90.0	2.4
5	4	36.5	96.0	2.3
6	5	36.1	92.0	2.3
7	6	36.5	92.0	2.3
8	7	36.4	91.0	2.1
9	8	36.0	91.0	2.2
10	9	id похожей записи6		2.1
11	10	название класса похожего пациентаAspirin		2.6
12	11	5		2.3
13	12	temperature36.5		2.2
14	13	pain91.0		2.2
15	14	cirrhosis2.2		2.3
16	15			2.0
17	16			2.0
18	17	36.1	93.0	2.5
19	18	36.1	94.0	2.2
20	19	36.1	93.0	2.6

Рисунок 3.2.7 – *Результат классификации записи исследуемой выборки в виде выдержки параметров ближайшей записи обучающей выборки*

3.3. Третий прототип программы (третья итерация)

Третий прототип – эта итерация посвящена проектированию интерфейсов. На этапе планирования необходимо реализовать режимы главного меню:

- Ввод данных в таблицу данных классифицируемой выборки.
- Классификация выборки, с выводом результата классификации и последующей возможностью добавления достоверных записей в классифицирующую выборку с моментальным обновлением весов классификатора.
- Полнофункциональное редактирование таблицы данных обучающей или/и классифицируемой.
- Создание пользовательских баз данных с определенным набором параметров и препаратов.

Разработка велась на языке Python 3 и библиотеки Tkinter для разработки кроссплатформенных приложений на этом языке.

