Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Дисциплина: Объектно-ориентированное программирование и проектирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ УЧЕТА МАШИНОЧАСОВ

БГУИР КП 1-53 01 02 06 029 ПЗ

Студент гр. 820601 А.Р. Шведов
Руководитель М.П. Ревотюк

Минск 2020

РЕФЕРАТ

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ УЧЕТА МАШИНО-ЧАСОВ: курсовой проект / А.Р. Шведов – Минск: БГУИР, 2019, – п.з. – 37 с., чертежей (плакатов) – 3 л. формата А1.

Курсовой проект представлен в виде графического материала на 2 листах формата А1 и пояснительной записки на 39 страницах, состоящей из 3 разделов.

Пояснительная записка к курсовому проекту состоит из введения, 3 разделов, включающих, анализ технического задания и создания спецификации к нему, проектирования системы, а именно создание различных диаграмм, реализацию системы, заключение, список использованных источников и приложение, содержащее листинг кода отдельных классов.

Для разработки проекта системы был выбран объектно-ориентированный язык проектирования UML. В первом разделе выполняется анализ предметной области и рассмотрение других языков моделирования, с помощью которых можно спроектировать данную систему. Во втором разделе описывается проектирование и моделирование системы. Рассматривается разработка диаграмм в теории. В третьем разделе представлена реализация всех диаграмм для данной системы. Заключение включает основные выводы по работе.

Проект представляет собой описание создания системы диагностики ЭВМ, её взаимодействие с окружающей средой: техническим персоналом, ЭВМ, базой данных. В нем присутствуют различные диаграммы, каждая из которых демонстрирует определенный аспект проекта.

В результате выполнения курсовой работы была спроектирована модель системы, написан глоссарий, а также составлены чертежи.

Результаты, полученные в ходе курсового проектирования, могут использоваться на производстве, где необходима регулярное тестирование различных ЭВМ в целях контроля их работоспособности. Поскольку проект создавался обобщенно, то его легко видоизменить. Созданная модель является также хорошим примером создания UML диаграмм и может служить примером написания моделей других проектов.

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
_____2020 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Шведову Андрею Робертовичу*

1. Тема проекта: Объектно-ориентированный анализ и проектирование программного обеспечения. Программная система учета машино-часов.
2. Срок сдачи студентом законченного проекта: *27 мая 2020г.*
3. Исходные данные к проекту: *реализовать систему автоматического контроля использования и степени загрузки персональных ЭВМ класса для диагностики непригодности конкретных ЭВМ к выполнению конкретных работ в классе Rational Rose. Назначение разработки: проектирование системы автоматического контроля использования и степени загрузки ЭВМ.*
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке):

Введение;

1. анализ технихнического задания на курсовое проектирование;

2. проектирование системы;

3. реализация системы;

заключение;

список использованных источников;

приложения.

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков):

1. Диаграмма вариантов использования (ПД, формат А1);

2. Диаграмма классов (ПД, формат А1).

6. Консультант по проекту (с обозначением разделов проекта): *М. П. Ревотюк*

7. Дата выдачи задания: *17 февраля 2020 г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

раздел 1 к 15.03 – 15 %;

раздел 2 к 15.04 – 50 %;

раздел 3 к 15.05 – 15 %;

оформление пояснительной записки и графического материала к 20. 05 – 20 %;

*Защита курсового проекта с **23.05** по **05.06.2020**г.*

Руководитель *М.П. Ревотюк*

(подпись)

Задание принял к исполнению *А.Р. Шведов*

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Определив требования к программному обеспечению, разработчик получает согласованный четкий план действий, график оплат и сроков, сокращает время разработки и повышает её качество.

В результате проектирования получается техническое задание с понятной и однозначной для заказчика и исполнителя (руководителя проекта, программистов, тестировщиков, дизайнеров и других участников процесса разработки) иллюстрацией ответов на вопросы:

- что делать (описание продукта, функционала, пользователей);
- как делать (архитектура);
- как проверить, что цель достигнута (тестирование, критерии оценки).

В силу специфичности производства ПО для коммерческих предприятий, мы используем язык проектирования *UML*. Он отлично подходит для данной задачи, так как обладает широким спектром различных инструментов.

UML находится в процессе стандартизации, проводимом консорциумом *OMG (Object Management Group)*, в настоящее время он принят в качестве стандартного языка моделирования и получил широкую поддержку. *UML* принят на вооружение практически всеми крупнейшими компаниями – производителями программного обеспечения (*Microsoft, IBM, Hewlett-Packard, Oracle, Sybase* и др.). Кроме того, практически все мировые производители *CASE*-средств, помимо *Rational Softwar*e (*Rational Rose*), поддерживают *UML* в своих продуктах (*Paradigm Plus (CA), System Architect (Popkin Software), Microsoft Visual Modeler* и др.) [1].

Преимущества *UML* среди других языков:

- объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках;
- позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;
- диаграммы *UML* сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом;
- расширяет и позволяет вводить собственные текстовые и графические стереотипы, что способствует его применению не только в сфере программной инженерии.

Для разработки проекта используется среда *Rational Rose*. 5 основных элементов интерфейса *Rose* – это браузер, окно документации, панели инструментов, окно диаграммы и журнал (*log*).[6]

Задачами курсового проекта (работы) как этапа подготовки к дипломному проектированию являются:

– освоение, углубление и обобщение знаний, полученных студентами в процессе обучения;

– приобретение практических навыков и развитие творческих подходов к решению конкретной инженерной или инженерно-экономической задачи;

– формирование умений использовать справочную литературу, нормативную, правовую, нормативно-техническую документацию, осуществлять патентный поиск;

– приобретение навыков по оформлению текстовой и графической документации согласно требованиям государственных стандартов и стандарта предприятия Дипломные проекты (работы). Общие требования. СТП 01-2017.

Целью курсового проекта является объектно-ориентированный анализ и проектирование программного обеспечения.

Темой курсового проекта является проектирование системы диагностики ЭВМ. Эта тема актуальна, потому что на сегодняшний день ни одно крупное предприятие не обходится без комплекса ЭВМ различных конфигураций, типов и назначений. С целью описания работы системы диагностики ЭВМ и дальнейшей ее оптимизации и был создан курсовой проект на данную тему.

1. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

1.1 Постановка задачи проектирования

Предметом исследования является система учета машино-часов. В курсовом проекте рассматриваются возможности контроля использования и степени загрузки ЭВМ. Цели внедрения системы учета машино-часов:

- диагностика и контроль ЭВМ;
- подсчет часов работоспособности ЭВМ;
- выявление неисправностей ЭВМ;
- замена непригодных ЭВМ.

Задачей данного курсового проекта является разработка модели программного обеспечения для диагностики различных ЭВМ. Диагностика ЭВМ является гарантом стабильности их работы, защитой от перегрузки. Она позволяет вовремя сообщать персоналу о непригодности ЭВМ и заменять их, что уменьшает вероятность отказа ЭВМ.

Система учёта машино-часов выступает в роли действующего лица и может осуществлять следующие операции (они же варианты использования):

- определить ЭВМ системы;
- подсчитать часы работоспособности этой ЭВМ;
- определить задачи ЭВМ;
- провести диагностику;
- по результатам диагностики составить отчет;
- проверить выполнение поставленных задач;
- занести отчёт в базу данных.

При обнаружении неполадок в работе или при несоответствии ЭВМ её задаче система осуществляет взаимодействие с другим действующим лицом-техническим персоналом (например, инженеры или системные администраторы) посредством сообщения о непригодности ЭВМ. Технический персонал может выполнить следующие действия: провести проверку ЭВМ, проверить диагностику, заменить непригодные ЭВМ, поставить задачу ЭВМ.

