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

Направление подготовки/специальность 09.03.04 Программная инженерия
(код и наименование направления подготовки/специальности)

Профиль/специализация Проектирование и разработка программного обеспечения
(наименование профиля/специализации)

КУРСОВОЙ ПРОЕКТ

по дисциплине Технологии разработки программного обеспечения

на тему „Спиральная машина"

Выполнил студент Воропай Александр Андреевич курса 2 группы 19-КБ-ПР1
(фамилия, имя, отчество)

Допущен к защите 15.12.2020
(дата)

Руководитель (нормоконтролер) работы Попова О.Б. Фамилия И.О.

Защищена 17.12.2020 Оценка отл
(дата)

Члены комиссии: Кушнир Н.В.

Тотухов К.Е.

(должность, подпись, дата, расшифровка подписи)

Краснодар
2020 г.

Институт Компьютерных систем и информационной безопасности
Кафедра Информационных систем и программирования
Направление подготовки/специальность 09.03.04. Программная инженерия
(код и наименование направления подготовки/специальности)

Профиль/специализация Проектирование и разработка программного обеспечения
(наименование профиля/специализации)

УТВЕРЖДАЮ
Зав. Кафедрой
«08» 09 2020 г.

ЗАДАНИЕ
на курсовой проект

Студенту Воропаю Александру Андреевичу курса 2 группы 19-КБ-ПР1
Тема работы: «Виртуальная машина»
(утверждена указанием директора института №___от____.20__ г.)

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

Объем работы:
а) пояснительная записка 62 с.
б) иллюстративная часть _____ лист(-ов)

Рекомендуемая литература:
1 Роберт А. Максимчук. UML для простых смертных
2 Гради Буч, Джеймс Рамбо, Ивар Якобсон. Объектно-ориентированный анализ и проектирование сис.

Срок выполнения: с «01» 09 по «30» 12 2020 г.

Срок защиты: «21» 12 2020 г.

Дата выдачи задания: «08» 09 2020 г.

Дата сдачи работы на кафедру: «21» 12 2020 г.

Руководитель работы доц. Попова О.Б
(должность, подпись)

Задание принял студент Воропай А.А.
(подпись)

Реферат

Курсовая работа: 62 страниц, 39 рисунков, 5 используемых источников, 1 таблицу.

Ключевые слова: ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, ПОСЛЕДОВАТЕЛЬНОСТИ, МОДЕЛЬ, КЛАСС, СТИРАЛЬНАЯ МАШИНА, UML, FORM, ГАНТ, IDEF0, IDEF1X, IDEF3, ДИАГРАММЫ.

Объектом исследования является программное обеспечение и симулятор стиральной машины, который способен принимать команды от пользователя и выполнять процесс стирания одежды.

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

В результате были получены диаграммы, обладающие исчерпывающей информацией о программном обеспечении стиральной машины. К ним относятся: диаграмма Ганта, сетевая диаграмма этапов, временная диаграмма распределения работников по этапам, IDEF0-диаграмма, IDEF1X-диаграмма, IDEF3-диаграмма, диаграмма классов, диаграмма компонентов и размещений, диаграмма пакетов, диаграмма последовательности, диаграмма кооперации, диаграмма состояний, диаграмма вариантов использования.

Содержание

Введение

В настоящее время стиральные машины служат одним из основных устройств в каждом доме. Столь желанная и нужная бытовая техника максимально облегчает процесс стирания одежды. Применение стиральных машин стала неотъемлемой частью человеческого бытия.

Однако, несмотря на повсеместное использование такого оборудования, машина не смогла бы выполнять свои функции на таком же эффективном уровне без тщательного исследования технической области и проведения различных тестов, учитывающих всевозможные взаимодействия с компонентами системы. Для уменьшения денежных затрат стали использовать симуляторы – «имитаторы» (обычно механическое или компьютерные), задача которых состоит в имитировании управления каким-либо процессом, аппаратом или транспортным средством.

Таким образом, симулятор стиральной машины – имитатор системы, который позволяет проследить взаимодействие программных и механических блоков.

Воропай А. А. выполняет построение диаграммы Ганта, сетевую диаграмму этапов, временную диаграмму распределения работников по этапам, IDEF0, IDEF1X, IDEF3, диаграмму классов, диаграмму вариантов использования, диаграммы компонентов и размещений, диаграмму пакетов, диаграмму последовательности, диаграмму кооперации, диаграмму состояний.

Кравцов О. Ю. выполняет тщательное тестирование программы, в частности: тестирование по принципу черного и белого ящика, модульное тестирование, выполнил профилирование программы, благодаря которой можно было отследить загрузку CPU и оперативной памяти. Также занимался разработкой программного обеспечения, начиная с визуального оформления, заканчивая кодированием программы.

1 Формулировка задачи

Задачей данного курсового проекта является разработка модели программного обеспечения встроенной системы управления работой стиральной машины.

Стиральная машина предназначена для выполнения процесса стирания одежды. Она подключена к электрической сети. В ней имеется кнопочная панель, дисплей, барабан, таймер. В начальном состоянии таймер остановлен на нулевой отметке, машина не реагирует на нажатия кнопок, кроме открытия/закрытия. При этом цвет кнопки открытия барабана – красный, что указывает, что барабан закрыт. После нажатия кнопки цвет становится зеленым и происходит открытие барабана. При добавлении одежды в барабан появится дисплей, имеющий цветовой фон, зависящий от загруженного веса одежды, а также вес одежды, отображенный поверх этого фона. Стиральная машина имеет максимальный вес загрузки и не начнет стирку при перегрузке, на что указывает красный цвет фона. После загрузки одежды нужно повторно нажать на кнопку открытия/закрытия барабана и выбрать тип ткани, температуру и режим работы машины и нажать на «Пуск», цвет кнопки «Пуск» изменится на зеленый. В зависимости от всех предыдущих факторов будет рассчитано время, которое будет отображаться на таймере до окончания процесса стирания одежды. После окончания таймера барабан будет открыт, цвет кнопочной панели вернется к стандартному варианту, дисплей отображающий вес одежды в барабане будет отключен.

2 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. 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.

Ключевым понятием диаграммы Ганта является «веха» — метка значимого момента в ходе выполнения работ, общая граница двух или более задач. Вехи позволяют наглядно отобразить необходимость синхронизации, последовательности в выполнении различных работ. Вехи, как и другие границы на диаграмме, не являются календарными датами. Сдвиг вехи приводит к сдвигу всего проекта. Поэтому диаграмма Ганта не является, строго говоря, графиком работ. Кроме того, диаграмма Ганта не отображает значимости или ресурсоемкости работ, не отображает сущности работ (области действия). Для крупных проектов диаграмма Ганта становится чрезмерно тяжеловесной и теряет всякую наглядность.

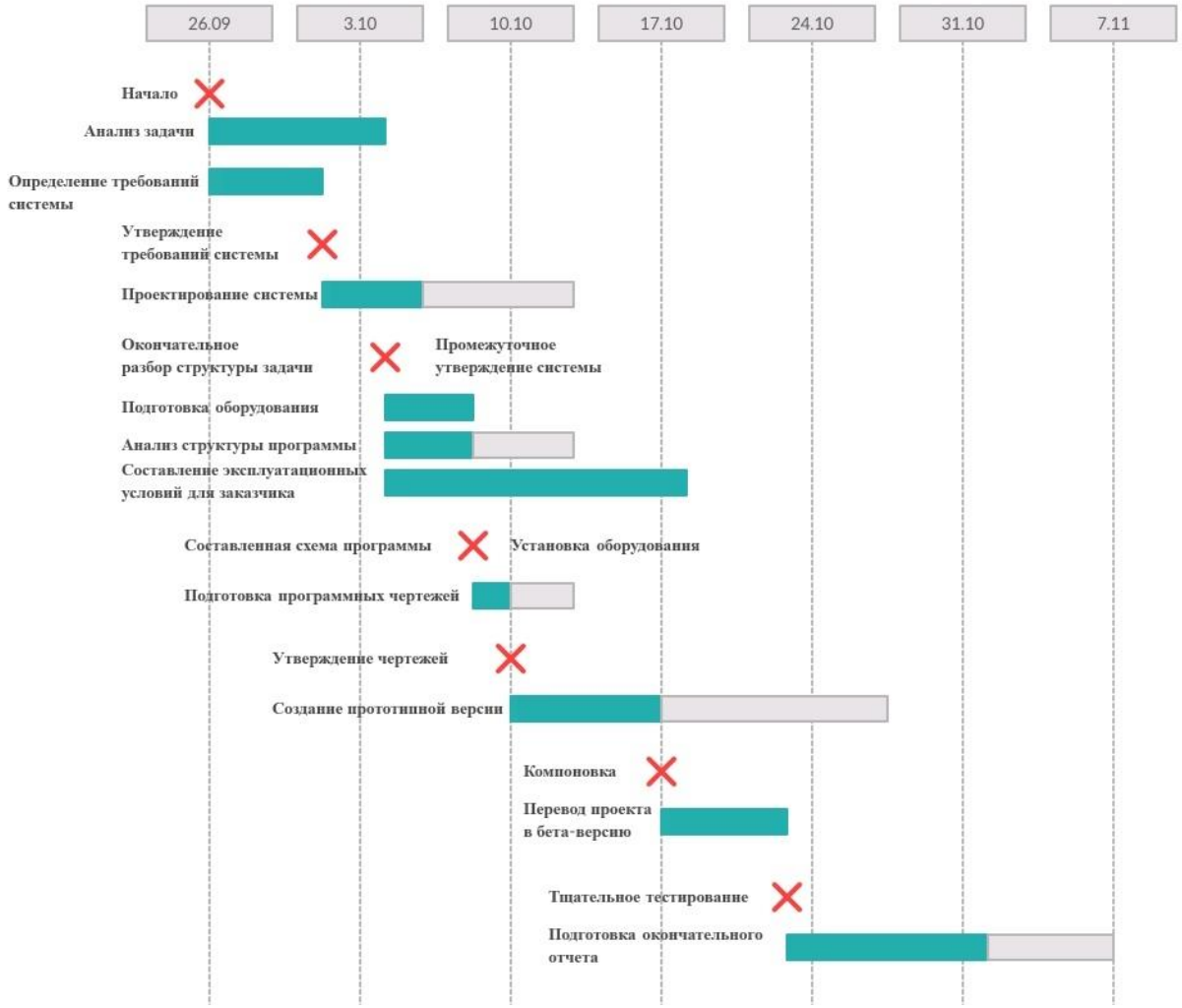


Рисунок 1 – Диаграмма Ганта

Рисунок 2 – Временная диаграмма распределения работников по этапам

3 Сетевая диаграмма

Сетевая диаграмма отображает зависимости между различными этапами проекта. Если для создания сетевой диаграммы используются программные средства поддержки управления проектом, каждый этап должен заканчиваться контрольной отметкой. Очередной этап может начаться только тогда, когда будет получена контрольная отметка (которая может зависеть от нескольких предшествующих этапов).

Любой этап не может начаться, пока не выполнены все этапы на всех путях, ведущих от начала проекта к данному этапу. Минимальное время выполнения всего проекта можно рассчитать, просуммировав в сетевой диаграмме длительности этапов на самом длинном пути от начала проекта до его окончания. Таким образом, общая продолжительность реализации проекта зависит от этапов работ, находящихся на критическом пути. Любая задержка в завершении любого этапа на критическом пути приведет к задержке всего проекта.

Задержка в завершении этапов, не входящих в критический путь, не влияет на продолжительность всего проекта до тех пор, пока суммарная длительность этих этапов на каком-нибудь пути не превысит продолжительности работ на критическом пути.

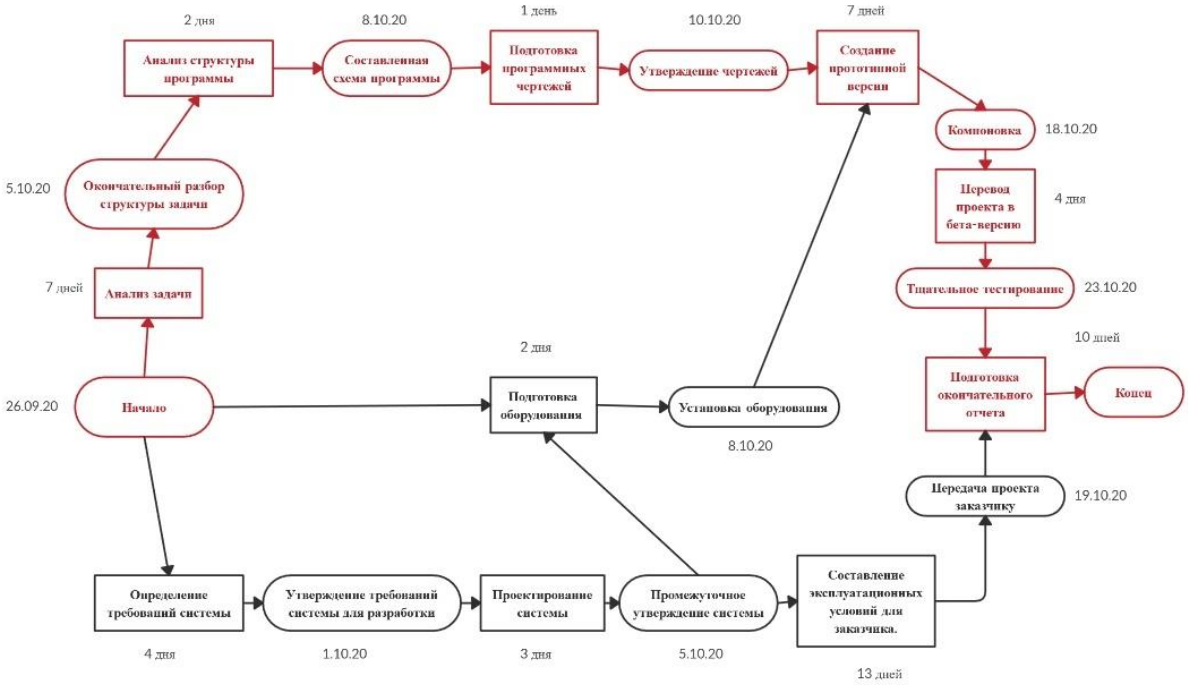


Рисунок 3 – Сетевая диаграмма

4 Создание модели As-Is в стандарте IDEF0

Чтобы оценить возможности разрабатываемой системы необходимо 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 графическая нотация, предназначенная для описания бизнес-процессов. Система, описываемая в данной нотации, проходит через

декомпозицию или, иными словами, разбиение на взаимосвязанные функции. 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 4 была составлена модель As-Is проекта «Стиральная машина».

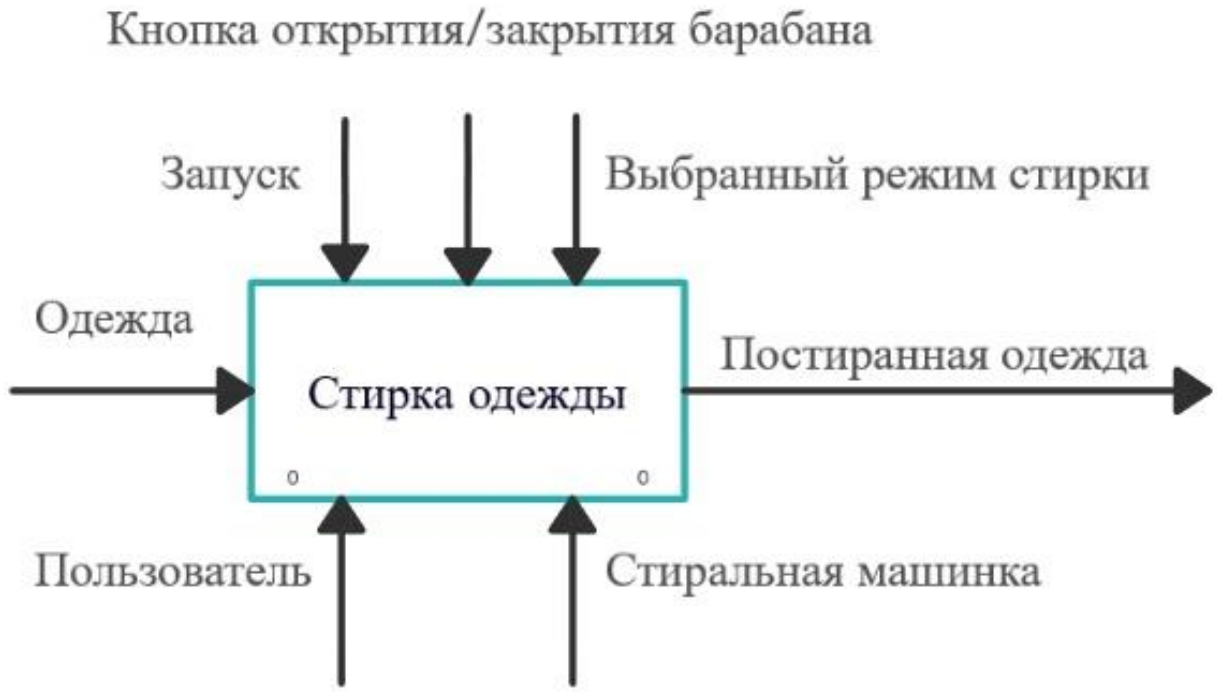


Рисунок 4 – Модель As-Is проекта «Стиральная машина»

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

На рис. 5 изображена модель «Стирка одежды» после декомпозиции.