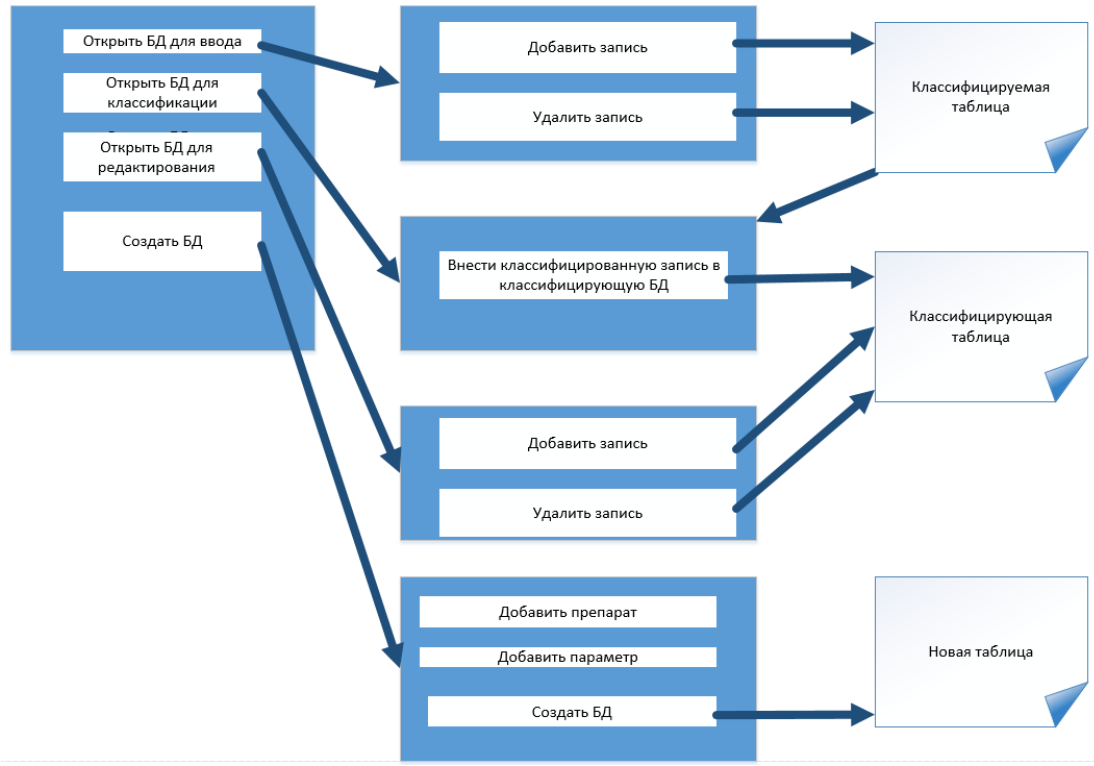


Рисунок 3.3.1 – *Схема программы*

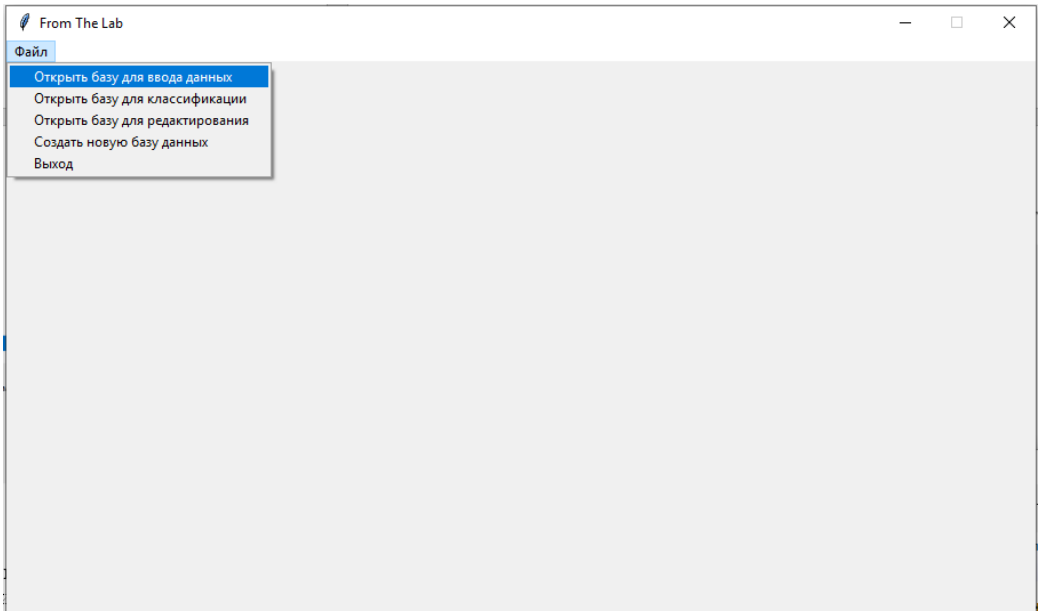


Рисунок 3.3.2 – *Меню «Файл» для запуска режимов работы с БД*

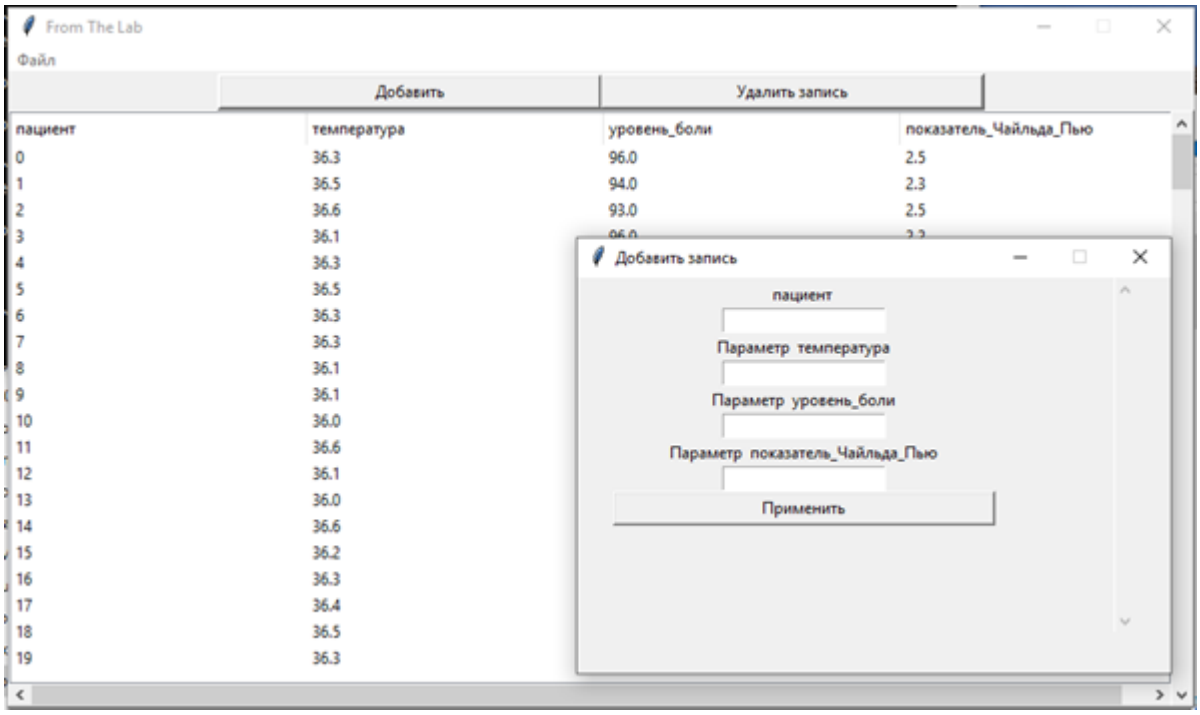


Рисунок 3.3.3 – *Режим ввода с формой для создания записи с параметрами пациента*

