Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт Компьютерных систем и информационной безопасности
Кафедра Информационных систем и программирования
Специальность 10.05.03 Информационная безопасность автоматизированных систем
Профиль Защищенные автоматизированные системы управления

КУРСОВАЯ РАБОТА

по дисциплине Технологии и методы программирования
(наименование дисциплины)

на тему: «Стиральная машина»
(тема курсовой работы)

Выполнил студент 2 курса группы 18-К-АС1

Кальянов Константин Николаевич
(Ф.И.О.)

Допущен к защите_____

Руководитель (нормоконтролер) работы _____О.Б. Попова

Защищен _____ Оценка _____
(дата)

Члены комиссии Н.В. Кушнир_____

К.Е. Тотухов_____

Краснодар
2020

ФГБОУ ВО «Кубанский государственный технологический университет»
(ФГБОУ ВО КубГТУ)

Институт Компьютерных систем и информационной безопасности
Кафедра Информационных систем и программирования
Специальность 10.05.03 Информационная безопасность автоматизированных систем
Профиль Защищенные автоматизированные системы управления

УТВЕРЖДАЮ
Зав. кафедрой _____М.В. Янаева
«12» февраля 2020 г.

ЗАДАНИЕ
на курсовую работу

Студенту: Кальянову К. Н. группы 18-К-АС1 курса 2
(Ф.И.О.) (№ группы и курса)

Тема проекта: «Стиральная машина»

План работы:
1. Изучение предметной области
2. Проектирование
3. Описание реализованных диаграмм

Объем работы:
а) пояснительная записка 49 с.

Рекомендуемая литература
1. Йордон. «Объектно-ориентированный анализ и проектирование систем»
2. Роберт А. Максимчук. «UML для простых смертных»
3. «Автоматизация проектирования вычислительных систем.»ред. М.Брейер

Срок выполнения: с «15» февраля по «28» марта 2020г.
Срок защиты: с «11» мая по «14» июня 2020 г.
Дата выдачи задания «15» февраля 2020г.

Дата сдачи работы на кафедру «01» июня 2020 г.

Руководитель работы _____ к.т.н., доцент Попова О.Б.
(должность, подпись,)

Задание принял студент _____ Ф.И.О.

Реферат

Курсовая работа: 49 страниц, 18 рисунков, 12 используемых источников.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, ПОСЛЕДОВАТЕЛЬНОСТИ, МОДЕЛЬ, BPMN, DFD, IDEF0, FURPS+, СТИРАЛЬНАЯ МАШИНА, UML ДИАГРАММЫ.

Объектом исследования является программное обеспечение стиральной машины, которая способна стирать вещи по заданным параметрам при выборе определенной программы, при этом пользователь не может вносить изменения в программу, автоматически закрывать и открывать дверь в начале и конце стирки, автоматически греть воду до нужной температуры, остановить стирку по требованию пользователя.

Цель работы состоит в разработке проекта программного обеспечения «стиральная машина» с использованием UML диаграмм, в полной мере описывающих как внутреннее устройство исследуемой системы, так и всевозможные взаимодействия между её компонентами.

В итоге были получены UML диаграммы, обладающие исчерпывающей информацией о программном обеспечение стиральной машины. К ним относятся: диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма физического аспекта системы, диаграмма коммуникации, диаграмма состояний и диаграмма компонентов.

Содержание

Введение

На сегодняшний день стиральная машина есть в 99% домов. Появление стиральных машин позволило людям сэкономить много времени на стирке. Более того, ручная стирки подразумевает под собой нелегкий физический труд, так как приходится наклоняться, активно работать руками и проводить много времени с водой, что со временем неблагоприятно сказывается на коже.

Но, несмотря на практически повсеместное использование этого «маминого помощника», он никогда не смог бы обеспечить надлежавший уровень удобства без тщательного исследования предметной области и проведения различных тестов, учитывающих всевозможные взаимодействия с компонентами системы. Именно для этих целей был создан унифицированный язык моделирования UML, позволяющий описать любую систему в максимально удобном и понятном виде, даже для незнакомого с этой областью человека. Поэтому задачей данного курсового проекта является не только разработка симулятора стиральной машины, но и проектирование основных видов UML диаграмм, позволяющих в полной мере раскрыть все аспекты исследуемой системы. В результате стиральная машина, созданная на основании всех проведённых исследований, будет соответствовать основным требованиям для обеспечения удобной, стабильной и долговременной работы.

1. Формулировка задачи

Задачей данного курсового проекта является разработка модели программного обеспечения встроенного микропроцессора для стиральной машины, Машина предназначена для автоматической стирки белья. Машина включает в себя следующие устройства: бак для белья, клапаны для забора и слива воды, мотор, устройство подогрева воды, термометр, таймер, дверца для доступа в бак, несколько емкостей для различных моющих средств, панель управления с кнопками и индикатором. В памяти машины хранятся 5 программ стирки, заданные изготовителем. Пользователи не могут вносить в них изменения. Каждая программа определяет температуру воды, длительность стирки, используемые моющие средства (номер емкости и время подачи), скорость вращения бака во время стирки и отжима. Вариант задания предусматривает разработку схемы базы данных для хранения программ стирки в памяти машины.

Для использования машины необходимо открыть дверцу, поместить белье в бак, поместить моющие средства в емкости, закрыть дверцу, выбрать программу стирки и нажать на кнопку «Пуск». Перед тем как приступить к стирке машина открывает клапан для забора воды, набирает необходимое количество воды, после чего закрывает клапан. Далее, машина действует по выбранной пользователем программе:

− Подогревает, если необходимо воду до нужной температуры.

− Включает таймер и запускает вращение бака для стирки.

− По таймеру подает в бак моющие средства, предусмотренные программой.

− По окончании стирки сливает воду и запускает отжим.

Во время работы на индикаторе высвечивается время, прошедшее от начала стирки (минуты и секунды), текущий режим работы (стирка или отжим), номер текущей программы стирки. В целях безопасности дверца бака блокируется до окончания стирки. Машина не воспринимает нажатий на кнопки, за исключением одной – пользователь имеет возможность в любой момент нажать на кнопку «Останов», чтобы принудительно остановить стирку и слить воду.

2. Язык проектирования UML

Язык UML — это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем. При проектировании на данном языке создаются специальные диаграммы, подробно описывающие функционал, разрабатываемой системы, а также все возможные взаимодействия с ней.

Словарь UML включает три вида строительных блоков:

– Диаграммы;

– Сущности;

– Связи.

Сущности – это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей.

Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). Язык UML включает 13 видов диаграмм, среди которых на первом месте в списке — диаграмма классов, о которой и пойдет речь.

Диаграммы классов показывают набор классов, интерфейсов, а также их связи. Диаграммы этого вида чаще всего используются для моделирования объектно-ориентированных систем. Они предназначены для статического представления системы.

Большинство элементов UML имеют уникальную и прямую графическую нотацию, которая дает визуальное представление наиболее важных аспектов элемента. Диаграмма UML для стиральной машины находится в «Приложении А».

3. Создание модели As-Isв стандарте IDEF0

Чтобы оценить все возможности разрабатываемой системы, следует построить её базовую модель, которую можно представить в виде диаграммы As-Is.

Диаграмма As-Is – это функциональная модель системы «как есть», позволяющая узнать, где находятся слабые места, в чём будут состоять преимущества и недостатки, протекающих в ней бизнес-процессов относительно конкурентов. Применение данной модели позволит чётко зафиксировать какие информационные объекты принимают участие в жизненном цикле системы, какая информация будет поступать на вход и что будет получаться на выходе. Модель As-Is, строится с использованием нотации IDEF0.

IDEF0 – это методология функционального моделирования (англ. function modeling), графическая нотация, предназначенная для описания бизнес-процессов. Система, описываемая в данной нотации, проходит через декомпозицию или, иными словами, разбиение на взаимосвязанные функции. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. Для каждой функции существует правило сторон:

– стрелкой слева обозначаются входные данные;

– стрелкой сверху – управление;

– стрелкой справа – выходные данные;

– стрелкой снизу – механизм.

Учитывая всё вышеперечисленное, на рисунке 1 была составлена модель As-Is системы «Стиральная машина».

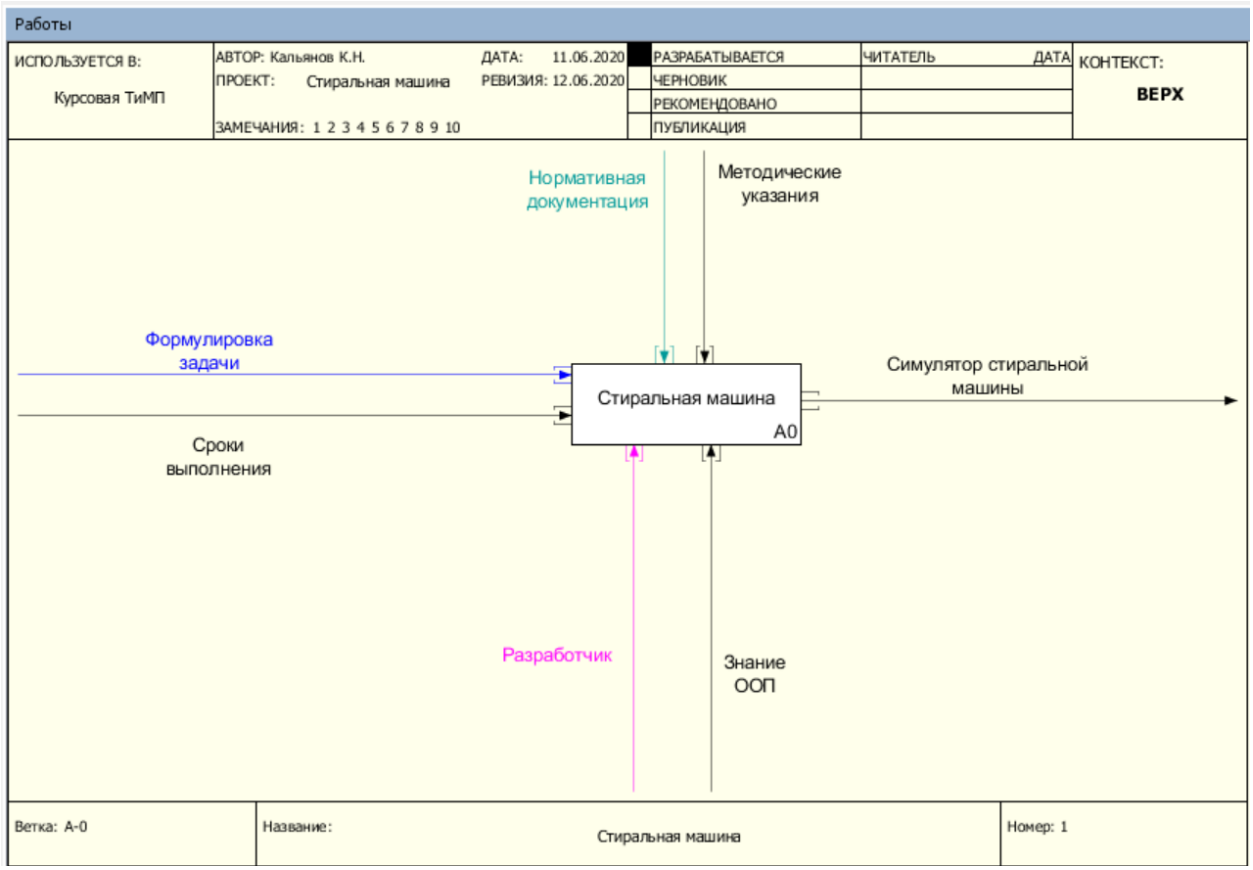


Рисунок 1 – Модель As-Is системы «Стиральная машина»

