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: «Телевизор»

(тема курсовой работы)

Выполнил студент 2 курса группы 18-К-АС1

Шевченко П.С.

(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: Шевченко П.С. группы 18-К-АС1 курса 2

(Ф.И.О.) (№ группы и курса)

Тема проекта: «Телевизор»

План работы:

1. Изучение предметной области

2. Проектирование

3.  Описание реализованных диаграмм

Объем работы:

а) пояснительная записка 52 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 «11» мая 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 Шевченко П.С. Ф.И.О.

Реферат

Курсовая работа: 52 страниц, 18 рисунков, 8 используемых источников.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, ПОСЛЕДОВАТЕЛЬНОСТИ, МОДЕЛЬ, КЛАСС, ТЕЛЕВИЗОР, UML ДИАГРАММЫ, IDEF0 ДИАГРАММА, BPMN ДИАГРАММА, DFD ДИАГРАММА, EPC ДИАГРАММА, ДИАГРАММА ГАНТА.

Объектом исследования является разработка программного обеспечения ПО телевизора, которое способно обеспечивать комфортный просмотр каналов, изменение параметров и функцию первоначальной настройки оборудования.

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

По итогу проделанной работы был разработан симулятор телевизора, написанный на языке программирование С, получены EPC, UML, DFD, FURPS+ диаграммы, обладающие достаточной информацией о ПО телевизора. Была проведена работа с веб-ресурсом Github. Создан свой репозиторий с исходным кодом программы и всеми файлами программного продукта.

**Содержание**

[Введение 6](#_Toc40464235)

[1. Формулировка задачи 7](#_Toc40464236)

[2. Язык проектирования UML 8](#_Toc40464237)

[3. Создание модели Аs-Isв стандарте IDEF0 9](#_Toc40464238)

[4. Диаграмма потоков данных (DFD) 14](#_Toc40464239)

[5. Диаграмма Ганта 15](#_Toc40464240)

[6. Проектирование диаграммы EPC 17](#_Toc40464241)

[7. Проектирование диаграммы BPMN 18](#_Toc40464242)

[8. Требования к системе: классификация FURPS+ 19](#_Toc40464243)

[9. Проектирование диаграммы вариантов использования 20](#_Toc40464244)

[10. Проектирование диаграммы последовательности 22](#_Toc40464245)

[11. Результаты машинного тестирования программы 25](#_Toc40464246)

[12. Системные требования 27](#_Toc40464247)

[13. Руководство пользователя 28](#_Toc40464248)

[14. Сопровождение ПО 30](#_Toc40464249)

[Заключение 31](#_Toc40464250)

[Список использованных источников 32](#_Toc40464251)

[Приложение А – Диаграмма Ганта 33](#_Toc40464252)

[Приложение Б – Антиплагиат 34](#_Toc40464253)

[Приложение В – Код программы 35](#_Toc40464255)

Введение

В данный момент телевизор есть почти в каждом доме, скорее редкость если его нет, чем то, что их может быть даже несколько, по одному в каждой комнате. Люди привыкли, приходя домой, закончив все свои рабочие и домашние дела ложиться на диван и включать телевизор, неважно фильмы это или сериалы, музыка или новости, таким образом, многие привыкли отдыхать. И это не удивительно, человек уже не задумывается, что будет делать, придя домой и делает это на автомате.

Есть категория людей, которые смотрят телевизор достаточно редко, чтобы посмотреть какой-то отдельный интересный для него фильм, или, чтобы узнать последние новости в мире. Это люди, которые адекватно воспринимают телевидение и разумно его используют, понимая истинное его назначение. Активное использование телевизоров в домах людей, организациях и предприятиях ещё раз доказывает, что данная технология стала неотъемлемой частью человеческой жизни.

Впрочем, несмотря на частое использование данной технологии, она никогда не смогла бы обеспечить должный уровень удобства без тщательного исследования предметной области и проведения различных тестов, учитывающих всевозможные взаимодействия с компонентами системы. Именно поэтому задачей данного курсового проекта является разработка симулятора пробной модели, проектирование основных видов диаграмм, позволяющих в полной мере раскрыть все аспекты исследуемой системы. В результате проделанной работы телевизор, разработанный на основании всех проведённых исследований, будет готов к удобному использованию.