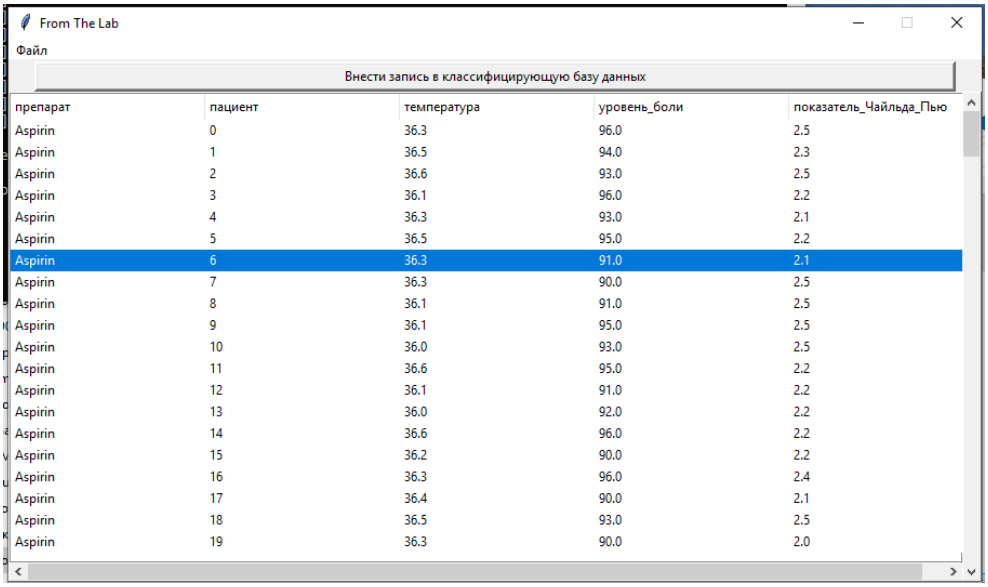
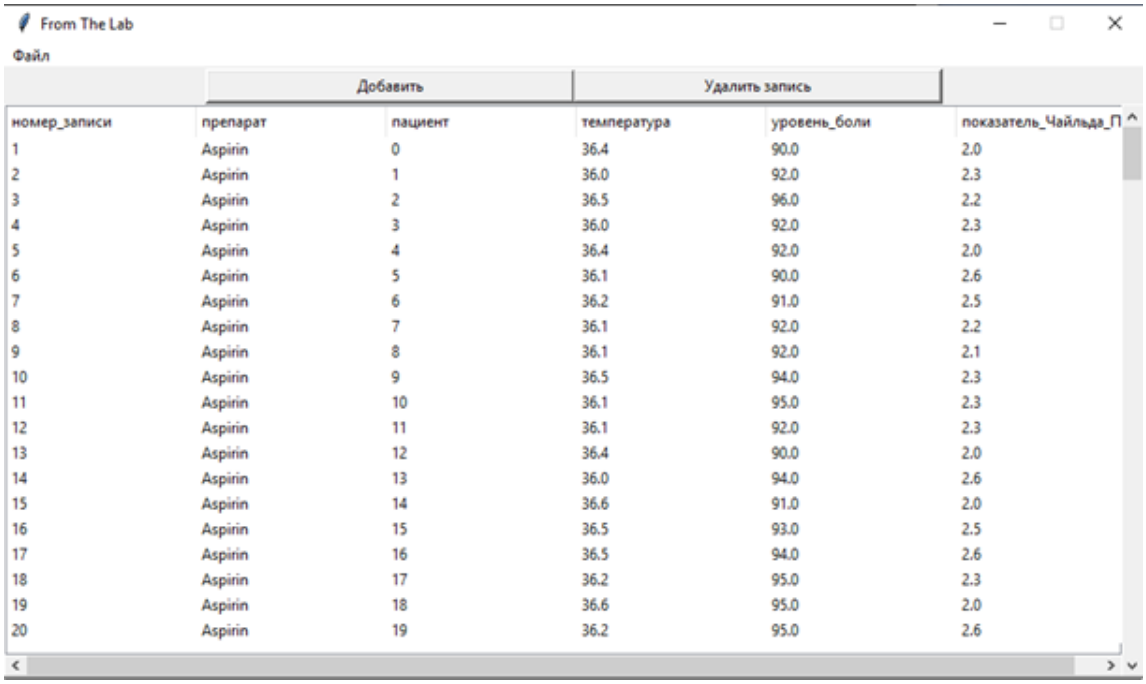