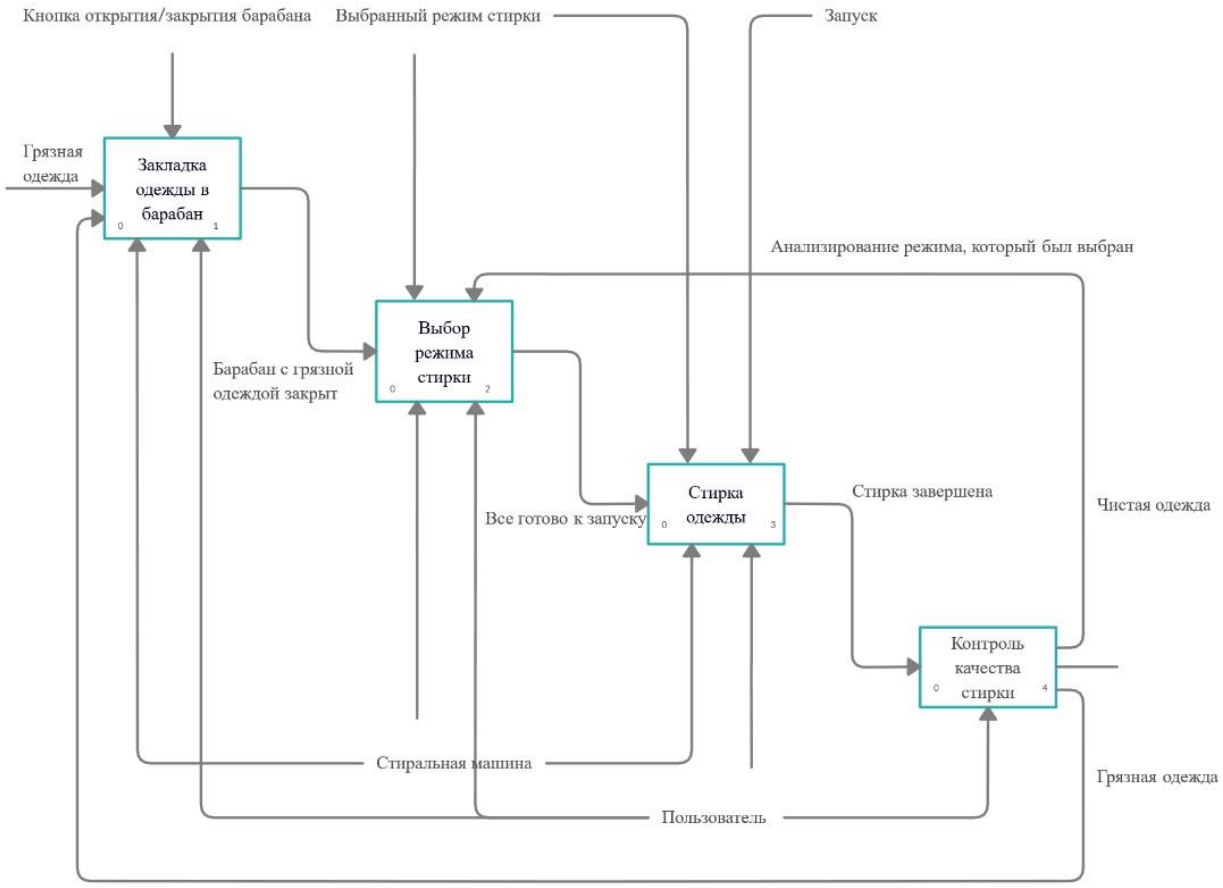


Рисунок 5 – Модель As-Is после декомпозиции

Таким образом работа, рассматриваемой системы разбивается на четыре основанных функции:

1. Закладка одежды в барабан
2. Выбор режима стирки
3. Стирка одежды
4. Контроль качества стирки

5 Метод описания процессов IDEF3

Для описания логики взаимодействия информационных потоков наиболее подходит IDEF3, называемая также workflow diagramming – методологией моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов.

IDEF3 – это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Основные описательные блоки диаграммы IDEF3:

1. Работы – являются центральными компонентами модели, изображаются прямоугольниками с прямыми углами и имеют имя, обозначающее процесс действия.

2. Связи – показывают взаимоотношение работ.

3. Перекрестки. Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончание нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Типы перекрестков проиллюстрированы на рис. 6.

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
&	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
&	Synchronous AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
O	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
O	Synchronous OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
X	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Рисунок 6 – Типы перекрестков

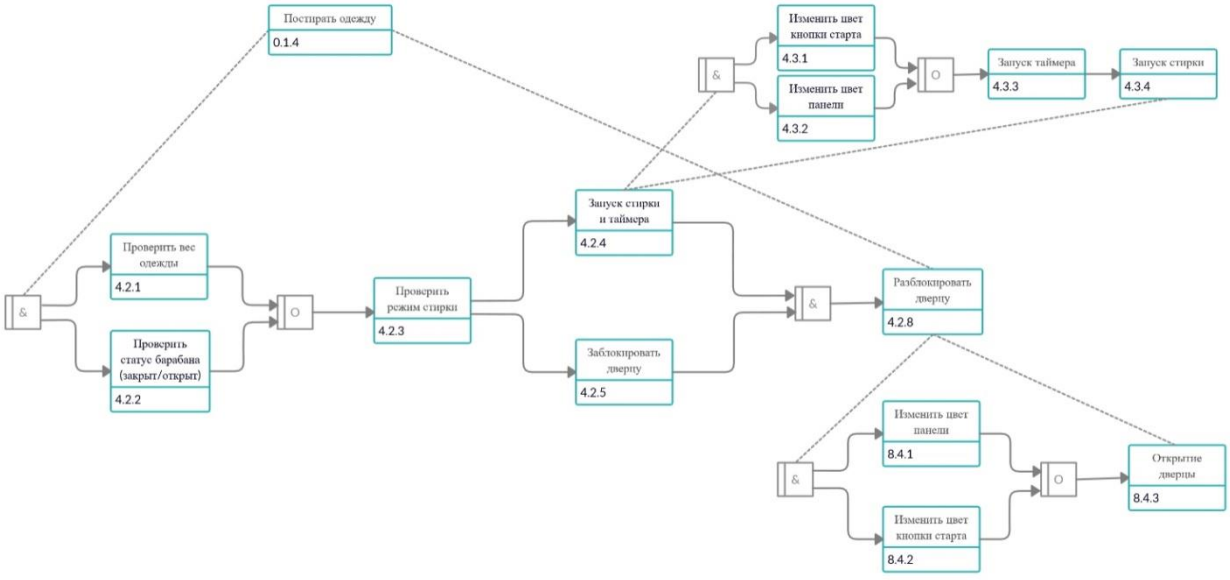


Рисунок 7 – Диаграмма IDEF3

6 Методология IDEF1X

Методология моделирования IDEF1X предназначена для описания данных. Чаще всего такая методология используется для описания данных в целях последующей автоматизации их обработки с помощью систем управления базами данных.

Основные элементы модели IDEF1X:

1. Сущности – это множество объектов, обладающих общими характеристиками.
2. Атрибуты – характеристики сущности.
3. Отношения – это связи между двумя и более сущностями.

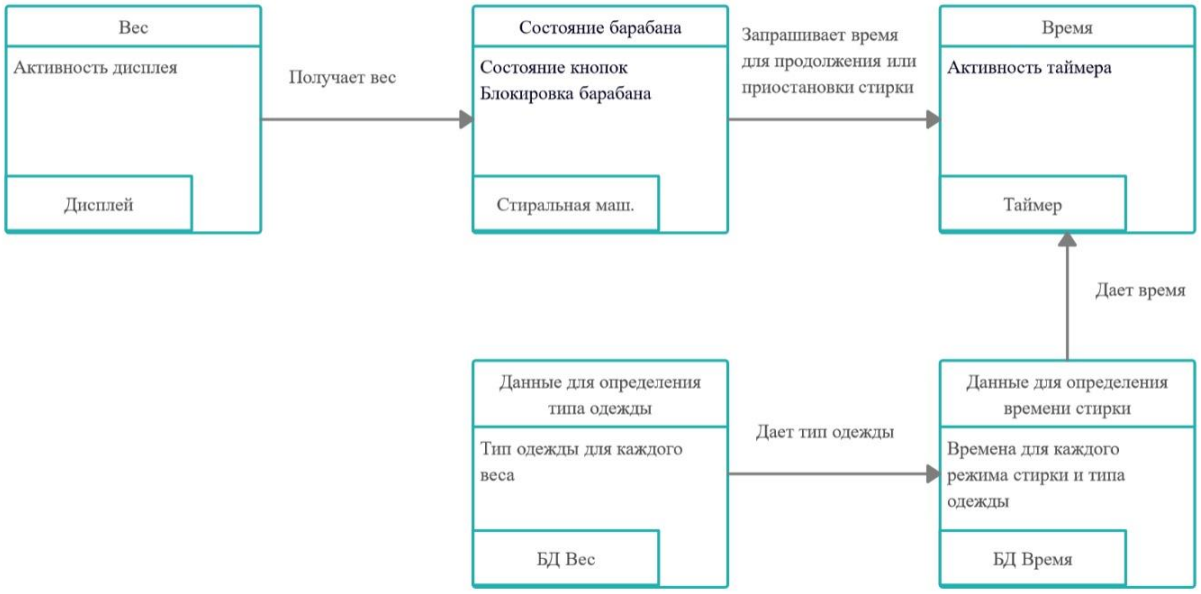


Рисунок 8 – Диаграмма IDEF1X

7 Расчет трудоемкости методом функциональных точек

Оценка трудоемкости создания программного обеспечения является важной составляющей в определении сроков реализации программного проекта и его выполнимости. Модели и методы оценки трудоемкости

используются для разработки бюджета проекта, анализа степени риска и выбор компромиссного решения, планирования и управления проектом.

Общее количество функциональных точек (FP-оценок) рассчитывается по формуле $FP = X * (0{,}65 + 0{,}01 * \sum_{i=1}^{14} Fi)$, где X – суммарное количество функциональных точек для каждого бизнес-процесса; Fi – коэффициенты регулировки сложности.

На рисунке 8 изображена таблица для расчета коэффициентов регулировки сложности.

Факторы, влияющие на сложность программного продукта

Наименование фактора	Показатель оценки фактора
1. Передача данных	Количество средств связи, необходимых для передачи или обмена информацией с приложением или системой
2. Распределенная обработка данных	Количество и сложность вычислительных процессов, требующих распределенной обработки данных
3. Производительность	Потребность пользователя в фиксации времени ответа или производительности
4. Распространенность используемой конфигурации	Уровень распространенности аппаратно-программной платформы, на которой будет выполняться приложение
5. Частота транзакций	Частота выполнения транзакций (ежеминутно, ежедневно, ежемесячно)
6. Оперативный ввод данных	Доля информации, которую необходимо вводить в режиме реального времени
7. Эффективность работы конечного пользователя	Уровень повышения производительности работы пользователя, использующего приложение
8. Оперативное обновление	Количество внутренних файлов, обновляемых в онлайновой транзакции
9. Сложность обработки	Способность приложения выполнять интенсивную логическую или математическую обработку данных
10. Повторная используемость	Возможность использования отдельных компонентов программного продукта в будущих проектах
11. Простота инсталляции	Трудность преобразования и инсталляции приложений
12. Простота эксплуатации	Эффективность и/или уровень автоматизации процедур запуска, резервирования и восстановления
13. Разнообразные условия размещения	Возможность применения программного продукта в разных местах, разных организациях
14. Простота изменений	Наличие в приложении процедуры поддержки внесения изменений с максимальной простотой

Рисунок 9 – Факторы, влияющие на сложность программного продукта

Чтобы определить количество FP-оценок надо обратиться к диаграммам IDEF0 (рис. 5) и IDEF1X (рис. 8).

Таблица расчета количества функциональных точек изображена на рисунке 9.

Наименование функции	Количество функциональных точек
1. Определение количества выводов	4*5
2. Определение количества вводов	4*5
3. Определение количества опросов вывода	0
4. Определение количества опросов ввода	12*5
5. Определение количества файлов	0
6. Определение количества интерфейсов	0
Общее количество функциональных точек	100

Рисунок 10 – Таблица расчета количества функциональных точек

Расчет происходил следующим образом:

1. При использовании для получения FP-метрик моделей Idef0 и Idef1x выводы можно определять на основе стрелок, исходящих из рассматриваемого процесса модели Idef0 и соответствующих им сущностей модели Idef1x.

2. При использовании для получения FP-метрик моделей Idef0 и Idef1x вводы можно определять на основе стрелок, входящих в рассматриваемый процесс модели Idef0 и соответствующих им сущностей модели Idef1x.

3,4. Под запросами при расчете FP-оценок следует понимать диалоговый ввод/вывод, который немедленно приводит к немедленному программному ответу.

5. Количество внешних файлов с данными.

6. Под интерфейсами следует понимать структуры данных, получаемых из внешних программных систем и структуры данных, передаваемые во внешние программные системы.

Не стоит забывать и о коэффициентах сложности, которые прямо влияют на расчет количества функциональных точек. Таблицы расчетов изображены на рисунке 11.

Весовые коэффициенты сложности выводов

Количество структур данных	Значение коэффициента α в зависимости от количества элементов данных		
	от 1 до 5, α_{11}	от 6 до 19, α_{12}	20 и более, α_{13}
1	4	4	5
2–3	4	5	7
4 и более	5	7	7

Весовые коэффициенты сложности вводов

Количество структур данных	Значение коэффициента α в зависимости от количества элементов данных		
	от 1 до 5, α_{11}	от 6 до 19, α_{12}	20 и более, α_{13}
1	4	4	5
2–3	4	5	7
4 и более	5	7	7

Весовые коэффициенты сложности запросов

Количество структур данных	Значение коэффициента α в зависимости от количества элементов данных		
	от 1 до 5, α_{11}	от 6 до 19, α_{12}	20 и более, α_{13}
1	4	4	5
2–3	4	5	7
4 и более	5	7	7

Весовые коэффициенты сложности структур данных

Количество логических взаимосвязей	Значение коэффициента α в зависимости от количества элементов данных		
	от 1 до 5, α_{11}	от 6 до 19, α_{12}	20 и более, α_{13}
1	7	7	7
2–5	7	10	10
6 и более	10	15	15

Весовые коэффициенты сложности интерфейсов

Количество логических взаимосвязей	Значение коэффициента α в зависимости от количества элементов данных		
	от 1 до 5, α_{11}	от 6 до 19, α_{12}	20 и более, α_{13}
1	5	5	7
2–5	7	7	10
6 и более	7	10	10

Рисунок 11 – Таблицы расчетов весовых коэффициентов

Относительно этого и диаграмм Idef0 и Idef1x мы получаем количество функциональных точек, которые позже складываются.

Производим вычисления:

W=0.65+(0.01*20)

Уточненное количество функциональных точек

R(F)=100*0.85=85

Размерность ПО для C#

R(LOC)=85*5,4=459

8 Диаграмма вариантов использования

Понятие варианта использования впервые ввел Ивар Якобсон. В настоящее время вариант использования превратился в основной элемент разработки и планирования проекта.

Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым действующим лицом.

Действующее лицо – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ.

При этом на диаграмме вариантов использования существует три типа связи:

1. Связь коммуникации – это связь между вариантом использования и действующим лицом.
2. Связь включения применяется, когда имеется фрагмент поведения системы, который повторяется больше чем в одном варианте использования.

3. Связь расширения позволяет варианту использования при необходимости использовать функциональные возможности другого.

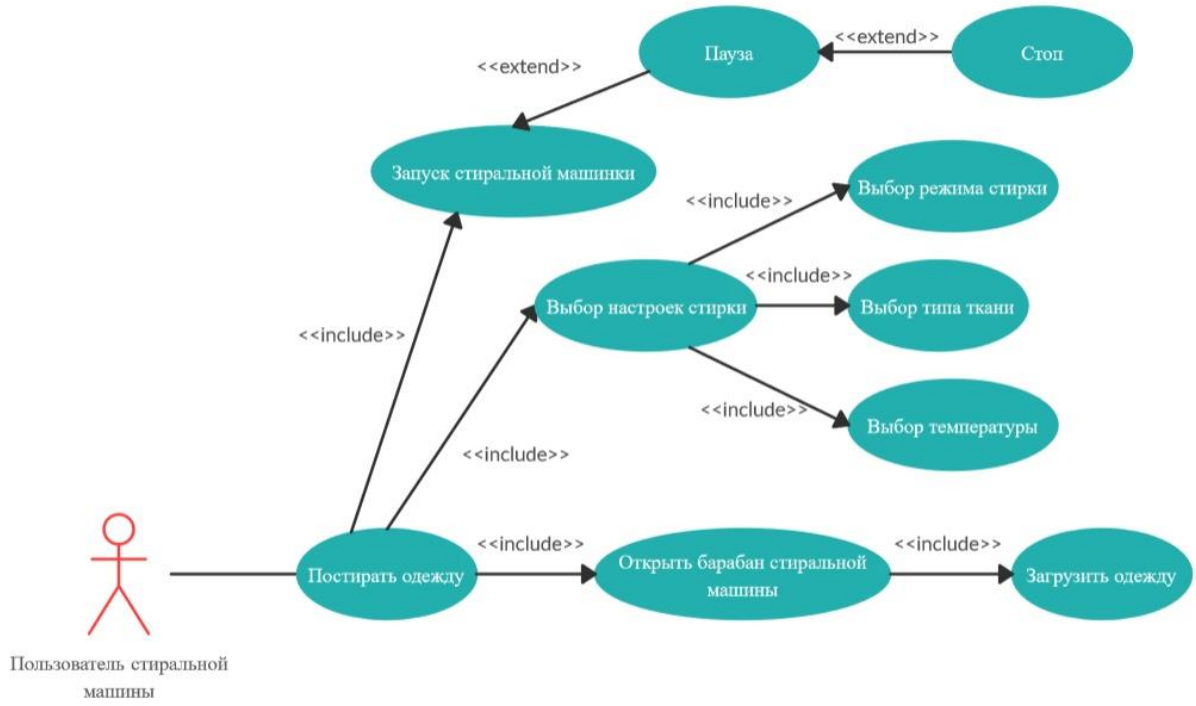


Рисунок 12 – Диаграмма вариантов использования

9 Диаграмма последовательности

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектами для выполнения требуемых функций.

На диаграмме объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия (линия жизни).

