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

Институт информационных технологий
Кафедра промышленной информатики

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине
«Проектирование баз данных»

Студент группы ИКБО-32-21 Хабибуллин О.Р. _____
(учебная группа, фамилия, имя, отчество студента) *(подпись студента)*

Преподаватель Баев И.Б. _____
(должность, ученая степень, звание, фамилия, имя, отчество преподавателя) *(подпись преподавателя)*

Работа выполнена «___» _____ 2023 г.

«Зачтено» «___» _____ 2023 г.

Москва, 2022 г.

СОДЕРЖАНИЕ

ПРЕДМЕТНАЯ ОБЛАСТЬ

На первом этапе проектирования базы данных необходимо определить цель создания базы данных, основные ее функции и информацию, которую она должна содержать

Предметная область «Процесс разработки нейронных сетей» – одна из самых сложных областей в сфере IT. Для получения качественной и продвинутой нейронной сети необходимо учитывать много различных факторов и требований.

1 МОДЕЛЬ НОТАЦИИ IDEF0

В рамках данной предметной области была разработана модель такого процесса, как **«Процесс разработки нейронных сетей»** в нотации IDEF0. На Рисунке 1 представлена контекстная диаграмма данного процесса.

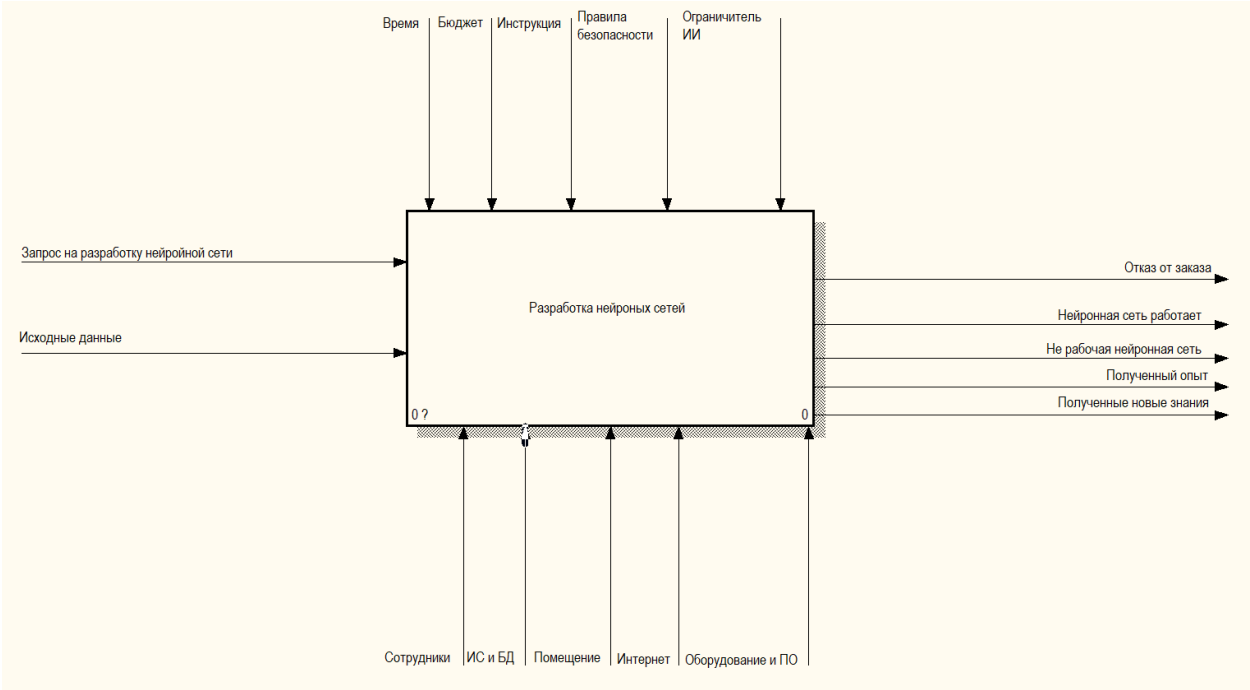


Рисунок 1 – Контекстная диаграмма «Процесс разработки нейронных сетей» в методологии IDEF0

Основной блок:

- разработка нейронных сетей.

Входной информацией системы являются:

- запрос на разработку нейройной сети;
- исходные данные.

Выходной информацией системы являются:

- Отказ от заказа;
- Нейронная сеть работает;
- Не рабочая нейронная сеть;
- Полученный опыт;
- Полученные новые знания.

Управляющей информацией системы являются:

- Время;
- Бюджет;
- Инструкция;
- Правила безопасности;
- Ограничитель ИИ.

Механизмы системы являются:

- Сотрудники;
- ИС и БД;
- Помещение;
- Интернет;
- Оборудование и ПО.

Декомпозируем общий блок «Разработка нейронных сетей» на связанные между собой элементы. Получим 5 основных этапов (Рис. 2):

- Рассмотрение запроса заказчика;
- Подготовка к разробтке нейронной сети;
- Разработка нейронной сети;
- Обучение нейронной сети;
- Тестированпие.

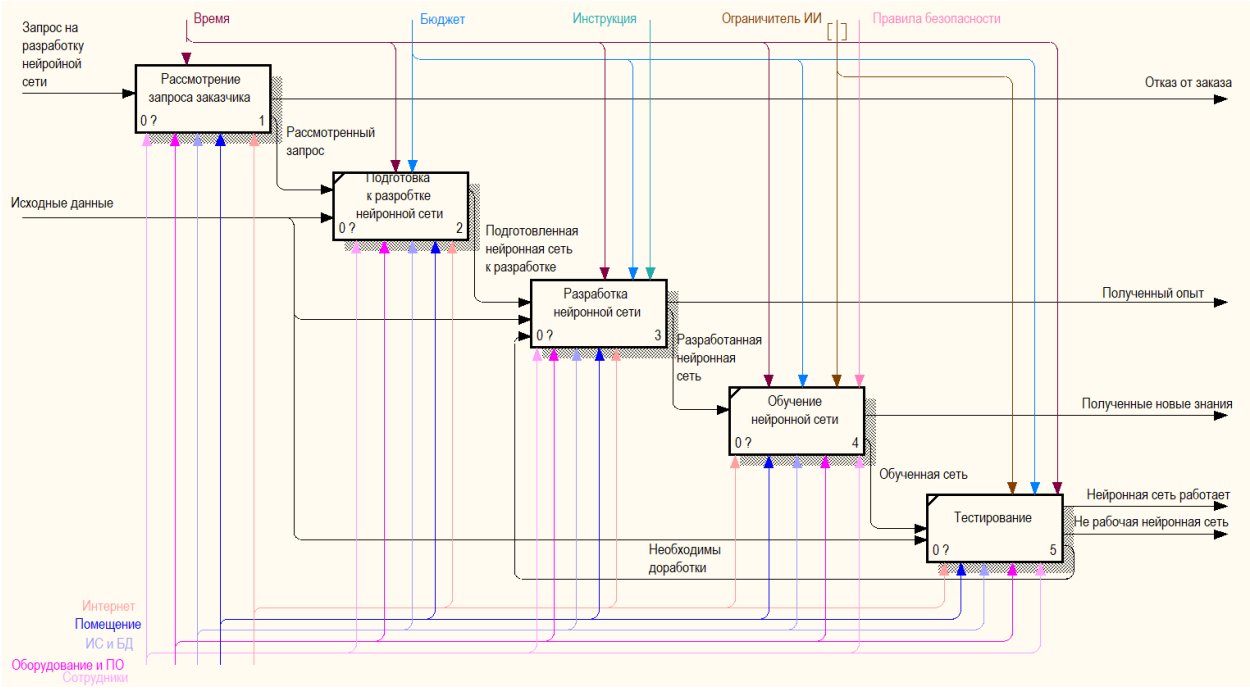


Рисунок 2 – Декомпозиция контекстной диаграммы «Разработка нейронной сети» в методологии IDEF0

Блок «Рассмотрение запроса» декомпозируем еще на 3 этапа (Рисунок 3):

- Рассмотрение запроса;
- Согласование ТЗ с заказчиком;
- Принятие решения на счет разработки.