Рисунок 3.3.4 – *Режим классификации с открытой БД, для которой автоматически вычислены предполагаемые препараты с кнопкой для добавления данной записи с полем препарата в классифицирующую БД*



номер_записи	препарат	пациент	температура	уровень_боли	показатель_Чайльда_П
1	Aspirin	0	36.4	90.0	2.0
2	Aspirin	1	36.0	92.0	2.3
3	Aspirin	2	36.5	96.0	2.2
4	Aspirin	3	36.0	92.0	2.3
5	Aspirin	4	36.4	92.0	2.0
6	Aspirin	5	36.1	90.0	2.6
7	Aspirin	6	36.2	91.0	2.5
8	Aspirin	7	36.1	92.0	2.2
9	Aspirin	8	36.1	92.0	2.1
10	Aspirin	9	36.5	94.0	2.3
11	Aspirin	10	36.1	95.0	2.3
12	Aspirin	11	36.1	92.0	2.3
13	Aspirin	12	36.4	90.0	2.0
14	Aspirin	13	36.0	94.0	2.6
15	Aspirin	14	36.6	91.0	2.0
16	Aspirin	15	36.5	93.0	2.5
17	Aspirin	16	36.5	94.0	2.6
18	Aspirin	17	36.2	95.0	2.3
19	Aspirin	18	36.6	95.0	2.0
20	Aspirin	19	36.2	95.0	2.6

Рисунок 3.3.5 – *Режим редактирования БД*

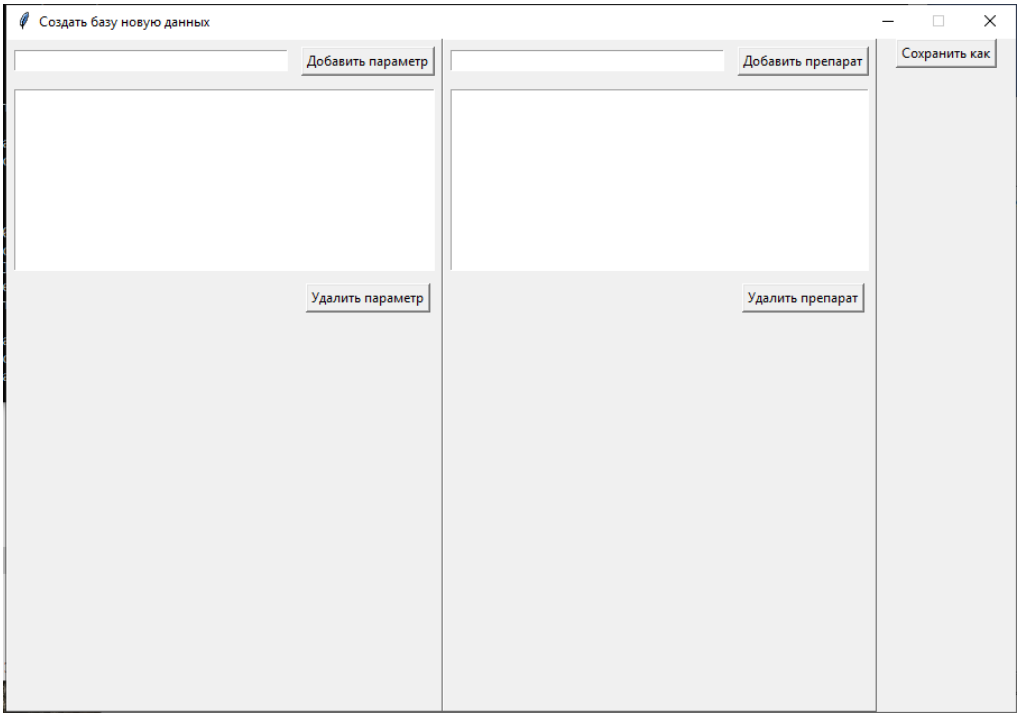


Рисунок 3.3.6 – *Режим создания новой БД с пользовательскими названиями препаратов и параметров*

3.4. Четвертый прототип программы (четвертая итерация)

На четвертой итерации было решено внедрить подсказку в панели состояния. Это должно повысить понятность интерфейса без размещения на начальном экране кнопок открытия файлов, оставив их в подменю «Файл».