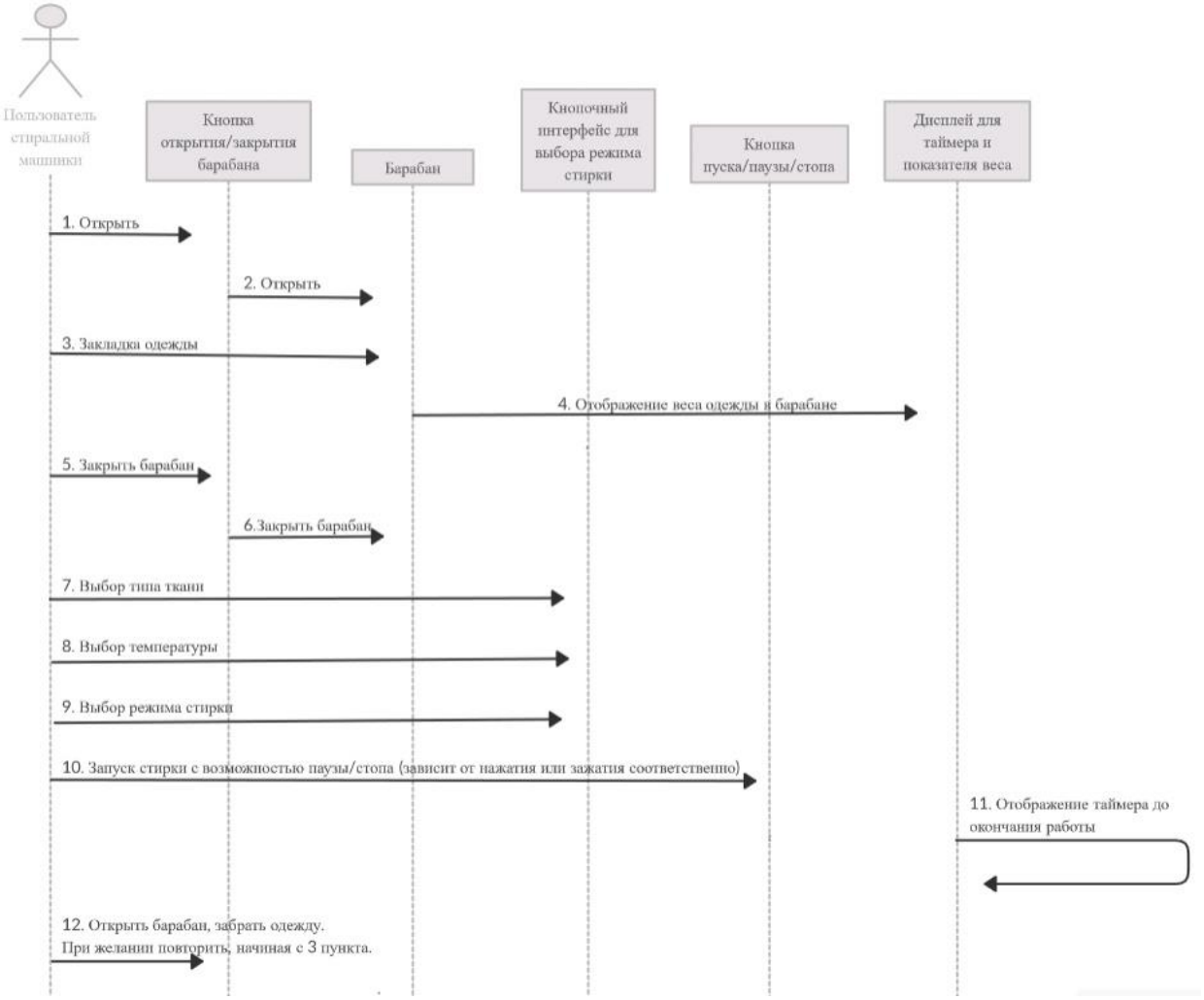


Рисунок 13 – Диаграмма последовательности

10 Диаграмма кооперации

На диаграмме кооперации представлена вся та информация, которая есть и на диаграмме последовательности, но она по-другому описывает поток событий. Из нее легче понять связи между объектами, но труднее понять последовательность событий.

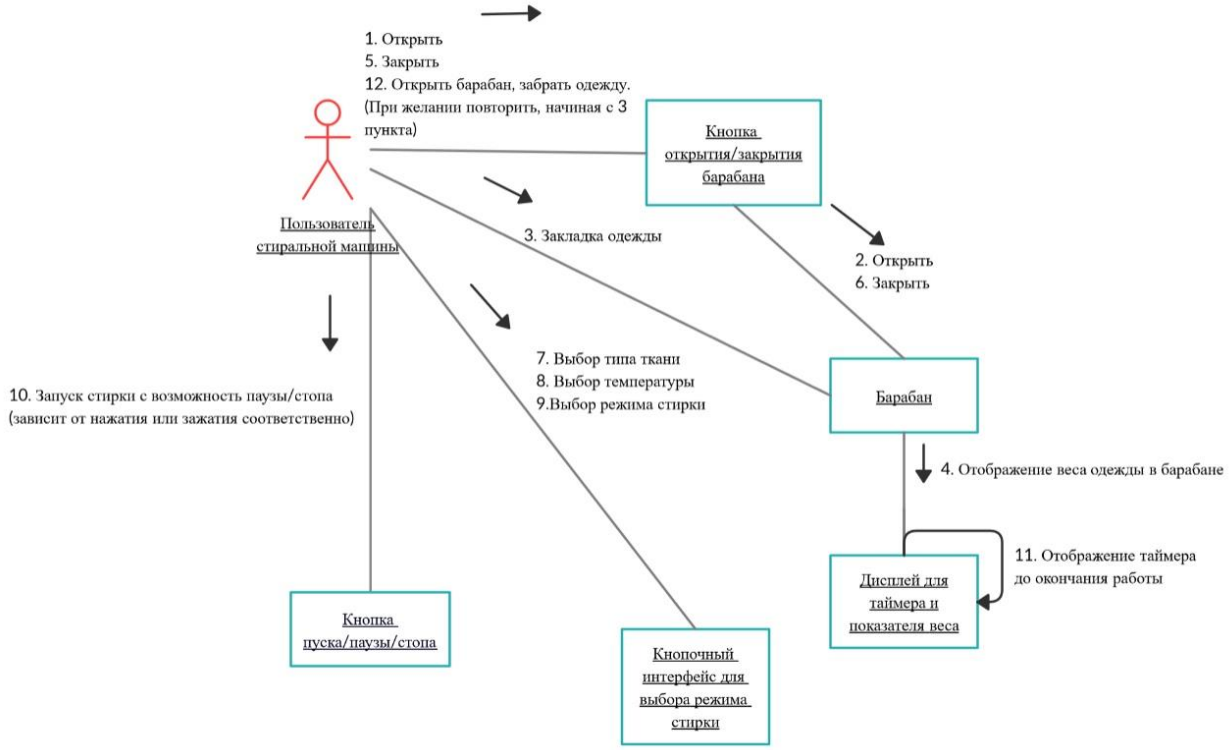


Рисунок 14 – Диаграмма кооперации

11 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграмме классов также изображаются атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов – это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами – отношение между узлами.

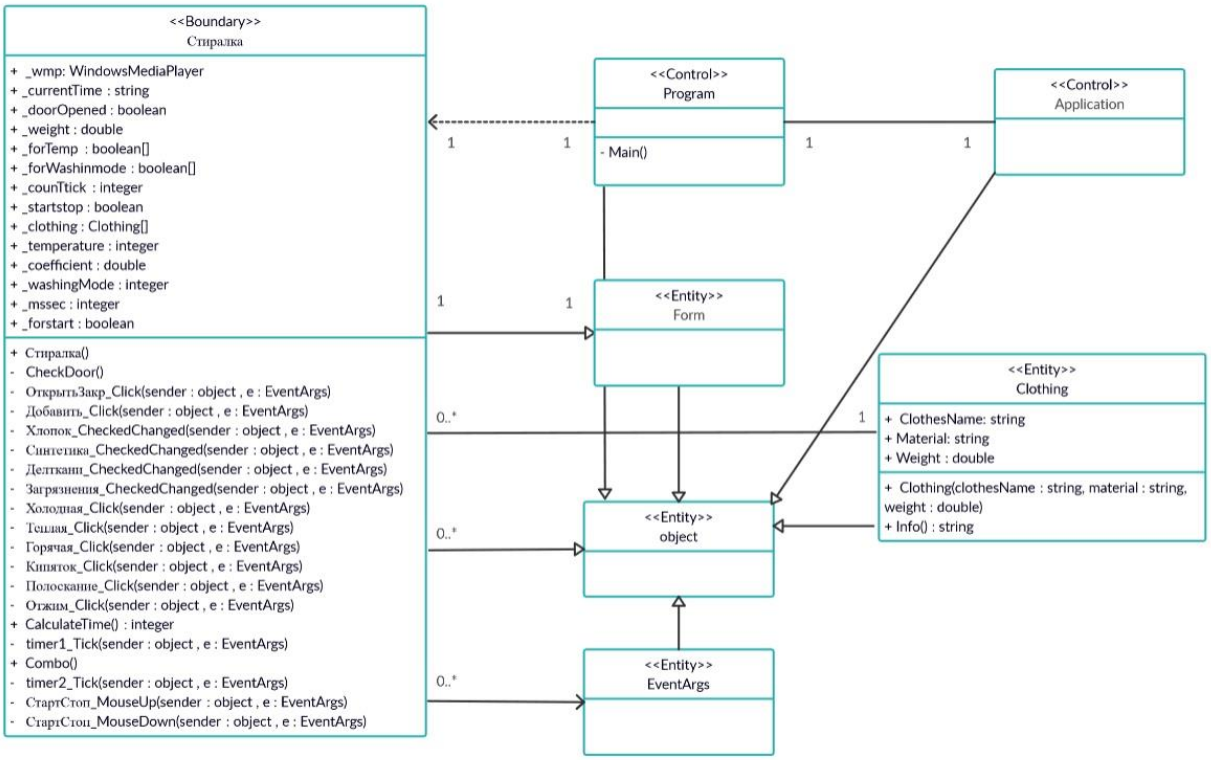


Рисунок 15 – Диаграмма классов

В контексте диаграмм классов, пакет – это вместилище для некоторого набора классов и других пакетов.

Существуют некоторые общепринятые случаи деления классов по пакетам. Например, тесно взаимодействующие классы, или разбиение системы на подсистемы.

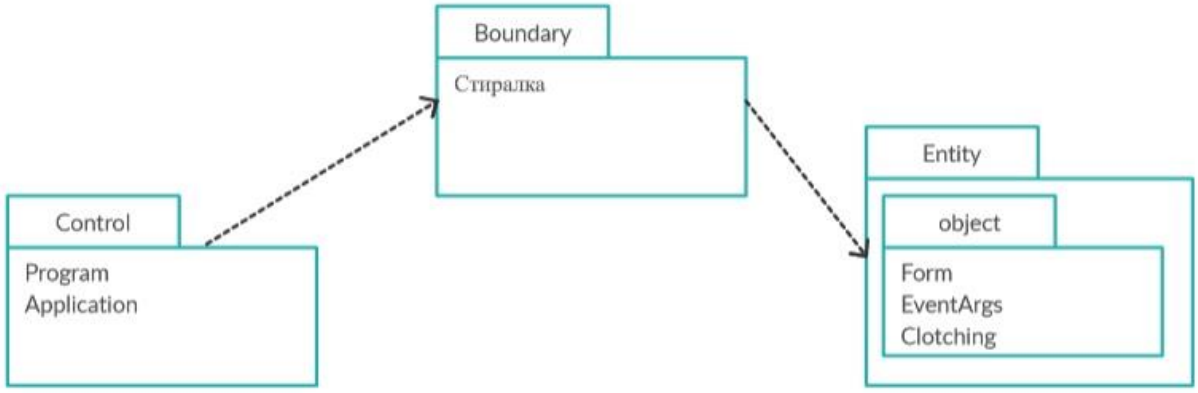


Рисунок 16 – Диаграмма пакетов

12 Диаграмма состояний

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

На диаграмме имеются два специальных состояния – начальное и конечное. Начальное состояние – это выделенная черная точка, конечное состояние – черная точка в белом круге. На диаграмме может отсутствовать конечное состояние.

Диаграмма состояний состоит из:

1. Деятельность – поведение, реализуемое объектом, пока он находится в данном состоянии.
2. Входное действие – поведение, которое выполняет объект при переходе в данное состояние.
3. Выходное действие – поведение, которое выполняет объект при выходе из данного состояния.
4. Событие – то, что вызывает переход из одного состояния в другое.
5. Ограждающие условия определяют, когда переход может осуществиться, а когда нет.
6. Действие – часть перехода. Рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

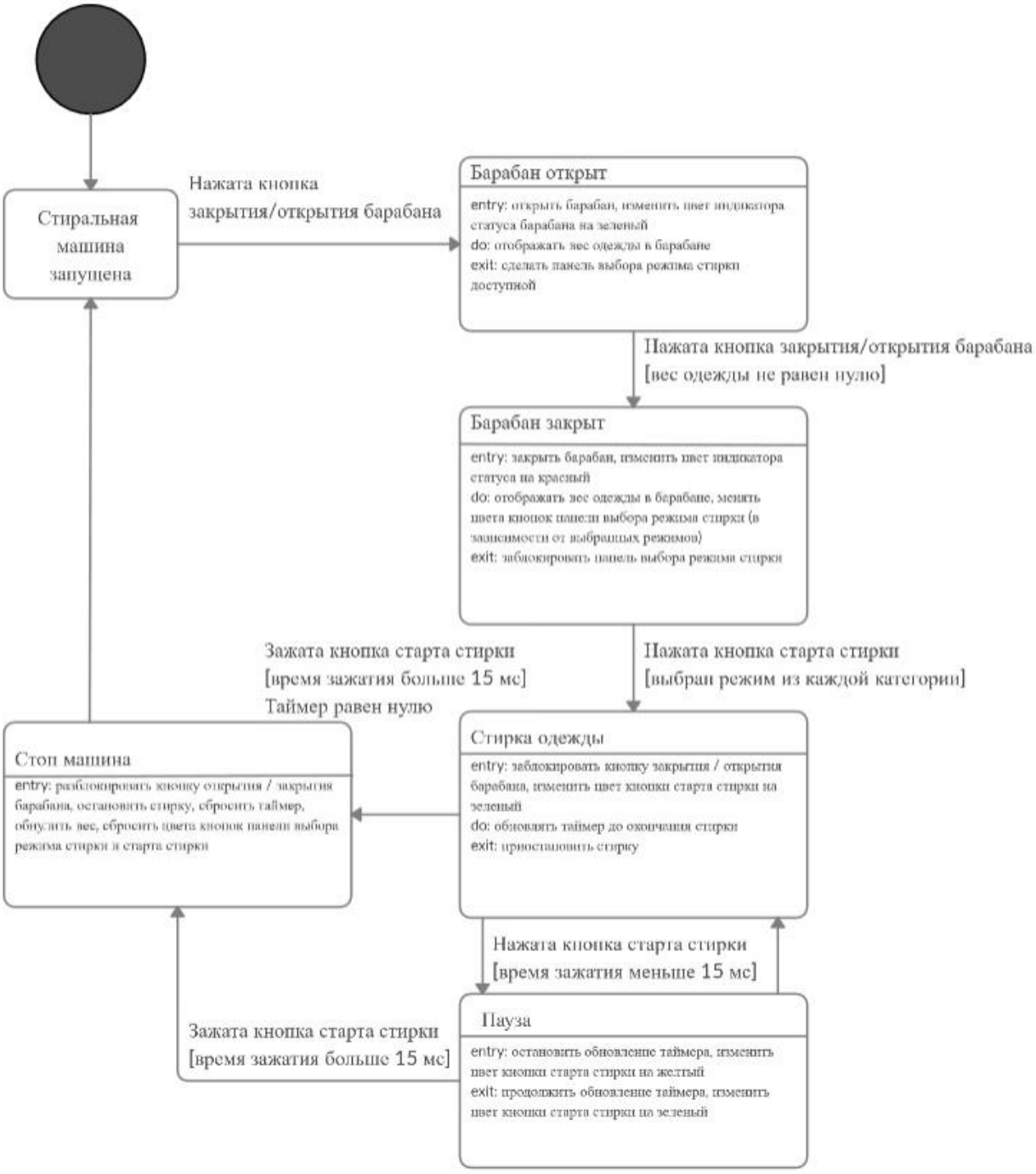


Рисунок 17 – Диаграмма состояний

13 Диаграмма размещения

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы. Диаграмма

размещения показывает физическое расположение сети и местонахождение в ней различных компонентов.

Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства – в большинстве случаев, часть аппаратуры.