При запросе на проведение проверки конкретной ЭВМ от технического персонала, система создает объект класса *Диагностика*, который обращается к объектам классов *ЭВМ*, *КонфигурацияЭВМ*, при вызове их собственных методов *СборДанныхОКонфигурацииЭВМ(), СборДанныхОКлассахЭВМ()* и других осуществляется сбор необходимых для проведения диагностики данных. Метод *ОбработкаПолученныхДанных()* позволяет проверить

собранные данные о ЭВМ сети. Далее создаётся объект класса Результаты Дмагностики с данными о диагностике, затем объект класса Отчёт, который при помощи своего метода *ДанныеРезультатовДиагностики()* обращается к объекту класса *Результаты Диагностики*, обрабатывает полученные из него данные и выводит их в форме отчета.

Далее система проверяет, справляется ли ЭВМ со своей задачей на основе полученных в результате диагностики данных. В случае непригодности ЭВМ конкретного класса для конкретной задачи, система посылает сообщение о непригодности техническому персоналу.

Отчёт заносится системой контроля машино-часов в реестр.

При получении техническим персоналом сообщении о непригодности ЭВМ возможно заменить ЭВМ или их части на основе данных отчета и сообщения о непригодности. Также возможно поставить новую задачу для конкретной ЭВМ и провести повторную диагностику (используется связь <<*extend*>> Проведение дополнительных тестов).

1.2 Составление глоссария проекта

Для описания терминологии предметной области составим глоссарий проекта. Он может быть использован как неформальный словарь данных системы. Глоссарий приведен в таблице 1.

Таблица 1 – Глоссарий проекта.

Техническая система	Искусственно созданная система, предназначенная для удовлетворения определенной потребности. Существует как изделие производства, как устройство, потенциально готовое совершить полезный эффект, как процесс взаимодействия с компонентами окружающей среды, в результате которого образуется полезный эффект
Технический персонал	Физические лица, выполняющие работы по техническому обслуживанию, ремонту, монтажу, диспетчерскому контролю, осмотру, управлению ЭВМ.
Техническая диагностика	Процесс определения технического состояния объекта. Обеспечение безопасности, функциональной надёжности и эффективности работы технического объекта, а также сокращение затрат на его техническое обслуживание и уменьшение потерь от простоев в

	результате отказов и преждевременных выводов в ремонт.
ЭВМ	Комплекс технических средств, в котором основные функциональные элементы (логические, запоминающие, индикационные и др.) выполнены на электронных элементах, предназначенных для автоматической обработки информации в процессе решения вычислительных и информационных задач.
Конфигурация	Определенный набор комплектующих и характеристик ЭВМ. Например аппаратное и программное обеспечения, прошивка, сопроводительная документация.

1.3 Описание дополнительных спецификаций к проекту

Функциональные возможности

Система должна обеспечивать многопользовательский режим работы. Она должна иметь возможность работать одновременно с несколькими пользователями. Так же система должна позволять пользователю следить за каждым шагом выполнении своей работы.

Удобство использования

Пользовательский интерфейс должен быть *Windows* 7-10, *Mac OS*-совместимым. Так же если система будет создана для мобильных приложений, интерфейс должен быть *Android, IOS* - совместим.

Надежность

Система должна быть в работоспособном состоянии 24 часа в день 7 дней в неделю. *SLA* – 99% (не более 14 мин простоя в сутки). Необходимо наличие backup для серверов базы данных.

Производительность

Система должна поддерживать одновременную работу с 20 ЭВМ различных классов и конфигураций, а также с 3 пользователями.

Безопасность

Система не должна позволять пользователям, у которых нет доступа к системе, выполнять какие-либо действия, изменять результаты диагностики, а также данные отчета о проведении диагностики. Должна проводить контроль поставленных техническим персоналом задач на корректность. Система должна обеспечивать безопасность хранения данных и недоступность просмотра, редактирования и удаления информации сторонними лицами.

Проектные ограничения

Система должна быть интегрирована с существующей системой хранения информации о ЭВМ системы, функционирующей на основе реляционной СУБД.

2. ПРОЕКТИРОВАНИЕ и МОДЕЛИРОВАНИЕ СИСТЕМЫ

2.1 Создание модели вариантов использования

Созданная модель диаграммы вариантов использования на основе полученного варианта задания, представлена на рисунке 1.

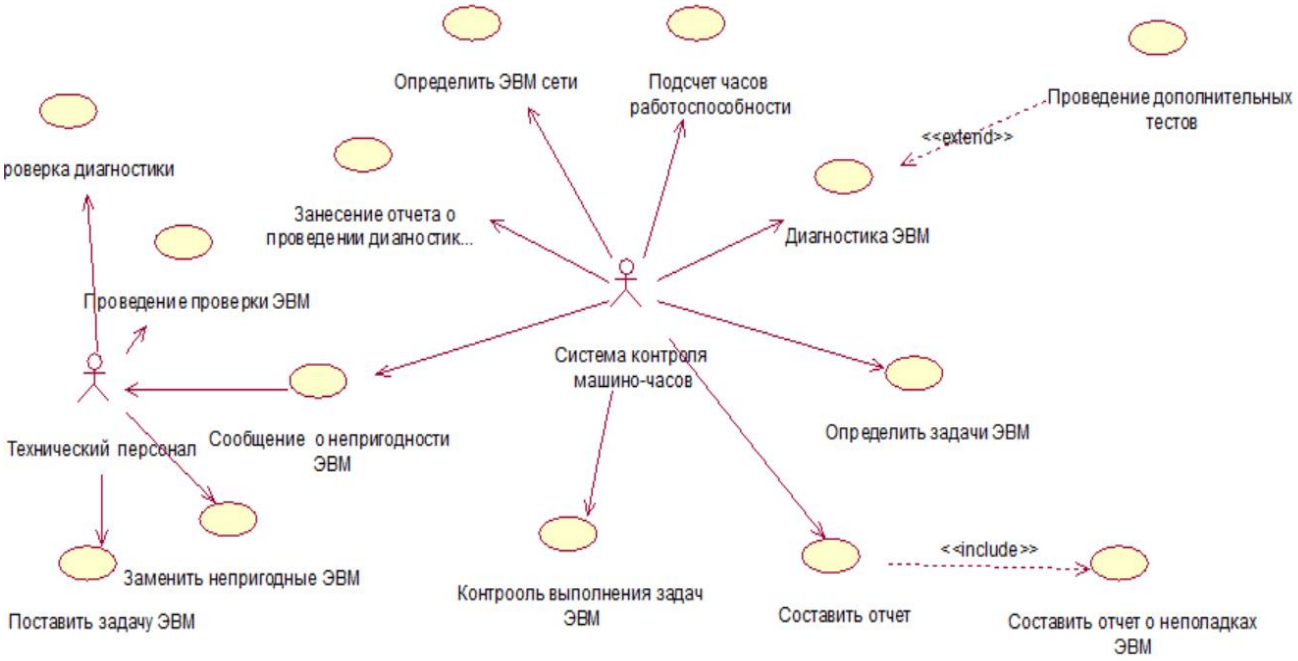

Рисунок 1 – Диаграмма вариантов использования

Примечание. Диаграммы вариантов использования применяются при бизнес-анализе для моделирования видов работ, выполняемых организацией, и для моделирования функциональных требований к ПС при ее проектировании и разработке. Построение модели требований при необходимости дополняется их текстовым описанием. При этом иерархическая организация требований представляется с помощью пакетов *use-cases* [2].

На диаграмме вариантов использования показаны 2 действующих лица:

1. Технический персонал-обращается к системе для проверки ЭВМ, ставит и изменяет задачи ЭВМ, заменяет непригодные ЭВМ, посылает запрос на повторное проведение диагностики .

2. Система контроля машино-часов осуществляет сбор данных о ЭВМ и их работоспособности, определяет задачу, проводит диагностику ЭВМ, составляет отчёт, заносит его в реестр. В случае непригодности ЭВМ посылает

Техническому персоналу сообщение о непригодности. При запросе на повторную диагностику проводит дополнительные тесты.

В курсовом проекте представлены 15 вариантов использования, выполняемых моделируемой системой:

- проверка диагностики;
- проведение проверки ЭВМ;
- поставить задачу ЭВМ;
- заменить непригодные ЭВМ;
- занесение отчета о проведении диагностики в реестр;
- сообщение о непригодности;
- контроль выполнения задач ЭВМ;
- определить ЭВМ сети;
- подсчет часов работоспособности;
- диагностика ЭВМ;
- определить задачи ЭВМ;
- составить отчет;
- составить отчёт о неполадках ЭВМ;
- проведение дополнительных тестов;
- проведение диагностики.