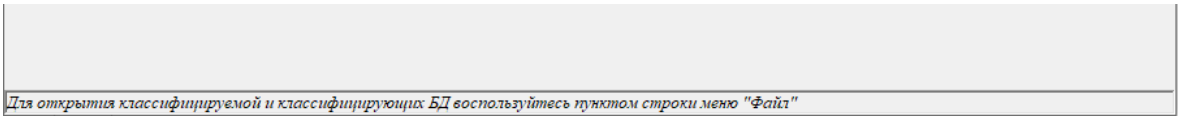


Рисунок 3.4.1 – *Строка подсказки в панели состояния*

На последней итерации было воплощено последнее требование. На этой итерации цикл разработки завершается.

3.5. Нагрузочное тестирование

При работе с базами данных большое значение имеют задержки при выполнении запросов и прочих действий [18] [19]. Как правило, в бизнес-логике могут присутствовать алгоритмы, требующие оптимизации и действия, инициирующие множественные запросы. А при выводе табличных данных из БД через графический интерфейс может возникать задержка вывода. Для сокращения задержек в программе была реализована буферизация таблиц в ОЗУ, но не были проведены оптимизации алгоритмов бизнес-логики. Значения по задержкам для разных размеров таблиц приведены в таблице.

Таблица 3.5.1 – *Данные нагрузочного тестирования*

Количество записей	Действие	t1, с	2, с	t3, с	t4, с	t5, с	Среднее время отклика, с	Среднеквадратичное отклонение
5	Открытие	0,004	0,003	0,002	0,003	0,007	0,004	0,002

Количество записей	Действие	t1, с	2, с	t3, с	t4, с	t5, с	Среднее время отклика, с	Среднеквадратичное отклонение
5	Добавление	0,020	0,023	0,023	0,022	0,024	0,020	0,019
5	Удаление	0,015	0,013	0,017	0,016	0,014	0,015	0,011
5	Классификация	0,075	0,079	0,076	0,076	0,074	0,075	0,072
50	Открытие	0,005	0,004	0,004	0,004	0,004	0,005	0,001
50	Добавление	0,034	0,035	0,041	0,039	0,040	0,034	0,034
50	Удаление	0,035	0,034	0,032	0,030	0,029	0,035	0,028
50	Классификация	0,087	0,087	0,087	0,087	0,089	0,087	0,083
500	Открытие	0,016	0,014	0,014	0,015	0,015	0,016	0,011
500	Добавление	0,180	0,183	0,173	0,183	0,171	0,180	0,174
500	Удаление	0,185	0,179	0,180	0,182	0,183	0,185	0,178
500	Классификация	0,224	0,228	0,225	0,236	0,239	0,224	0,227

Тестирование показало, что при использовании больших баз данных больше и задержки, растущие линейно с малым разбросом. Однако задержки при открытии зависят от размера таблицы линейно и для увеличения быстродействия менять алгоритм открытия БД на другой алгоритм не требуется. Прочие задержки пренебрежимо малы.

Раздел 4. Организационно – экономическая часть

В экономическом разделе описаны и рассчитаны объемы затрат в разрезе статей затрат на разработку продукта. Последний представляет собой программу, конечным пользователем которой является врач больницы. Оптимальная разработка программы потребует усилий команды из трех человек:

- Менеджер для работы с заказчиком (product owner). В его обязанности должны входить задачи общения с заказчиком об особенностях предметной области, сбора требований и формулировке технического задания. Его задача – выступать в проекте на стороне бизнеса, дать клиенту условия для формулировки требований. Он должен доносить до программистов требования заказчика, при необходимости, принимать решения за него или связываться с ним для уточнения работы.
- Старший программист. Его роль в проекте противоположна роли менеджер по связи с заказчиком и состоит в облегчении разработки. Он решает, как интерпретировать и реализовать техническое задание и вместе с менеджером решает - реализованы ли требования заказчика. Организует работу младшего программиста, работает над программой сам и тестирует её.
- Младший программист выполняет задачи, выделенные на него старшим программистом. Задачи могу состоять из тестирования, программирования или участия в обсуждении технической проблемы.