Входными данными для начала работы являются сроки выполнения и формулировка задачи, они необходимая для того, иметь представления какое программное обеспечение нужно разрабатывать и в какой срок нужно уложиться.

Управление происходит с помощью нормативной документации.

Механизмом реализации работы системы является разработчик, знание и применение ООП.

Результатом деятельности системы является готовый программный продукт.

Полученная модель системы может быть представлена в более развернутом виде путём разбиения на большее количество составных элементов.

На рисунке 2 можно видеть модель системы «Стиральная машина» после декомпозиции.

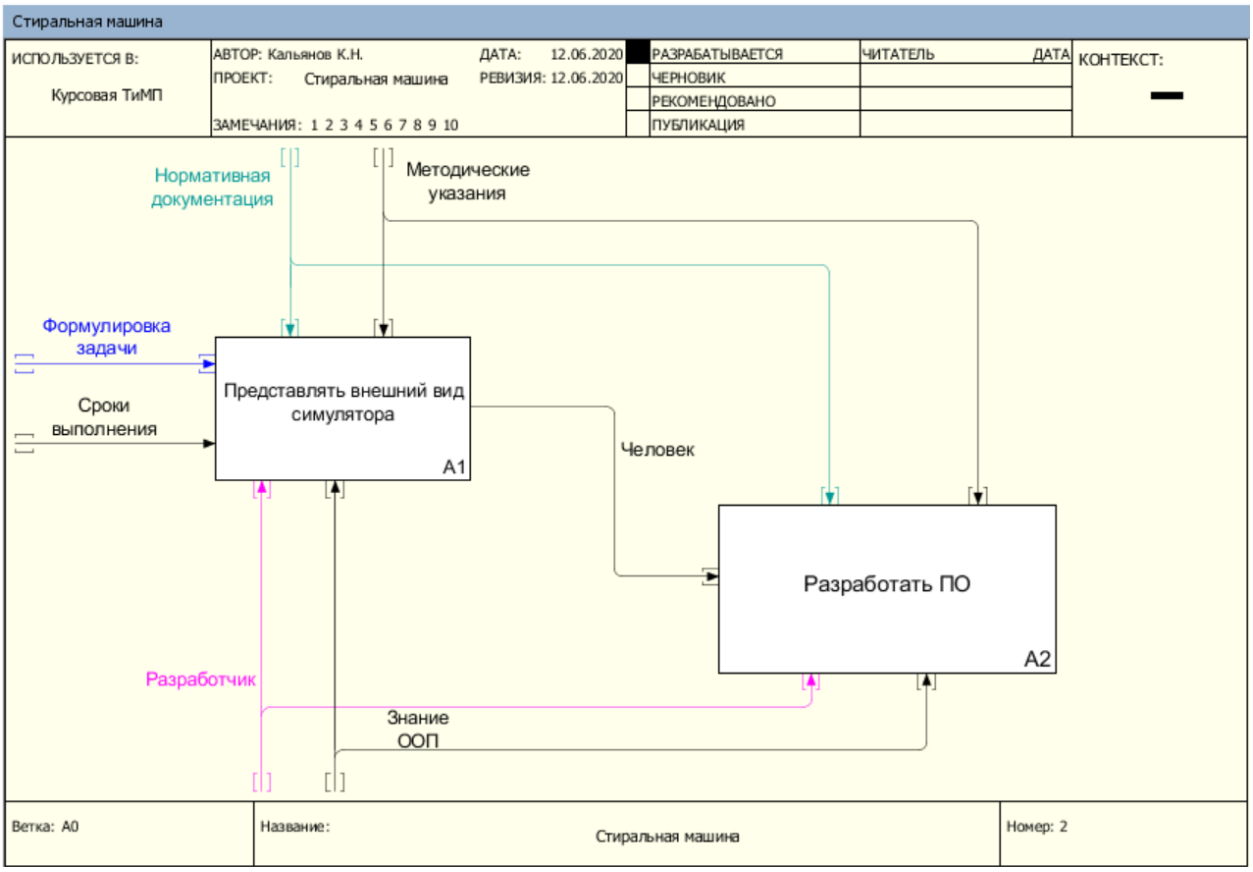


Рисунок 2 – Декомпозиция системы «Стиральная машина»

Полученная модель системы может быть представлена в более подробном виде путём разбиения на большее количество составных элементов.

На рисунке 3 можно видеть модель ПО «Стиральная машина» после декомпозиции.

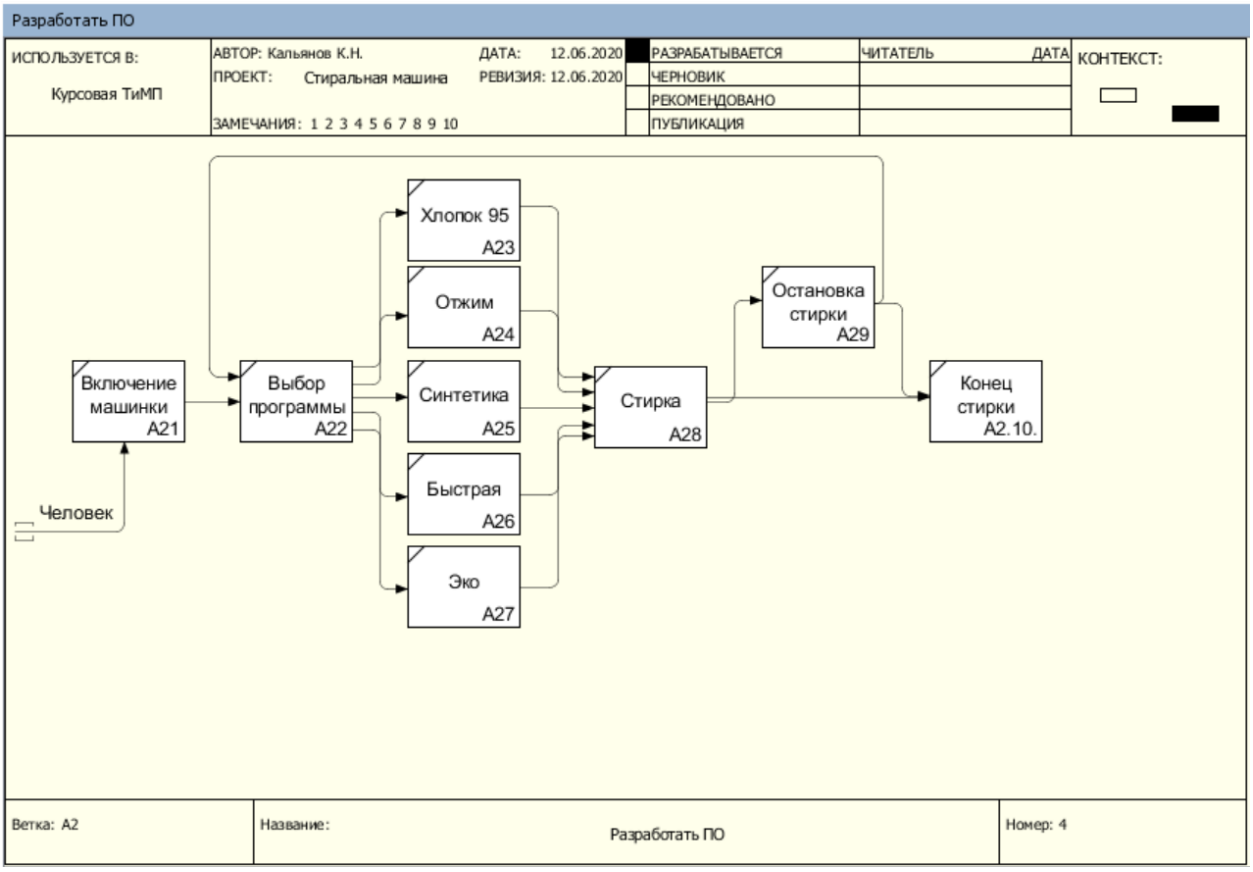


Рисунок 3 – Декомпозиция кода ПО «Стиральная машина»

Входными данными для данной системы являются действия человека, необходимые для того, чтобы включить стиральную машину.

Управление происходит при помощи кнопок на панели стиральной машины, а также правилам, описанным в инструкции.

Механизмом реализации работы системы человек перед стиральной машинкой, панель от стиральной машины и ПО стиральной машины.