На диаграмме описаны основные требования к системе. С помощью данной диаграммы можно понять, каким функционалом должно обладать программное обеспечение для системы контроля машино-часов, какие действующие лица могут осуществлять взаимодействие с ней. На ней также описаны возможные варианты ответных действий системы. Это поможет как при составлении последующих диаграмм и моделировании всей системы, так и при написании кода для программного продукта.

2.2 Добавление описаний к вариантам использования

Вариант использования «Определить ЭВМ сети»

Краткое описание

Данный вариант использования описывает определение ЭВМ сети системой контроля машино-часов.

Основной поток событий

Данный вариант использования начинает выполняться при поступлении от пользователя запроса на диагностику. Запрос поступает с кодом ЭВМ сети, по коду система определяет ЭВМ.

Альтернативные потоки

При неправильном введении просит ввести корректный код ЭВМ.

Предусловия

Перед этим должен был выполняться вариант использования *«Проверка диагностики»* или *«Проведение проверки ЭВМ»*.

Постусловия

При неправильном введении на экран выводится сообщение о повторном введении кода.

Вариант использования «Подсчёт часов работоспособности»

Краткое описание

Данный вариант использования описывает подсчёт часов работоспособности системой контроля машино-часов.

Основной поток событий

Данный вариант использования начинает выполняться при обращении к нему системы. Т.к. ЭВМ системы определена, для неё из базы данных считываются часы работоспособности.

Альтернативные потоки

Отсутствуют.

Предусловия

Должна быть определена ЭВМ сети (вариант использования *«Определить ЭВМ сети»).*

Постусловия

Возвращает пользователю количество часов работоспособности.

Вариант использования «Диагностика ЭВМ»

Краткое описание

Данный вариант использования описывает процесс диагностики ЭВМ системой контроля машино-часов.

Основной поток событий

Данный вариант использования начинает выполняться, когда технический персонал совершил обращение к системе учёта машино-часов для проведения диагностики ЭВМ.

Он включает в себя проверку корректности работы всех каналов и компонентов ЭВМ, определение назначения и конфигурации ЭВМ, сохранение результатов проведения диагностики в БД Результаты Диагностики.

Альтернативный поток А1

Если технический персонал послал запрос на проверку предыдущей диагностики, то после проведения диагностики будут проведены дополнительные тесты- переход к варианту использования «Проведение дополнительных тестов».

Альтернативный поток А2

В ходе диагностики были выявлены неполадки – переход к варианту использования «Сообщение о неполадках ЭВМ».

Предусловия

Должна быть определена ЭВМ сети (вариант использования *«Определить ЭВМ сети»).*

Постусловия

Если вариант использования выполнен успешно, результаты диагностики сохраняются в сущности (базе данных) Результаты Диагностики, на экран выводится сообщение об успешном проведении диагностики. В противном случае техническому персоналу оправляется файл с сообщением об ошибке.

Вариант использования «Проведение дополнительных тестов»

Краткое описание

Данный вариант использования описывает процесс проведения дополнительных тестов.

Основной поток событий

Данный вариант использования начинает выполняться, когда технический персонал обращается к системе с целью проверки предыдущей диагностики. Он включает себя подсчёт нагрузки на ЭВМ, среднего времени решения задач, время простоя и другие показатели.

Альтернативные потоки

Отсутствуют

Предусловия

1. Пользователь обратился к системе (вариант использования *«Проверка диагностики»).*
2. Должна быть определена ЭВМ сети (вариант использования *«Определить ЭВМ сети»).*

Постусловия

Передаёт результаты повторной диагностики объектам в вариант использования Диагностика ЭВМ.

Вариант использования «Составить отчёт»

Краткое описание

Данный вариант использования описывает поведение системы при выборе пользователем варианта использования - Составить отчёт.

Основной поток событий

Данный вариант использования начинает выполняться, когда пользователь выбирает вариант – *Составить отчёт*. По результатам диагностики составляется текстовый файл – объект класса *Отчёт*.

Альтернативный поток А1

При наличии в Результатах Диагностики ошибок автоматически переходит на вариант использования *«Составление отчёта о неполадках»*.

Альтернативный поток А2

Если диагностика не была проведена, на экран выведется сообщение с просьбой сначала провести диагностику.

Предусловия

Ранее должен был выполниться вариант использования *«Проведение диагностики»*.

Постусловия

При неудачном выполнении варианта использования переходит к выбору пользователем списка действий. При удачном завершении варианта использования предоставляет отчёт пользователю.

2.3 Создание диаграммы последовательности и диаграммы кооперации

Данный этап разработки модели программного обеспечения системы контроля машино-часов будет сопровождаться созданием *диаграмм взаимодействия*, а в частности, диаграммы последовательности «Диагностика» и диаграммы кооперации «Диагностика». Этот вид диаграмм предназначен для моделирования отношений между объектами (ролями, классами, компонентами) системы в рамках одного варианта использования. В нашем случае в рамках проведения диагностики ЭВМ.

Диаграмма взаимодействия отражает следующие аспекты проектируемой Системы:

- обмен сообщениями между объектами (в том числе в рамках обмена сообщениями со сторонними Системами);
- ограничения, накладываемые на взаимодействие объектов;
- события, инициирующие взаимодействия объектов.

Диаграмма последовательности предназначена для моделирования взаимодействия объектов Системы во времени, а также обмена сообщениями между ними [3]. На диаграмме последовательности для варианта использования *«Диагностика»* представлен актер *«Технический персонал»*, десять сообщений и следующие объекты:

- *Database «Client»* (класс Диагностика_Диагностика).
- *Form Order* (класс Конфигурация ЭВМ).
- *Order Manager* (класс OrderManager).
- *Order #1* (класс ЭВМ).
- *Items* (класс DB_Результаты Диагностики).
- Присутствует само-делегирование.

Данная диаграмма представлена на рисунке 2.

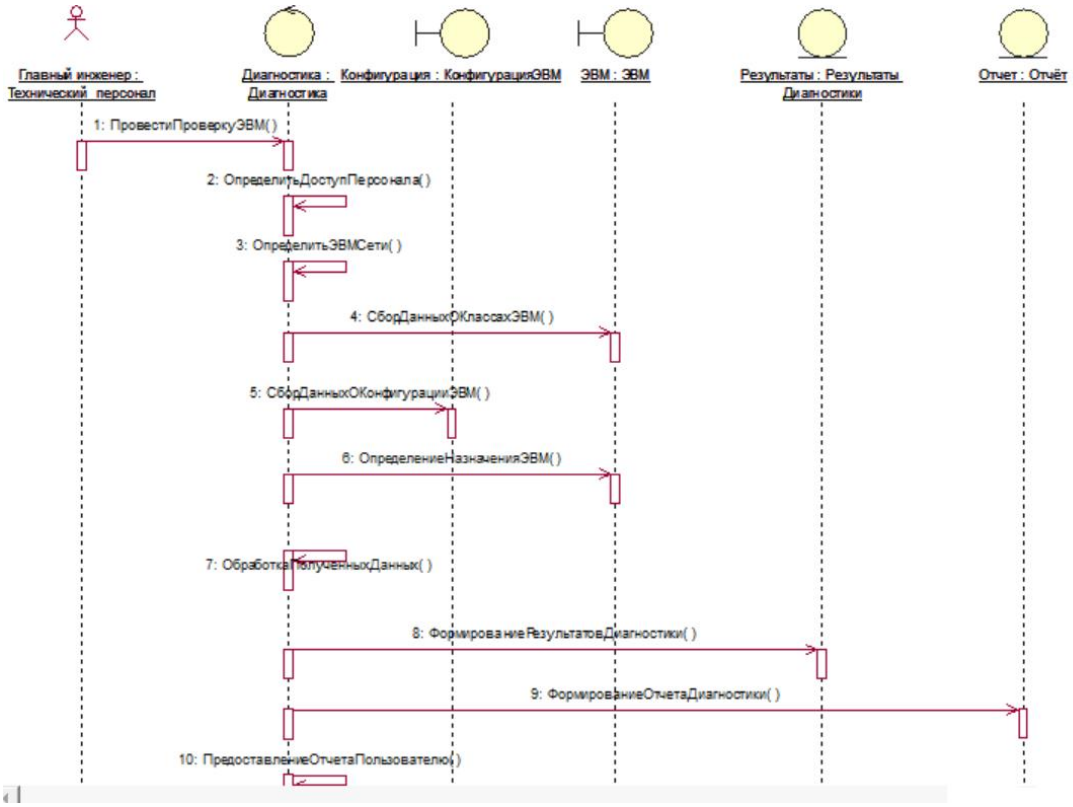

Рисунок 2 – Диаграмма последовательности «Диагностика»