1. Формулировка задачи

Задачей данного курсового проекта является разработка программного обеспечение встроенной системы управления работой телевизора. Телевизор должен состоять из следующих компонентов:

– приемника телевизионного сигнала;

– устройства отображения картинки;

– памяти каналов;

– управляющих кнопок;

– пульта.

Управление телевизором осуществляется при помощи кнопок на корпусе (их четыре: «ВКЛ / ВЫКЛ», «-», «+», кнопка начальной установки) и пульта ДУ. Кнопка «ВКЛ / ВЫКЛ» позволяет включать и выключать телевизор. После включения телевизора на экран отображается передача, идущая по каналу №1, при этом используются параметры изображения и значение громкости, сохраненные в памяти настроек. Память каналов телевизора хранит до 60 каналов. Каналы нумеруются, начиная с нуля. Последовательное переключение каналов осуществляется при помощи кнопок «-» и «+». Нажатие на «+» переключает телевизор на канал с номером, на единицу большим (с 59-го канал телевизор переключается на 0-ой). Нажатие на «-» переключает телевизор на канал с номером, на единицу меньшим (с 0-го канала телевизор переключается на 59-ый).

2. Язык проектирования UML

UML – унифицированный язык моделирования (Unified Modeling Lаnguаge) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

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.

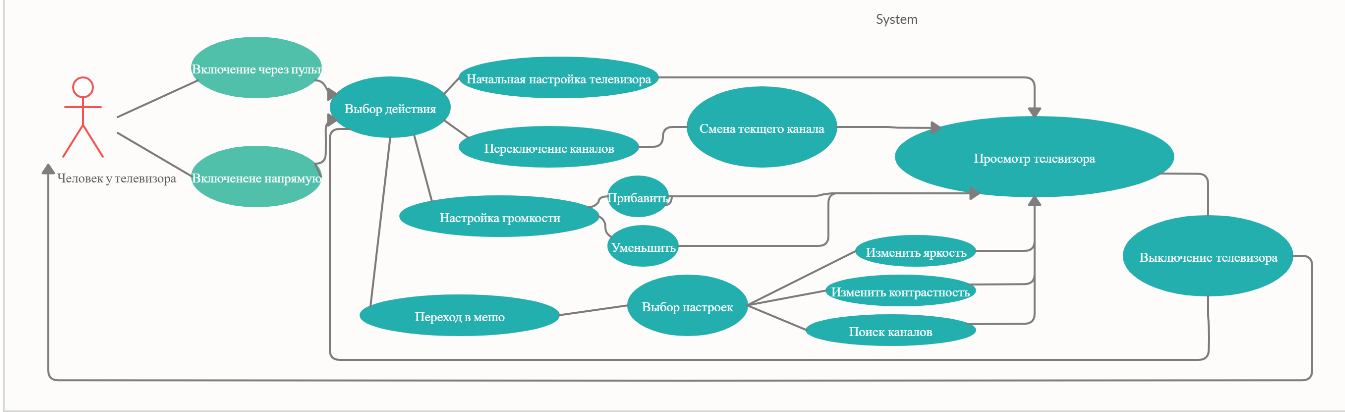


Рисунок 1 – UML-диаграмма системы «Телевизор»

3. Создание модели Аs-Isв стандарте IDEF0

Для того чтобы хорошо оценить возможности, разрабатываемого ПО нужно построить её базовую модель, которую можно представить в виде диаграммы Аs-Is.

Диаграмма Аs-Is – это функциональная модель системы «как есть», позволяющая узнать, где находятся уязвимые места, какие будут плюсы и минусы, протекающих в ней бизнес-процессов относительно конкурентов. Использование данной модели поможет чётко зафиксировать какие информационные объекты принимают участие в жизненном цикле системы, какая информация будет поступать на вход и что будет получаться на выходе. Модель Аs-Is, строится с использованием нотации IDEF0.

IDEF0 –  [методология](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) функционального моделирования ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *function modeling*) и графическая нотация, предназначенная для формализации и описания [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность ([поток работ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D1%80%D0%B0%D0%B1%D0%BE%D1%82)).