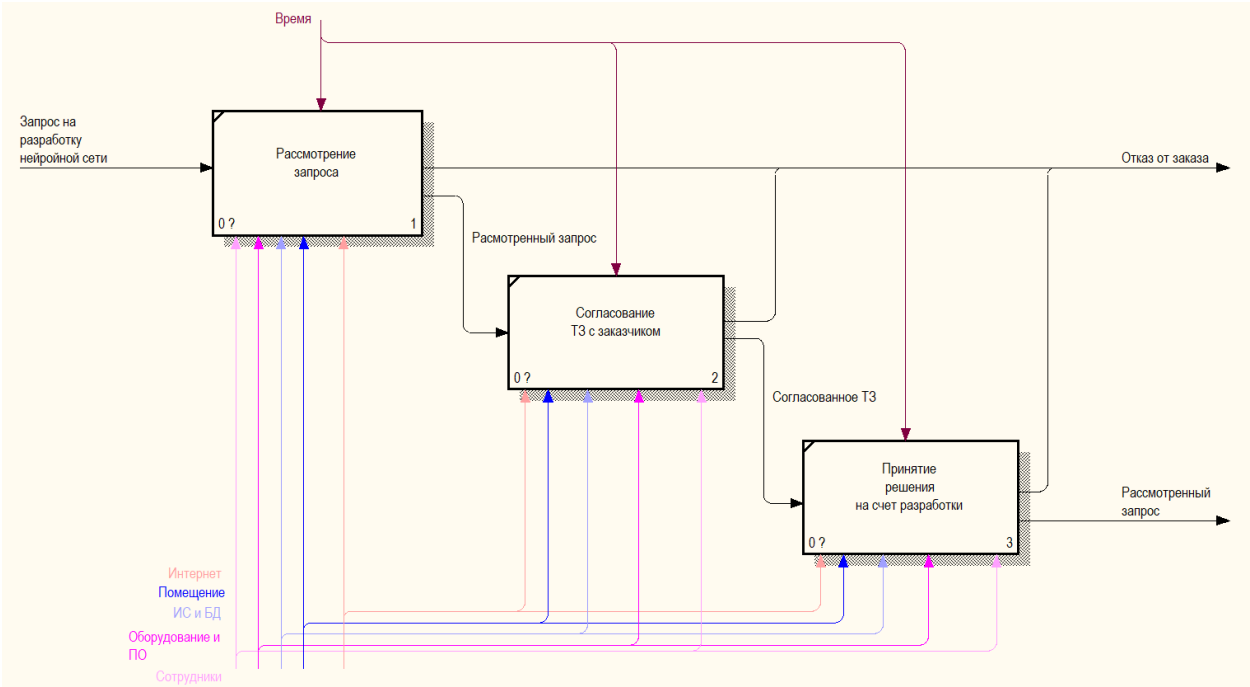


Рисунок 3 – Декомпозиция блока «Рассмотрение запроса» в методологии IDEF0

Блок «Разработка нейронной сети» декомпозируем еще на 5 этапов (Рисунок 4):

- Разобрать, как устроенная нейронная сеть;
- Подготовить компьютер к нейронной сети;
- Установить необходимые программы;
- Добавляем классификатор;
- Добавить набор данных.

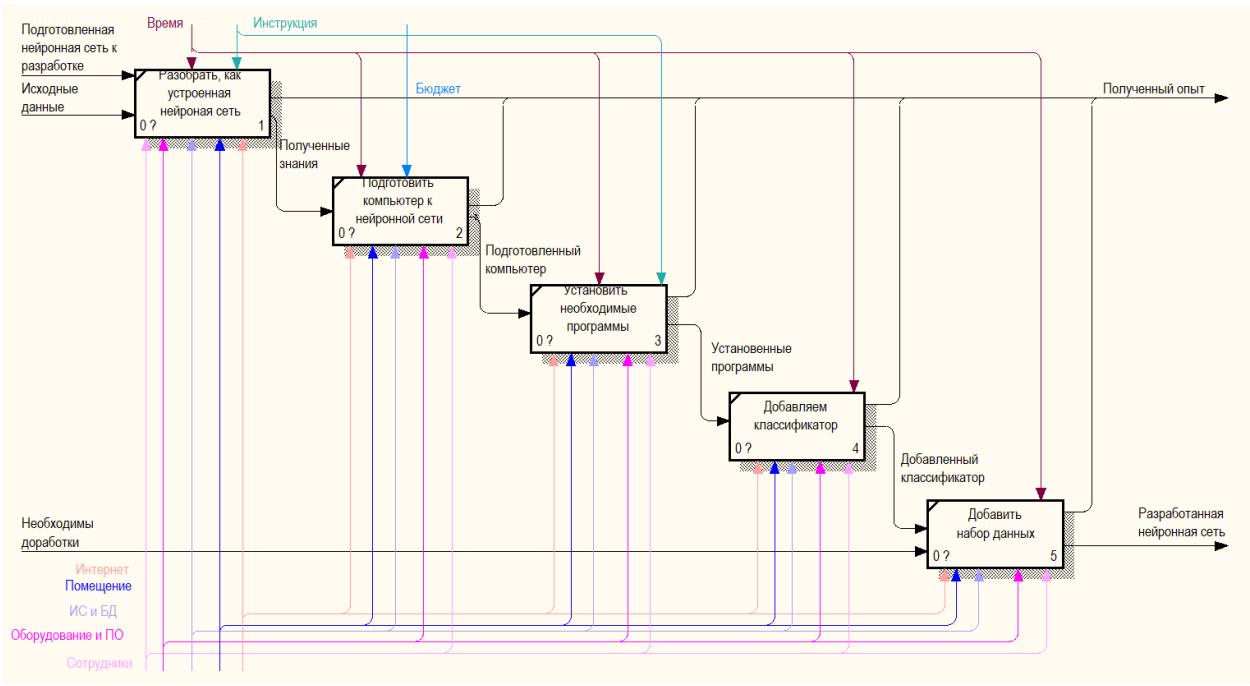


Рисунок 4 – Декомпозиция блока «Разработка нейронной сети» в методологии IDEF0

2 МОДЕЛЬ НОТАЦИИ DFD

Была разработана модель DFD по предметной области **«Разработка нейронных сетей»** (Рисунки 5-8).