На данной диаграмме представлены:

- информационное сообщение – сообщение, снабжающее объект–получатель некоторой информацией для обновления его состояния (провести проверку ЭВМ- Сообщение 1);

- сообщение-запрос – это сообщение, запрашивающее выдачу некоторой информации об объекте–получателе (запрос информации от персонала - Сообщение 2,3);
- императивное сообщение – это сообщение, запрашивающее у объекта–получателя выполнение некоторых действий (ожидание действия - Сообщение 12 [2];
- само-делегирование – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни [3] (на данной диаграмме Сообщения 2, 3, 7, 10).

Диаграмма кооперации имеют возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии. То есть они отображают поток событий через конкретный сценарий варианта использования. Но в отличие от диаграмм последовательности больше внимания заостряют на связях между объектами.

На кооперативной диаграмме так же, как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность указывается путем нумерации сообщений.

Диаграмма кооперации «Диагностика» была создана автоматически с помощью нажатия клавиши *F5* на основе уже построенной диаграммы последовательности «Диагностика».

При необходимости редактирования порядка следования сообщений из двух диаграмм следует выбрать диаграмму последовательности. Так как в этом случае достаточно нажать левую кнопку мыши на стрелке соответствующего сообщения и, не отпуская её, перетащить вверх или вниз данное сообщение. Кроме того, имеется возможность добавления новых классов на диаграмму с произвольным дальнейшим их размещением.

Данная диаграмма представлена на рисунке 3.

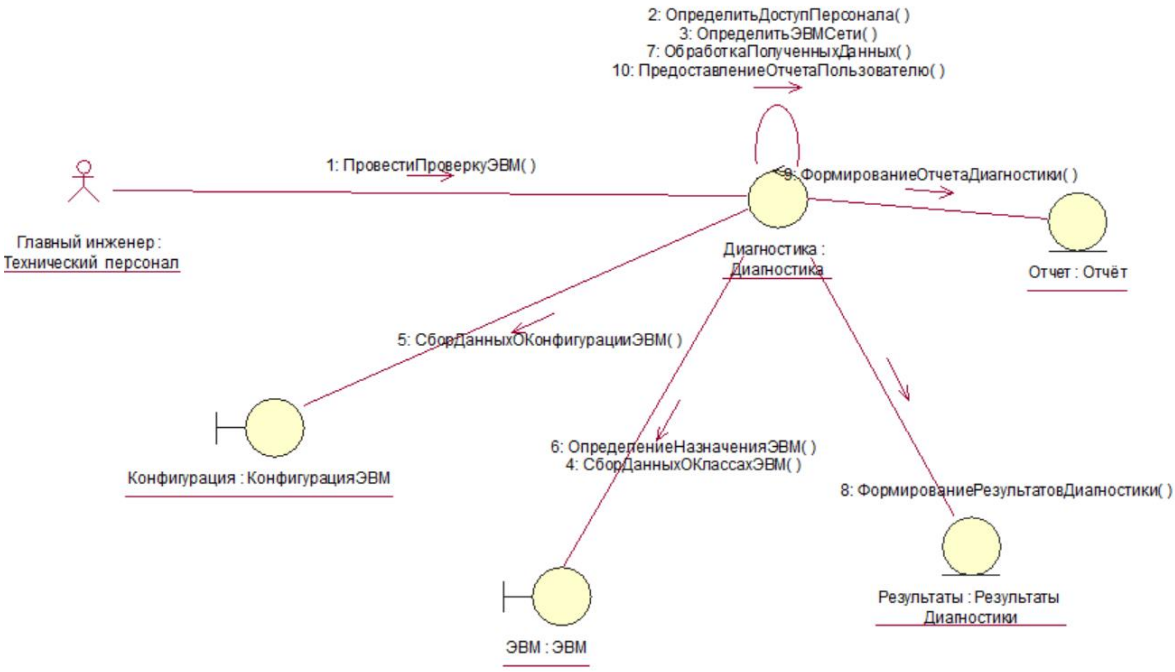

Рисунок 3 – Кооперативная диаграмма «Диагностика»

2.4 Создание диаграммы классов

Диаграмма классов (*class diagram*) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений [3].

Для подробного описания внутренней работы составляющих системы контроля машино-часов необходимо определить основные классы для работы такие, как «*Диагностика», «ЭВМ», «КонфигурацияЭВМ», «Результаты Диагностики», «Отчет»*.

На диаграмме классов показаны множественности:

- один отчет ко многим результатам диагностики (1…n);
- одна диагностика к одному отчету (1…1);
- одна диагностика ко многим результатам диагностики (1…n);
- одна ЭВМ ко многим характеристикам ЭВМ (1…n);
- многие ЭВМ ко многим диагностикам (n…n).

В ходе выполнения данного задания были созданы диаграммы классов. Общий вид данной диаграммы представлен на рисунке 4.

В результате исходная диаграмма классов «Диагностика» содержит 5 классов, каждый – со своим индивидуальным набором спецификаций и определёнными взаимосвязями.

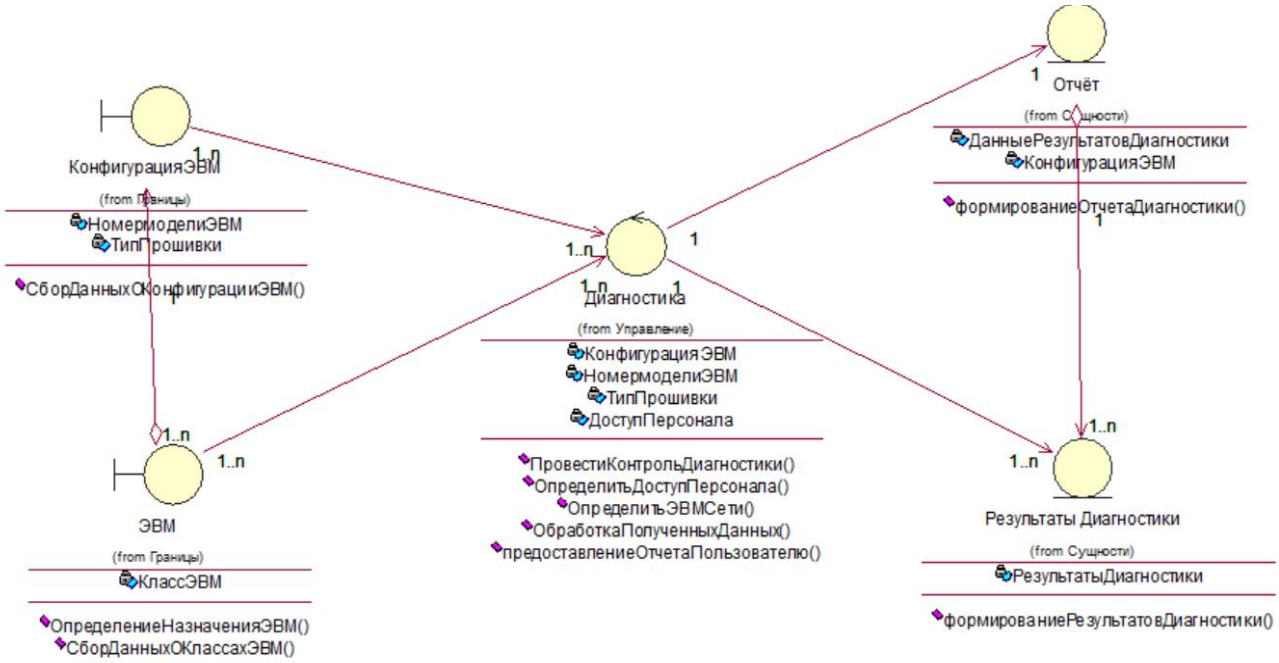

Рисунок 4 – Диаграмма классов «Диагностика»

На данных диаграммах пакеты сгруппированы по стереотипу: один пакет с классами-сущностями, один с граничными классами, один с управляющими.