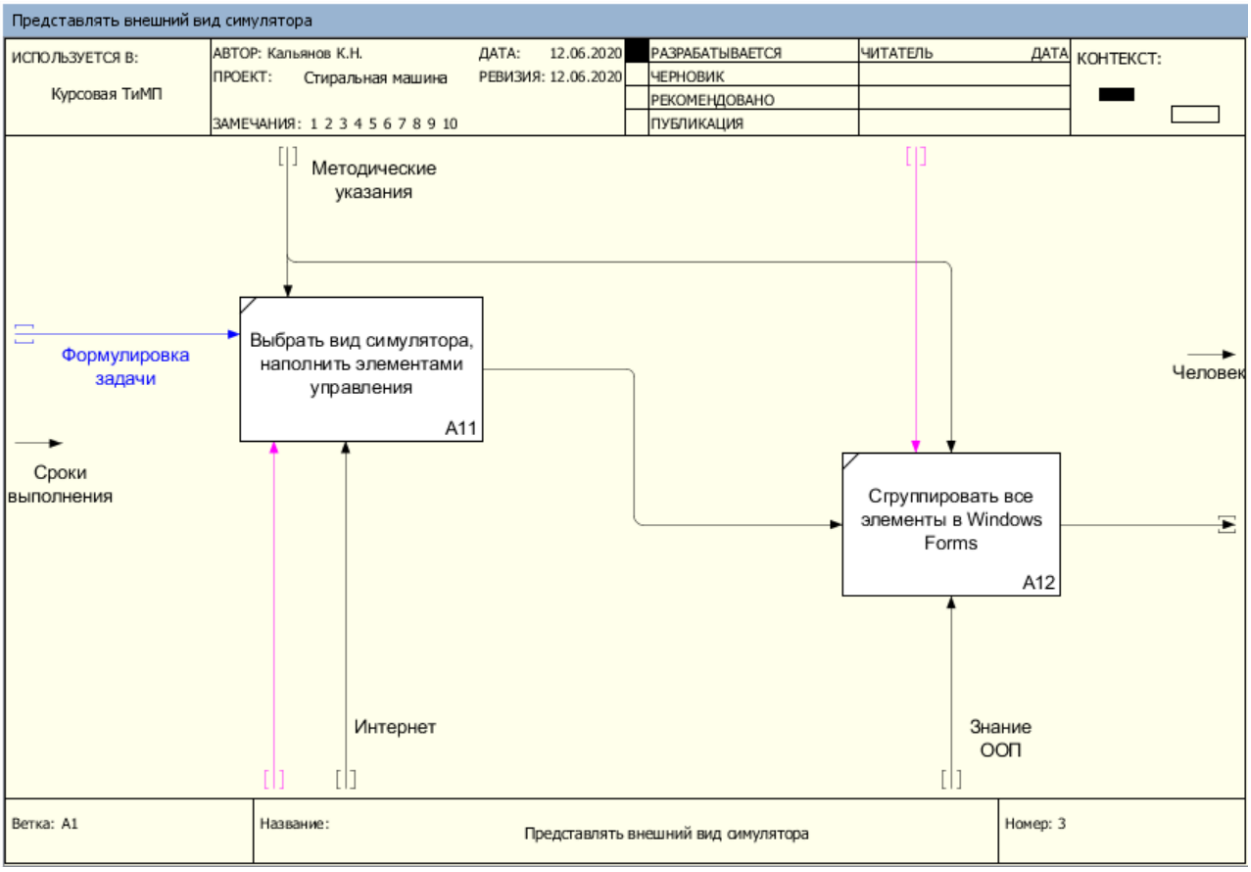


Рисунок 4 – Декомпозиция представления как должен выглядеть симулятор телевизора

Результатом деятельности системы является стирка с определенными параметрами, заложенными в программе.

4. Диаграмма потоков данных (DFD)

DFD — общепринятое сокращение от англ. data flow diagrams — диаграммы потоков данных. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Диаграмма потоков данных — один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML.

В результате декомпозиции системы «Стиральная машина» была получена следующая диаграмма DFD (рис. 5).

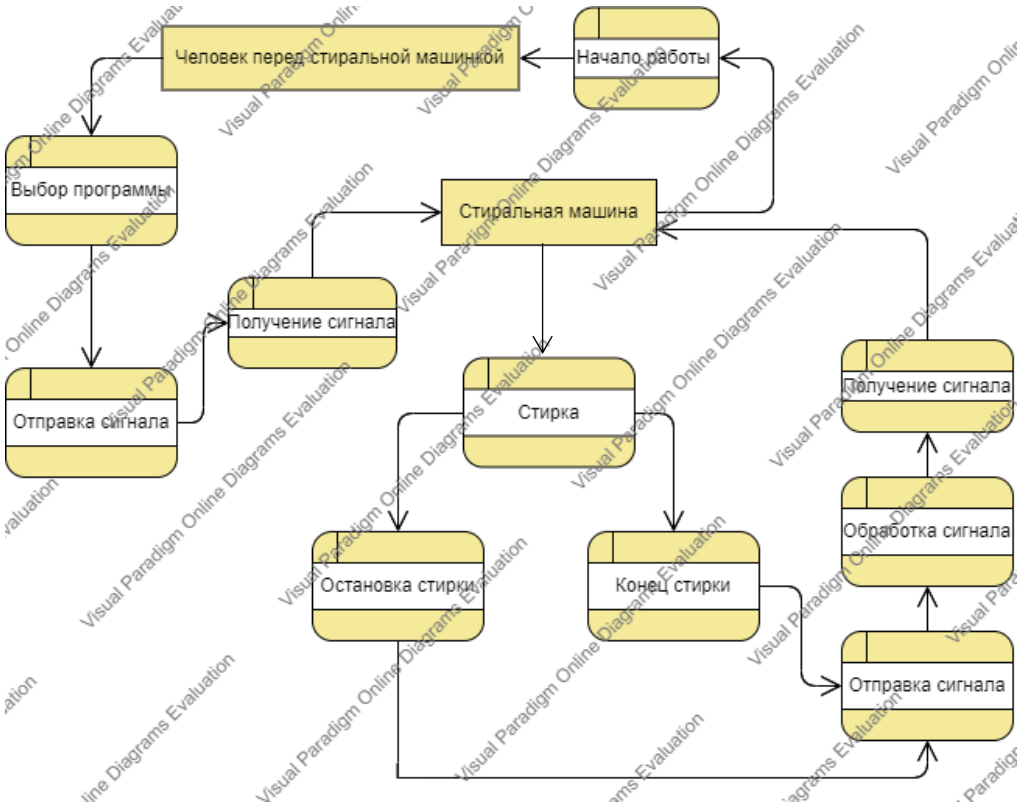


Рисунок 5 – Диаграмма DFD системы «Стиральная машина»

Внешними сущностями данной системы является человек перед стиральной машинкой, а также стиральная машина. Система также содержит одну базу данных для программ стирок, закодированных изготовителем.

5. Диаграмма Ганта

Диаграмма Ганта — это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Используется в приложениях по управлению проектами.

Диаграмма Ганта — это инструмент планирования, управления задачами, который придумал американский инженер Генри Гант (Henry Gantt). Выглядит это как горизонтальные полосы, расположенные между двумя осями: списком задач по вертикали и датами по горизонтали.

На диаграмме видны не только сами задачи, но и их последовательность. Это позволяет ни о чём не забыть и делать всё своевременно.

Общий вид диаграммы Ганта – это обычный график, состоящий из горизонтальных полос, которые ориентированы между двумя осями:

– Вертикальная представляет собой список задач. Каждая полоса – это отдельный процесс, часть проекта. Последовательное расположение позволяет помнить о всех процессах и отслеживать сроки их реализации.

– Горизонтальная – временные даты. На графике можно увидеть момент начала и окончания работы, ее общую продолжительность.

График Ганта может также отражать процентный показатель завершения работ, совокупные процессы и способы их объединения, содержать метки или вехи ключевых моментов. На многих таблицах указывают и ответственных за каждую задачу.

Преимущества диаграммы как инструмента:

– Визуализация обеспечивает четкое понимание того, в какой стадии находится проект, сколько времени осталось на выполнение задач, где расположены критические точки.

– Графики позволяют оптимизировать процесс планирования и распределения задач между сотрудниками.

– Это отличный инструмент презентации, который помогает наглядно продемонстрировать приоритетные задачи проекта.

– Составление диаграмм не требует специальных знаний – их можно составить от руки или воспользоваться специальными приложениями, среди которых есть и бесплатные.

Диаграмма Ганта для проекта «Стиральная машина» находится в «Приложении Б».

6. Проектирование диаграммы EPC

Событийная цепочка процессов (EPC, event-driven process chain) - тип диаграмм, используемых для моделирования, анализа и реорганизации бизнес-процессов (функционального моделирования). В то же время EPC-диаграммы могут использоваться для моделирования поведения отдельных частей системы при реализации функций и служить заменой традиционных блок-схем (поведенческого моделирования). EPC-метод был разработан Августом-Вильгельмом Шеером в рамках работ над созданием методологии ARIS (Architecture of Integrated Information Systems - Архитектура интегрированных информационных систем) в начале 1990-х годов.

Диаграмма процесса (функции) в нотации EPC представляет собой упорядоченную комбинацию событий и функций. Для каждой функции могут быть определены начальные и конечные события, участники, исполнители, материальные и информационные потоки, сопровождающие её, а также проведена декомпозиция на более низкие уровни.

Как и в случае с DFD, методология EPC «разрослась» интерпретациями, поддерживающими различные нотации (синтаксис и семантику элементов). К этому «приложили руку» как сам автор методологии, так и производители ПО, в котором реализована возможность моделирования бизнес-процессов посредством EPC (ARIS, Microsoft Visio, Business Studio, Bflow). По аналогии с блок-схемами, символы (элементы) графической нотации можно сгруппировать по назначению. Бизнес-процессы в нотации EPC описываются в виде последовательности событий и функций. Для каждой функции могут быть определены начальные и конечные события, участники, материальные и информационные потоки. Функции можно декомпозировать на более низкие уровни. Декомпозиция может производиться в нотациях EPC или BPMN.