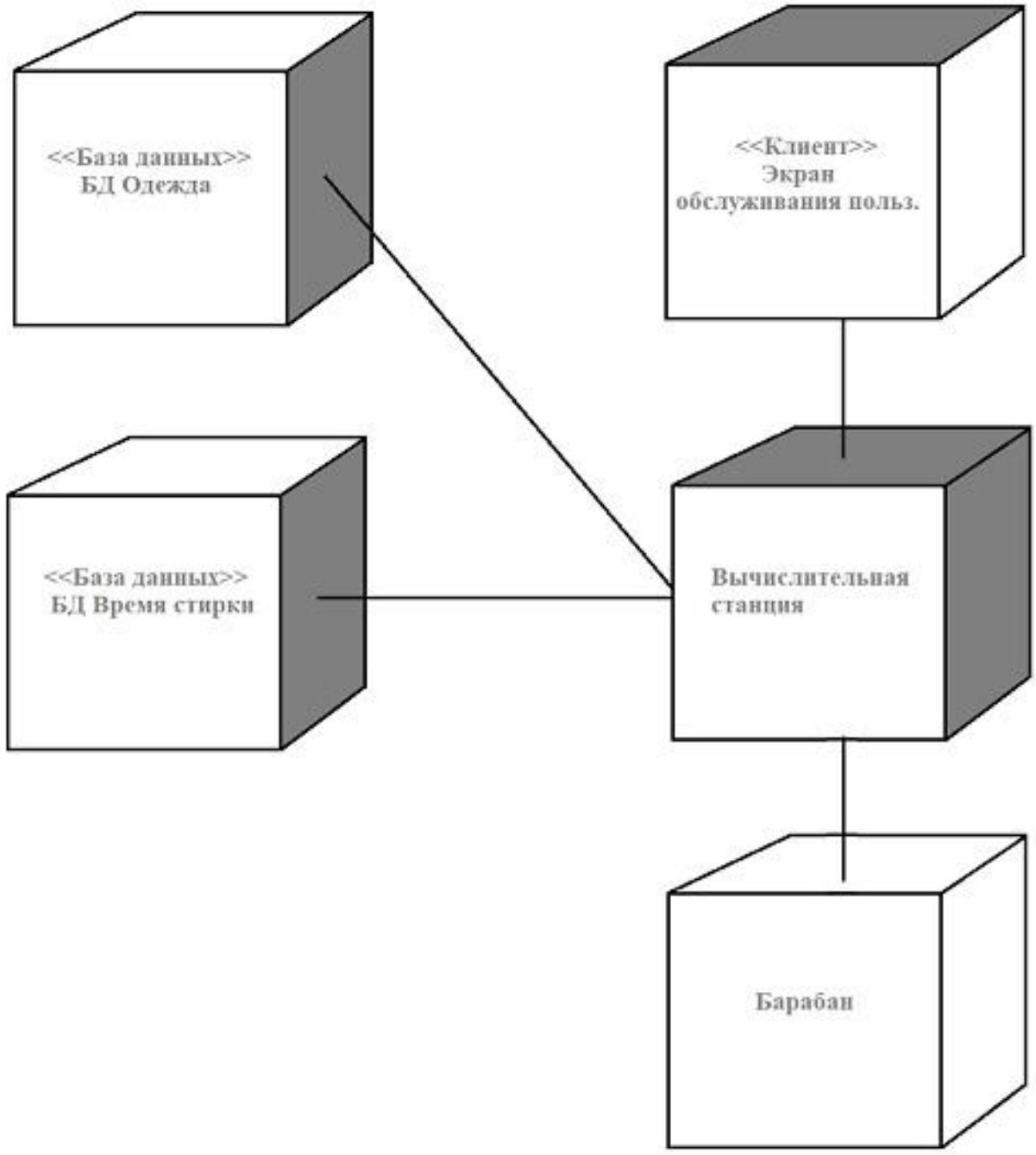


Рисунок 18 – Диаграмма размещения

14 Диаграмма компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними.

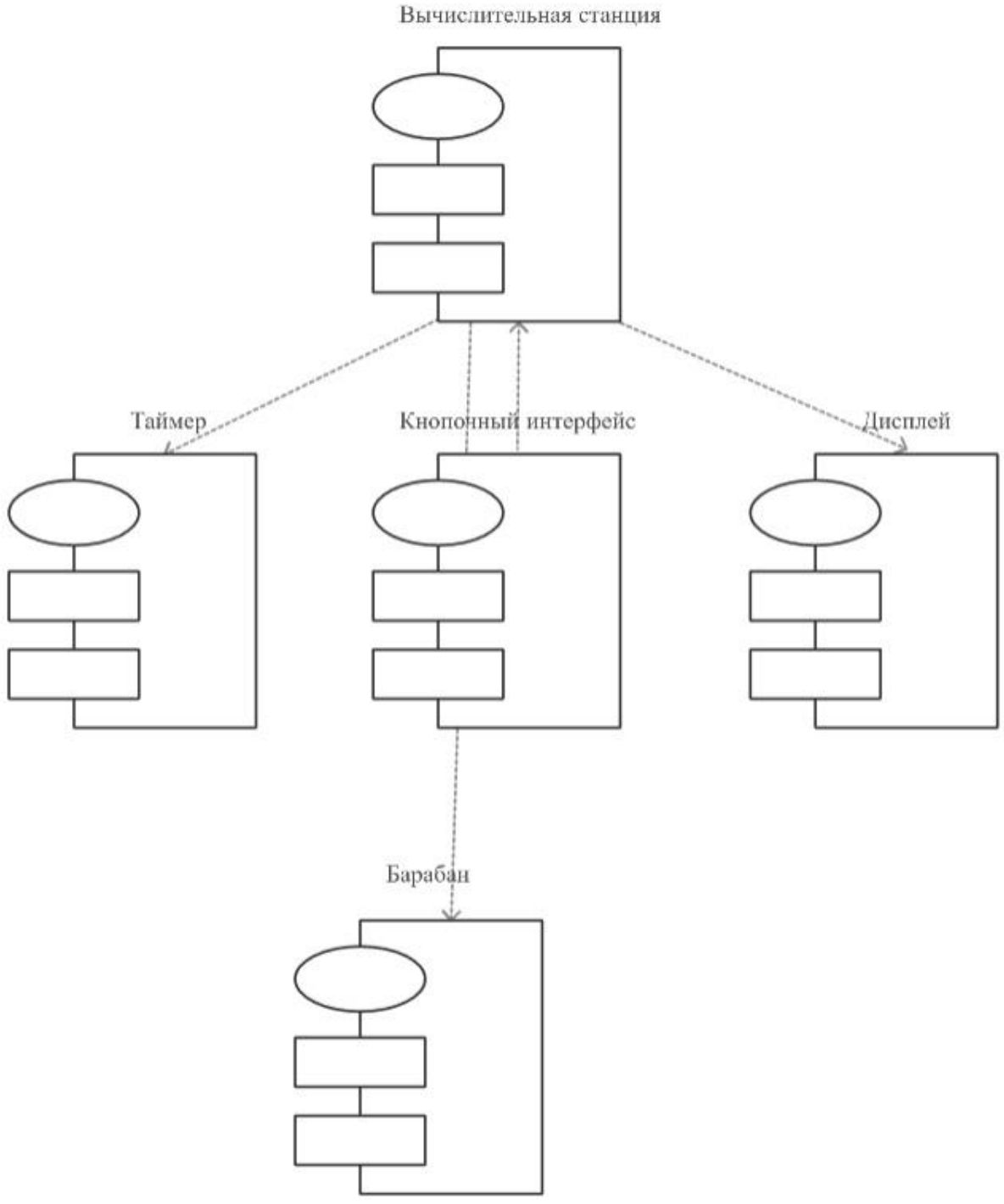


Рисунок 19 – Диаграмма компонентов

15 Объединенная диаграмма компонентов и развертывания

В некоторых случая допускается размещать диаграмму компонентов на диаграмме развертывания. Это позволяет показать, какие компоненты выполняются и на каких узлах.

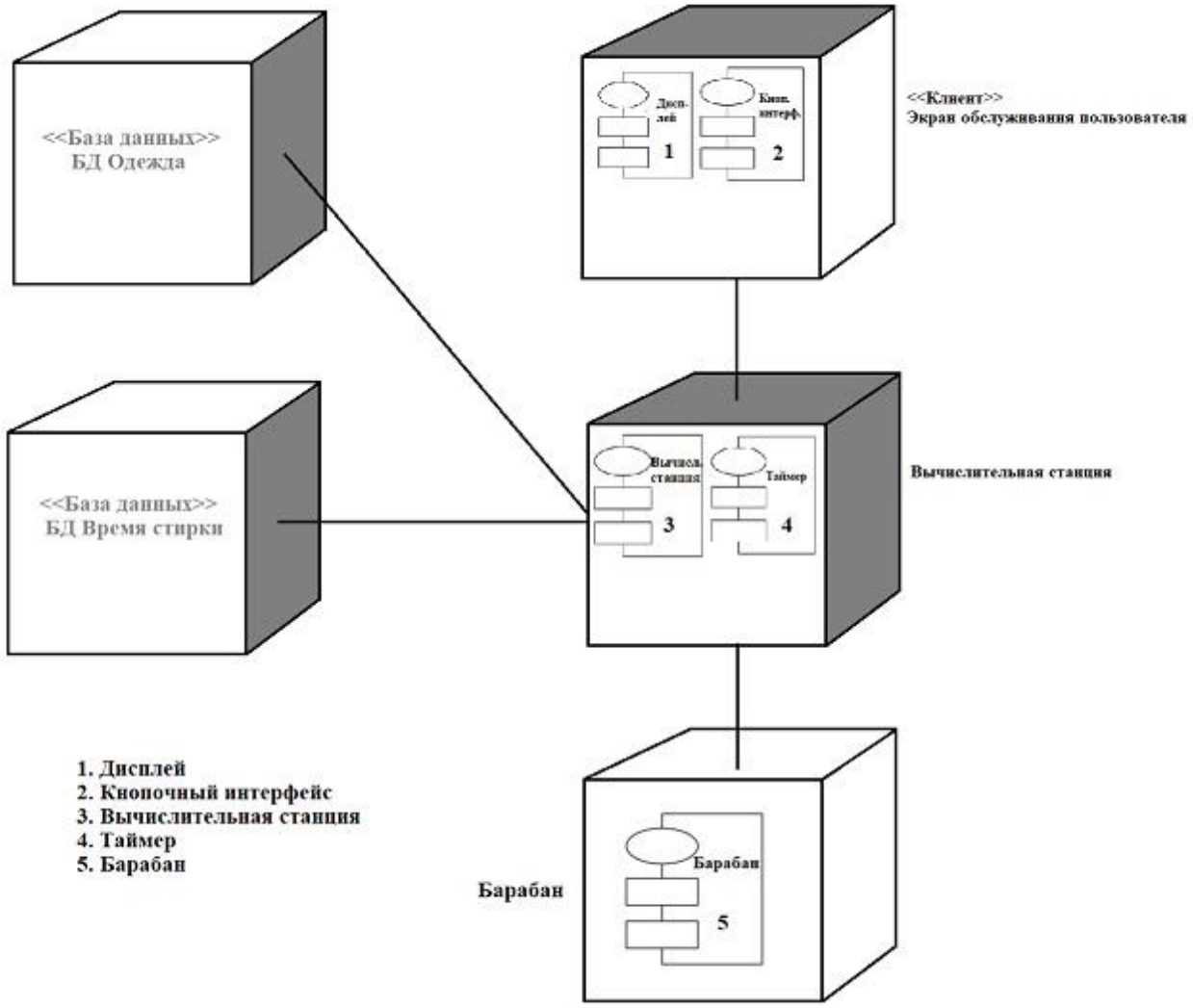


Рисунок 20 – Объединенная диаграмма компонентов и развертывания

16 Тестирование программы

16.1 Общее положение

Тестирование программы происходило по принципу «Черного и «Белого» ящика было проведено модульное тестирование, машинное тестирование и профилирование программы.

16.2 Тестирование по принципу «Черного» и «Белого» ящика

Тестирование по принципу черного ящика определяется как методика тестирования, при которой функциональность тестируемого приложения тестируется без учета внутренней структуры кода, деталей реализации и знания внутренних путей программного обеспечения. Этот тип тестирования полностью основан на требованиях и спецификациях программного обеспечения.

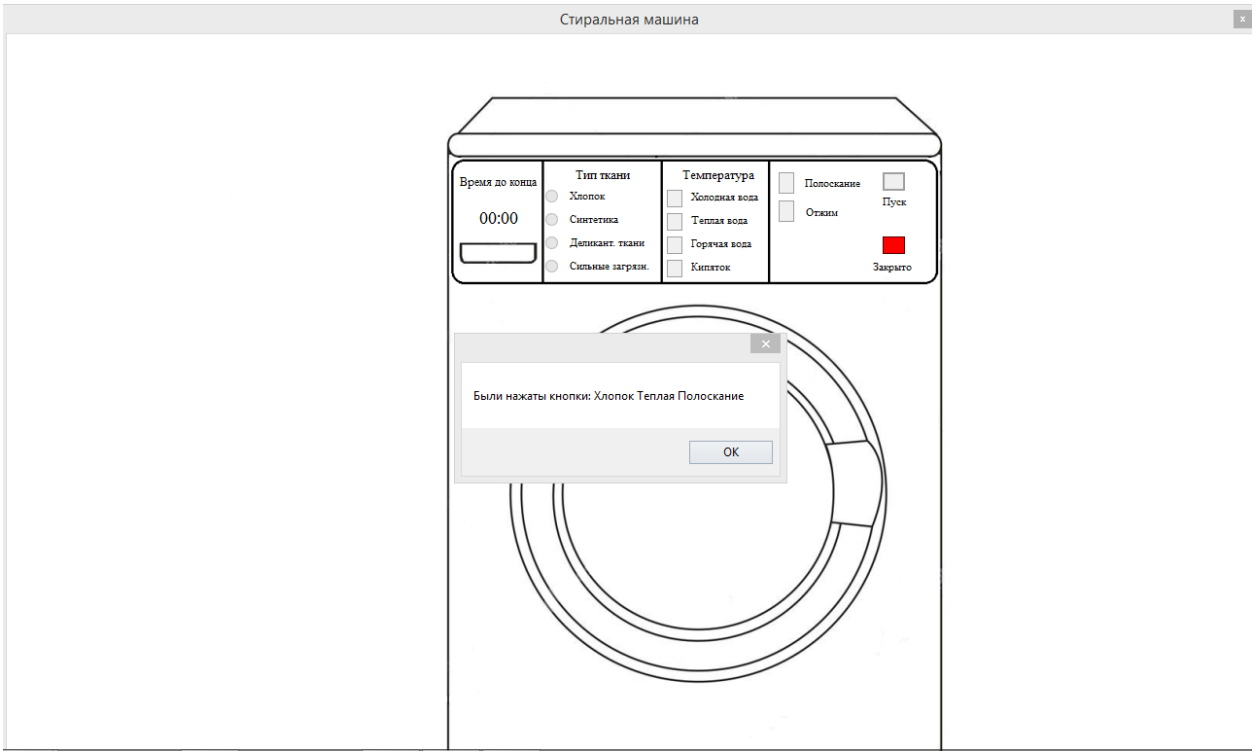


Рисунок 21 – Тестирование по принципу «Черного» ящика

В качестве входных данных в тестировании по принципу «Черного» ящика были кнопки, которые записывали в отдельную переменную информацию о том, какая кнопка была нажата и в качестве выходных данных была информация о том, какие кнопки были нажаты (когда уже заканчивался процесс стирки), которая выводилась на экран.

Тестирование по принципу белого ящика опирается на то, что тестировщик разрабатывает тесты, основываясь на знании исходного кода, к которому он имеет полный доступ.

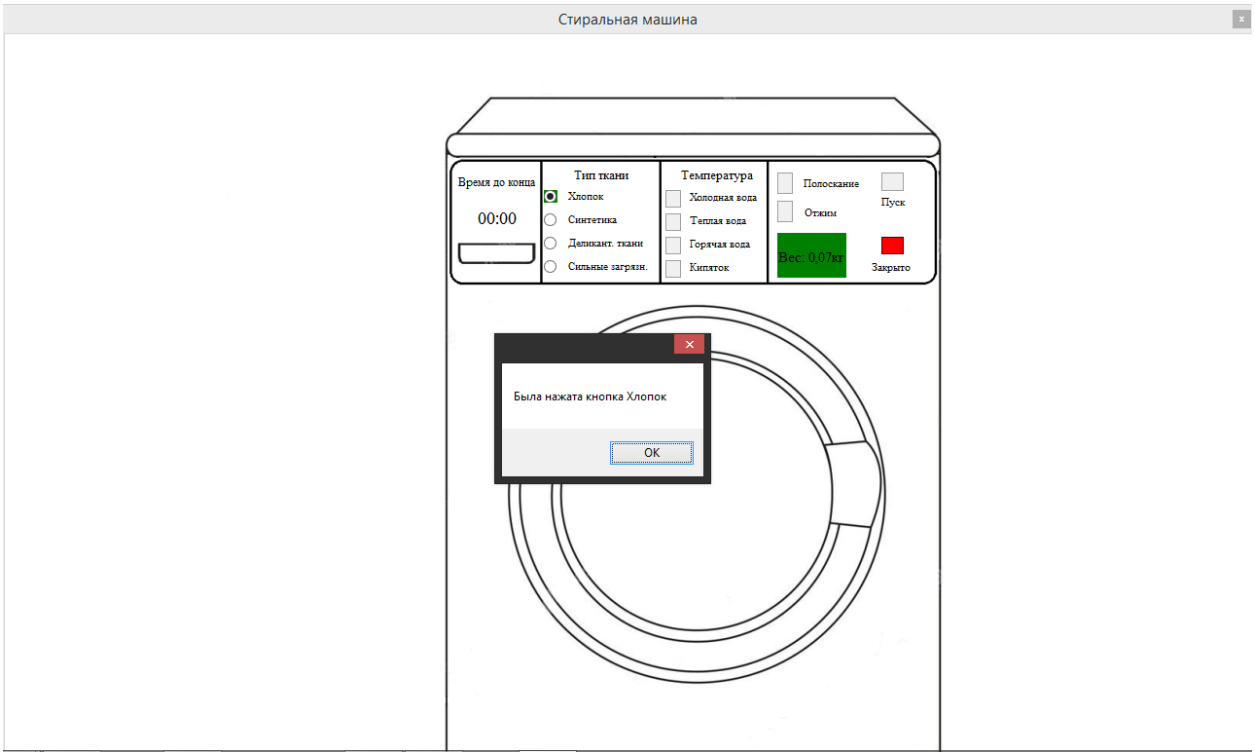


Рисунок 22 – Тестирование по принципу «Белого» ящика

Тестирование по принципу «Белого» ящика заключалось в том, что при нажатии на любую кнопку программы сразу выводилась на экран информация о том, какая кнопка была нажата.

16.3 Модульное тестирование программы

Модульное тестирование или юнит-тестирование – процесс в программировании, который позволяет проверить на корректность работы отдельные модули исходного кода программы

Тестирование отдельных блоков программы происходило благодаря встроенному тестировщику в VS Unit Test. Так, в качестве основного блока для тестирования был метод подсчета времени стирки.