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 Аs-Is системы «Телевизор».

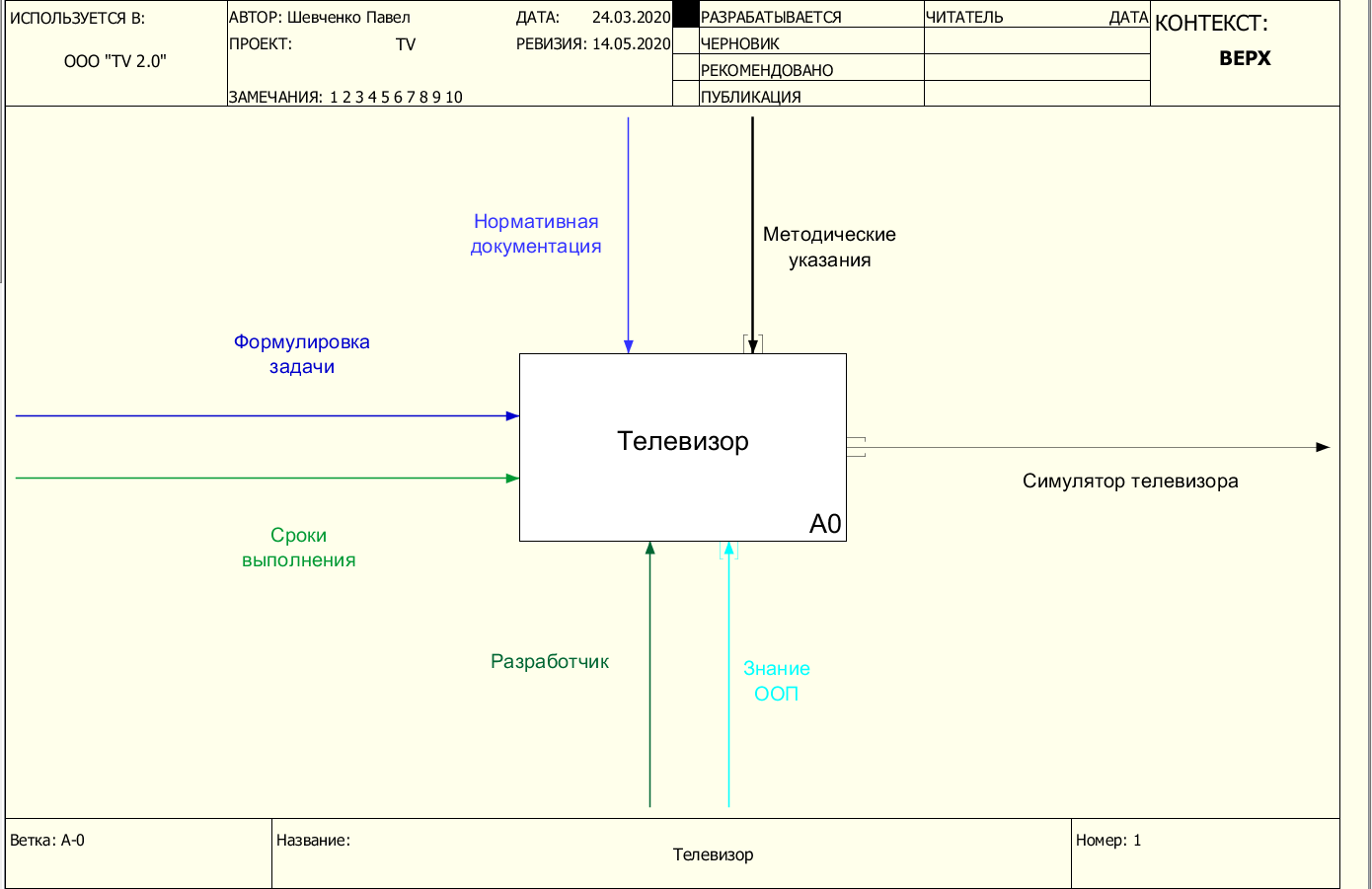


Рисунок 2 – Модель Аs-Is системы «Телевизор»

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 3 можно видеть модель системы «Телевизор» после декомпозиции.