Для составления сметы затрат на разработку программы автоматизированного определения лекарственных препаратов на основе

данных пациента с использованием методов распознавания образов требуется определить затраты по следующим статьям:

- Заработная плата исполнителей и страховые отчисления.
- Расход электроэнергии.
- Амортизация оборудования.
- Затраты на аренду офиса.

Три члена команды выполняют разные функции, что порождает разницу в человеко-часах и использовании ресурсов. При работе по пятидневной рабочей неделе с восьмичасовым рабочим днем целесообразно распределить работу по часам. Основной объем работы выполняют программисты и их труд наиболее затратный по времени. Менеджер же проводит формулировку задач, связь с заказчиком, осуществляет контроль и прием работы программистов для представления заказчику. По этой причине, решено сократить время выполнения его обязанностей по проекту в середине разработки.

Таблица 4.1 – *График работы команды в рабочих днях*

Период	Менеджер	Старший программист	Младший программист
1 – 5 день	1 – 3 день	1 – 5 день	1 – 5 день
5 – 10 день	7 – 8 день	5 – 10 день	5 – 10 день
10 –15 день	13 – 15 день	10 – 15 день	10 – 15 день

Оплата труда состоит из основной заработной платы и дополнительной. Основная заработная плата вычисляется исходя из ставки сотрудника и времени, затрачиваемого на выполнение работы.

Таблица 4.2 – *Затраты на заработную плату*

Сотрудник	Ставка в час	Рабочих дней	Заработная плата, руб.
Менеджер	375	10	30000
Старший программист	312	15	37440
Младший программист	250	15	30000
Итого			97440

Полученные в таблице 4.2 числа рассчитаны при условии восьмичасового дня. Расчет заработной платы проводится по формулам 4.1 и 4.2.

$$ЗП = t_{час} * Т_{час}, \quad (4.1)$$

где, ЗП - заработная плата;

$t_{час}$ - часовая тарифная ставка;

$Т_{час}$ - фактически отработано часов.

НДФЛ в Российской Федерации на данный момент составляет 13%, а страховые взносы равны 30%. Также, по страховому тарифу по первому классу опасности согласно федеральному закону 179-ФЗ « Об обязательном социальном страховании от несчастных случаев на производстве и профессиональных заболеваний», требуется производить отчисления в размере 0,2% от заработной платы. При этом НДФЛ удерживается из зарплаты работника, а взносы платит работодатель. Поэтому, последние стоит рассчитать по формуле 4.2 и внести в список затрат.

$$З_{стр} = Ф_{зп} * 0{,}302. \quad (4.2)$$

$З_{стр} = 97440 * 0{,}302 = 29427$ руб.

Для разработки программы не требуется иного оборудования кроме настольного компьютера (или ноутбука) под управлением операционной системы Microsoft Windows 8 или новее, либо операционной системы семейства

GNU/Linux с установленными лицензионно свободными и бесплатными пакетами интерпретатора Python3 и Tkinter. В качестве редактора текста программы может использоваться текстовый редактор Microsoft Visual Studio Code с бесплатными расширениями для языка Python. В качестве программы для создания документации можно применять бесплатную и свободную программу LibreOffice Writer или более распространенный платный Microsoft Word. Иными словами, затраты на программное обеспечение отсутствуют или сделаны фирмой единожды ранее, в случае покупки компьютера с предустановленными Microsoft Windows и Microsoft Office или при их доустановке.

Стоимость аренды составляет 20000 руб. за кв. м в год. При площади зоны офиса в 36 кв. м, за 15 дня будет потрачено 30000 руб. Затраты на электроэнергию считаются при использовании потолочного освещения электрочайника и компьютеров работниками. По СНиП (строительным нормам и правилам) освещенность офиса, где не производятся чертежные работы должна составлять 300 Люкс, что равно 300 Люменам на квадратный метр. В пересчете на площадь рабочей зоны требуется суммарный световой поток в 10800 Люмен. В настоящее время самыми распространенными лампами являются люминесцентные под патрон g13. Лампа на 18 Вт потребляемой мощности обеспечивает световой поток в 1200 Лм. Исходя из вышесказанного, потребуется 9 ламп с суммарным энергопотреблением в 162 Вт.