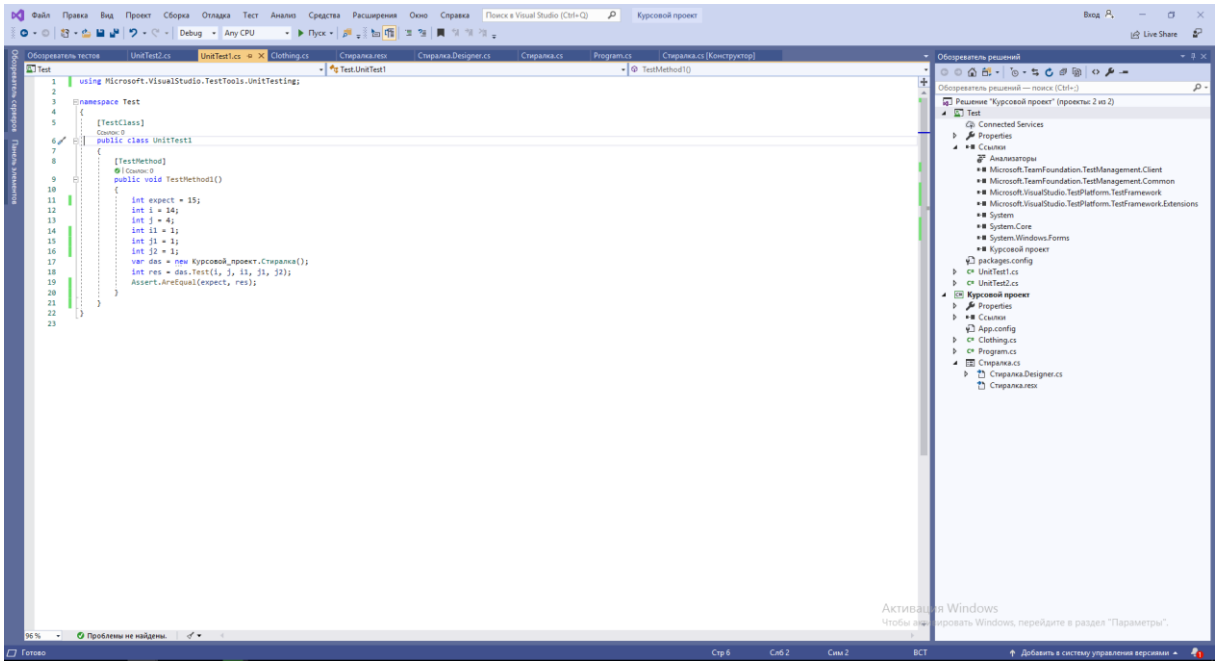


Рисунок 23 – Основной блок тестирования программы

Модульное тестирование программы

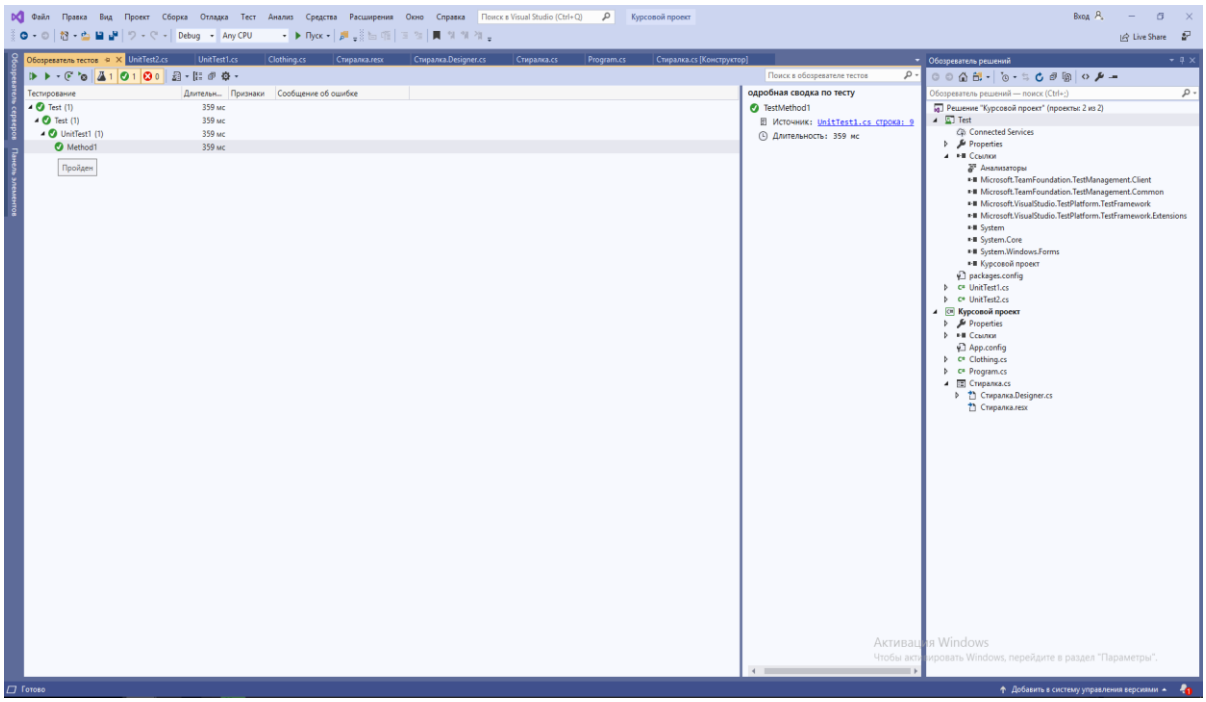


Рисунок 24 – Исходные данные для первого теста

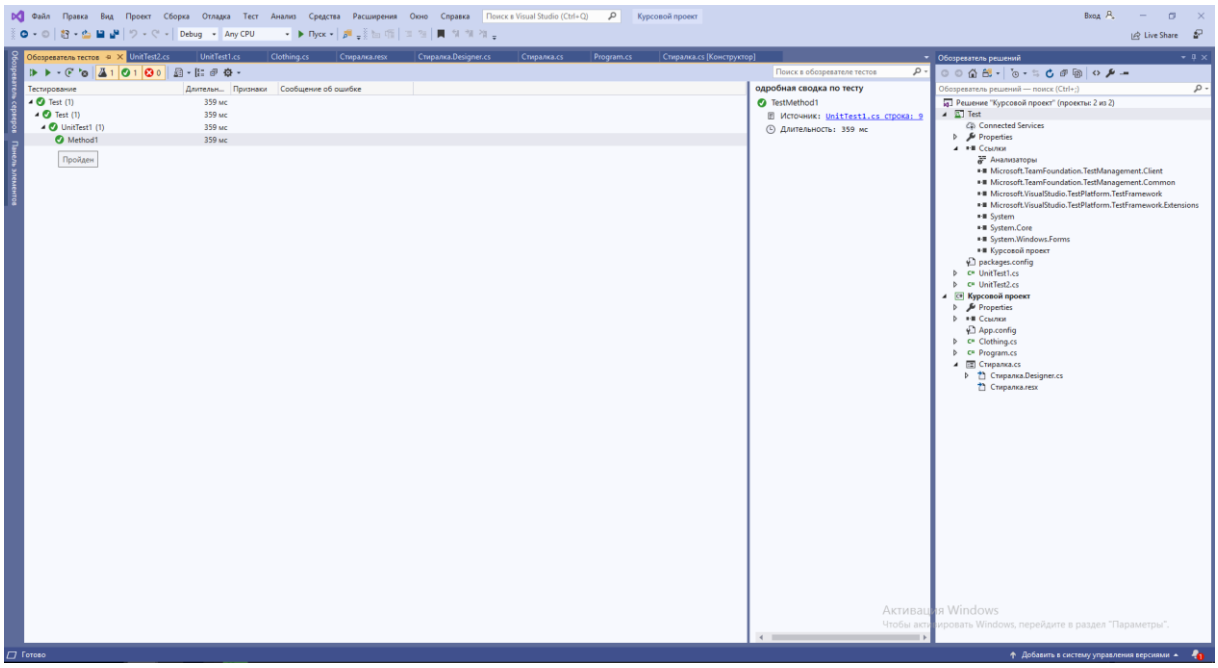


Рисунок 25 – Результаты первого модульного тестирования

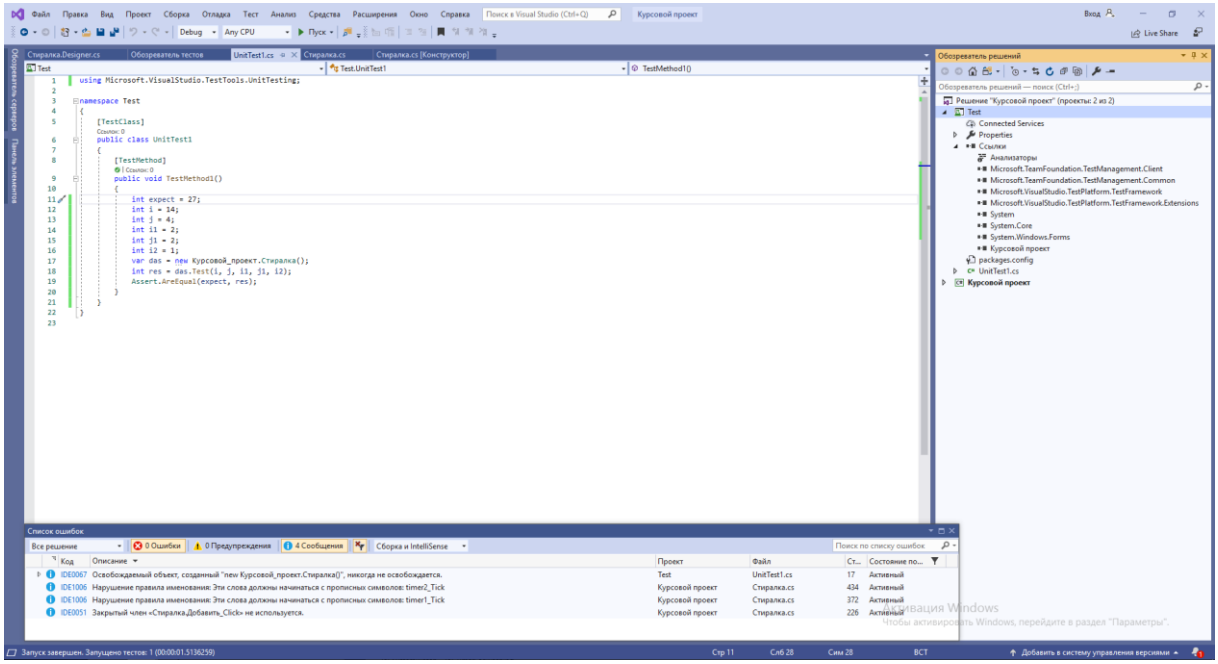


Рисунок 26 – Исходные данные для второго теста

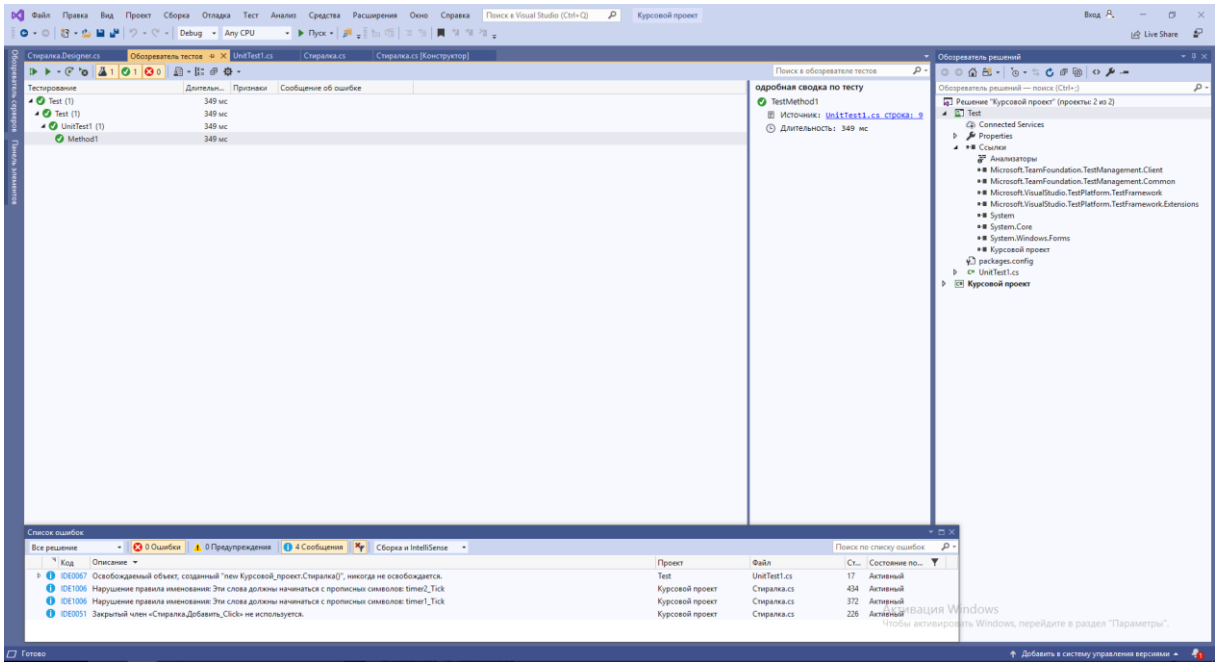


Рисунок 27 – Результаты второго модульного тестирования

Во время тестирования не было выявлено ошибок, т.к были внесены правильные данные. Что же случится, если внести искусственно ошибку?

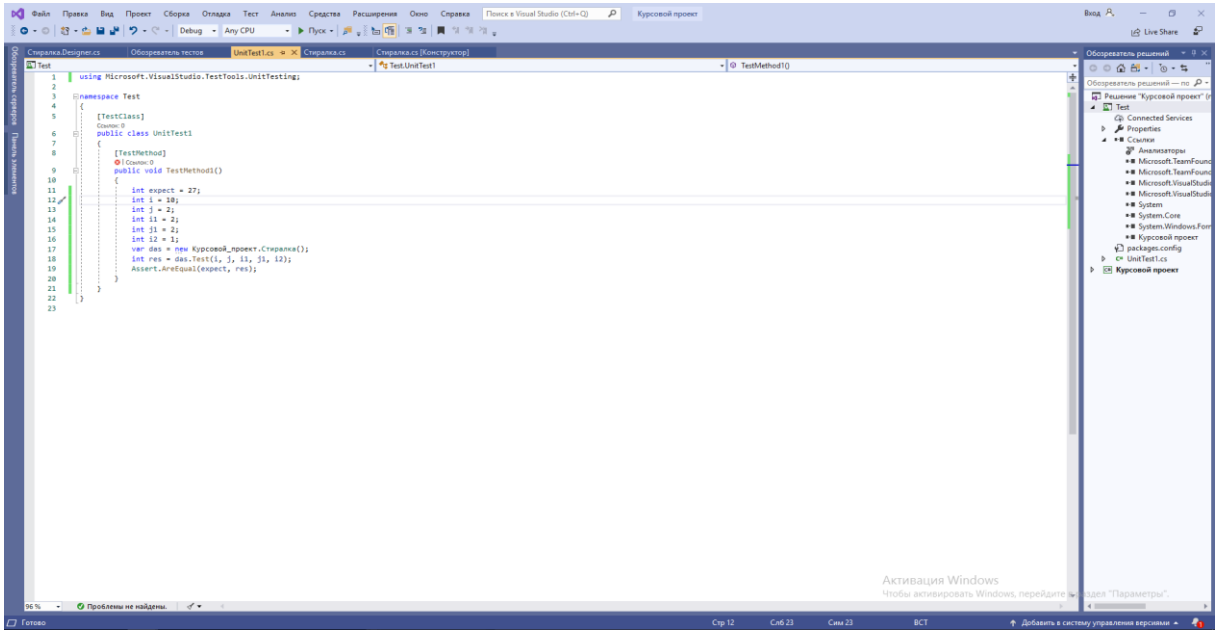


Рисунок 28 – Искусственное внедрение ошибки

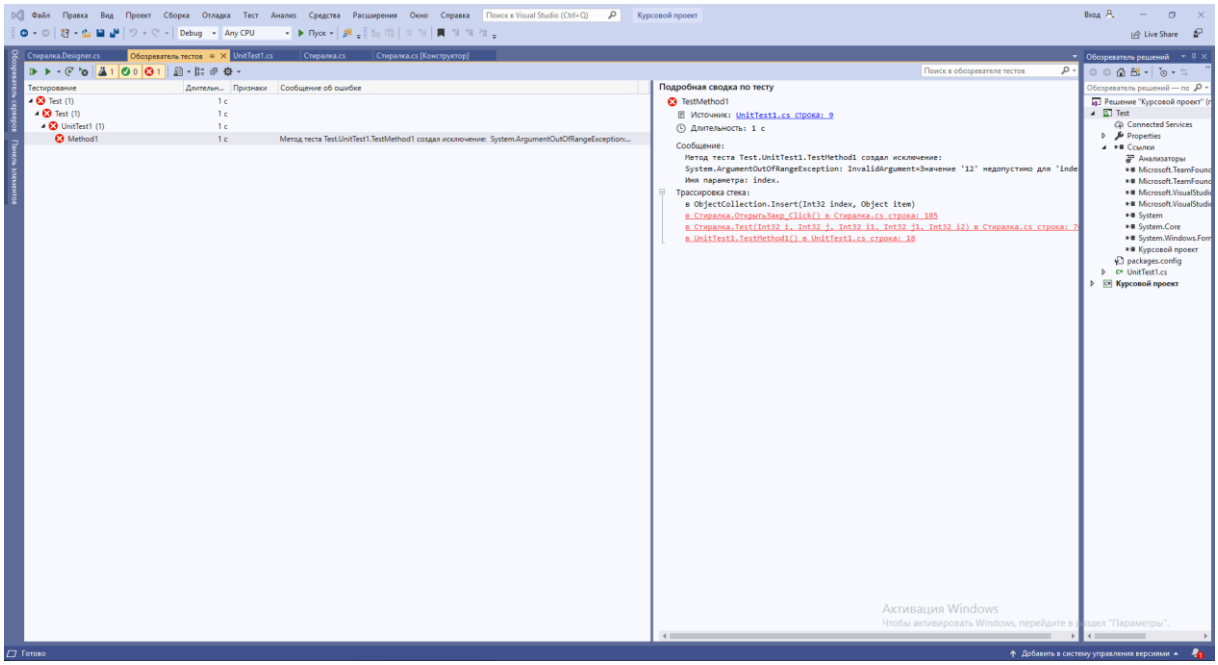


Рисунок 29 – Результат тестирования при внедрении ошибки

Ошибка заключалась в том, что был удален элемент в листе, который раньше был в индексе i.