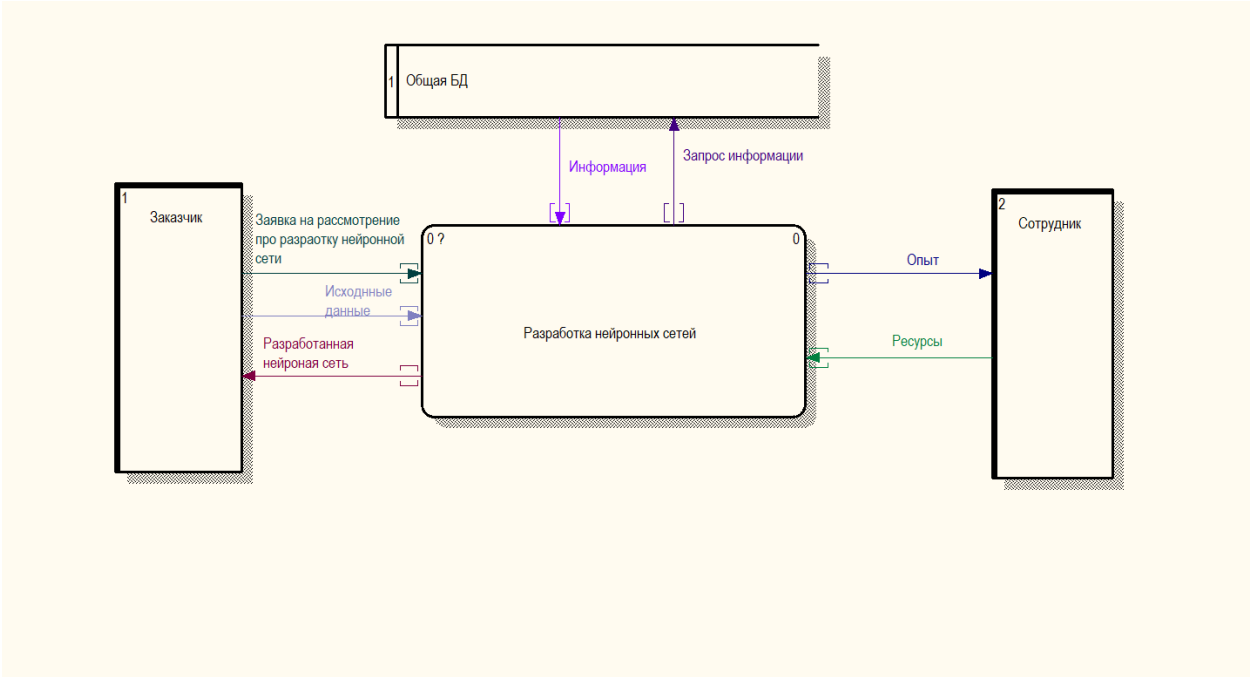


Рисунок 5 – Контекстная диаграмма «Разработка нейронных сетей» в нотации DFD

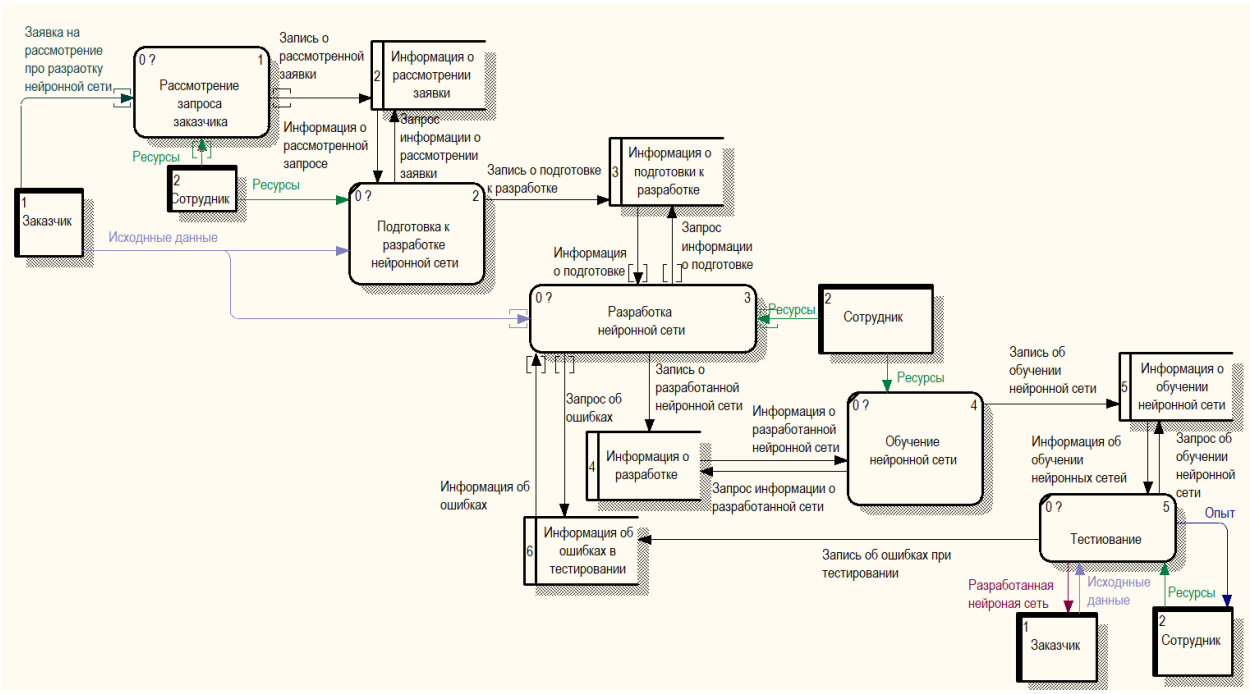


Рисунок 6 – Декомпозиция контекстной диаграммы «Разработка нейронных сетей» в методологии DFD

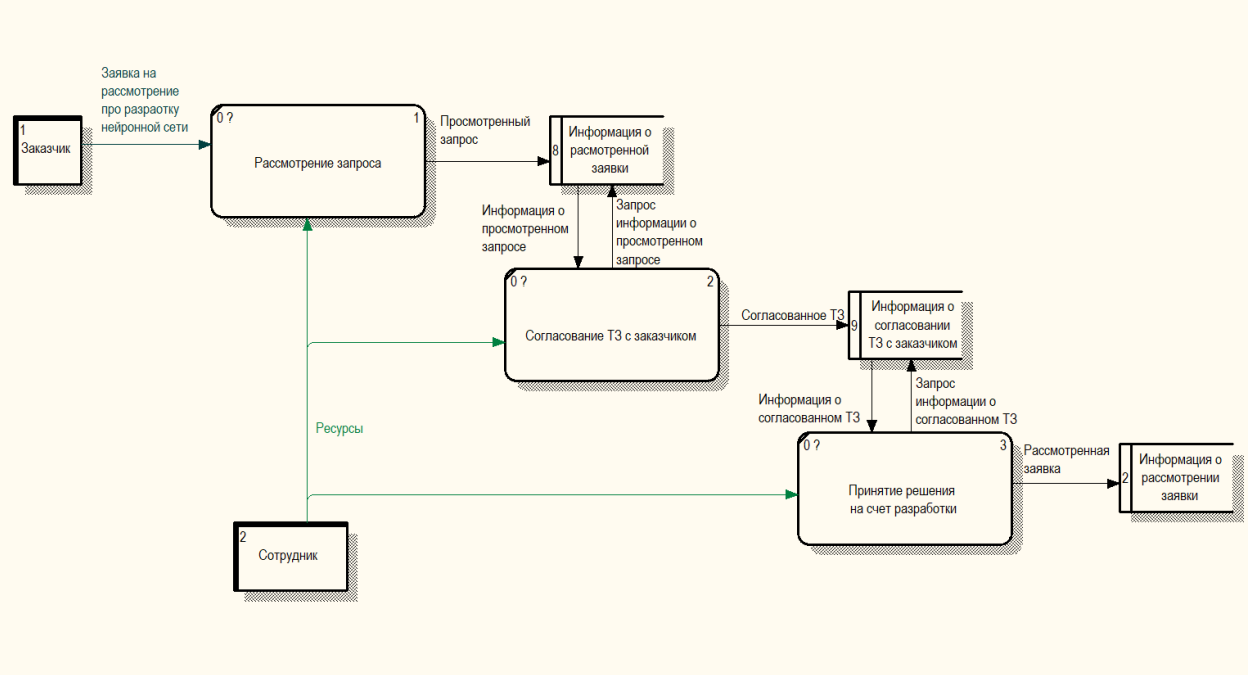


Рисунок 7 – Декомпозиция блока «Рассмотрение запроса заказчика» в методологии DFD