Затраты на электроэнергию зависят от стоимости машинного часа и времени работы оборудования и определяются по формуле (4.3).

$$З_{эл}=Р*Ц_{эл}*Т_{и}\ , \qquad (4.3)$$

где Р – потребляемая мощность оборудования, кВт/ч;

$Ц_{эл}$ – стоимость одного кВт/ч, руб.;

$Т_{и}$ – время использования оборудования при проведении работ, ч.

Таблица 4.3 – *Затраты на электроэнергию*

Наименование оборудования	Мощность электрооборудования, кВт/ч	Время использования, ч.	Цена одного кВт/ч, руб.	Сумма затрат на эл. энергию, руб.
Ноутбук Lenovo ThinkBook 15 менеджера	76	80	5,50	33,5
Ноутбук Lenovo ThinkBook 15 старшего программиста	76	120	5,50	50,2
Ноутбук Lenovo ThinkBook 15 младшего программиста	76	120	5,50	50,2
Электрочайник REDMOND RK-G127-E	2000	0,117	5,50	1,3
Потолочное освещение	162	120	5,50	107,5
Итого				241,4

Рассчитаем затраты на амортизацию оборудования:

$$A = \left(\frac{Ф_{перв} * Н_а * Т}{Ф_{эф}}\right), \quad (4.4)$$

где $Ф_{перв}$ – первоначальная стоимость оборудования и приборов, руб.;

$Н_а$ – годовая норма амортизации, Т – время использования оборудования, дни;

$Ф_{эф}$ – годовой эффективный фонд времени работы оборудования, для односменной работы он составляет 256 дней.

Таблица 4.4 – *Затраты на амортизацию оборудования*

Наименование оборудования	Стоимость единицы оборудования, руб.	Время использования, дни	Норма амортизации оборудования	Сумма амортизационных отчислений, руб.
Ноутбук Lenovo ThinkBook 15 менеджера	49990	10	0,5	977
Ноутбук Lenovo ThinkBook 15 старшего программиста	49990	15	0,5	1465
Ноутбук Lenovo ThinkBook 15 младшего программиста	49990	15	0,5	1465
Электрочайник REDMOND RK-G127-E	3799	0,015	0,1	0,03
Итого				3907,03

Таблица 4.5 – *Таблица итоговых рассчитанных затрат*

Наименовании статьи затрат	Сумма затрат, руб.
Затраты на заработную плату	97440
Страховые взносы	29427
Затраты на электроэнергию	241,4
Затраты на амортизацию оборудования	3907,03
Затраты на аренду офиса	30000
Итого	161015,43

Затраты финансовых средств на создание системы, проводящей дополнительный анализ и прогнозирование эффективности лечения различных

заболеваний препаратами по параметрам пациентов, дают возможность иметь данные об оптимальных дозировках и случаях применения лекарств, что повысит как эффективность их применения, так оперативность проведения медико-биологических исследований соблюдая принцип доказательной медицины. Применение системы в рамках одной больницы даст возможность адаптировать лечение под ее географическое положение, экологическую обстановку и особенности пациентов. А использование программы в нескольких медицинских учреждениях позволит проводить сравнительный анализ эффективности их работы и создаст предпосылки к обмену свежей клинической информацией.

Заключение

В ходе работы проделано моделирование процессов системы в нотации IDEF0 на верхнем уровне с нулевого до первого уровня декомпозиции. На нижнем уровне моделирование процессов велось с применением нотации IDEF3. Построена карта процессов до внедрения системы и предполагаемая в случае ее внедрения. Проведен анализ требований по созданию медицинской информационной системы. По их списку был составлен план разработки по итерационной модели.

Были изучены методы распознавания и принято решение выбрать и усовершенствовать метод ближайшего соседа. Путем введения весов параметров была повышена его точность. Создан и протестирован прототип программы, реализующей требования за 4 итерации: проектирование данных, проектирование методов и алгоритмов, проектирование интерфейсов, создание подсказок для пользователя. Проведен тест классификации исследуемой выборки улучшенным методом ближайшего соседа с использованием записей классифицирующей выборки с последующим добавлением классифицированной записи в классифицирующую выборку с автоматическим расчетом новых весов. Нагрузочное тестирование созданной программы показало, что скорость отклика для базовых операций (открытие базы данных, добавление записи, удаление записи, классификация всех записей таблицы база данных) составляет десятые доли секунды для выборок до пятисот человек при трех параметрах, что является комфортными значениями, несмотря на использование высокоуровневого интерпретируемого языка программирования.