Классы-сущности (*entity classes*) содержат хранимую информацию. Они имеют наибольшее значение для пользователя, и потому в их названиях часто используют термины из предметной области.

Граничными классами (*boundary classes*) называются такие классы, которые расположены на границе системы и всей окружающей среды.

Управляющие классы (*control classes*) отвечают за координацию действий других классов.

Создана диаграмма классов и главная диаграмма пакетов Main, на которой расположены три пакета: (*Entities*, *Boundaries*, *Control*) Границы, Сущности, Управление (созданы главные диаграммы классов для каждого пакета).

В пакете *Сущности* находятся два класса Результаты диагностики и Отчет (классы-сущности).

В пакете *Границы* находится класс ЭВМ и Конфигурации ЭВМ (граничный класс).

В пакете *Управления* находится класс Диагностика (управляющий класс).

На рисунках 5 продемонстрировано распределение классов по папкам, в соответствии с их стереотипом.

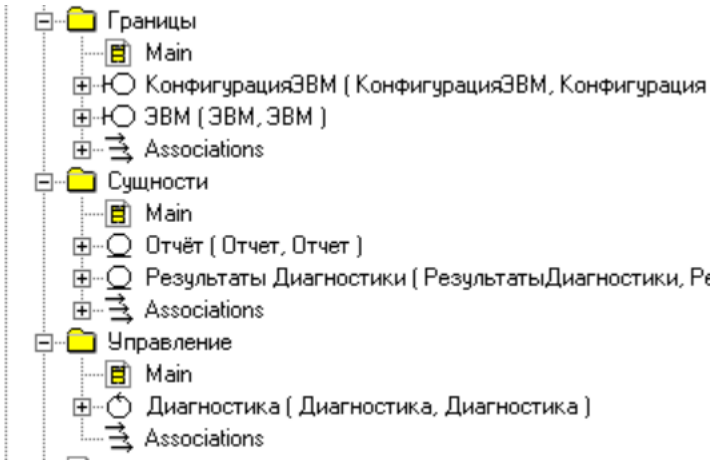

Рисунок 5 – Распределение классов по пакетам

Главная диаграмма классов создана для демонстрации взаимодействия между пакетами классов системы и представлена на рисунке 6. Пакеты применяют, чтобы сгруппировать классы, обладающие некоторой общностью. Активное взаимодействие происходит между классами управление и границы.

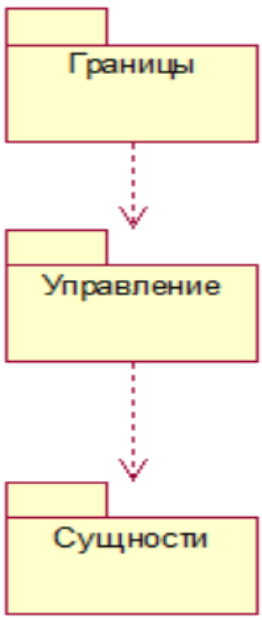

Рисунок 6 – Диаграмма пакетов «Главная диаграмма классов»

Среди всех графических элементов среды *Rational Rose* класс обладает максимальным набором свойств, главными из которых являются его атрибуты и операции. Например, на главной диаграмме классов видно, что класс ЭВМ содержит атрибут *Класс ЭВМ* и операции *Определение Назначения ЭВМ, Сбор Данных О Классах ЭВМ*.

Кроме того, к каждому классу добавляются связи. **Связь** представляет собой семантическую взаимосвязь между ними. Она дает возможность узнавать об атрибутах, операциях и связях другого класса. Существуют четыре типа связей, которые могут быть установлены между классами: ассоциации, зависимости, агрегации и обобщения. В данном курсовом проекте используются ассоциации и агрегации.

Ассоциация *(association)* – это семантическая связь между классами. Их рисуют на диаграмме классов в виде обыкновенной линии. Ассоциации могут быть однонаправленные, двунаправленные и рефлексивные. Для данного проекта использованы только однонаправленные ассоциации: Диагностика-ЭВМ, Диагностика- Конфигурация ЭВМ, Диагностика- Отчет: а, Диагностика - РезультатыДиагностики: Результаты.

Агрегации *(aggregations)* представляют собой более тесную форму ассоциации. Агрегация – это связь между целым и его частью: связи ЭВМ - КонфгиурацияЭВМ; Отчет: а – РезультатыДиагностики: Результаты [2].

2.5 Создание диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий [2].

Основная задача диаграмм состояний – объяснить, каким образом работают сложные объекты, то есть они показывают, каким образом объект переходит из одного состояния в другое. Делая вывод, можно сказать, что диаграммы состояний предназначены для моделирования динамических аспектов системы в целом. Данный вид диаграммы представлен на рисунке 7.

Основные элементы, применяемые для построения данного вида диаграмм, следующие: состояние, суперсостояние, переход.

Состояние – абстрактный класс, используемый для моделирования отдельной ситуации, в течение которой выполняются некоторые условия. На

диаграмме состояний «Проведение диагностики» имеется 4 состояния, исключая начальное и конечные.

На диаграмме присутствуют два вида специальных состояния – начальное *(start)* и конечное *(stop)*. Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением.

Переход – перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может перемещаться между своими состояниями. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим [2].

На диаграмме все состояния связаны межу собой асинхронными сообщениями, т.е. ответ на такое сообщение не требуется и вызывающий объект может продолжить свою работу.

У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события. На диаграмме состояний *«Проведение Диагностики»* используются такие спецификации, как события: «*Проведение диагностики», «Составление отчёта» и «Сообщение персоналу о непригодности».*

Событие *(event)* – это то, что вызывает переход из одного состояния в другое. На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу. В нашем проекте события описаны обычными фразами.

Действие *(action)* — непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния. Пример для состояния «*Проведение диагностики*»: входное действие– *Определение ЭВМ*, действие–*Диагостика*, выходное действие– *Диагностика завершена*.

Ограждающие условия *(guard conditions)* определяют, когда переход может, а когда не может осуществиться. Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода.

Таким образом, диаграмма состояний «*Проведение диагностики*» определяет последовательный набор операций, в ходе которых происходит описание динамики системы выбора отдельных вариантов использования. Данная диаграмма представлена на рисунке 7.