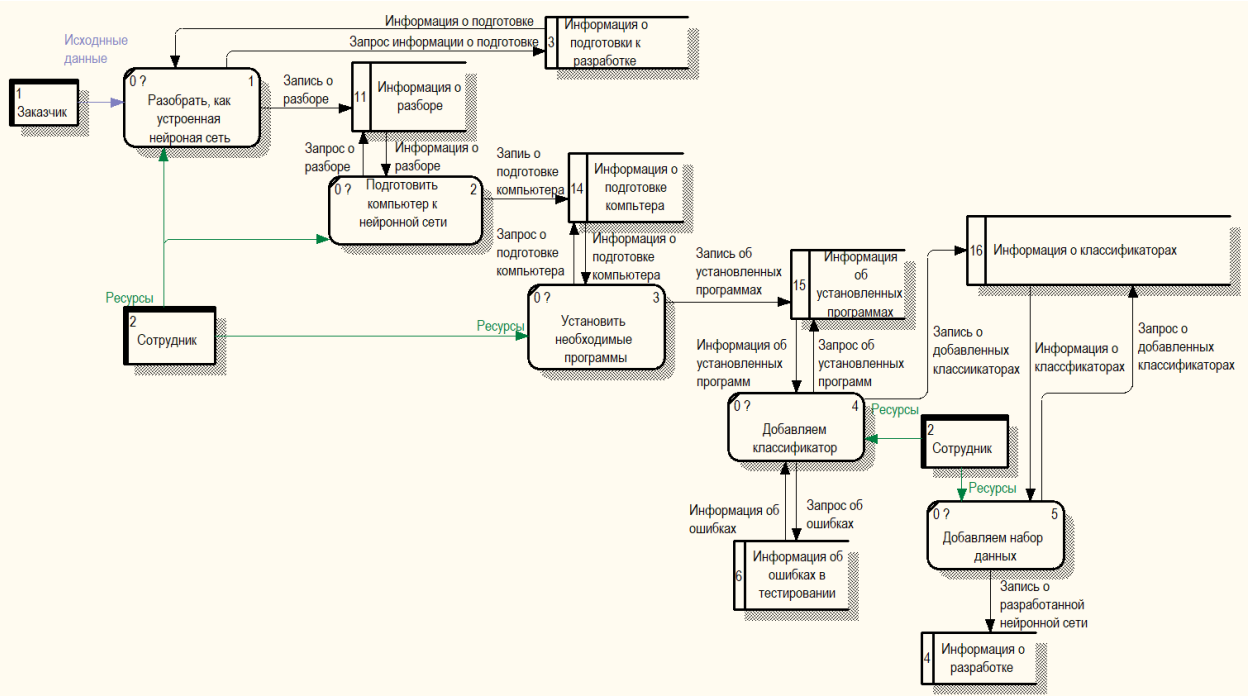


Рисунок 8 – Декомпозиция блока «Разработка нейронной сети» в методологии DFD

На данной модели отображается основной процесс (сама система в целом) и ее связи с внешней средой (внешними сущностями). Это взаимодействие показывается через потоки данных.

Внешние сущности изображают входы в систему и/или выходы из нее. Для процесса «Разработка нейронных сетей» были выделены такие внешние сущности, как:

- заказчик;
- сотрудники.

Стрелки (потоки данных) описывают движение объектов из одной части системы в другую.

Хранилище данных. В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое.

3 МОДЕЛЬ НОТАЦИИ IDEF3

Была разработана модель IDEF3 по предметной области **«Разработка нейронной сетей»** (Рисунок 9).

Методология IDEF3 позволяет декомпозировать работу многократно, т. е. работа может иметь множество дочерних работ. Возможность множественной декомпозиции отражается в нумерации работ: номер работы состоит из номера родительской работы, номера декомпозиции и номера работы на текущей диаграмме.

Слабые связи переходов изображаются сплошными одинарными стрелками.

Сильные связи переходов изображаются двойными однонаправленными стрелками.

Объектный поток – тип связи, обозначающийся двойными однонаправленными стрелками, значат, что выход исходного действия является входом конечного действия. Из этого, в частности, следует, что исходное действие должно завершиться, прежде чем конечное действие сможет начаться.

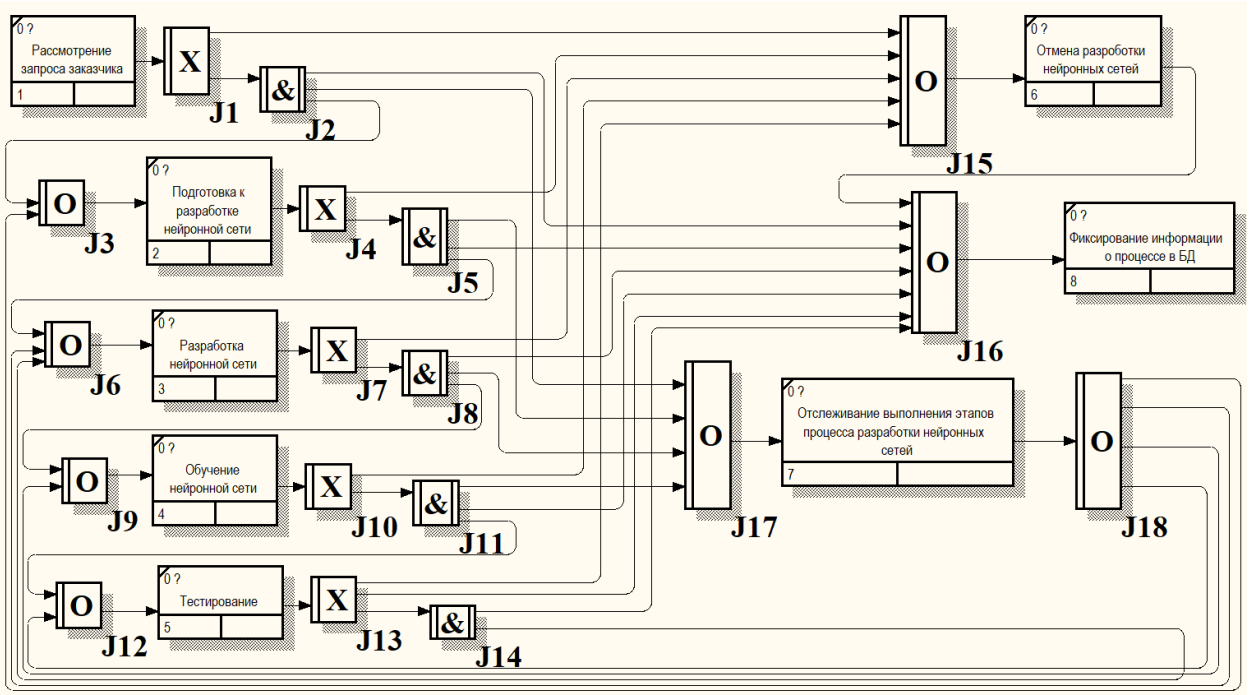


Рисунок 9 – Диаграмма «Разработка нейронных сетей» в методологии IDEF3

На данном рисунке представлена контекстная диаграмма процесса «Разработка нейронной сети», на 8 подпроцессов:

- Рассмотрение запроса заказчика;
- Подготовка к разработке нейронной сети;
- Отмена разработки нейронных сетей;
- Фиксирование информации о процессе в БД;
- Отслеживание выполнения этапов процесса разработки нейронных сетей;
- Разработка нейронной сети;
- Обучение нейронной сети;
- Тестирование.