Оценка экономической состоятельности коммерческой разработки показала, что это возможно сделать силами небольшой команды фирмы по производству программного обеспечения за период меньше квартала. Для

разработки программы не требуется иного оборудования кроме настольного компьютера (или ноутбука) под управлением операционной системы Microsoft Windows 8 или новее, либо операционной системы семейства GNU/Linux с установленными лицензионно свободными и бесплатными пакетами интерпретатора Python3 и Tkinter. Исходя из результатов можно говорить, не только о нужности внедрения систем подобной этой, но и о высокой скорости и простоте их внедрения в практику работы лечебных учреждений. Наличие в программном обеспечении адаптивных алгоритмов ведет к повышению его точности с увеличением объема использования, а быстрый цикл внедрения свидетельствует о потенциале подобного программного обеспечения к дальнейшему развитию.

Список использованных литературных источников

1. Швец М.Ю. Монотонные классификаторы для задач медицинской диагностики. Бакалаврская диссертация / МФТИ.-М., 2015.
2. Федорова Г.Н. Информационные системы: учебник для студ. учреждений сред. проф. образования // Издательский центр «Академия», 2013 – 208 с.
3. Невлюдов И. Ш., Евсеев В. В., Бортникова В. О. Модели жизненного цикла программного обеспечения при разработке корпоративных информационных систем технологической подготовки производства. – 2011.
4. Анисимов В. В. Проектирование информационных систем. Конспект лекций. М.: Дальневосточный государственный университет путей сообщения. URL:https://sites.google.com/site/anisimovkhv/publication/umr/pris (дата обращения 09.12.2019).
5. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.
6. Fieldboom, статья «The Kano Model Explained: Analysis and Examples». URL: https://www.fieldboom.com/kano-model (дата обращения: 22.05.20).
7. Свободная энциклопедия Википедия, статья «Iterative and incremental development».URL:https://en.wikipedia.org/wiki/Iterative_and_incremental_development (дата обращения: 22.05.2020) https://en.wikipedia.org/wiki/Iterative_and_incremental_development.
8. Барышникова М.Ю. Инженерный менеджмент и информационные технологии. Конспект лекций. М.: МГТУ им. Н. Э. Баумана, 2009. - 252 с.
9. Карпенко С.Н., Вершинина Е.В., Гонченко М.С. Обзор моделей жизненного цикла разработки программного обеспечения// Математические и

программные технологии для современных компьютерных систем (Информационные технологии), 2017 – 69 с.

10. Лешек А. Мацяшек Анализ требований и проектирование систем. // Издательский дом «Вильямс», 2005 – 432 с.
11. Идентификация требований и определения спецификаций. Конспект лекций. М.: Томский Политехнический Университет. URL: https://studfiles.net/preview/4241597/page:8/ (дата обращения: 22.05.2020).
12. Свободная энциклопедия Википедия, статья «Анализ требований».URL:https://ru.wikipedia.org/wiki/Анализ_требований (дата обращения: 22.05.2020).
13. Ротер М., Шук Дж. Учитесь видеть бизнес-процессы. Построение карт потоков создания ценности / Ротер М. – М.: Альпина Паблишер, 2017. – 144 с.
14. Степанов Д. Ю. Анализ, проектирование и разработка корпоративных информационных систем: уровень бизнес-процессов / МИРЭА. - М., 2017.
15. Половодов Д. А., Половодова Е. А., Сергин С. Е. Применение технологии IDEF для моделирования медицинских СУБД // Роль науки в развитии общества. – 2015. – С. 7.
16. Клебанов Б.И. Проектирование автоматизированных систем обработки информации и управления. Методические указания к выполнению курсового проекта М.: Уральский государственный технический университет, 2004. – 66 с.
17. Дейт К. Дж. Введение в системы баз данных, 8-е издание. // Издательский дом «Вильямс», 2010 – 1328 с.
18. Проектирование программного обеспечения : учеб. пособие / Т. В. Черушева. – Пенза : Изд-во ПГУ, 2014. – 84 с.

19. Куликов С.С. Тестирование программного обеспечения. Базовый курс. – Минск: Четыре четверти, 2017. – 312 с.
20. Fieldboom, статья «The Kano Model Explained: Analysis and Examples». URL: https://www.fieldboom.com/kano-model (дата обращения: 02.11.2018).