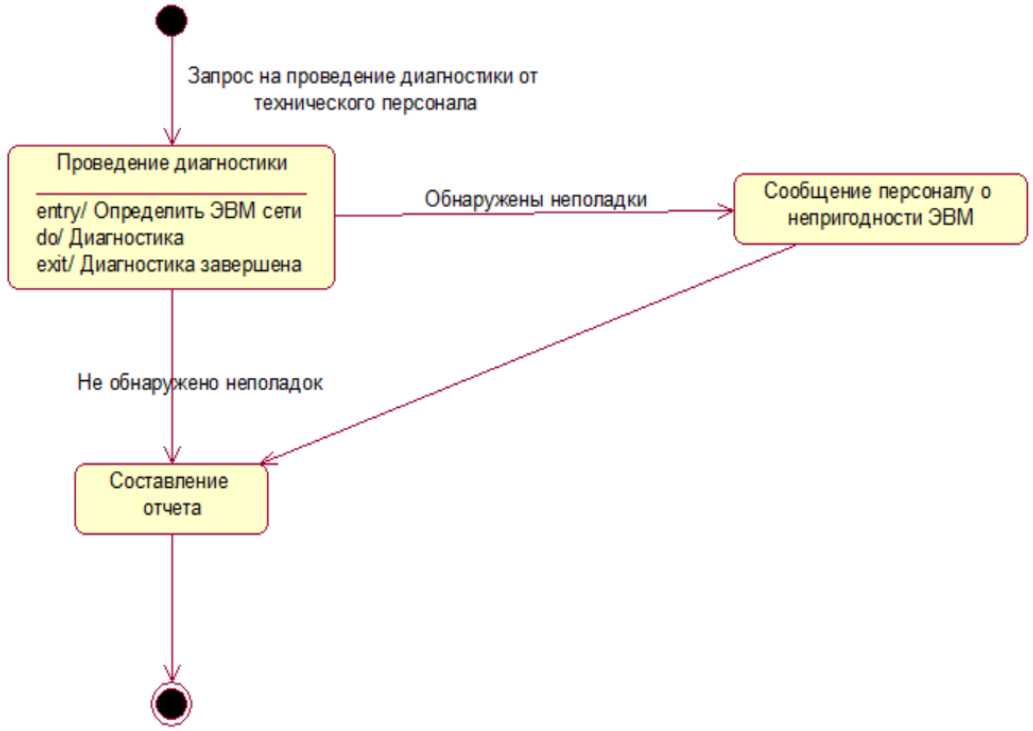

Рисунок 7 - Диаграмма состояний «Проведение Диагностики»

2.6 Создание диаграммы активности

Данная диаграмма создана для демонстрации выполнения задачи в системе и особенно полезна в описании поведения, включающего большое количество параллельных процессов. Подобно большинству других средств, моделирующих поведение, диаграммы активности обладают определенными достоинствами и недостатками, поэтому их лучше всего использовать в сочетании с другими средствами.

В данном случае рассматривается диаграмма активности «*Действия при диагностике*», которая определяет последовательный набор операций, в ходе которых происходит описание динамики системы выбора отдельных вариантов использования.

Из связей видно, что, прежде всего, проверяются результаты диагностики на предмет ошибок. Это сделано для разграничения действий,

которые могут происходить на протяжении программы в зависимости от варианта развития событий. Поэтому сначала используется условие проверки возникновения в ходе диагностики ошибок, если они были, происходит переход к варианту использования «*Сообщение персоналу о непригодности*». Далее осуществляется переход к варианту использования «*Составить отчёт*». Происходит повторная проверка для определения дальнейшего пути развития событий. Если ошибки возникали, то дополнительно будет составлен Отчёт о неполадках. Деятельность завершается после составления отчёта.

Данный вид диаграммы представлен на рисунке 8.

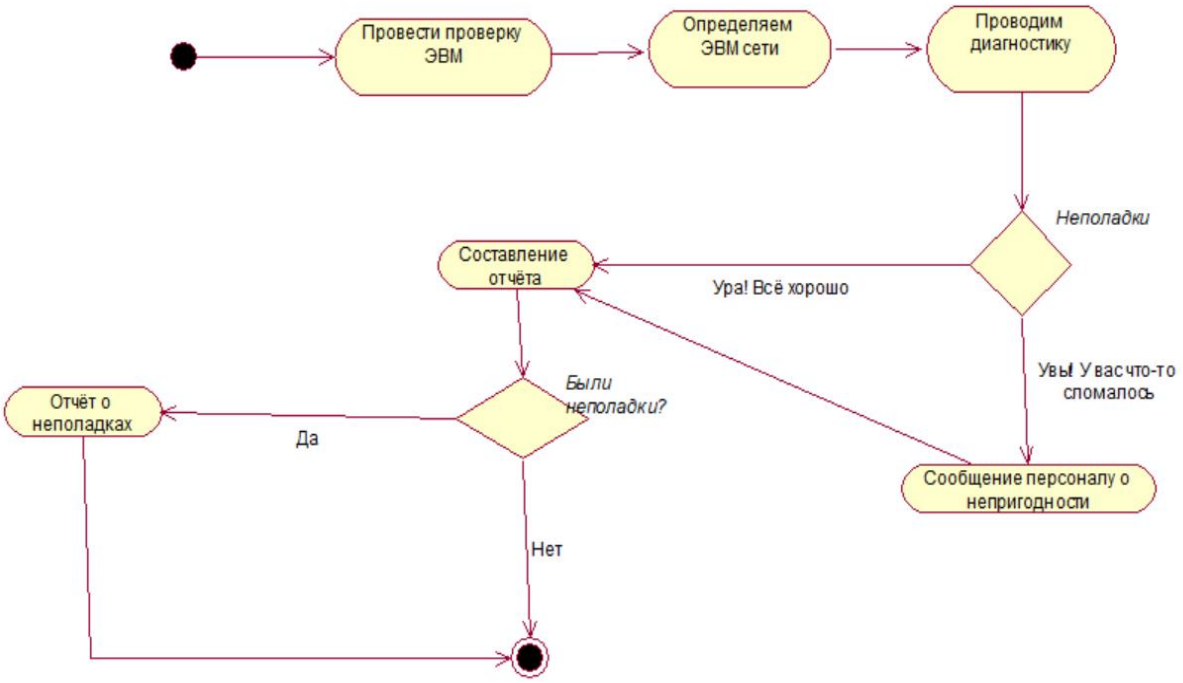

Рисунок 8- Диаграмма активности «Действия при диагностике»

Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. *Конечное состояние* обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением.

Переходом *(Transition)* называется перемещение из одного состояния в другое.

Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события.

Событие *(event)* – это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода. На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу. У событий могут быть аргументы [2].

На диаграмме активностей «Действия при диагностике» можно проследить действия пользователя в системе учёта машино-часов : он может отправить запрос на проверку ЭВМ, определить ЭВМ, провести диагностику, в зависимости от результатов диагностики могут происходить несколько параллельных процессов (отправка сообщения о непригодности техническому персоналу и возможность составить отчёт о неполадках в случае возникновения неполадок и возможность составить отчёт в случае успешной диагностики). Например, переход к варианту использования *«Составление отчёта о неполадках»* невозможен, если не были обнаружены ошибки.

2.7 Создание диаграммы Базы Данных

Одна из задач нашей системы – хранение данных о результатах предыдущих диагностик и отчётов о проведении диагностик. Для этого нам необходима база данных- реестр. В ней представлены все компоненты пакета Сущности. Диаграммы БД были сгенерированы с помощью модулю *Data Model*. Модель базы данных представлена на рисунке 9.

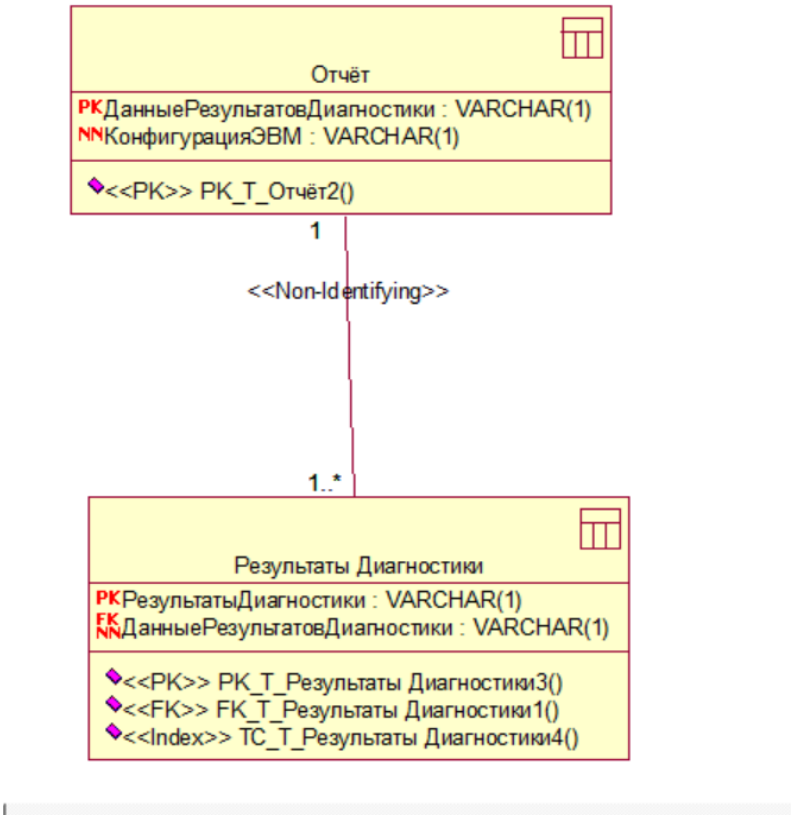

Рисунок 9 – Диаграмма схемы базы данных «Схема БД»

На данной диаграмме отображаются 2 класса системы – Результаты Диагностики и Отчёт. Их атрибуты Данные Результатов Диагностики, Результаты Диагностики, Конфигурация ЭВМ содержат данные о проведенной диагностики. Результаты Диагностики и Данные Результатов Диагностики заносятся в таблицы, множество таких таблиц вместе с атрибутом Конфигурация ЭВМ может содержаться в таблице Отчёты.