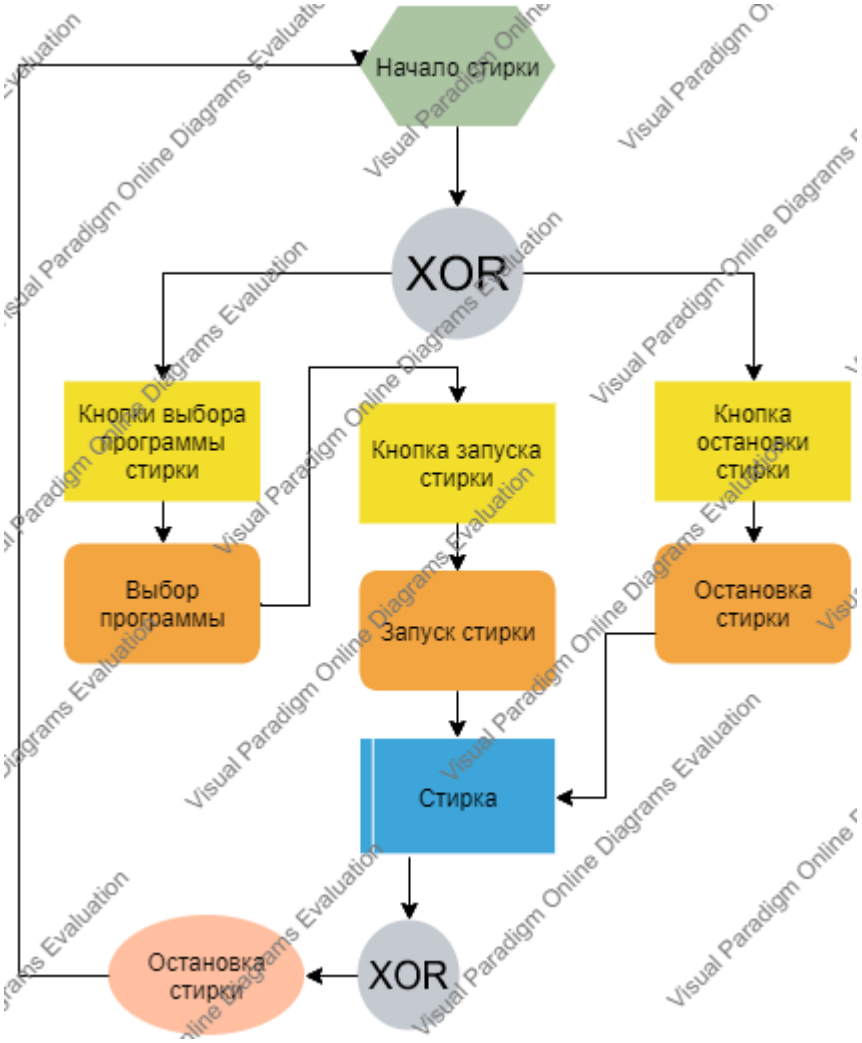


Рисунок 6 – EPC-диаграмма системы «Стиральная машина»

7. Проектирование диаграммы BPMN

BPMN (Business Process Management Notation) – это язык моделирования бизнес-процессов, который является промежуточным звеном между формализацией/визуализацией и воплощением бизнес-процесса.

Основной целью BPMN является обеспечение доступной нотацией описания бизнес-процессов всех пользователей: от аналитиков, создающих схемы процессов, и разработчиков, ответственных за внедрение технологий выполнения бизнес-процессов, до руководителей и обычных пользователей, управляющих этими бизнес-процессами и отслеживающих их выполнение. Таким образом, BPMN нацелен на устранение расхождения между моделями бизнес-процессов и их реализацией.

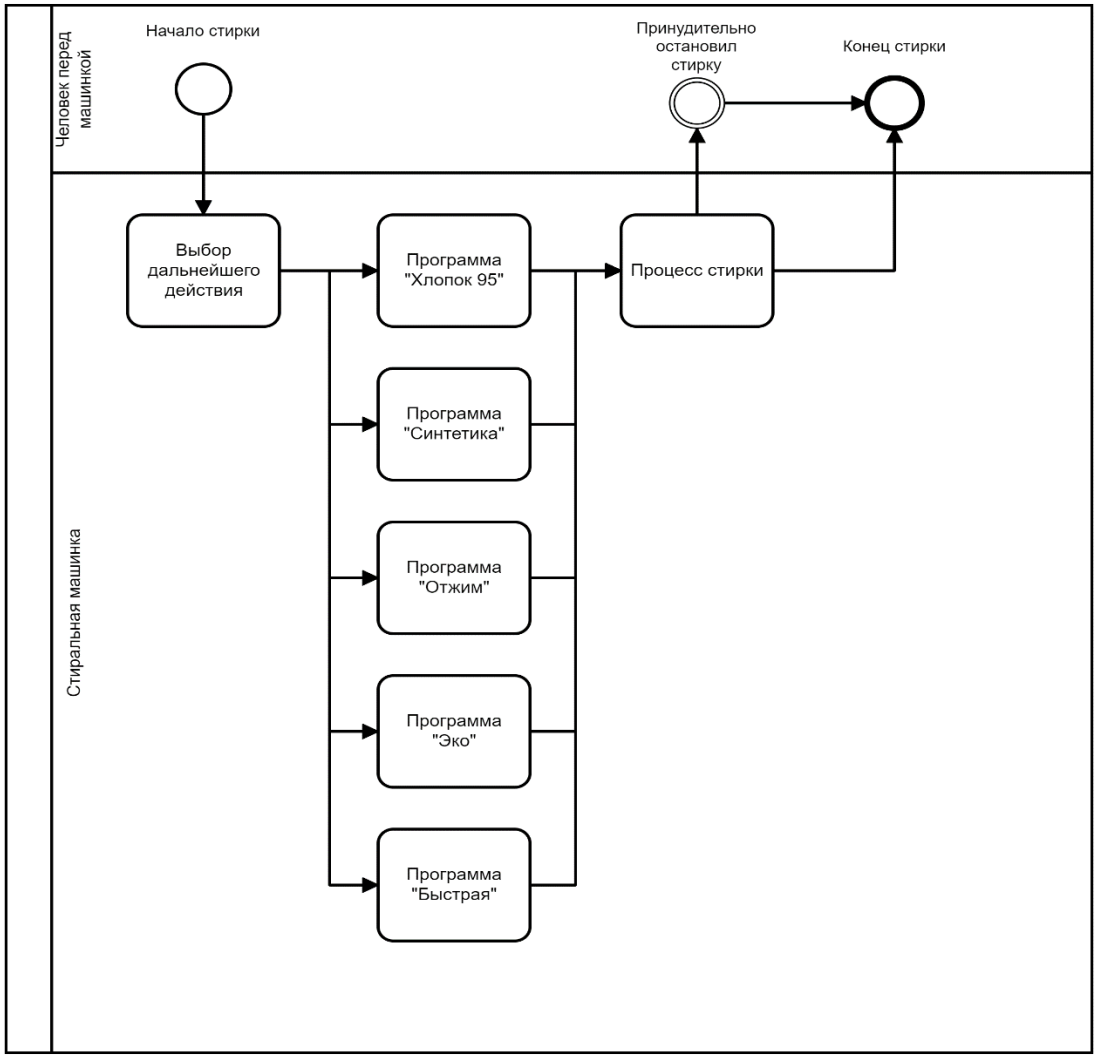


Рисунок 7 – Диаграмма BPMN «As-Is» и «To be»

8. Требования к системе: классификация FURPS+

Сокращение FURPS расшифровывается так:

1. Functionality, функциональность
2. Usability, удобство использования
3. Reliability, надежность
4. Performance, производительность
5. Supportability, поддерживаемость
6. + необходимо помнить о таких возможных ограничениях, как:
    - ограничения проектирования, design
    - ограничения разработки, implementation
    - ограничения на интерфейсы, interface
    - физические ограничения, physical

Если применить к этой классификации популярное разделение требований на функциональные и нефункциональные, то к последним следует отнести все перечисленные выше группы кроме первой, т.е. URPS+.

F – стандартный набор функций;

U – приятный дизайн, интуитивно понятный интерфейс;

R – 1 сбой/5 лет; среднее время сбоя – 2 секунды; время готовности системы к работе – 10 мсек.

P – время отклика системы 0.01 сек, 100% эффективность работы, пропускная способность 10 запросов в минуту; потребление ресурсов – 1 750 Вт/ час;

S – легкая настройка, простая установка, совместимость со всеми схожими моделями по функционалу;

+ - память программ ограничена (максимальный размер 5 программ); нельзя изменить стандартную программу стирки по таким параметрам, как «Температура», «Обороты».

9. Проектирование диаграммы вариантов использования

Диаграмма вариантов использования описывает функциональные требования системы с точки зрения способов её применения. Данная модель позволяет связать то, что нужно разработчикам от системы, с тем, как система удовлетворяет эти потребности. Данная диаграмма должна состоять из актеров, вариантов использования и связей между ними.

Ее суть состоит в том, что система представляется в виде актеров, которые взаимодействуют с системой. Они могут взаимодействовать исходя из различных вариантов использования. Актером может быть любой объект, который может взаимодействовать в исследуемой системой извне. Вариант использования – это набор функций, которая система может предоставить актеру.

Согласно UML актера графически можно изобразить в виде «человечка». Варианты использования обозначаются на диаграмме эллипсом, внутри которого содержится его описание. Вариант использования представляет собой конечную последовательность действий, которую может выполнить актер по отношению к системе.

В данной работе необходимо было реализовать ПО для Стиральной машины, которое представляет собой хранилище программ стирки. Для каждой программы хранятся свои параметры. Пользователь не может изменить начальные настройки, но может остановить стирку и в дальнейшем решить, продолжать текущую стирку или начать новую. Таким образом, можно сделать вывод, что Стиральная машина является одним из актеров. В качестве второго актера можно выделить человека, который будет пользоваться данным устройством.

Учитывая всё вышеперечисленное была построенная диаграмма, вариантов использования, на которой описаны всевозможные действия.

Рисунок 8 – Диаграмма вариантов использования

10. Проектирование диаграммы последовательности

Диаграмма последовательности отражает взаимодействие определенного набора объектов на некоторой временной оси.

Основными ее элементами являются обозначения объектов, линии жизни объектов и стрелки, показывающие обмен сигналами или сообщениями между ними.

На этой диаграмме изображаются только те объекты, которые непосредственно участвуют во взаимодействии. Для диаграммы последовательности ключевым моментом можно назвать именно динамику взаимодействия объектов во времени. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни.

Линия жизни объекта изображается пунктирной вертикальной линией. Она служит для обозначения периода времени, в течение которого объект существует в системе. Отдельные объекты, выполнив свою роль в системе, могут быть уничтожены путем обрыва линии жизни.