4 ПРОЕКТИРОВАНИЕ НА ЯЗЫКЕ UML

Целью данной работы является освоение технологии проектирования информационных систем с позиции объектно-ориентированного проектирования на основе языка UML.

В процессе выполнения работы строятся диаграммы логического проектирования, не имеющие прямого отношения к языку программирования. Это диаграммы концептуального моделирования. Реализация диаграмм производится с помощью программного обеспечения Rational Rose.

4.1. Диаграмма прецедентов

Use case diagram (диаграммы прецедентов) – этот вид диаграмм, позволяющий создать список операций, которые выполняет система. Каждая такая диаграмма – это описание сценария поведения, которому следуют действующие лица (Actors).

Данный тип диаграмм используется при описании бизнес-процессов предметной области, определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области, и задачи, ими выполняемые.

Разработанная диаграмма прецедентов изображена на рисунке 10. На диаграмме находятся Use Case (Вариант использования/Прецедент), Actor (Действующее лицо). Так же были добавлены ассоциации и описания ко всем элементам диаграммы.

На Рисунке 10 представлена диаграмма для предметной области «Разработка нейронной сети».

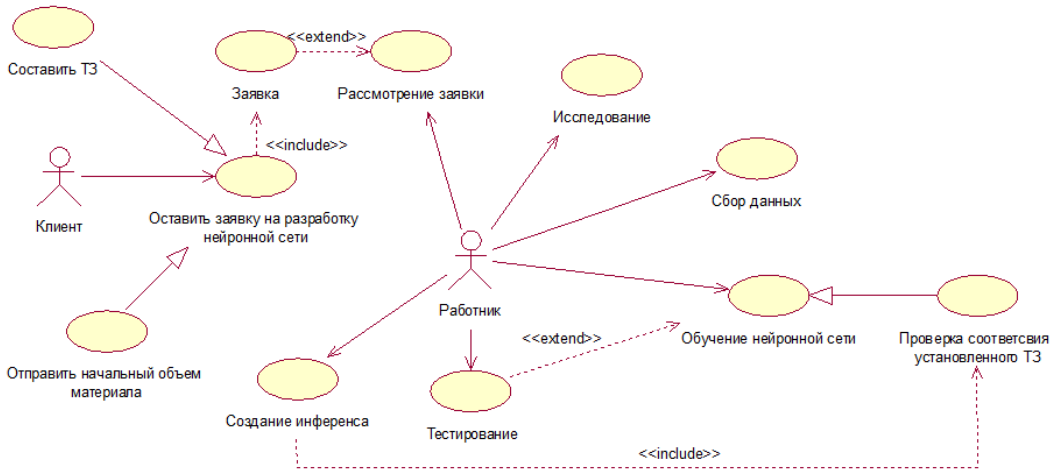


Рисунок 10 – Диаграмма прецедентов для «Разработки нейронных сетей»

4.2. Диаграмма классов

На данном этапе создается диаграмма классов, её создание состоит из следующих этапов: добавление классов, их расположение на рабочем пространстве, создание связей между ними, добавление атрибутов и операций. Происходит подробное описание операций и атрибутов, указываются их типы и названия.

Class diagram (диаграмма классов) – тип диаграмм, позволяющий создавать логическое представление системы, на основе которого создается исходный код описанных классов. Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов (Classes) и интерфейсов (Interfaces). Происходит подробное описание операций и атрибутов, а именно указываются их типы.

На данной диаграмме на Рисунке 11 используется 4 вида стереотипов: граничный класс (boundary), класс сущности (entity), управляющий класс (control) и интерфейс (interface). Каждый из этих классов содержит свои методы и атрибуты.

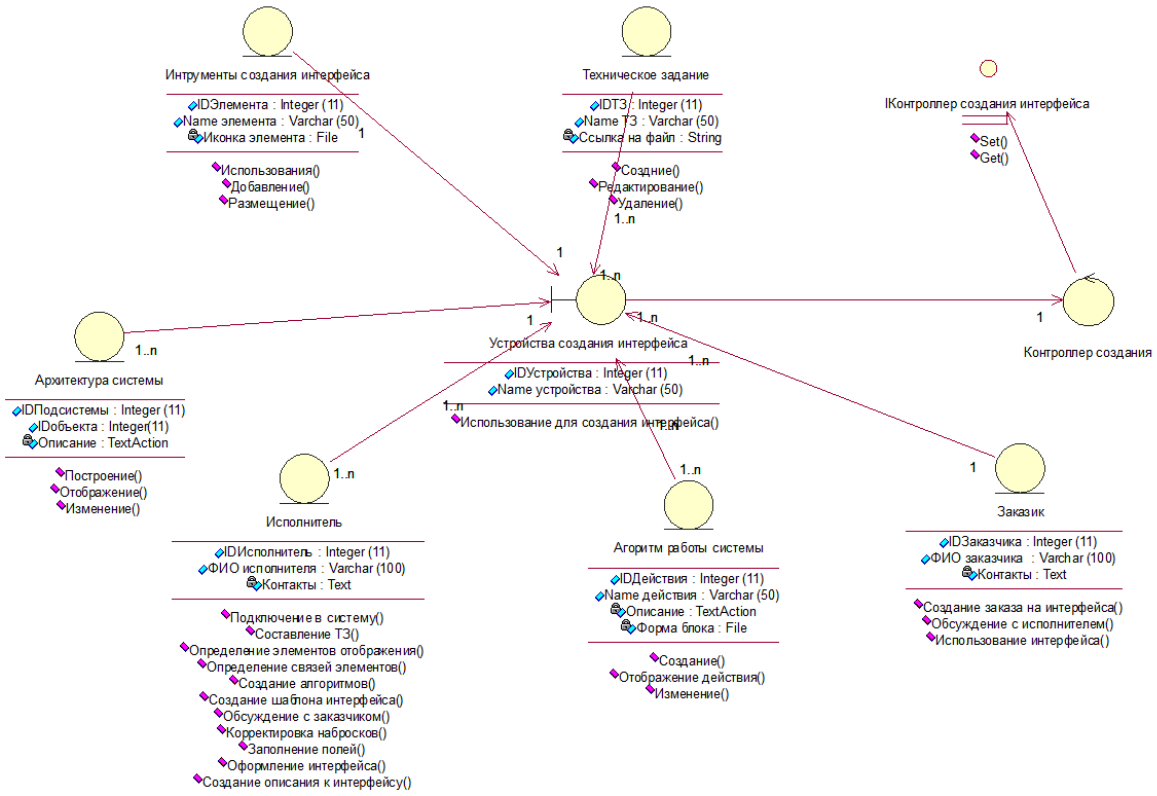


Рисунок 11 – Диаграмма классов для прецедента «Создание интерфейса»

4.3. Диаграмма кооперацій

Collaboration diagram (диаграммы кооперацій) – тип диаграмм, позволяющий описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта и типы этих сообщений.