Во время тестирования выявлено несколько ошибок, которое было связано с подсчетом времени программы.

16.4 Машинное тестирование

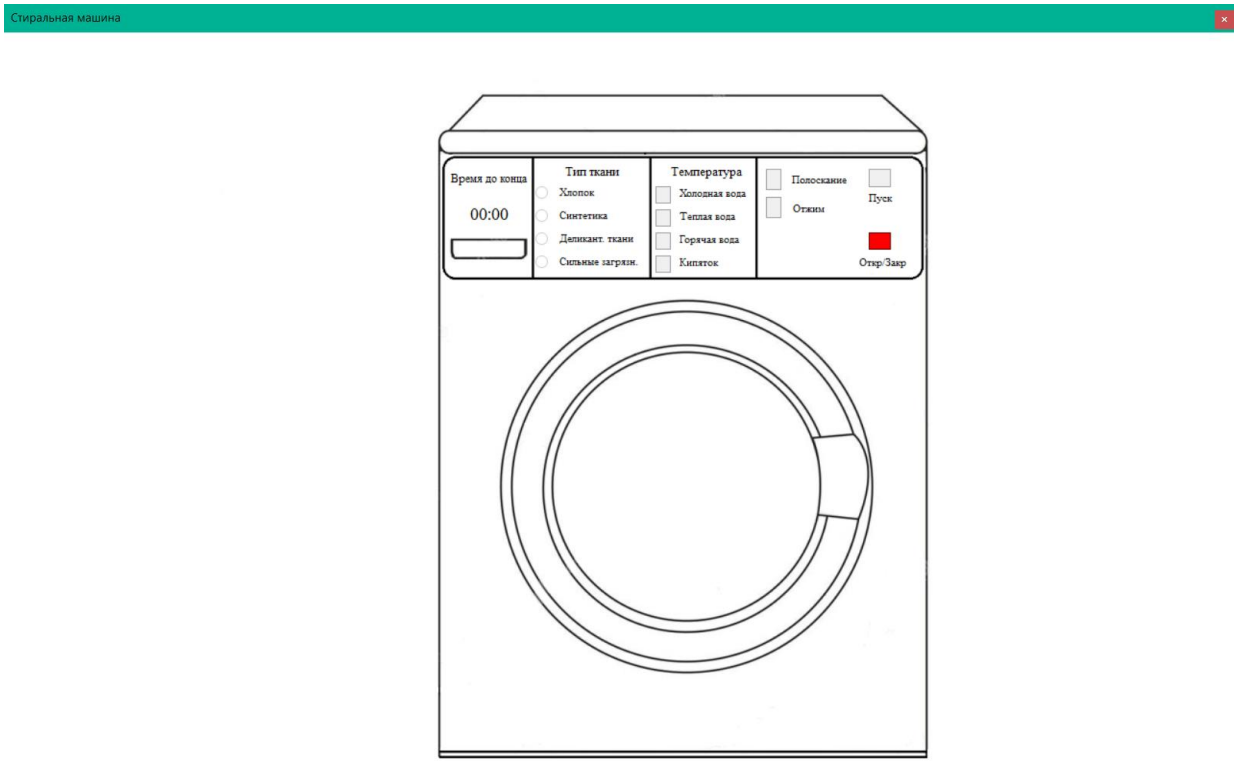


Рисунок 30 – Готовая к работе стиральная машина

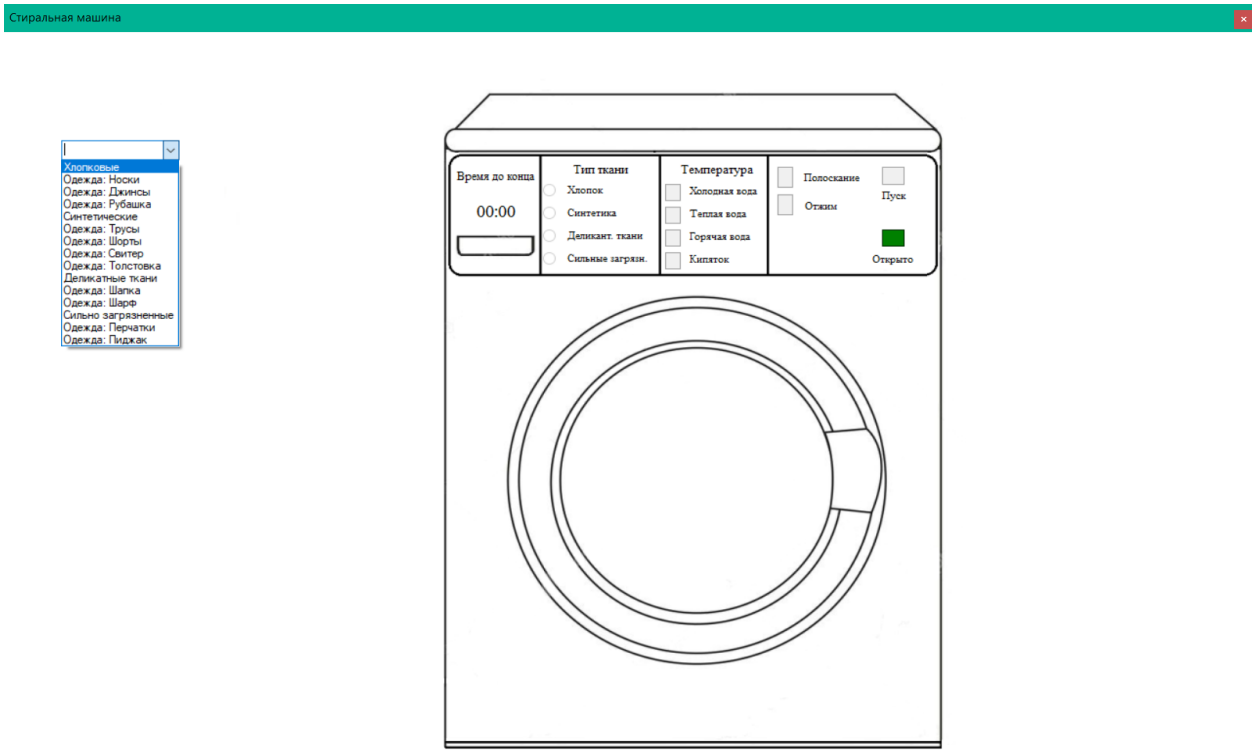


Рисунок 31 – Открыт барабан

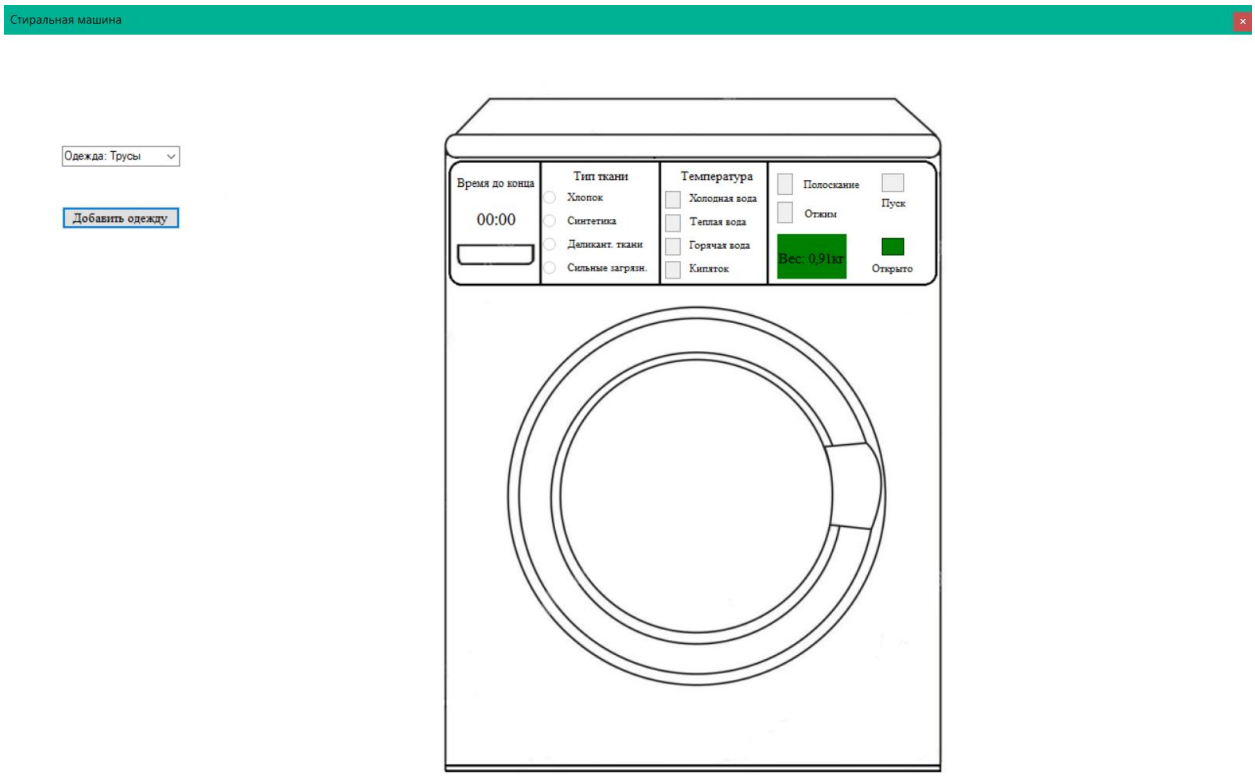


Рисунок 32 – Закладка одежды

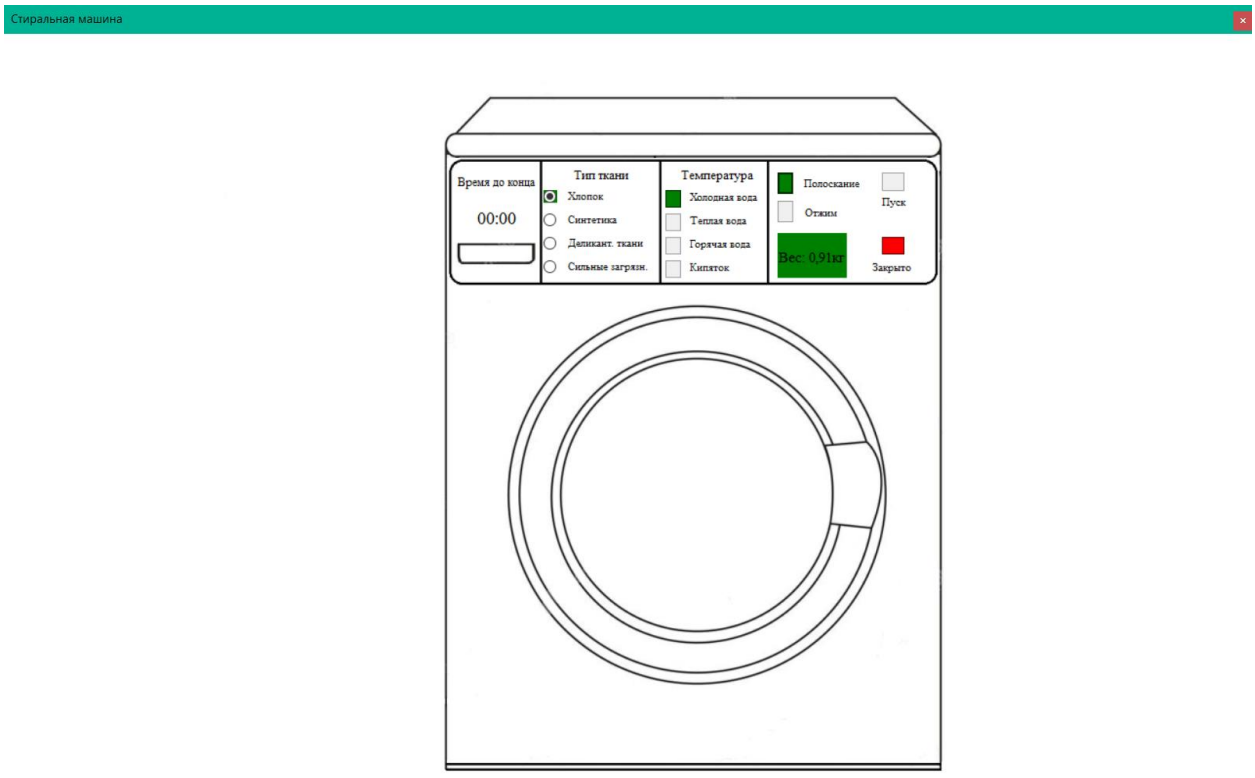


Рисунок 33 – Выбор типа ткани, температуры, режима стирки

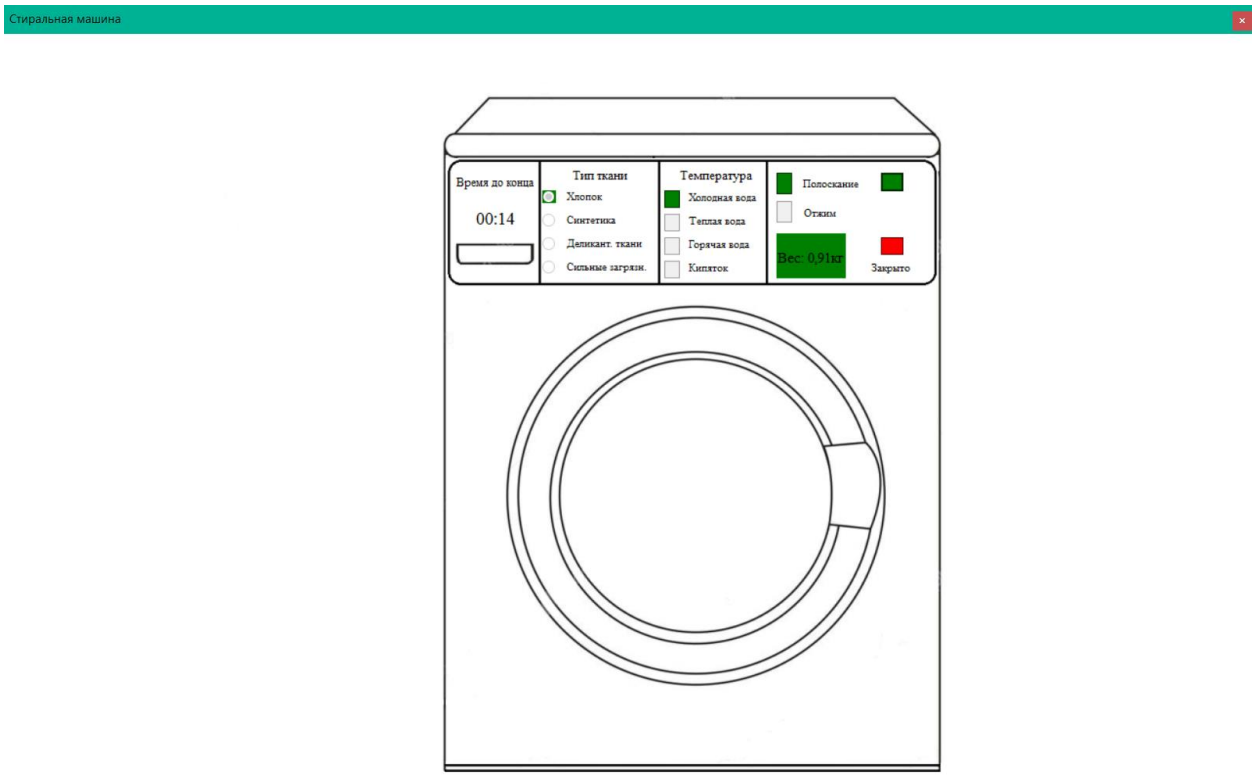


Рисунок 34 – Запуск стирки

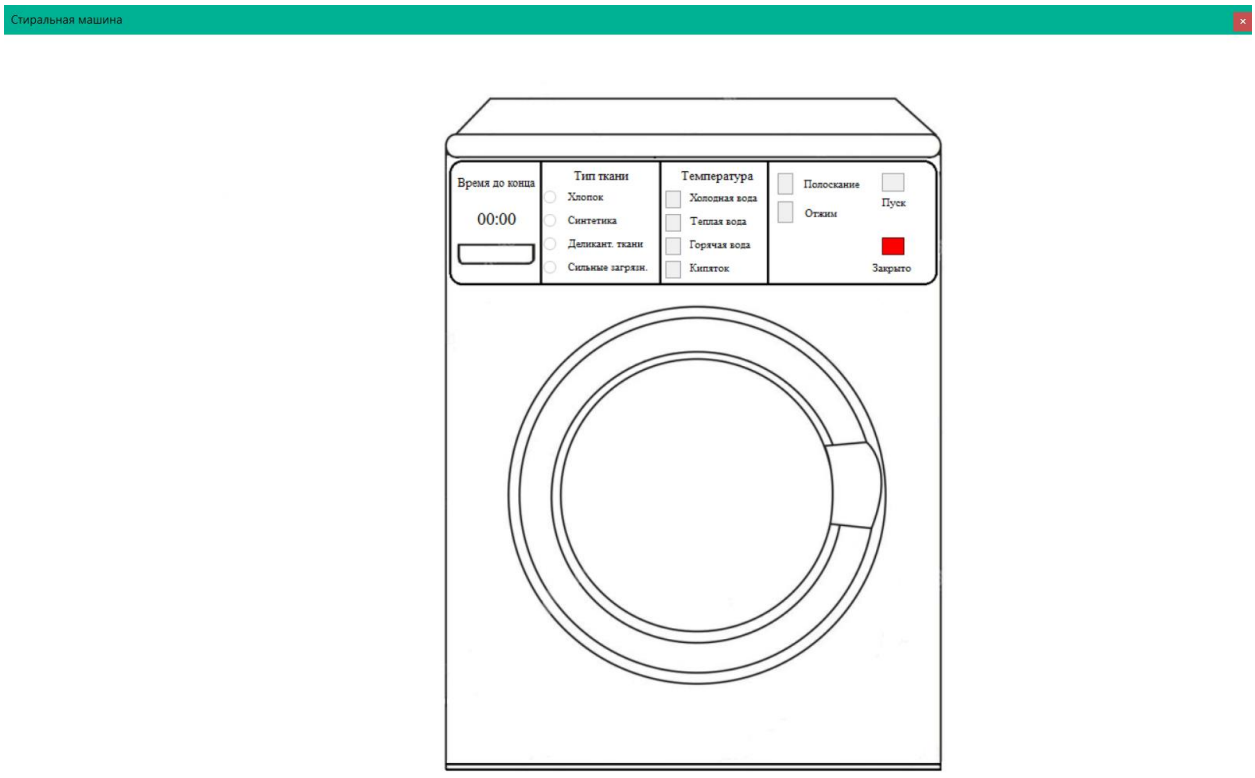


Рисунок 35 – Возращение к начальному состоянию после окончания таймера

16.5 Профилирование программы

Профилирование нужно для того, что отслеживать время выполнения отдельных фрагментов программы, загрузку процессора и оперативной памяти, число верно предсказанных условных переходов, число кэш-промахов и т. д. Профилирование программы происходило благодаря встроенному профилировщику в Virtual Studio.