Для программы «Стиральная машина», моделируемой в данном курсовом проекте можно выделить несколько вариантов развития событий. Первый – человек хочет включить стиральную машину. Этот процесс показан на рисунке 10:

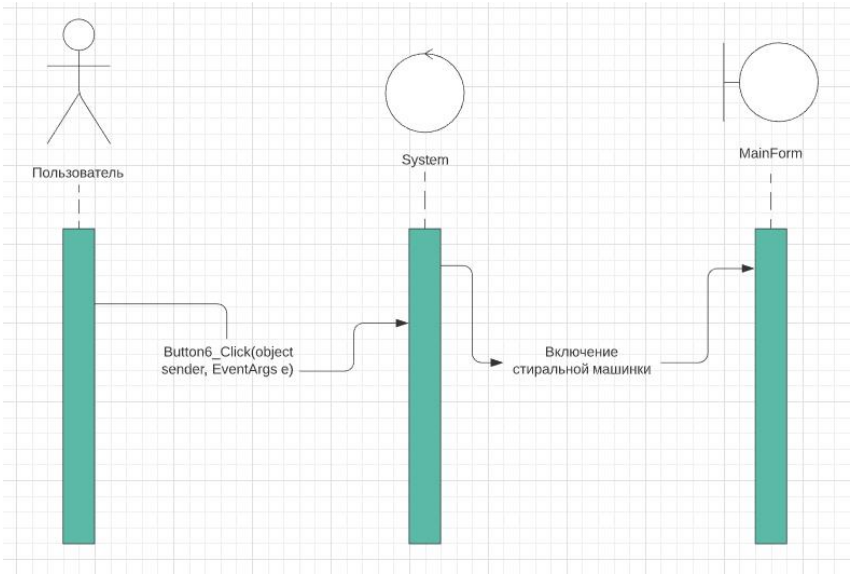


Рисунок 9 – Диаграмма последовательности (включение)

Следующий вариант развития событий показывает, цепочку действий для выбора программы стирки и запуска стирки. Этот процесс показан на рисунке 11:

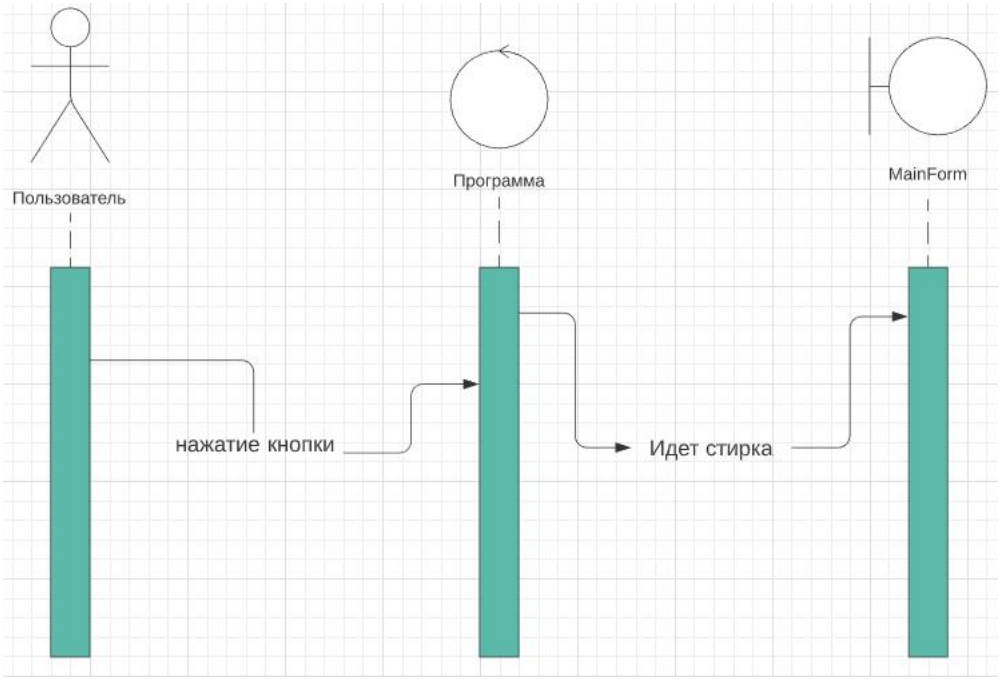


Рисунок 10 – Диаграмма последовательности (выбор программы стирки)

Третий вариант развития событий демонстрирует последовательность действий, необходимую для принудительной остановки стирки. Этот процесс показан на рисунке 12:

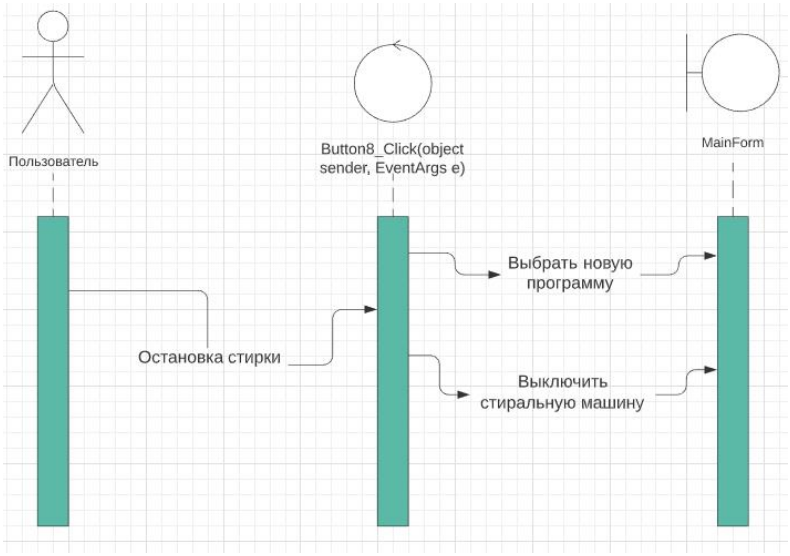


Рисунок 11 – Диаграмма последовательности (принудительная остановка)

Таким образом при помощи диаграммы последовательности были описаны цепочки действий, происходящие между объектами системы при всевозможных развитиях событий.