Реализация данной диаграммы для прецедента «Создание интерфейса» представлена на Рисунке 12. На данной диаграмме также задействованы участники – клиент и работник (Actor), объекты (Objects) и сообщения с двумя видами свойств синхронизации:

- simple (Простое) – Данное сообщение выполняется в одном потоке управления. Это свойство задается добавляемому на диаграмму сообщению по умолчанию;
- return (Возврат) – Данное сообщение посылается клиенту после окончания выполнения вызова процедуры.

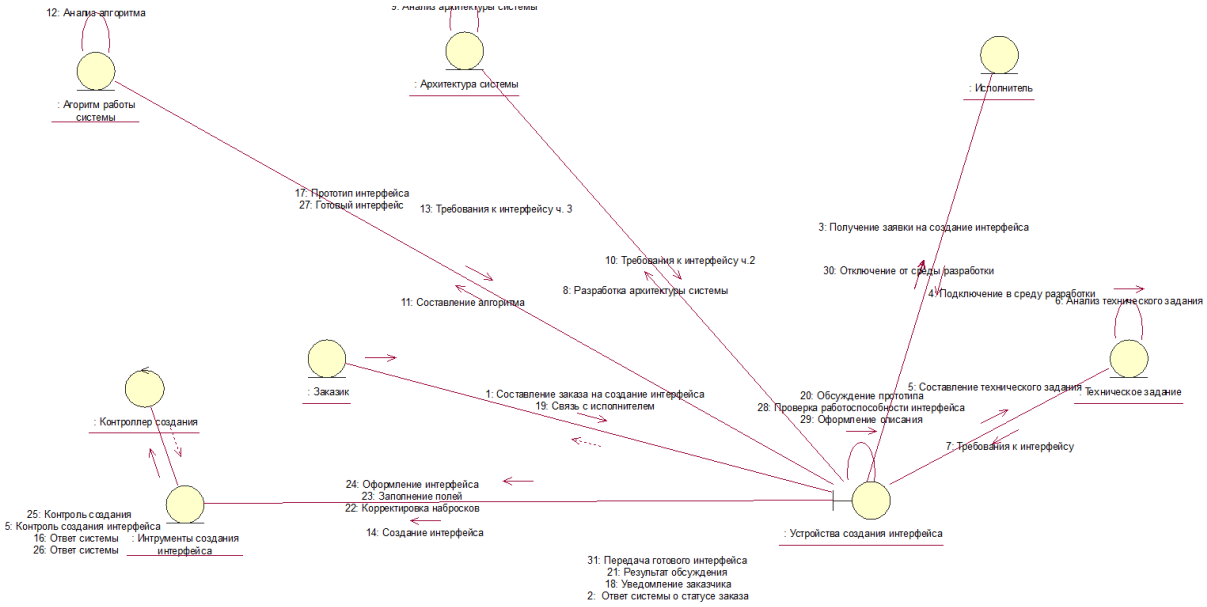


Рисунок 12 – Диаграмма кооперaций для прецедента «Создание интерфейса»

4.4. Диаграмма последовательности

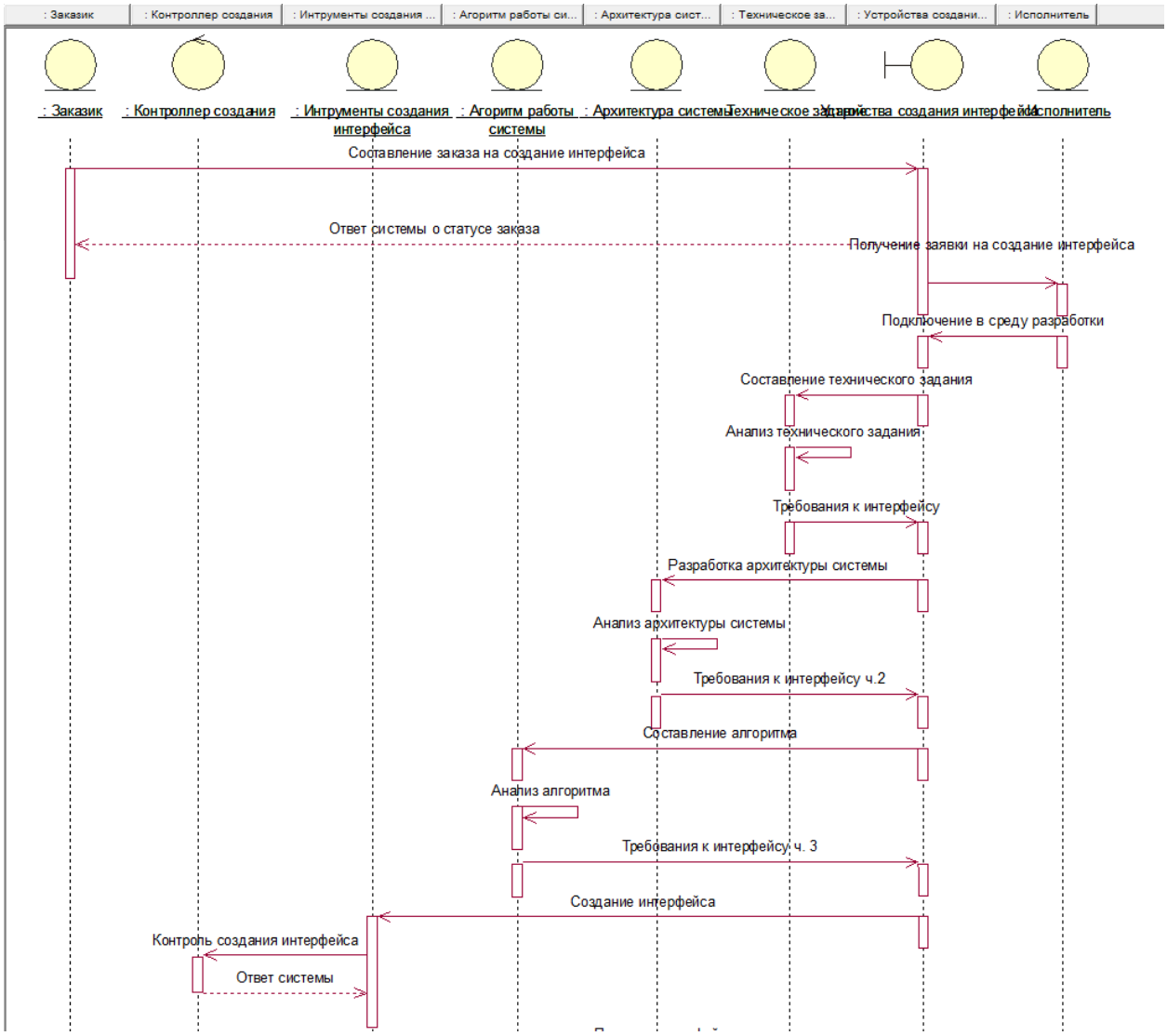
Sequence diagram (диаграммы последовательностей действий) – тип диаграмм, позволяющий отразить последовательность передачи сообщений между объектами. Этот тип диаграммы не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений.