Отслеживание загрузки CPU во время профилирования программы. (длительность сеанса диагностики: 51с.)

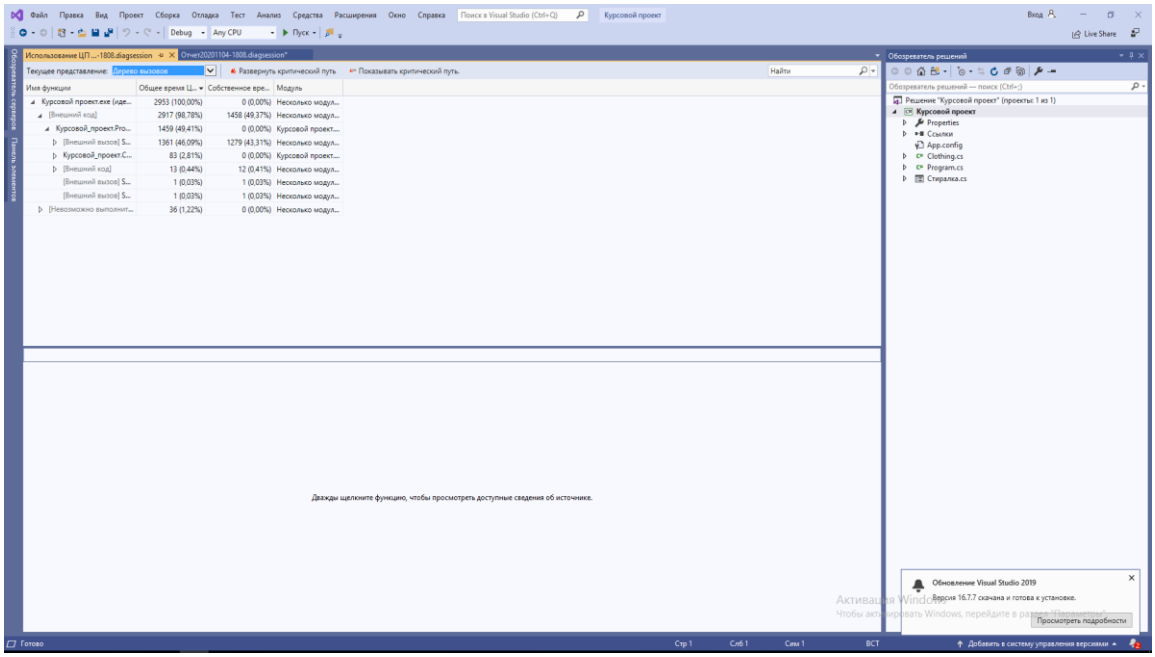


Рис 36 – Профилирование программы. Загрузка CPU

Отслеживание загрузки оперативной памяти во время профилирования программы

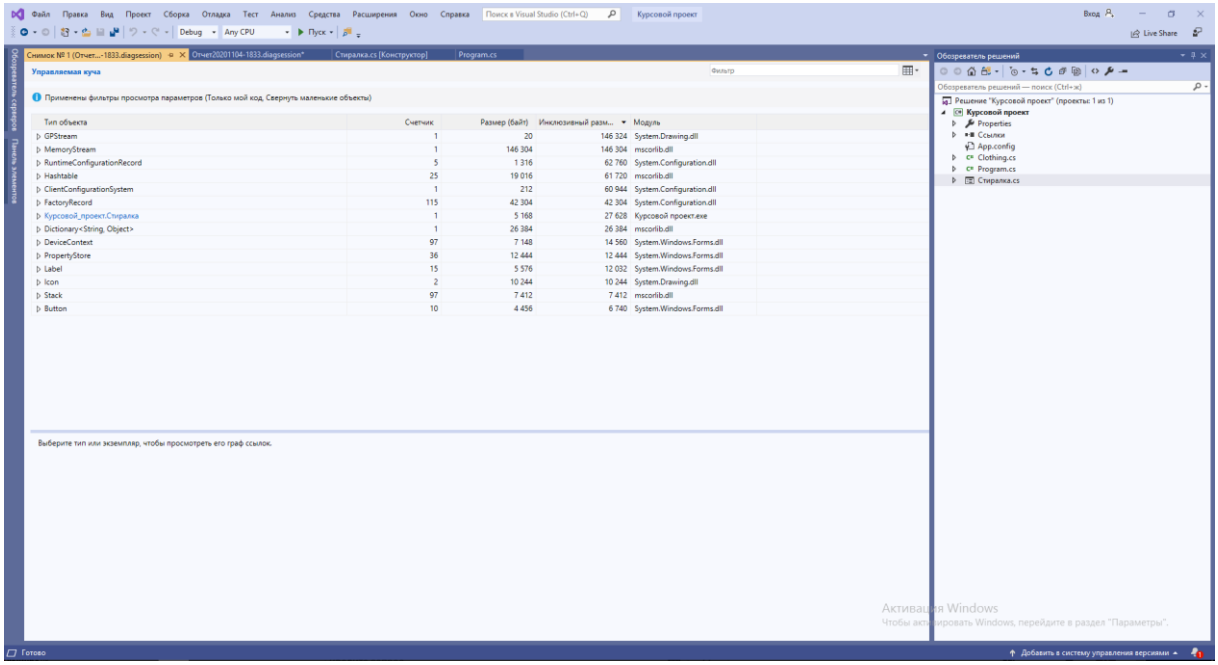


Рис 37- Профилирование программы. Расходование оперативной памяти во время простоя программы

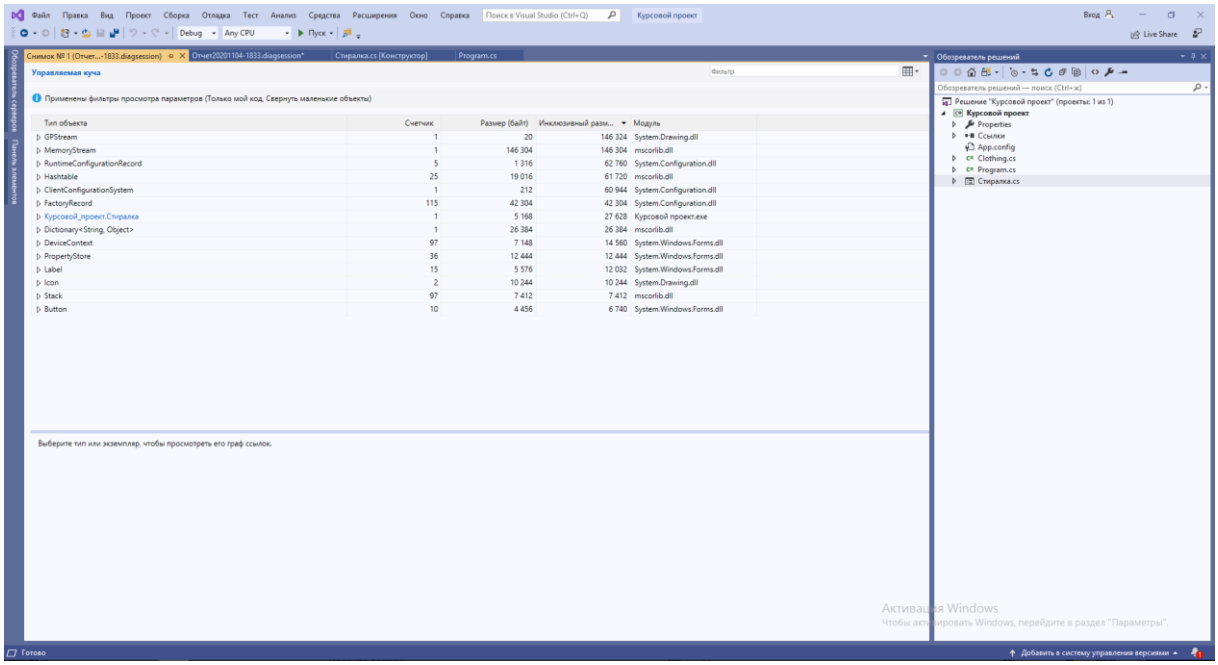


Рис 38 – Профилирование программы. Расходование оперативной памяти во время нажатия на кнопки стиральной машины

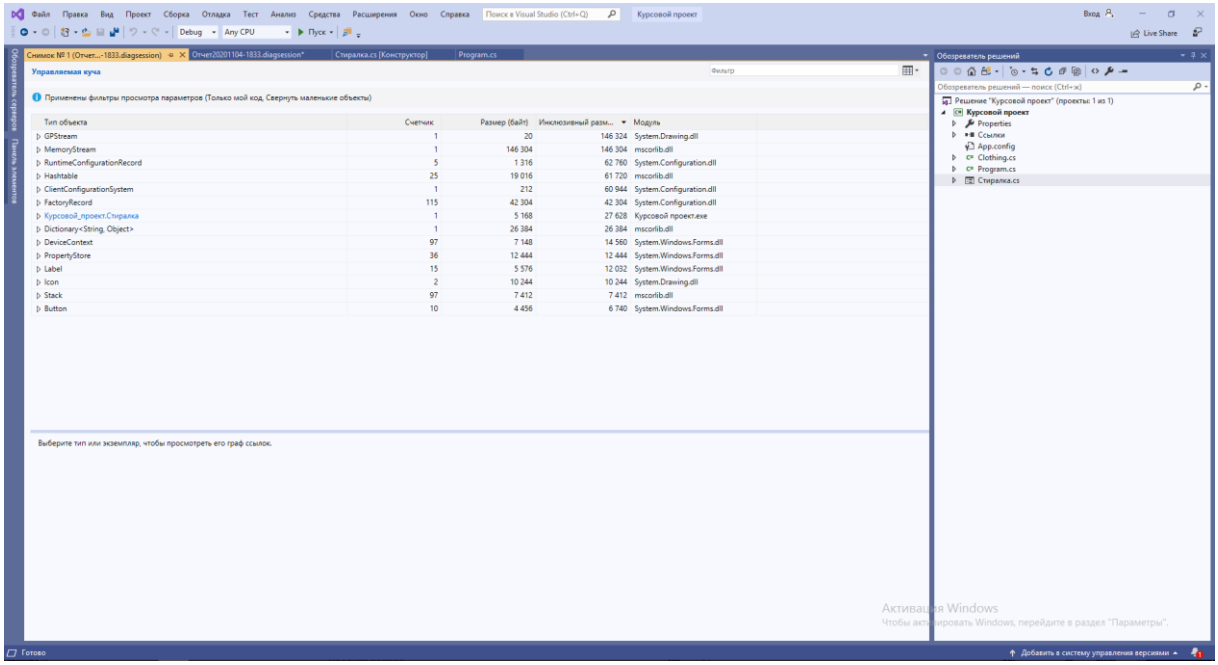


Рис 39 – Профилирование программы. Расходование оперативной памяти во время стирки

Профилирование программы помогло отследить загрузку CPU и расходование оперативной памяти на разных этапах выполнения программы.

17 Системные требования

Процессор	2.5 ГГц
Оперативная память	150 Мб
Монитор	1920 x 1080
Свободное место на носителе	5 Мб
Устройства взаимодействия с пользователем	Клавиатура и мышь
Программное обеспечение	Visual Studio 2019 года последней версии

Таблица 1 – Системные требования программы

18 Руководство пользователя

Запуск программы осуществляется при помощи открытия исполняемого файла:

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

Теперь перед нами интерфейс программы: стиральная машина с дисплеем, таймером и кнопочным интерфейсом. Начальный интерфейс программы изображен на рис. 21.

Для начала пользователь должен открыть барабан и загрузить одежду. После того как будет загружена первая часть одежды появится дисплей, отображающий вес одежды внутри. Цвет фона дисплей отображает степень нагрузки барабана. При красном цвете вы не сможете больше добавлять одежду.

С помощью кнопочного интерфейса выберите тип ткани, температуру, режим стирки. Система не позволит нажать на две кнопки из каждого раздела. Если необходимо изменить что-либо из вышенаписанного, то просто нажмите на нужный вам вариант. Предыдущая кнопка будет отжата, а нужная вам будет выбрана.

Стиральная машина перестает работать и возвращается к начальному состоянию только после того как окончится таймер стирки.

Заключение

В результате выполнения данного курсового проекта была спроектирована система «Стиральная машина» на языке высокого уровня C#, позволяющая наглядно продемонстрировать работу всех её компонентов. Полученные диаграммы обладают простой и понятной, даже для человека незнакомого с данной областью, структурой, описывающей каждый аспект системы с различных сторон. Также было проведено детальное тестирование программы, начиная с тестирования по «Черному» и «Белому» ящику заканчивая профилированием программы в области загрузки оперативной памяти и процессора.

При построении диаграмм использовались 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.

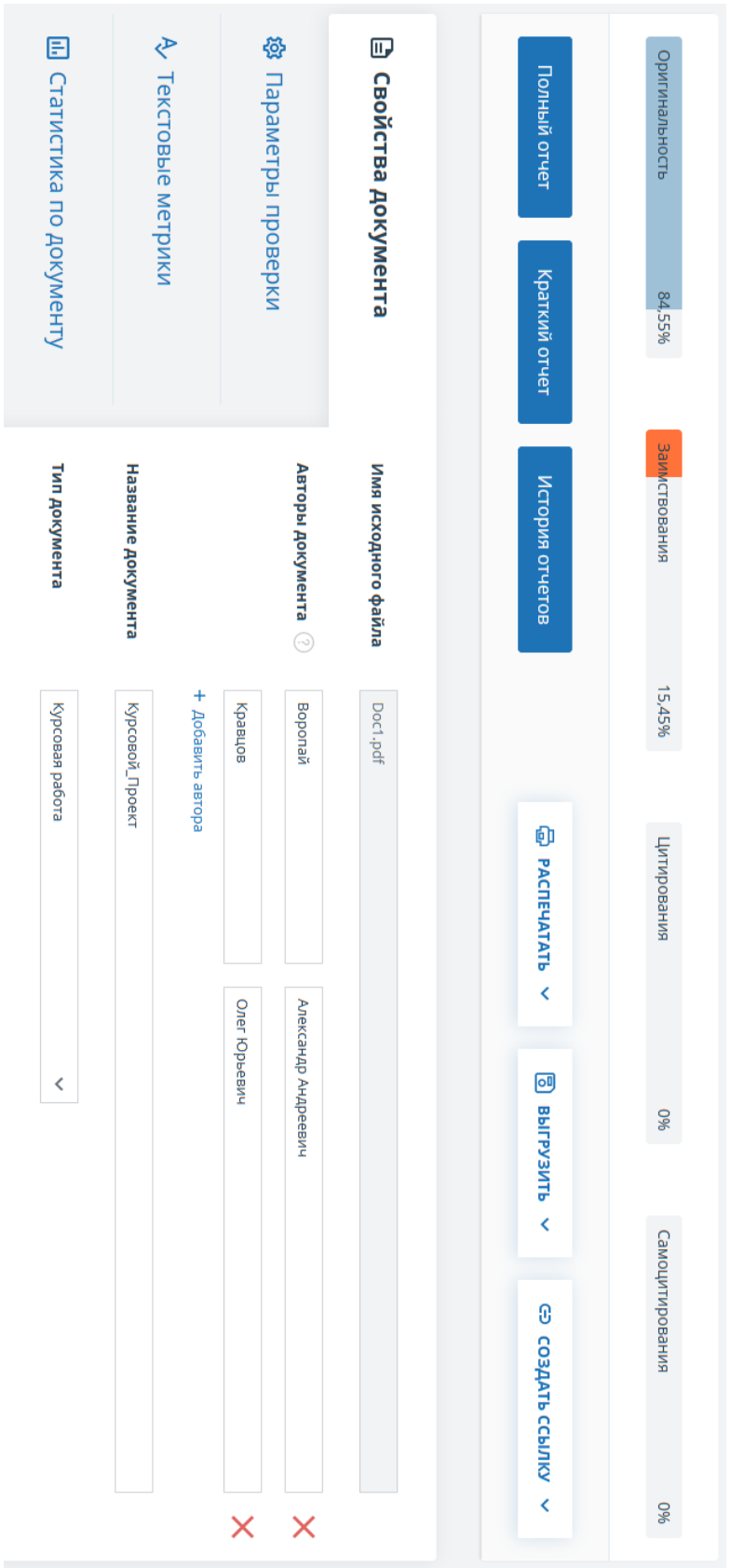
Благодаря детальному разбору проекта при помощи диаграмм проектирования, полученных в процессе разработки, можно с уверенностью сказать, что полученная система «Стиральная машина» полностью соответствует современным стандартам и способна выполнять задуманные для неё задачи.