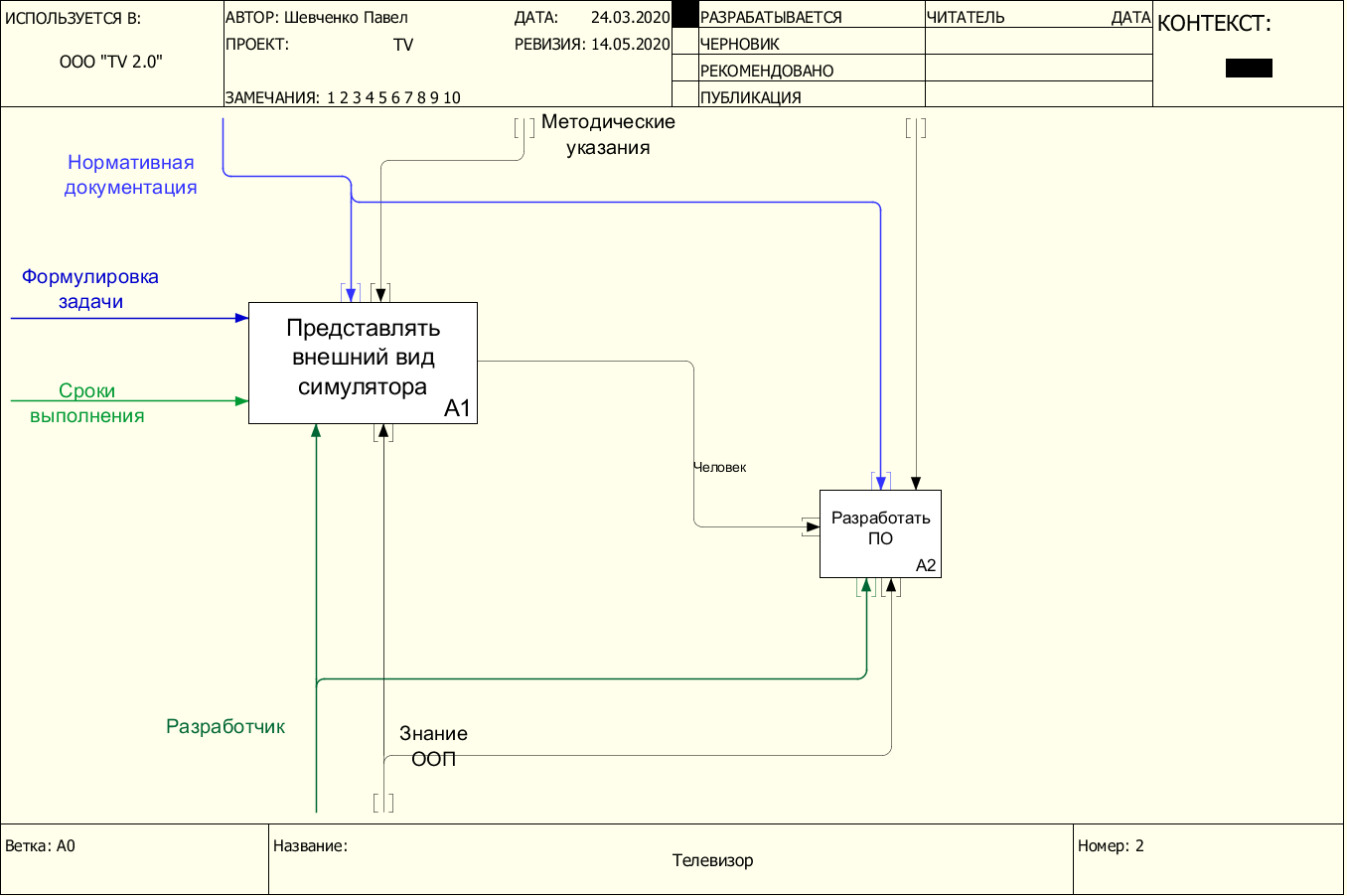


Рисунок 3 – Декомпозиция системы «Телевизор»

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

На рисунке 4 можно видеть модель ПО «Телевизор» после декомпозиции.