Таким образом, схема БД отражает организацию данных, представляющих результаты работы системы машино-часов. Это обеспечивает лёгкий и удобный анализ данных предыдущих диагностик и облегчение проведения следующих.

3. РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1 Создание диаграммы компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода [4].

Диаграмма показывает основные спецификации и их взаимодействие. Это позволяет правильно организовать написание кода, тем самым увеличивая продуктивность и уменьшая шанс возникновения ошибки. Она демонстрирует физический смысл исходной модели системы.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы.

На рисунке 10 представлена построенная диаграмма компонентов.

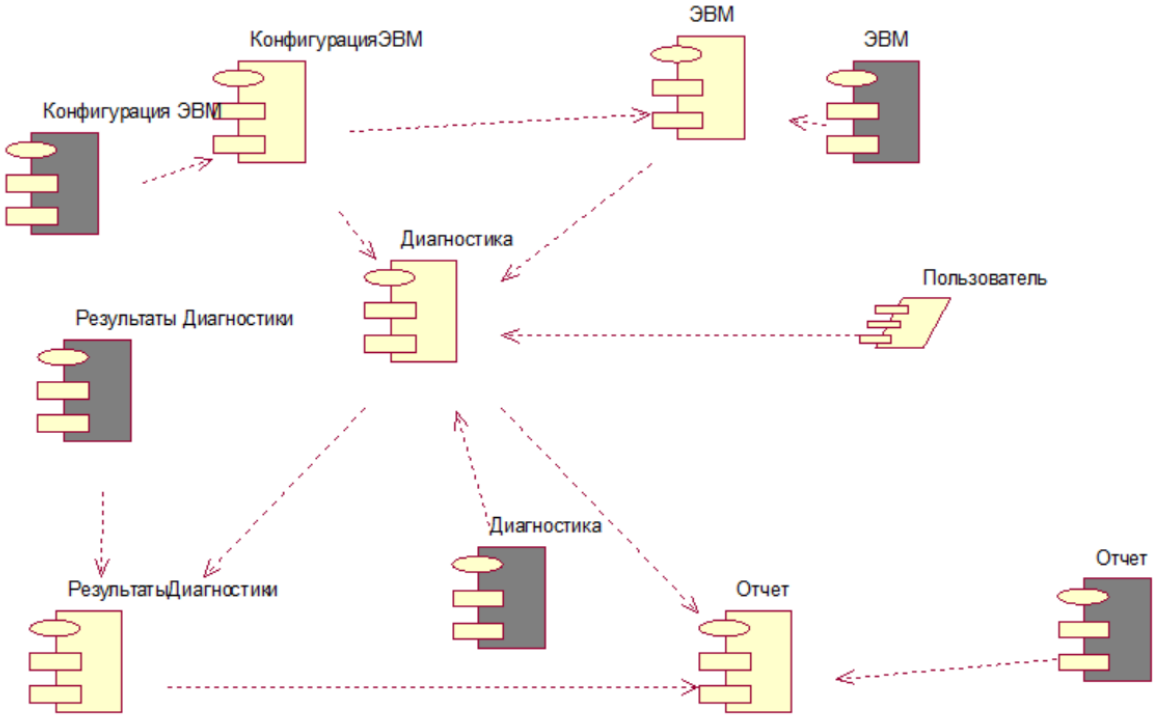

Рисунок 10 - Диаграмма компонентов

Каждый компонент данной диаграммы размещён в соответствующие пакеты (*Границы, Управление, Сущности*) в зависимости от назначенного ранее стереотипа и выполняемой им функцией. Также имеются соотнесения с определёнными классами логического представления браузера. Например, компонент *«ЭВМ»* реализует работу класса *«ЭВМ»*.

Взаимодействие компонентов диаграммы обеспечивается за счёт пунктирных однонаправленных линий связи (*Dependency*).

Данная диаграмма компонентов системы содержит следующие компоненты (включают тело и заголовок):

- конфигурация ЭВМ;
- ЭВМ;
- диагностика;
- пользователь;
- отчет.

3.2 Диаграмма размещения

Диаграмма размещения (*deployment diagram*) отражает физические взаимосвязи между программными и аппаратными компонентами системы [4]. Диаграмма размещения данного курсового проекта содержит 1 устройство (Экран) и 3 процессора:

1. система контроля ЭВМ;
2. пользовательская рабочая станция;
3. сервер с БД Отчётов Диагностики).

Взаимодействие отдельных структурных частей системы управления лифтами обеспечивается с помощью связей *Connection*.

Данная диаграмма изображена на рисунке 11.

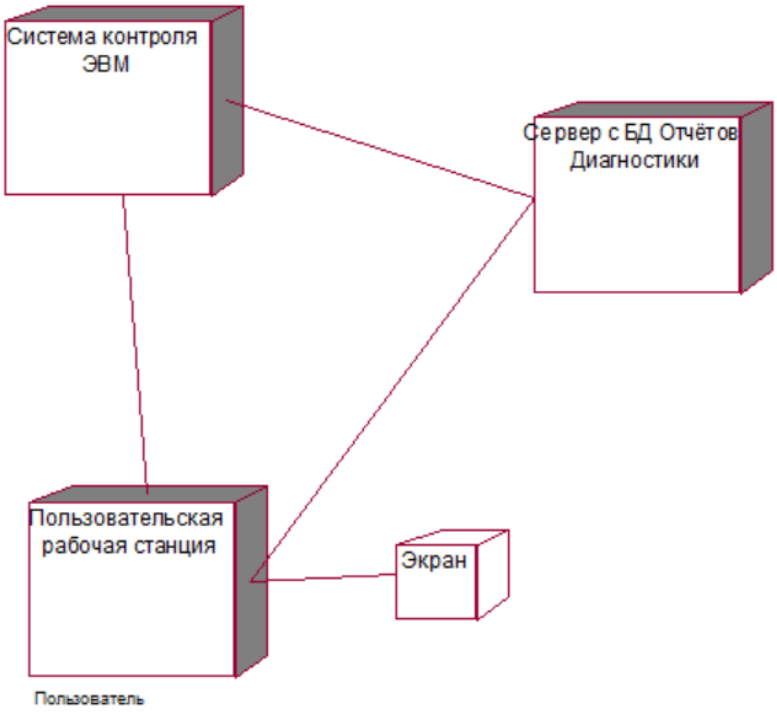

Рисунок 11 - Диаграмма размещения *«Deployment view»*

3.3 Генерация кода C++

Воспользуемся возможностью *Rational Rose* для генерации макетов кода для созданных в диаграммах классов. На основе полученных данных были созданы классы, в их спецификациях мы можем указать принадлежность к *C++* коду. Это позволяет нам облегчить работу программистов, направляя их работу в нужном направлении.

Для этого необходимо выбрать все компоненты диаграммы компонентов «Диаграмма компонентов» и установить соответствующий язык программирования, вызвав для этого окно спецификации каждого элемента [5]. Выбор класса или компонента для генерации программного кода означает выделение соответствующего элемента модели в браузере проекта.