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. О.Б.Попова. Технологии разработки программного обеспечения. Методические указания по выполнению лабораторных работ для студентов всех форм обучения направления подготовки 09.03.04 Программная инженерия. Краснодар, 2019 – 69 с.
2. Н. И. Шапченко. Оценка трудоемкости разработки программного продукта. Методические указания по выполнению лабораторных работ. Ульяновск, УлГТУ, 2015 – 40 с.
3. Основы методологии IDEF1X [Электронный ресурс] : - Режим доступа: https://www.cfin.ru/vernikov/idef/idef1x.shtml (Дата обращения 18.11.2020)
4. GitHub – AlexanderWoropai/LT [Электронный ресурс] : - Режим доступа: https://github.com/AlexanderWoropai/LT (Дата обращения 18.11.2020)
5. GitHub - Alex-KISLOV/Payphone [Электронный ресурс] : - Режим доступа: https://github.com/Alex-KISLOV/Payphone (Дата обращения 18.11.2020)

Приложение А – Проверка на антиплагиат



Приложение Б – Листинг программы

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Курсовой_проект
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Стиралка());
        }
    }
}
```

Файл Clotching.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Курсовой_проект
{
```

```
    public class Clothing
    {
        public string ClothesName;
        public string Material;
        public double Weight;
        public Clothing (string clothesName, string material,
double weight)
        {
            ClothesName = clothesName;
            Material = material;
            Weight = weight;
        }
        public string Info()
        {
            string Show = String.Format($"Одежда:
{ClothesName}");
            return Show;
        }

    }
}
```

Файл Стиралка.cs

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

```
namespace Курсовой_проект
{
    public partial class Стиралка : Form
    {
        public static WindowsMediaPlayer _wmp = new
WindowsMediaPlayer();
        public string _currentTime = "  00:00";
        public static bool _doorOpened;
        public static double _weight;
        public static bool[] _forTemp = new bool[4];
        public static bool[] _forWashinmode = new bool[2];
        public static int _counTtick;
        public static bool _startstop;
        public static Clothing[] _clothing = new Clothing[15];
        public static int _temperature;
        public static double _coefficient;
        public static int _washingMode;
        public static int _mssec;
        public static bool _forstart;
        public Стиралка()
        {
            InitializeComponent();
            Таймер.Text = "\n" + _currentTime;
            Добавить.Hide();
            Одежда.Hide();
            ОткрытьЗакр.BackColor = Color.Red;
            Хлопок.Enabled = false;
            Синтетика.Enabled = false;
            Делткани.Enabled = false;
            Загрязнения.Enabled = false;
            Холодная.Enabled = false;
            Теплая.Enabled = false;
            Горячая.Enabled = false;
            Кипяток.Enabled = false;
            Отжим.Enabled = false;
            Полоскание.Enabled = false;
            СтартСтоп.Enabled = false;
            label16.Hide();
```

```
            button3.Hide();
        }
        private void CheckDoor()
        {
            _doorOpened = !_doorOpened;
            if (_doorOpened == true)
            {
                ОткрЗакр.Text = "Открыто";
                СтартСтоп.Enabled = false;
                if (_weight < 9)
                {
                    Одежда.Show();
                    Добавить.Show();
                }
            }
            else
            {
                ОткрЗакр.Text = "Закрыто";
                СтартСтоп.Enabled = true;
                Одежда.Hide();
                Добавить.Hide();
            }
        }
        private void ОткрытьЗакр_Click(object sender, EventArgs
e)
        {
            label16.Hide();
            button3.Hide();
            CheckDoor();
            Одежда.Items.Clear();
            if (_doorOpened == true || _weight == 0)
            {
                СтартСтоп.BackColor = SystemColors.Control;
                label11.Text = "Пуск";
                _wmp.URL = @"D:\sound1.mp3";
                _wmp.settings.volume = 100;
                _wmp.controls.play();
```

```
                _clothing[1] = new Clothing("Носки", "Хлопок",
0.1);
                _clothing[2] = new Clothing("Джинсы", "Хлопок",
0.6);
                _clothing[3] = new Clothing("Рубашка", "Хлопок",
0.2);
                _clothing[5] = new Clothing("Трусы", "Хлопок",
0.07);
                _clothing[6] = new Clothing("Шорты",
"Синтетика", 0.25);
                _clothing[7] = new Clothing("Свитер",
"Синтетика", 0.5);
                _clothing[8] = new Clothing("Толстовка",
"Синтетика", 0.3);
                _clothing[10] = new Clothing("Шапка",
"Деликантные ткани", 0.5);
                _clothing[11] = new Clothing("Шарф",
"Деликантные ткани", 0.13);
                _clothing[13] = new Clothing("Перчатки",
"Сильные загрязнения", 0.1);
                _clothing[14] = new Clothing("Пиджак", "Сильные
загрязнения", 0.5);
                Одежда.Items.Insert(0, "Хлопковые");
                Одежда.Items.Add(_clothing[1].Info());
                Одежда.Items.Add(_clothing[2].Info());
                Одежда.Items.Add(_clothing[3].Info());
                Одежда.Items.Insert(4, "Синтетические");
                Одежда.Items.Add(_clothing[5].Info());
                Одежда.Items.Add(_clothing[6].Info());
                Одежда.Items.Add(_clothing[7].Info());
                Одежда.Items.Add(_clothing[8].Info());
                Одежда.Items.Insert(9, "Деликатные ткани");
                Одежда.Items.Add(_clothing[10].Info());
                Одежда.Items.Add(_clothing[11].Info());
                Одежда.Items.Insert(12, "Сильно загрязненные");
                Одежда.Items.Add(_clothing[13].Info());
                Одежда.Items.Add(_clothing[14].Info());
                ОткрытьЗакр.BackColor = Color.Green;
                Хлопок.Enabled = false;
                Синтетика.Enabled = false;
                Делткани.Enabled = false;
```

```
                Загрязнения.Enabled = false;
                Холодная.Enabled = false;
                Теплая.Enabled = false;
                Горячая.Enabled = false;
                Кипяток.Enabled = false;
                Полоскание.Enabled = false;
                Отжим.Enabled = false;
            }
            else
            {
                _wmp.URL = @"D:\sound2.mp3";
                _wmp.settings.volume = 100;
                _wmp.controls.play();
                ОткрытьЗакр.BackColor = Color.Red;
                ОткрытьЗакр.Enabled = false;
                Добавить.Hide();
                Хлопок.Enabled = !false;
                Синтетика.Enabled = !false;
                Делткани.Enabled = !false;
                Загрязнения.Enabled = !false;
                Холодная.Enabled = !false;
                Теплая.Enabled = !false;
                Горячая.Enabled = !false;
                Кипяток.Enabled = !false;
                Полоскание.Enabled = !false;
                Отжим.Enabled = !false;
            }
            if (_weight == 0)
            {
                ОткрытьЗакр.Enabled = true;
            }
        }
        private void Добавить_Click(object sender, EventArgs e)
        {
            CountWeight.Text = " ";
            if (Одежда.SelectedIndex != 0 && Одежда.SelectedIndex != 4 && Одежда.SelectedIndex != 9 && Одежда.SelectedIndex != 12)
```

```
            {
                _weight                                                 +=
_clothing[Одежда.SelectedIndex].Weight;
                CountWeight.Text    +=    String.Format($"\nВес:
{_weight}кг");
            }
            if (_weight >= 9)
            {
                CountWeight.BackColor = Color.Red;
                Добавить.Enabled = false;
                Одежда.Hide();
            }
            if (_weight > 0 && _weight <= 5.5)
            {
                CountWeight.BackColor = Color.Green;
            }
            if (_weight > 6 && _weight < 9)
            {
                CountWeight.BackColor = Color.Yellow;
            }
        }
        private   void   Хлопок_CheckedChanged(object   sender,
EventArgs e)
        {
            if (Хлопок.Checked == true && _doorOpened == false)
            {
                Хлопок.BackColor = Color.Green;
                _coefficient += 1.3;
            }
            else
            {
                Хлопок.BackColor = SystemColors.ButtonHighlight;
                _coefficient = 0;
            }
        }
        private   void   Синтетика_CheckedChanged(object   sender,
EventArgs e)
        {
```

```
            if (Синтетика.Checked == true && _doorOpened ==
false)
            {
                Синтетика.BackColor = Color.Green;
                _coefficient += 1.2;
            }
            else
            {
                Синтетика.BackColor                                =
SystemColors.ButtonHighlight;
                _coefficient = 0;
            }
        }
        private void Делткани_CheckedChanged(object sender,
EventArgs e)
        {
            if (Делткани.Checked == true && _doorOpened ==
false)
            {
                Делткани.BackColor = Color.Green;
                _coefficient += 1;
            }
            else
            {
                Делткани.BackColor                                 =
SystemColors.ButtonHighlight;
                _coefficient = 0;
            }
        }
        private void Загрязнения_CheckedChanged(object sender,
EventArgs e)
        {
            if (Загрязнения.Checked == true && _doorOpened ==
false)
            {
                Загрязнения.BackColor = Color.Green;
                _coefficient = 1.7;
            }
            else
            {
```

```
                Загрязнения.BackColor                                    =
SystemColors.ButtonHighlight;
                _coefficient = 0;
            }

        }
        private void Холодная_Click(object sender, EventArgs e)
        {
            _forTemp[0] = !_forTemp[0];
            if (_forTemp[0] == true && _doorOpened == false)
            {
                Холодная.BackColor = Color.Green;
                Теплая.Enabled = false;
                Горячая.Enabled = false;
                Кипяток.Enabled = false;
                _temperature += 10;
            }
            else
            {
                Холодная.BackColor = DefaultBackColor;
                Теплая.Enabled = true;
                Горячая.Enabled = true;
                Кипяток.Enabled = true;
                _temperature = 0;
            }

        }
        private void Теплая_Click(object sender, EventArgs e)
        {
            _forTemp[1] = !_forTemp[1];
            if (_forTemp[1] == true && _doorOpened == false)
            {
                Теплая.BackColor = Color.Green;
                Холодная.Enabled = false;
                Горячая.Enabled = false;
                Кипяток.Enabled = false;
                _temperature += 30;
            }
```

```
            else
            {
                Теплая.BackColor = DefaultBackColor;
                Холодная.Enabled = true;
                Горячая.Enabled = true;
                Кипяток.Enabled = true;
                _temperature = 0;
            }
        }

        private void Горячая_Click(object sender, EventArgs e)
        {
            _forTemp[2] = !_forTemp[2];
            if (_forTemp[2] == true && _doorOpened == false)
            {
                Горячая.BackColor = Color.Green;
                Холодная.Enabled = false;
                Теплая.Enabled = false;
                Кипяток.Enabled = false;
                _temperature = 65;
            }
            else
            {
                Горячая.BackColor = DefaultBackColor;
                Холодная.Enabled = true;
                Теплая.Enabled = true;
                Кипяток.Enabled = true;
                _temperature = 0;
            }
            if (Хлопок.Checked == true || Синтетика.Checked ==
true || Делткани.Checked == true || Загрязнения.Checked == true
|| _forTemp[0] == true || _forTemp[1] == true || _forTemp[2] ==
true || _forTemp[3] == true || (_forWashinmode[0] == true ||
_forWashinmode[1] == true))
            {
                ОткрытьЗакр.Enabled = false;
            }
        }
```

```
        private void Кипяток_Click(object sender, EventArgs e)
        {
            _forTemp[3] = !_forTemp[3];
            if (_forTemp[3] == true && _doorOpened == false)
            {
                Кипяток.BackColor = Color.Green;
                Холодная.Enabled = false;
                Теплая.Enabled = false;
                Горячая.Enabled = false;
                _temperature += 100;
            }
            else
            {
                Кипяток.BackColor = DefaultBackColor;
                Холодная.Enabled = true;
                Теплая.Enabled = true;
                Горячая.Enabled = true;
                _temperature = 0;
            }
        }
        private void Полоскание_Click(object sender, EventArgs
e)
        {
            _forWashinmode[0] = !_forWashinmode[0];
            if (_forWashinmode[0] == true && _doorOpened ==
false)
            {
                _washingMode += 10;
                Полоскание.BackColor = Color.Green;
                Отжим.Enabled = false;
            }
            else
            {
                Полоскание.BackColor = DefaultBackColor;
                Отжим.Enabled = true;
                _washingMode = 0;
            }
        }
```

```
        private void Отжим_Click(object sender, EventArgs e)
        {
            _forWashinmode[1] = !_forWashinmode[1];
            if (_forWashinmode[1] == true)
            {
                Отжим.BackColor = Color.Green;
                Полоскание.Enabled = false;
                _washingMode += 20;
            }
            else
            {
                Отжим.BackColor = DefaultBackColor;
                Полоскание.Enabled = true;
                _washingMode = 0;
            }
        }
        public int CalculateTime()
        {
            int washingtime = (int)(0.5 * _temperature * _coefficient) + _washingMode;
            return washingtime;
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            if ((Хлопок.Checked == true || Синтетика.Checked == true || Делткани.Checked == true || Загрязнения.Checked == true) && (_forTemp[0] == true || _forTemp[1] == true || _forTemp[2] == true || _forTemp[3] == true) && (_forWashinmode[0] == true || _forWashinmode[1] == true))
            {
                _mssec++;
            }
        }
        public void Combo()
        {
            button3.Show();
            label16.Show();
            Таймер.Text = "\n" + _currentTime;
            TimerForWashing.Enabled = false;
```

```
            SystemWatch.Enabled = false;
            _wmp.controls.stop();
            CountWeight.Clear();
            Хлопок.Enabled = false;
            Синтетика.Enabled = false;
            Делткани.Enabled = false;
            Загрязнения.Enabled = false;
            Холодная.Enabled = false;
            Теплая.Enabled = false;
            Горячая.Enabled = false;
            Кипяток.Enabled = false;
            Отжим.Enabled = false;
            Полоскание.Enabled = false;
            Хлопок.Checked = false;
            Синтетика.Checked = false;
            Делткани.Checked = false;
            Загрязнения.Checked = false;
            СтартСтоп.Enabled = false;
            ОткрытьЗакр.Enabled = true;
            Хлопок.BackColor = SystemColors.ButtonHighlight;
            Синтетика.BackColor = SystemColors.ButtonHighlight;
            Делткани.BackColor = SystemColors.ButtonHighlight;
            Загрязнения.BackColor                                    =
SystemColors.ButtonHighlight;
            Холодная.BackColor = SystemColors.Control;
            Теплая.BackColor = SystemColors.Control;
            Горячая.BackColor = SystemColors.Control;
            Кипяток.BackColor = SystemColors.Control;
            Полоскание.BackColor = SystemColors.Control;
            Отжим.BackColor = SystemColors.Control;
            СтартСтоп.BackColor = SystemColors.Control;
            CountWeight.BackColor = SystemColors.Window;
            СтартСтоп.BackColor = SystemColors.Control;
            _forTemp[0] = false;
            _forTemp[1] = false;
            _forTemp[2] = false;
            _forTemp[3] = false;
            _forWashinmode[0] = false;
```

```
            _forWashinmode[1] = false;
            _weight = 0;
            _temperature = 0;
            _washingMode = 0;
            _coefficient = 0;
            _counTtick = 0;
            _forstart = false;
            label11.Text = "Пуск";
            _mssec = 0;
            button3.Enabled = false;
        }
        private void timer2_Tick(object sender, EventArgs e)
        {
            if ((Хлопок.Checked == true || Синтетика.Checked ==
true || Делткани.Checked == true || Загрязнения.Checked == true)
&& (_forTemp[0] == true || _forTemp[1] == true || _forTemp[2] ==
true || _forTemp[3] == true) && (_forWashinmode[0] == true ||
_forWashinmode[1] == true))
            {
                if (_counTtick != 0 && _mssec < 15)
                {
                    _counTtick--;
                }
                else
                {
                    Combo();

                }
            }
            Таймер.Text   =   String.Format("\n      {0:00:00}",
_counTtick);
        }
        private     void     СтартСтоп_MouseUp(object     sender,
MouseEventArgs e)
        {
            if ((Хлопок.Checked == true || Синтетика.Checked ==
true || Делткани.Checked == true || Загрязнения.Checked == true)
&& (_forTemp[0] == true || _forTemp[1] == true || _forTemp[2] ==
true || _forTemp[3] == true) && (_forWashinmode[0] == true ||
_forWashinmode[1] == true))
            {
```

```
                if (_mssec >= 15)
                {
                    Combo();
                }
                else
                {
                    TimerForWashing.Enabled                          =
!TimerForWashing.Enabled;
                }
                SystemWatch.Enabled = false;
                _mssec = 0;
                if (TimerForWashing.Enabled == false)
                {
                    label11.Text = "Пауза";
                    СтартСтоп.BackColor = Color.Yellow;
                    _wmp.controls.pause();
                }
                else
                {
                    label11.Text = "";
                    _wmp.controls.play();
                    СтартСтоп.BackColor = Color.Green;
                }
            }
        }
        private    void    СтартСтоп_MouseDown(object    sender,
MouseEventArgs e)
        {
            if ((Хлопок.Checked == true || Синтетика.Checked ==
true || Делткани.Checked == true || Загрязнения.Checked == true)
&& (_forTemp[0] == true || _forTemp[1] == true || _forTemp[2] ==
true || _forTemp[3] == true) && (_forWashinmode[0] == true ||
_forWashinmode[1] == true))
            {
                SystemWatch.Enabled = true;
                if (_forstart == false)
                {
                    _forstart = true;
                    if (_forWashinmode[0] == true)
                    {
```

```
                    _wmp.URL = @"D:\Rinsing.mp3";
                    _wmp.settings.volume = 70;
                }
                else
                {
                    _wmp.URL = @"D:\Spinning.mp3";
                    _wmp.settings.volume = 70;
                }
                SystemWatch.Enabled = true;
                TimerForWashing.Interval = 1000;
                if (_counTtick == 0)
                {
                    _counTtick = CalculateTime();
                }
                СтартСтоп.BackColor = Color.Green;
                Хлопок.Enabled = false;
                Синтетика.Enabled = false;
                Делткани.Enabled = false;
                Загрязнения.Enabled = false;
                Холодная.Enabled = false;
                Теплая.Enabled = false;
                Горячая.Enabled = false;
                Кипяток.Enabled = false;
                Отжим.Enabled = false;
                Полоскание.Enabled = false;
                ОткрытьЗакр.Enabled = false;
            }
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {

    }
}
}
```