На диаграмме на Рисунке 13 показана полная последовательность действий для выполнения прецедента «Создание интерфейса», начиная с включения программы, заканчивая получением готовой схемы. На ней представлены участники – клиент и работник (Actor) и объекты (Object). Также используется несколько свойств синхронизации сообщений:

- simple (Простое) – Данное сообщение выполняется в одном потоке управления. Это свойство задается добавляемому на диаграмму сообщению по умолчанию;
- return (Возврат) – Данное сообщение посылается клиенту после окончания выполнения вызова процедуры.

Согласно правилам построения диаграмм последовательности, начальное действие и конечное приходятся на одного актера, в данном

случае, клиента.



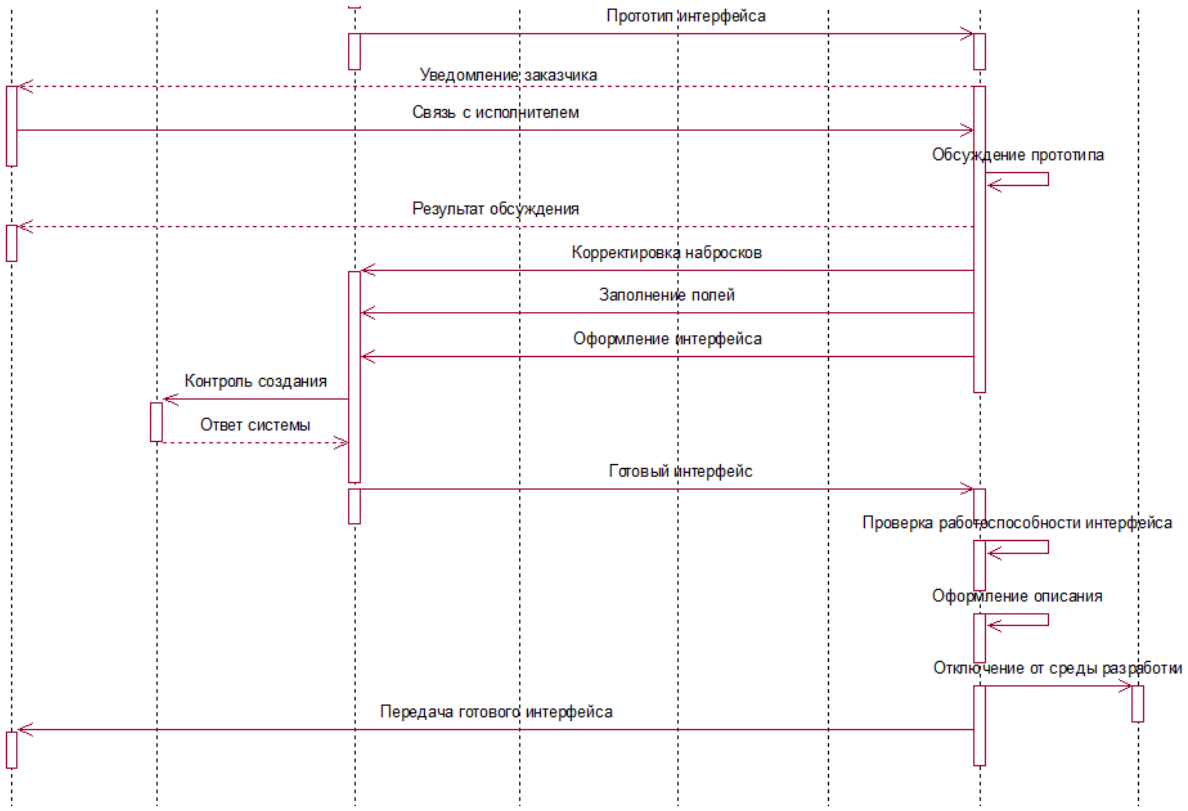


Рисунок 13 – Диаграмма последовательности для прецедента «Создание интерфейса»

4.5. Диаграмма состояния

Statechart diagram (Диаграмма состояний) – это тип диаграмм, предназначенный для отображения состояний объектов системы, имеющих сложную модель поведения.

Данный вид диаграммы для прецедента «Создание интерфейса» представлен на Рисунке 14. Диаграмма содержит в себе точку Начала и 3 точки Конца (Start State, End State, End State). Также данная диаграмма имеет 8 состояний, каждые из которых подключены друг к другу последовательно. Состояние может содержать только имя или имя и дополнительно список внутренних действий. Главное достоинство данной диаграммы – возможность моделировать условный характер реализации всех вариантов использования в форме изменения отдельных состояний разрабатываемой системы.

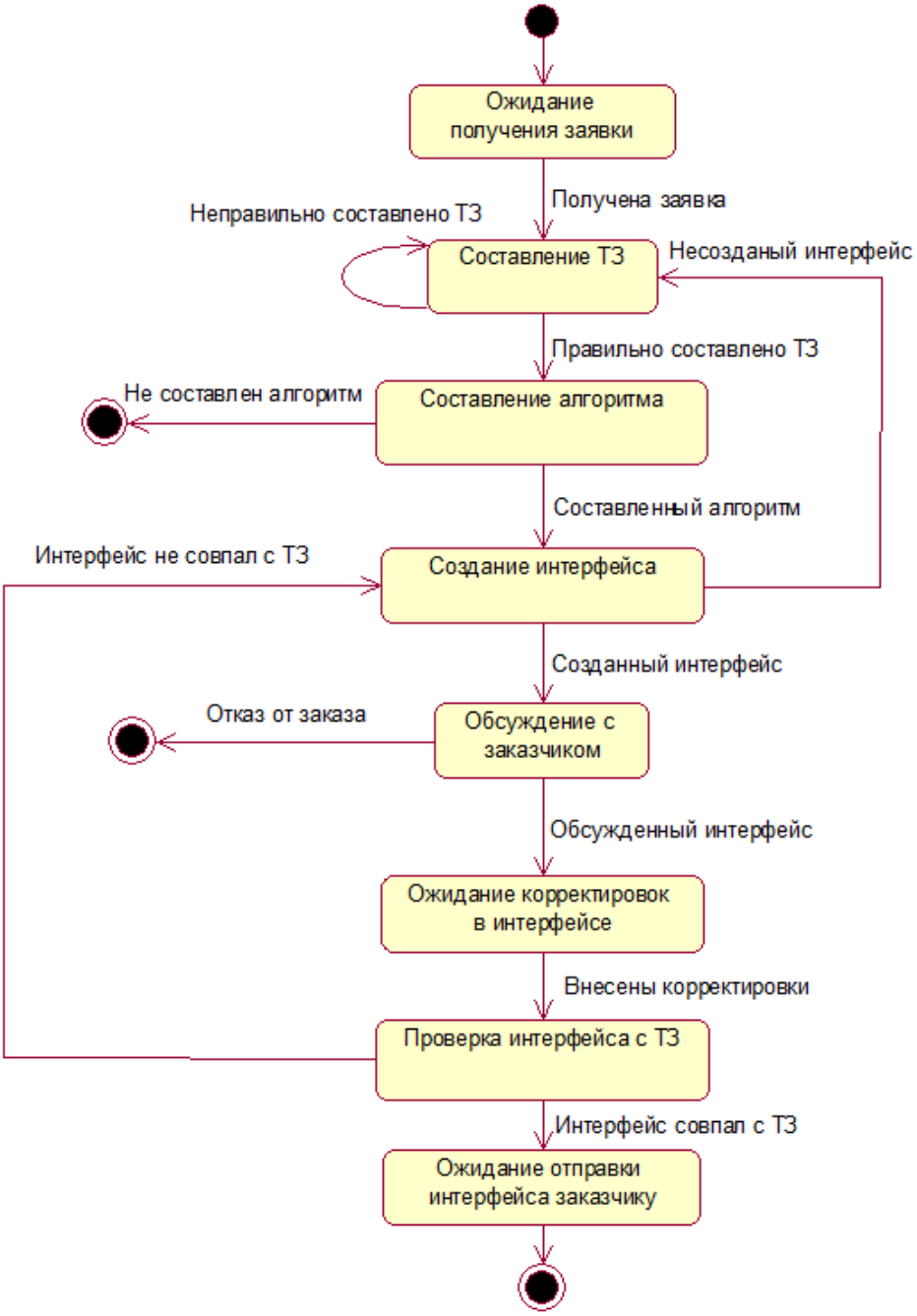


Рисунок 14 – Диаграмма состояний для прецедента «Создание интерфейса»