Сгенерированный код представлен в Приложении А.

ЗАКЛЮЧЕНИЕ

На основе полученных данных из задания была спроектирована система. Для этого была использована программа *Rational Rose* и язык объектно-ориентированного проектирования *UML*. В процессе работы были созданы основные диаграммы языка. Было создано описание к диаграммам.

Первой была построена диаграмма вариантов использования. На ней показаны основные требования к системе. Эта диаграмма является первым шагом к проектированию системы. Она изображает действующих лиц системы, возможные события и обобщает систему в целом. Исследовав диаграмму вариантов использования, был осуществлен переход к следующему шагу.

Далее была изображена диаграмма последовательности для процесса проведения пользователем диагностики. Это позволяет наглядно обозначить последовательность всех действий при взаимодействии технического персонала и системы контроля машино-часов.

Затем мы создали диаграммы классов. Общую для обозрения пакетов и их взаимодействия. Главную для проектировки самих классов и их ассоциаций. Здесь были описаны все классы системы.

В результате выполнения курсового проекта проведен анализ предметной области, исходя из которого, разработана функциональная модель системы «Машино-часы». В процессе проектирования модели данных автоматизированной системы «Машино-часы» определены сущности, описывающие объекты предметной области, выделены атрибуты, отображающие экземпляры данных сущностей, определены отношения и связи между сущностями. На основе чего построена логическая модель системы. Это позволило нам детально спроектировать программу, начиная от начальных требований диаграммы вариантов использования и заканчивая генерацией кода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Боггс У., Боггс М. *UML* и *Rational Rose*: Пер. с англ. – М.: Лори, 2000.

[2] Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2–е изд.: Пер. с англ. – М.: Издательство Бином, СПб.: Невский диалект, 1999.

[3] Буч Г., Рамбо Дж., Джекобсон *А.* Язык *UML*. Руководство пользователя: Пер. с англ. – М.: ДМК, 2000.

[4] Вендров *А. М.*, *CASE*-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998.

[5] Вендров *А. М.*, Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000.

[6] Вендров *А. М.*, Объектно-ориентированный анализ и проектирование с использованием языка *UML* и *Rational Rose*, 2004. – 54 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг кода

Диагностика.cpp

```
#include "Диагностика.h"

//##ModelId=5925E08C029C
Диагностика::Провести контроль Диагностики()
{
}

//##ModelId=5925E08F034E
Диагностика::Определить доступ персонала()
{
}

//##ModelId=5925E0B0028E
Диагностика::Определить ЭВМ сети()
{
}

//##ModelId=5925E0B9005D
Диагностика::Обработка полученных данных()
{
}

//##ModelId=5925E0C1014E
Диагностика::предоставление отчета пользователю()
{
}
```

Диагностика.h

```
ifndef ДИАГНОСТИКА_H_HEADER_INCLUDED_A6DA798C
#define ДИАГНОСТИКА_H_HEADER_INCLUDED_A6DA798C
```

```
//##ModelId=5925DE4601AB
class Диагностика
{
  public:
    //##ModelId=5925E08C029C
    Провести контроль диагностики();

    //##ModelId=5925E08F034E
    Определить доступ персонала();

    //##ModelId=5925E0B0028E
    Определить ЭВМ сети();

    //##ModelId=5925E0B9005D
    Обработка полученных данных();

    //##ModelId=5925E0C1014E
    предоставление отчета пользователю();

  private:
    //##ModelId=5925DF10032D
    Конфигурация ЭВМ;
    //##ModelId=5925DF1702A7
    Номер Модели ЭВМ;
    //##ModelId=5925DF1C002C
    Тип Прошивки;
    //##ModelId=5925DF1C002C
    Доступ персонала;
};
```

Конфигурация ЭВМ.cpp

```
#endif /* ДИАГНОСТИКА_H_HEADER_INCLUDED_A6DA798C */
#include "Конфигурация ЭВМ.h"

//##ModelId=5925E2C8009C
Конфигурация ЭВМ::Сбор данных о конфигурации ЭВМ()
```

```
{
}
```

Конфигурация ЭВМ.h

```
#ifndef КОНФИГУРАЦИЯ_ЭВМ_H_HEADER_INCLUDED_A6DA59FE
#define КОНФИГУРАЦИЯ_ЭВМ_H_HEADER_INCLUDED_A6DA59FE

//##ModelId=5925DE2E005B
class Конфигурация ЭВМ
{
  public:
    //##ModelId=5925E2C8009C
    Сбор данных о конфигурации ЭВМ();

  private:
    //##ModelId=5925DF330355
    Номер модели ЭВМ;
//##ModelId=5925DF330355
    Тип прошивки;
};
```

Отчет.cpp

```
#endif /* КОНФИГУРАЦИЯ_ЭВМ_H_HEADER_INCLUDED_A6DA59FE */
#include "Отчет.h"

//##ModelId=5925E0BE0235
Отчет::формирование отчета диагностики()
{
}
```

Отчет.h

```
#ifndef ОТЧЕТ_H_HEADER_INCLUDED_A6DA7BAC
#define ОТЧЕТ_H_HEADER_INCLUDED_A6DA7BAC
```

```
//##ModelId=5925DE7701BB
class Отчет
{
  public:
    //##ModelId=5925E0BE0235
    формирование отчета диагностики();

  private:
    //##ModelId=5925DED6026D
    Данные результатов диагностики;
    //##ModelId=5925DEE202DF
    Конфигурация ЭВМ;
};

#endif /* ОТЧЕТ_H_HEADER_INCLUDED_A6DA7BAC */
```

ЭВМ.cpp

```
#include "ЭВМ.h"

//##ModelId=5925E2940165
ЭВМ::Определение назначения ЭВМ()
{
}

//##ModelId=5925E2D502BD
ЭВМ::Сбор данных о типах ЭВМ()
{
}
```

ЭВМ.h

```
#ifndef ЭВМ_H_HEADER_INCLUDED_A6DA2FBD
#define ЭВМ_H_HEADER_INCLUDED_A6DA2FBD

//##ModelId=5925DE1E01DC
```

```
class ЭВМ
{
  public:
    //##ModelId=5925E2940165
    Определение назначения ЭВМ();

    //##ModelId=5925E2D502BD
    Сбор данных о типах ЭВМ();

  private:
    //##ModelId=5925DF2E0223
    Тип ЭВМ;
};

#endif /* ЭВМ_H_HEADER_INCLUDED_A6DA2FBD */
```

Результаты Диагностики.cpp

```
#include "Результаты Диагностики.h"

//##ModelId=5925E0BB02AD
Результаты Диагностики::формирование  результатов диагностики()
{
}
```

Результаты Диагностики.h

```
#ifndef
РЕЗУЛЬТАТЫ_ДИАГНОСТИКИ_H_HEADER_INCLUDED_A6DA739D
#define
РЕЗУЛЬТАТЫ_ДИАГНОСТИКИ_H_HEADER_INCLUDED_A6DA739D

//##ModelId=5925DE56006C
class Результаты Диагностики
{
  public:
    //##ModelId=5925E0BB02AD
```

```
    формирование  результатов диагностики();

  private:
    //##ModelId=5925DEF201F3
    Результаты диагностики;
};

#endif /*
РЕЗУЛЬТАТЫ_ДИАГНОСТИКИ_H_HEADER_INCLUDED_A6DA739D */
```

ВЕДОМОСТЬ КУРСОВОГО ПРОЕКТА

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1-53 01 02 06 029 ПЗ	Пояснительная записка	39с.
	Графические документы	
ГУИР.000000.001 ПЛ	Диаграмма вариантов	Формат А1
	использования	
ГУИР.000000.002 ПЛ	Диаграмма классов	Формат А1

					БГУИР КП 1-53 01 02 06 029 ПЗ
Изм.	Л	№ докум.	Подп.	Дата	
Разраб.		Шведов			Система учёта машино-часов Ведомость курсового проекта
Провер.		Ревотюк			

Лит	Лист	Листов
Т	39	39

Кафедра ИТАС
гр. 820601