11. Результаты машинного тестирования программы

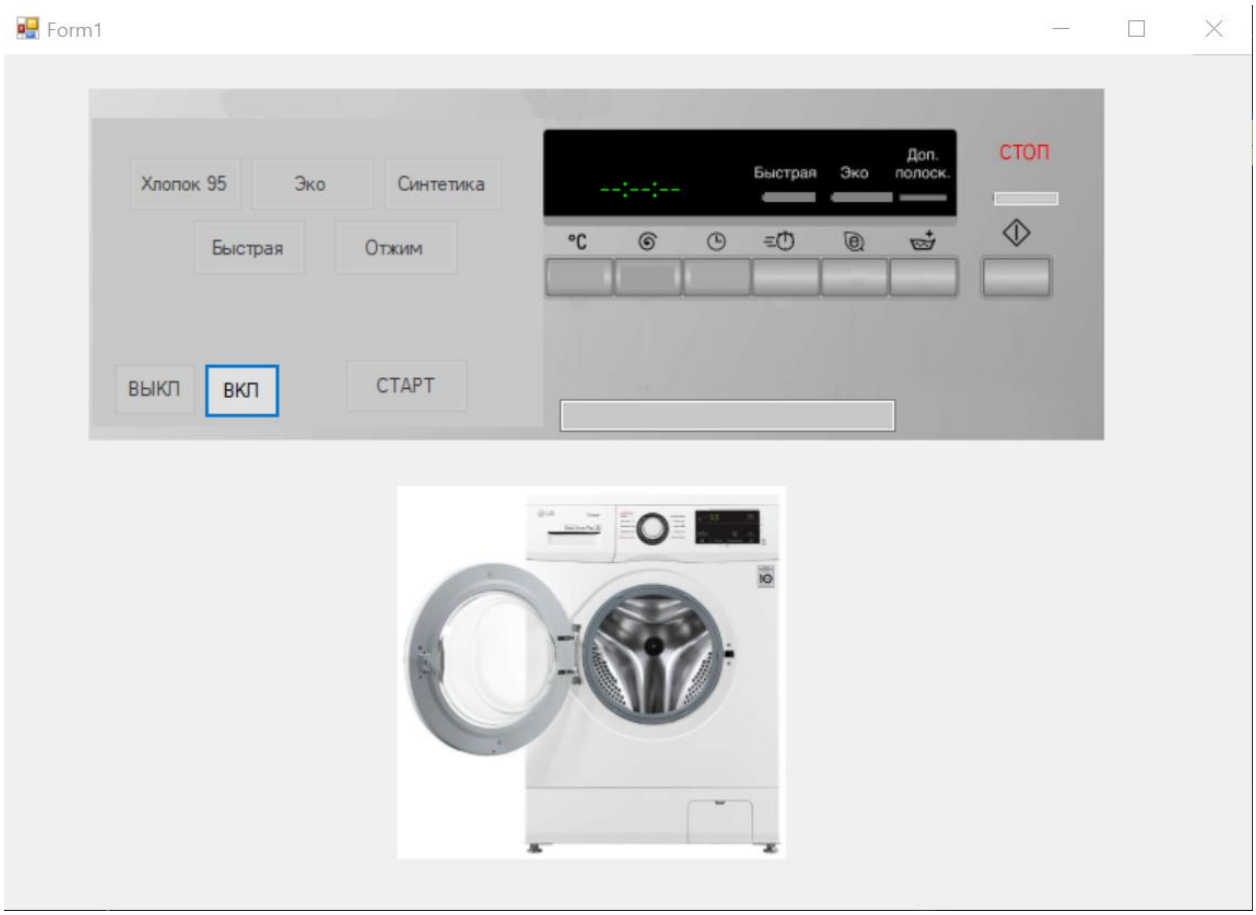


Рисунок 12 – Выключенная машинка

Рисунок 13 – Включенная машинка

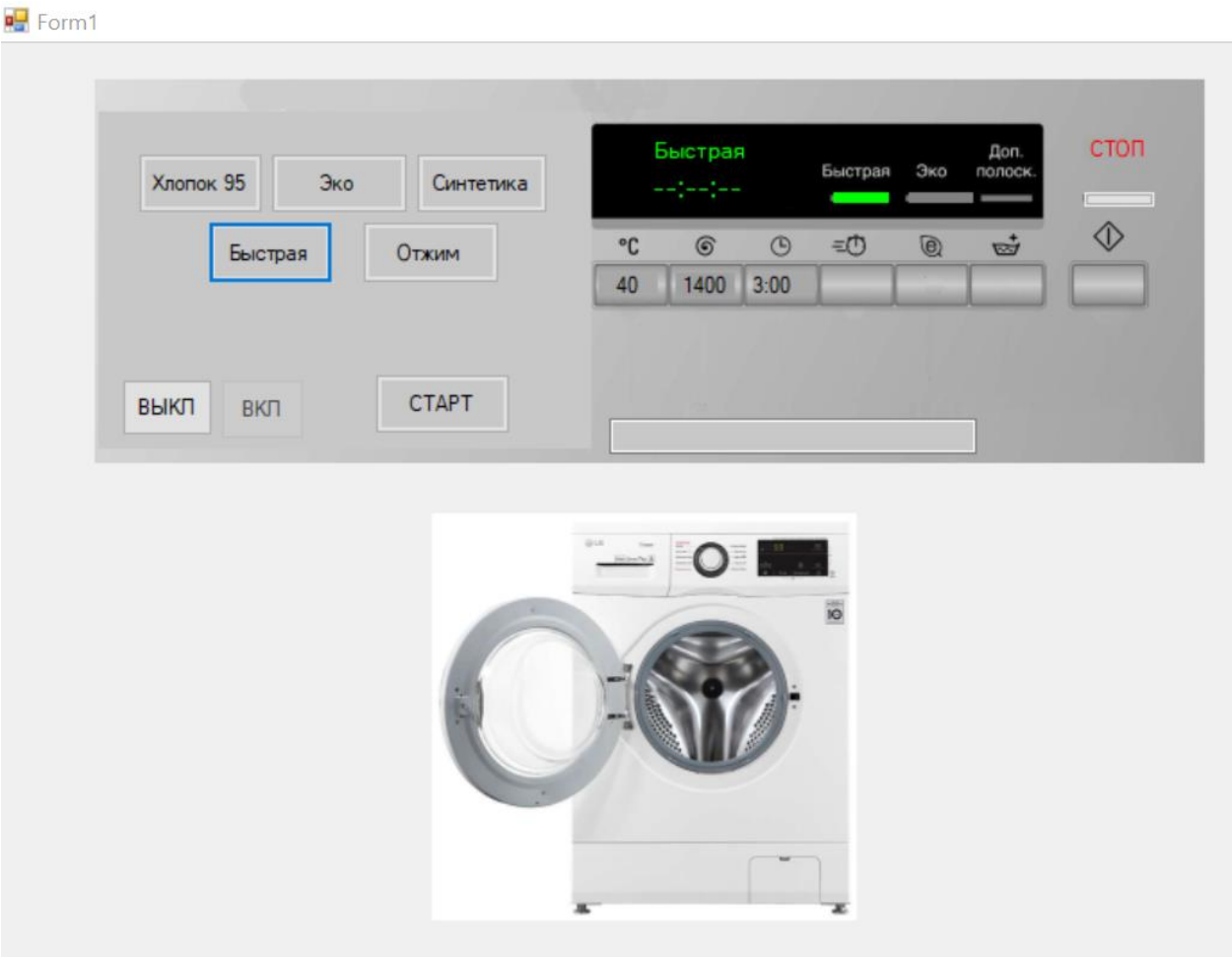


Рисунок 14 – Выбор программы стирки

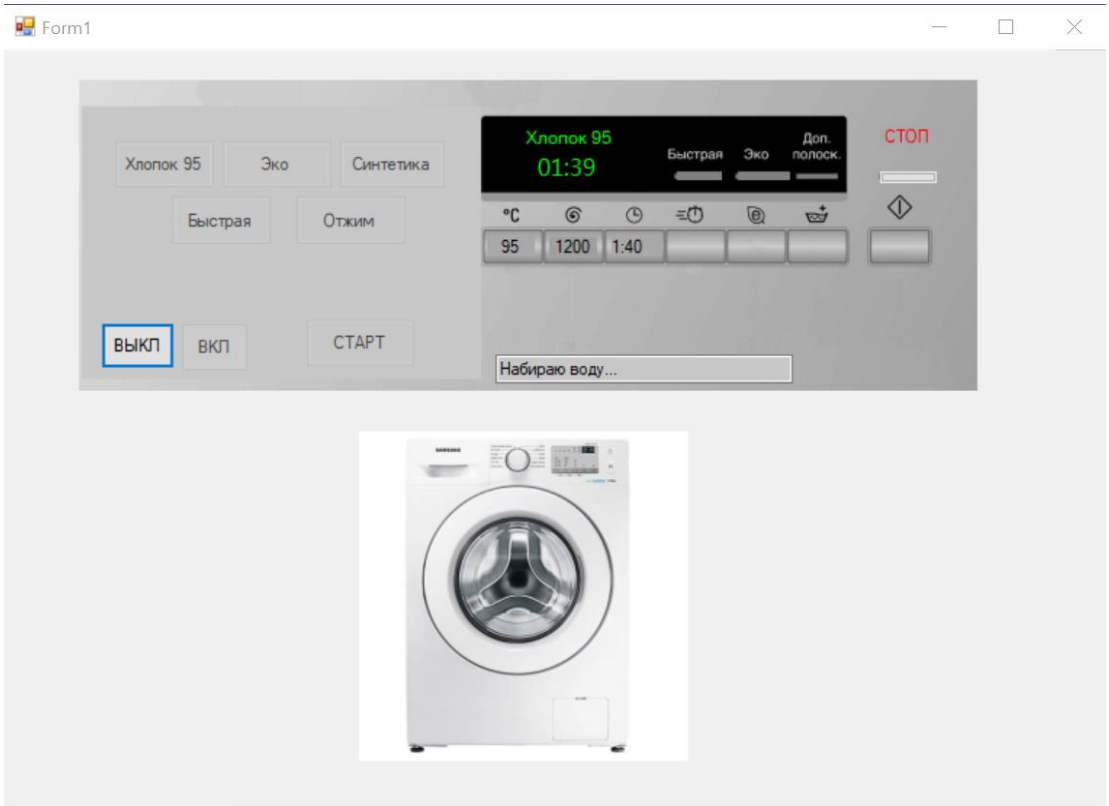


Рисунок 15 – Запуск стирки

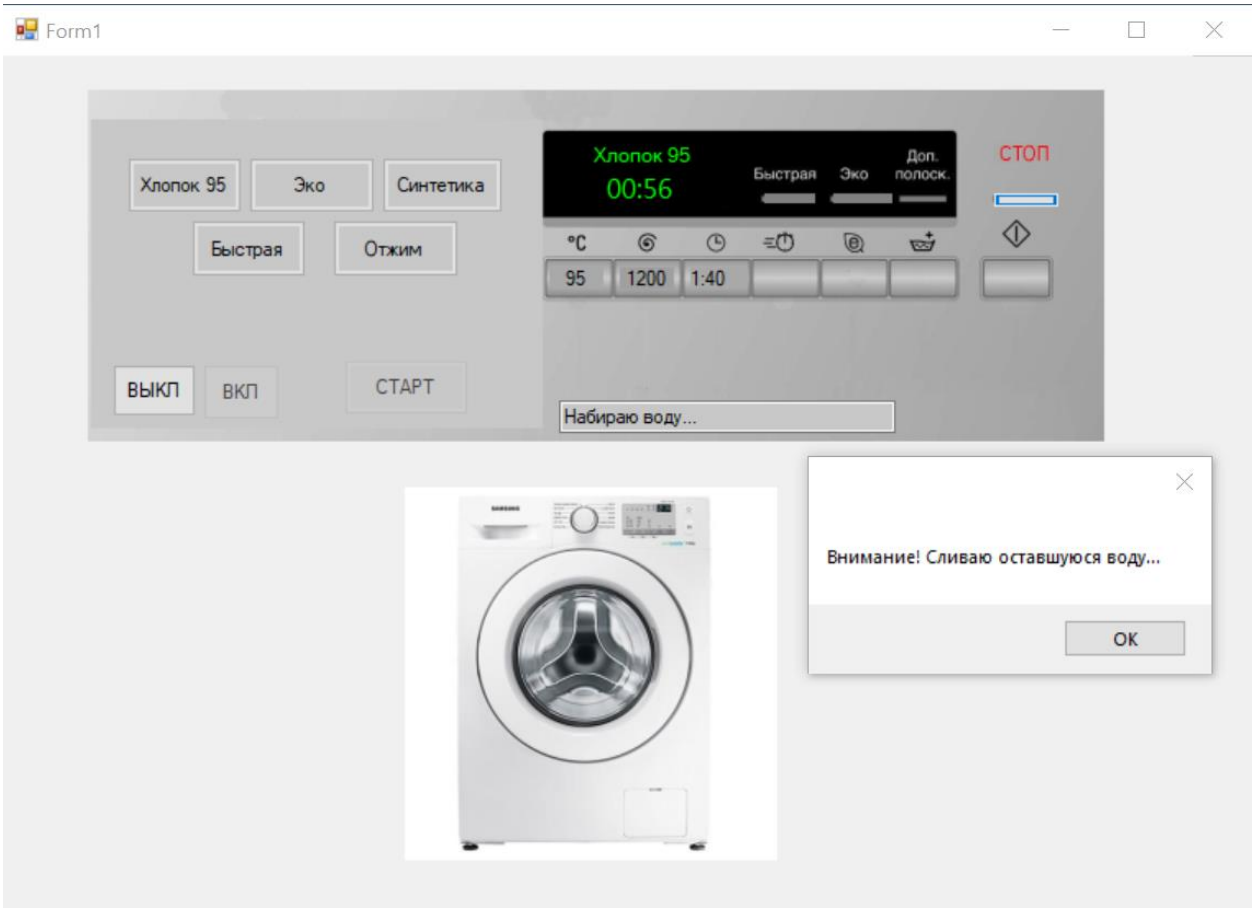


Рисунок 16 – Принудительная остановка стирки

12. Системные требования

Таблица 1 – Системные требования программы

Процессор	250 МГц
Оперативная память	26 Мб
Монитор	640 x 480
Свободное место на носителе	5 Мб
Устройства взаимодействия с пользователем	Клавиатура и мышь
Программное обеспечение	Visual Studio 2019 года последней версии

13. Руководство пользователя

Запустить программу можно с помощью Курсовая тимп.exe, либо открыть sln файл с названием Курсовая тимп.

Запуск программы через exe файл:

1. Находим исполняемый файл;
2. Запускаем исполняемый файл с форматом .exe.

Запуск программы через sln файл:

1. Находим папку проекта;
2. Запускаем файл с расширением .sln;
3. Попадаем в Visual Studio;
4. Далее попадаем в проект и запускаем его.

Теперь перед нами интерфейс программы: симулятор стиральной машины и главной панели управления.