4.6. Диаграмма деятельности

Activity diagram (диаграмма деятельности) – тип диаграмм, используемый для отражения состояний моделируемого объекта, однако, основное назначение Activity diagram в том, чтобы отражать бизнес-процессы объекта. Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление, и синхронизацию процессов.

Данная диаграмма для прецедента «Создание интерфейса» представлена на Рисунке 15.

Рисунок 15 – Диаграмма деятельности для прецедента «Создание интерфейса»

4.7. Диаграмма развертывания

Demployment diagram (диаграмма развертывания) – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая

такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, можно понять, как система будет физически развернута на аппаратном обеспечении.

Диаграмма развертывания представлена на Рисунке 16. На данной диаграмме общая сеть связывает устройства 3ех работников (Компьютер работника №1, Компьютер работника №2, Компьютер работника №3).

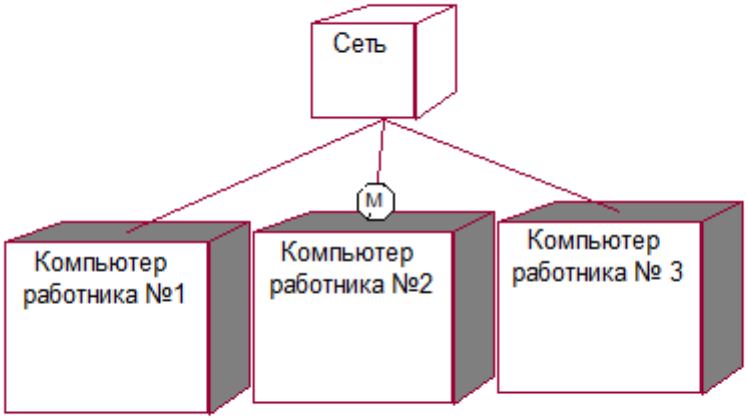


Рисунок 16 – Диаграмма развертывания разрабатываемой модели «Разработка нейронной сети»

4.8. Диаграмма компонентов

Component diagram (диаграмма компонентов) – тип диаграмм, предназначенный для распределения классов и объектов по компонентам при физическом проектировании системы.

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости. Основными графическими элементами диаграммы компонентов являются

компоненты, интерфейсы и зависимости между ними. Результат построения диаграммы представлен на Рисунке 17.

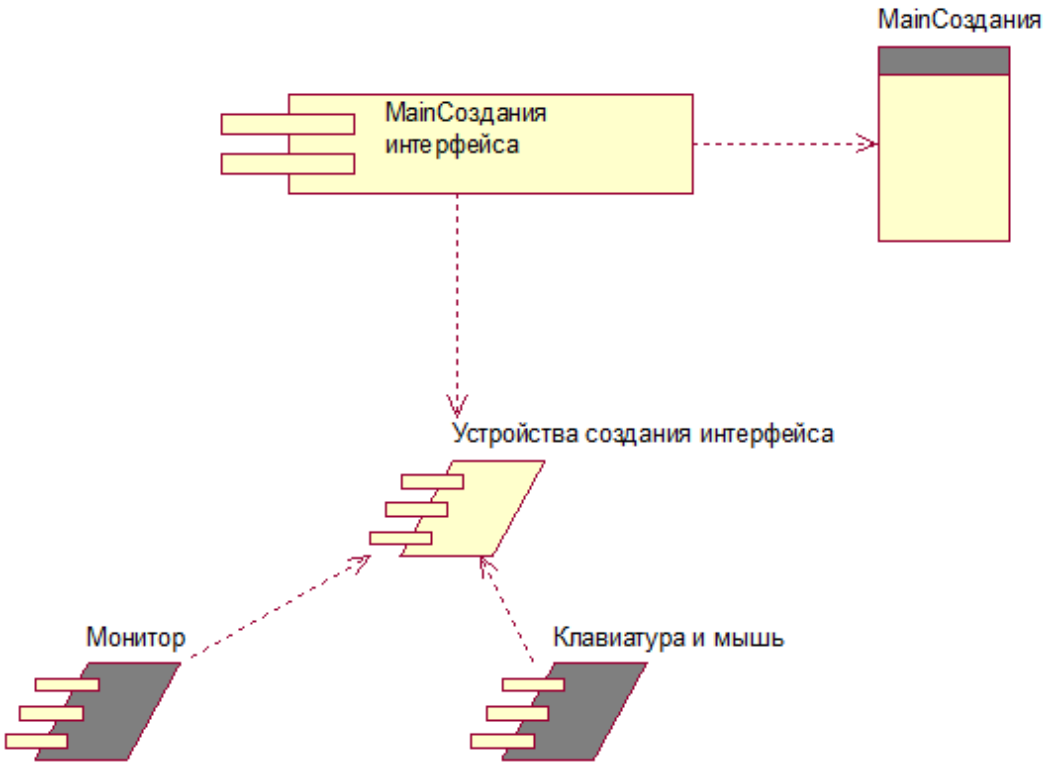


Рисунок 17 – Диаграмма компонентов разрабатываемой модели «Разработка нейронной сети»

ЗАКЛЮЧЕНИЕ

В ходе выполнения практик научился реализовывать ряд диаграмм необходимых для построения базы данных, а именно:

- IDEF0;
- DFD;
- IDEF3.

Также научился реализовывать следующие UML диаграммы внутри промышленных рабочих комплексов:

- Use case diagram (диаграммы прецедентов);
- Class diagram (диаграммы классов);
- Collaboration diagram (диаграммы кооперацій);
- Sequence diagram (диаграммы последовательностей действий);
- Statechart diagram (Диаграмма состояний);
- Activity diagram (диаграмма деятельности);
- Demployment diagram (диаграмма развертывания);
- Component diagram (диаграмма компонентов).

Мной были приобретены знания об основных принципах и правилах моделирования вышеперечисленных диаграмм. На основе полученных знаний и схем, была создана полная модель информационной области «Разработка нейронной сети», которая освящает этот процесс с разных сторон, раскрывая его под разными точками зрения.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. «Методические указания по DFD»//РТУ МИРЭА, 2023 г. – 8 с
2. «Методические указания по IDEF0»//РТУ МИРЭА, 2023 г. – 43 с.
3. «Методические указания по IDEF3»//РТУ МИРЭА, 2023 г. – 13 с
4. «Методические указания по языку UML»//РТУ МИРЭА, 2023 г. – 142 с