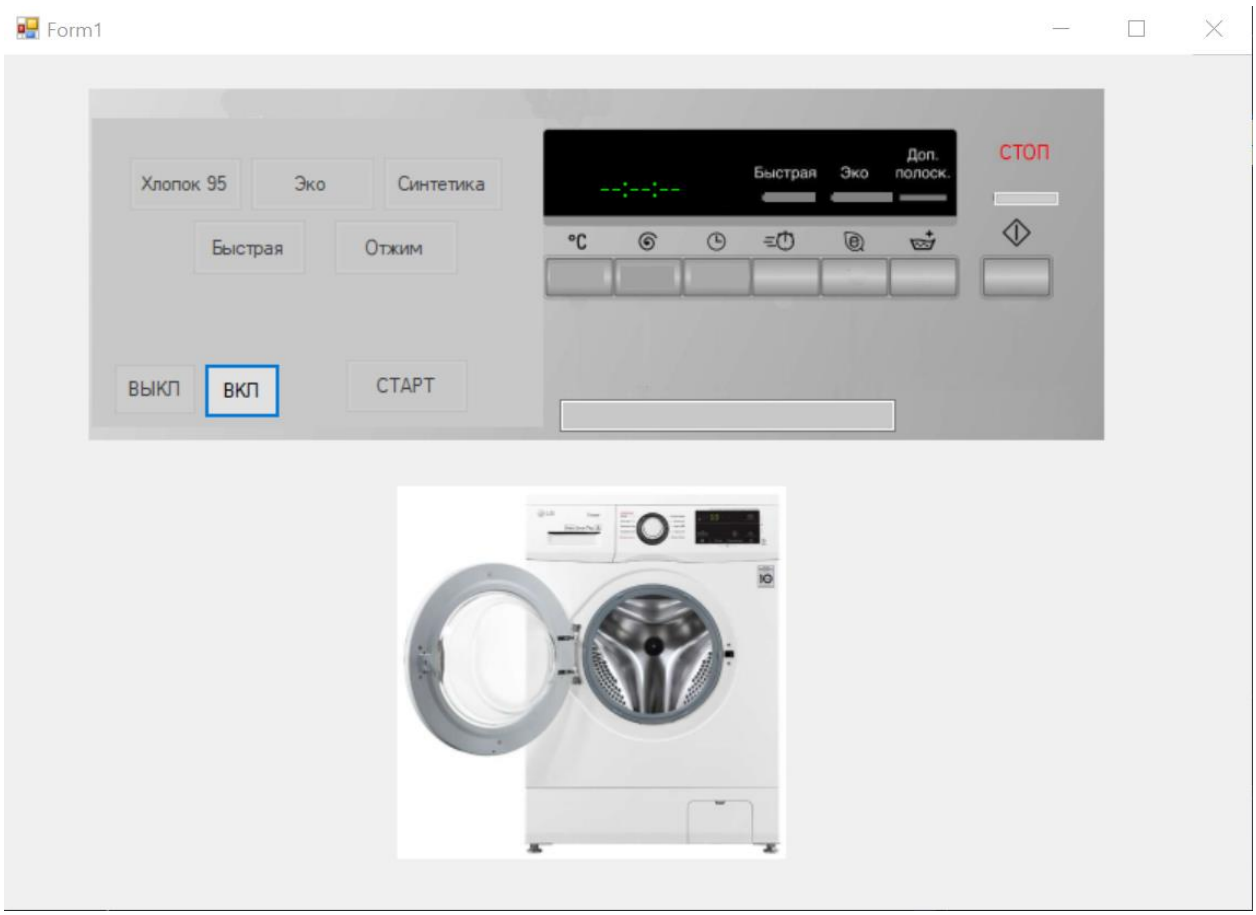


Рисунок 17 – Начальный интерфейс программы

Для начала работы программы пользователь должен включить стиральную машину. Выполнить это можно нажав кнопку «ВКЛ» на панели

управления. Далее следует выбрать нужную программу стирки, которые находятся на панели управления. После выбора программы стирки можно запускать стирку, сделать это можно нажав кнопку «СТАРТ», при этом кнопки с программами стирки станут недоступны.

В любой момент использования программы стиральную машину можно принудительно остановить кнопкой «СТОП», после этого все кнопки будет слита оставшаяся вода и станут доступны все кнопки. Также, в любой момент времени пользователь может отключить стиральную машину, тогда все кнопки станут нерабочими.

14. Сопровождение ПО

Представленное в курсовой работе программное обеспечение документируется в Wiki на гитхаб по данной ссылке:

https://github.com/KalyanovK/Washing-machine/wiki/ Оглавление

На рисунке 18 показано оглавление:

Рисунок 18 - Оглавление

Заключение

В результате выполнения данного курсового проекта было разработано ПО для симулятора стиральной машины. Программирование осуществлялось на языке высокого уровня C#, использование данного языка позволило удобно разработать симулятор и наглядно показать результат работы при помощи встроенного в Visual Studio интерфейса программирования приложений Windows Forms. Также были спроектированы диаграммы, которые позволяют детально изучить не только процесс машинного выполнения программы, но также и оценить процесс создания (проектирования и реализации) данного проекта.

При построении диаграмм применялись основные правила и принципы моделирования, включающие графическое представление объектов и связей между ними, иерархическое построение, а также названия, отражающие назначение той или иной сущности, или взаимодействия.

Благодаря детальному разбору проекта при помощи диаграмм проектирования, полученных в процессе разработки, можно с уверенностью сказать, что разработанное ПО удовлетворяет все потребности потребителя, позволяет комфортно использовать стиральную машину.

Были получены важные знания и практические навыки как в области использования объектно-ориентированных языков программирования в целом, так и в области построения диаграмм проектирования, отображающих поведение различных организационных структур.

Список использованных источников

1. Автоматизация проектирования вычислительных систем. Языки, моделирование и базы данных / ред. М. Брейер. - М.: Мир, 2015. - 463 с.

2. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / КрэгЛарман. - Москва: Гостехиздат, 2017. - 736 с.

3. Слепцов, А.И. Автоматизация проектирования управляющих систем гибких автоматизированных производств / А.И. Слепцов, А.А. Юрасов. - М.: Техника, 2015. - 110 с.

4. Роберт А. Максимчук. UML для простых смертных / Роберт А. Максимчук, Эрик Дж. Нейбург. - Москва: СИНТЕГ, 2014. - 272 с.

5. Йордон, Эдвард. Объектно-ориентированный анализ и проектирование систем / Эдвард Йордон , Карл Аргила. - М.: ЛОРИ, 2014. - 264 с.

6. Википедия - https://ru.wikipedia.org/wiki/DFD.

7. Учебная и научная деятельность с GOOGLE – Лекция «Разработка функциональной модели». Методология DFD - https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_3.

8. Вендров, А. М. Практикум по проектированию программного обеспечения экономических информационных систем / А.М. Вендров. - М.: Финансы и статистика, 2006. - 192 с.

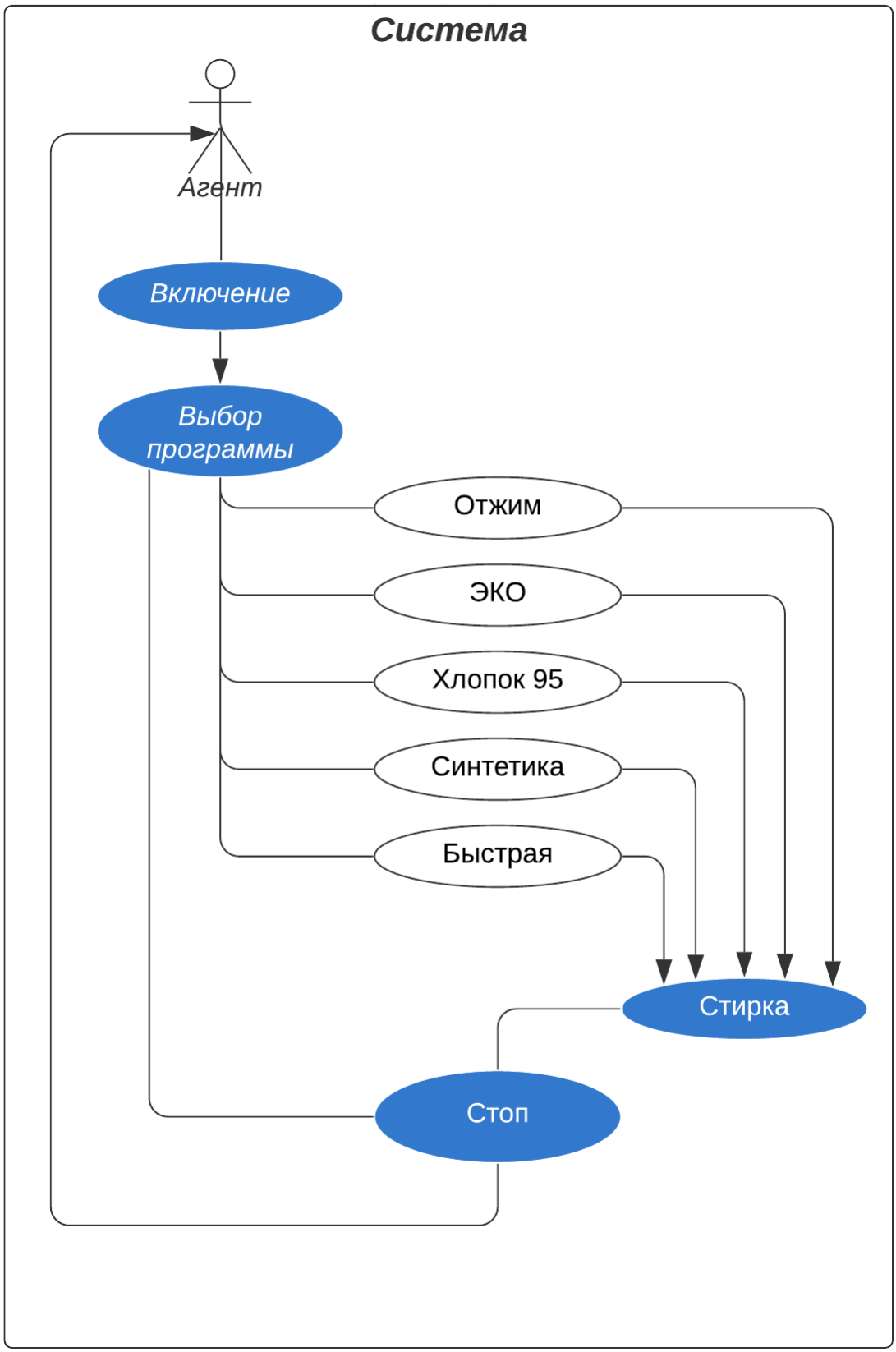
9. Е.Д. Емцева, К.С. Солодухин, С.В. Кучерова. МОДЕЛИРОВАНИЕ И АНАЛИЗ БИЗНЕС-ПРОЦЕССОВ [Текст]: учебное пособие / авт.-сост. Е.Д. Емцева, К.С. Солодухин, С.В. Кучерова. – Владивосток: Изд-во ВГУЭС, 2013. – 76 с. 2013.

10. Моделирование бизнес-процессов в нотации BPMN 2.0 [Текст] : монография, научно-практическое издание / И. Г. Федоров. - М. : МЭСИ , 2013. - 264 с. - ISBN 978-5-7764-0772-7.

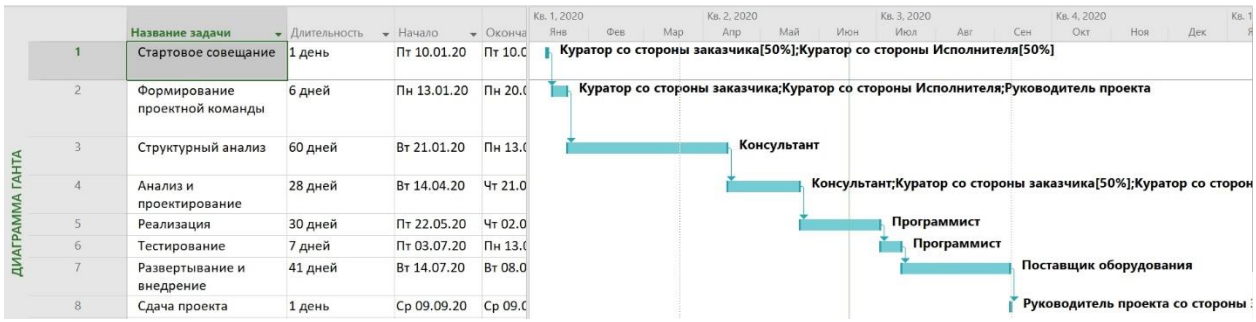
11. Comindware – Нотация BPMN 2.0 [Электронный ресурс]: - Режим доступа: https://comindware.com/ru/blog-нотация-bpmn-2-0-элементы-и-описание/

12. SysAna– Требования к системе: классификация FURPS+ [Электронный ресурс]: - Режим доступа: https://sysana.wordpress.com/2010/09/16/furps/

Приложение А – Диаграмма UML



Приложение Б – Диаграмма Ганта



	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Стартовое совещание	1 день	Пт 10.01.20	Пт 10.01.20		Куратор со стороны заказчика[50%];Куратор со стороны Исполнителя[50%]
2	Формирование проектной команды	6 дней	Пн 13.01.20	Пн 20.01.20	1	Куратор со стороны заказчика;Куратор со стороны Исполнителя;Руководитель проекта
3	Структурный анализ	60 дней	Вт 21.01.20	Пн 13.04.20	2	Консультант
4	Анализ и проектирование	28 дней	Вт 14.04.20	Чт 21.05.20	3	Консультант;Куратор со стороны заказчика[50%];Куратор со стороны Исполнителя[50%]
5	Реализация	30 дней	Пт 22.05.20	Чт 02.07.20	4	Программист
6	Тестирование	7 дней	Пт 03.07.20	Пн 13.07.20	5	Программист
7	Развертывание и внедрение	41 дней	Вт 14.07.20	Вт 08.09.20	6	Поставщик оборудования
8	Сдача проекта	1 день	Ср 09.09.20	Ср 09.09.20	7	Руководитель проекта со стороны Заказчика;Руководитель проекта со стороны Исполнит

Приложение В – Антиплагиат

АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Отчет о проверке на заимствования №1

Автор: kalianov43@gmail.com / ID: 6502269
Проверяющий: (kalianov43@gmail.com / ID: 6502269)

Отчет предоставлен сервисом «Антиплагиат»- http://users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 28
Начало загрузки: 17.06.2020 16:19:55
Длительность загрузки: 00:00:01
Имя исходного файла: ПЗ.pdf
Название документа: ПЗ
Размер текста: 1 кБ
Символов в тексте: 31683
Слов в тексте: 3691
Число предложений: 292

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 17.06.2020 16:19:57
Длительность проверки: 00:00:03
Комментарии: не указано
Модули поиска: Модуль поиска Интернет

ЗАИМСТВОВАНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
22,34%	0%	0%	77,66%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Ссылка	Актуален на	Модуль поиска
[01]	5,65%	Задания для семестровой работы по предмету "Технология разработки ...	http://pandia.ru	раньше 2011	Модуль поиска Интернет
[02]	0%	Экзамен	http://studopedia.net	14 Ноя 2015	Модуль поиска Интернет
[03]	0%	План лекций	http://studfiles.ru	13 Июл 2016	Модуль поиска Интернет

Еще источников: 17
Еще заимствований: 16,7%

Приложение Г – Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Курсовая_тимп
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            button1.Enabled = false;
            button2.Enabled = false;
            button3.Enabled = false;
            button4.Enabled = false;
            button5.Enabled = false;
            button9.Enabled = false;
            button7.Enabled = false;
            button8.Enabled = false;
        }
```

```
public void Lock (int i)
{
   if (i == 1)
   {
      button1.Enabled = false;
      button2.Enabled = false;
      button3.Enabled = false;
      button4.Enabled = false;
      button5.Enabled = false;
   }
   if(i==0)
   {
      button1.Enabled = true;
      button2.Enabled = true;
      button3.Enabled = true;
      button4.Enabled = true;
      button5.Enabled = true;
   }
}
bool voda = false;
int i;
int tk;
string c;
string t;
int interval;



public void Stirka(int time)
{
```

```
        while (time>time*0.95)
        {
           textBox9.Text = "Набираю воду";
           voda = true;

        }
        textBox9.Text = "Набираю воду";
        while (time > time * 0.9)
        {
           textBox9.Text = "Развожу порошок";
        }
        while (time > time * 0.2 && time < time * 0.89)
        {
           textBox9.Text = "Моем моем трубочиста, чисто чисто чисто чисто /n Будет будет трубочист чист чист чист чист";
        }
        while (time > time * 0.06 && time < time * 0.19)
        {
           textBox9.Text = "Идет отжим";
        }
        while (time > time * 0.01 && time < time * 0.05)
        {
           textBox9.Text = "Сливаю воду";
        }
        if (time == 0)
        {
           textBox9.Text = "Конец";
           pictureBox1.BackgroundImage = Properties.Resources.открытаяя_дверь;
        }
```

```
}

private void Form1_Load(object sender, EventArgs e)
{

}

// хлопок 95
private void Button2_Click_1(object sender, EventArgs e)
{

   textBox7.BackColor = Color.Gray;
   textBox6.BackColor = Color.Gray;
   textBox8.Text = "95";
   textBox2.Text = "Хлопок 95";
   textBox3.Text = "1200";
   t = "1:40";
   textBox4.Text = t;
   interval = 100;

}
```

```
// старт
private void Button7_Click(object sender, EventArgs e)
{

   pictureBox1.BackgroundImage = Properties.Resources.закрыто;
   Lock(1);
   button7.Enabled = false;
   button8.Enabled = true;
   i = interval;
   c = t;
   textBox9.Text = "Набираю воду...";
   timer1.Interval = 1000;
   timer1.Enabled = true;
   timer1.Start();




}
// эко
private void Button3_Click(object sender, EventArgs e)
{

      textBox7.BackColor = Color.Lime;
      textBox6.BackColor = Color.Gray;
      textBox8.Text = "35";
      textBox2.Text = "Эко";
      textBox3.Text = "800";
      t = "7:30";
```

```
        textBox4.Text = t;
        interval = 450;



}

// вкл
private void Button6_Click(object sender, EventArgs e)
{
    Lock(0);
    button6.Enabled = false;
    button9.Enabled = true;
    button8.Enabled = true;
    button7.Enabled = true;

}

// выкл
private void Button9_Click(object sender, EventArgs e)
{

    Lock(1);
    button6.Enabled = true;
    button8.Enabled = false;
    button9.Enabled = false;
}

private void Timer1_Tick(object sender, EventArgs e)
{
```

```
    tk = --i;
    TimeSpan span = TimeSpan.FromMinutes(tk);
    string label = span.ToString(@"hh\:mm");
    textBox1.Text = label.ToString();
    if (i < 0)
        timer1.Stop();
}

private void TextBox1_TextChanged(object sender, EventArgs e)
{

}

private void TextBox2_TextChanged(object sender, EventArgs e)
{

}

private void TextBox6_TextChanged(object sender, EventArgs e)
{

}

private void Button5_Click(object sender, EventArgs e)
{
    textBox7.BackColor = Color.Gray;
    textBox6.BackColor = Color.Lime;
    textBox8.Text = "40";
    textBox2.Text = "Быстрая";
```

```
    textBox3.Text = "1400";
    t = "3:00";
    textBox4.Text = t;
    interval = 180;
}

private void Button4_Click(object sender, EventArgs e)
{
    textBox7.BackColor = Color.Gray;
    textBox6.BackColor = Color.Gray;
    textBox8.Text = "50";
    textBox2.Text = "Синтетика";
    textBox3.Text = "1000";
    t = "5:00";
    textBox4.Text = t;
    interval = 300;
}

private void Button1_Click(object sender, EventArgs e)
{
    textBox7.BackColor = Color.Gray;
    textBox6.BackColor = Color.Gray;
    textBox8.Text = "/--/";
    textBox2.Text = "Отжим";
    textBox3.Text = "1600";
    t = "1:30";
    textBox4.Text = t;
    interval = 90;
}
```

```
        private void Button8_Click(object sender, EventArgs e)
        {
            Lock(0);
            timer1.Stop();
            MessageBox.Show("Внимание! Сливаю оставшуюся воду...");
            if (voda == false)
                pictureBox1.BackgroundImage = Properties.Resources.открытаяя_дверь;
            else
                pictureBox1.BackgroundImage = Properties.Resources.закрыто;
        }
    }
}
```