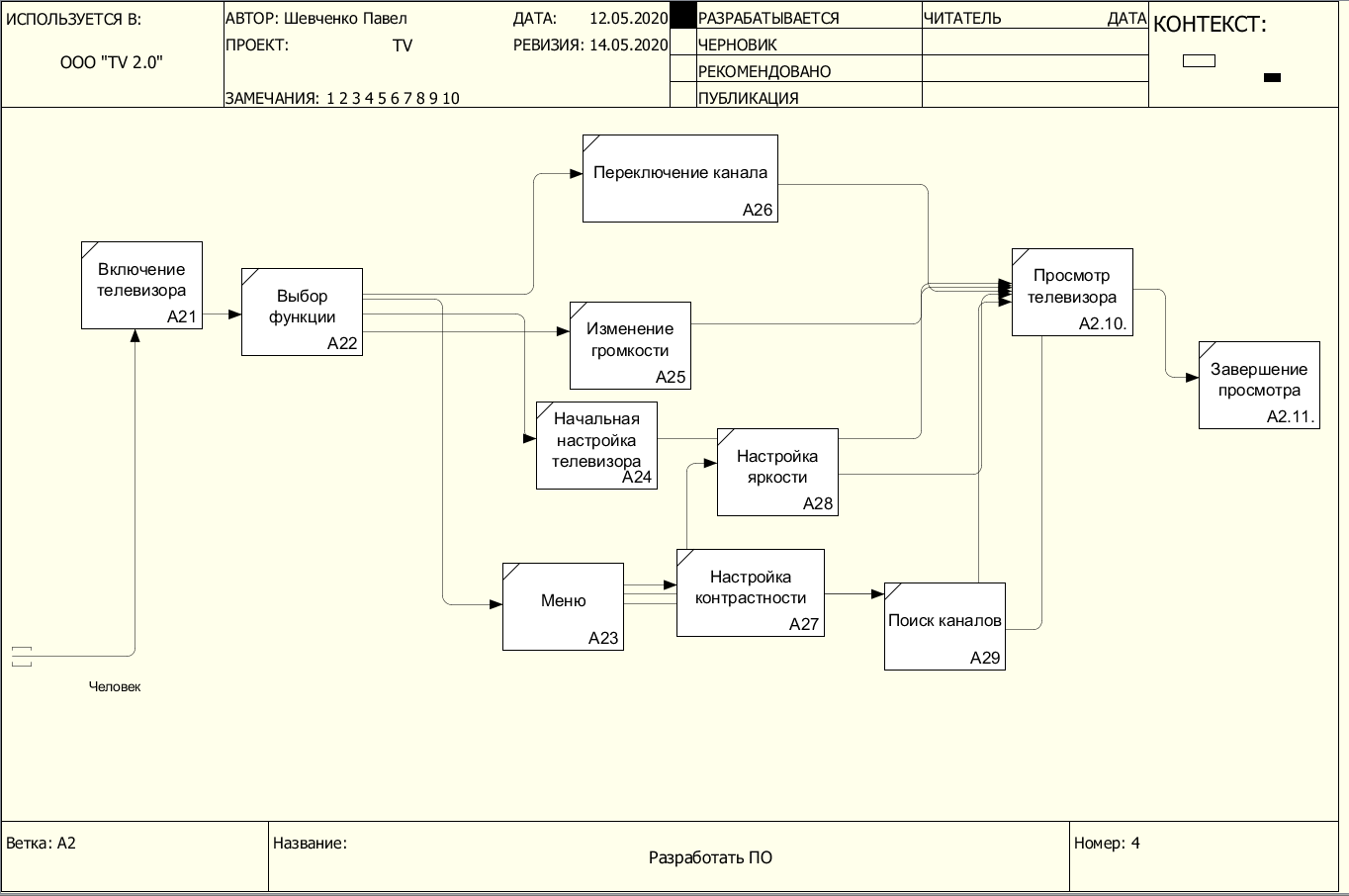


Рисунок 4 – Декомпозиция кода ПО «Телевизор»

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

Управление происходит при помощи пульта и кнопок на корпусе телевизора, а также правилам, описанным в инструкции.

Механизмом реализации работы системы человек перед телевизором, пульт от телевизора и ПО телевизора.

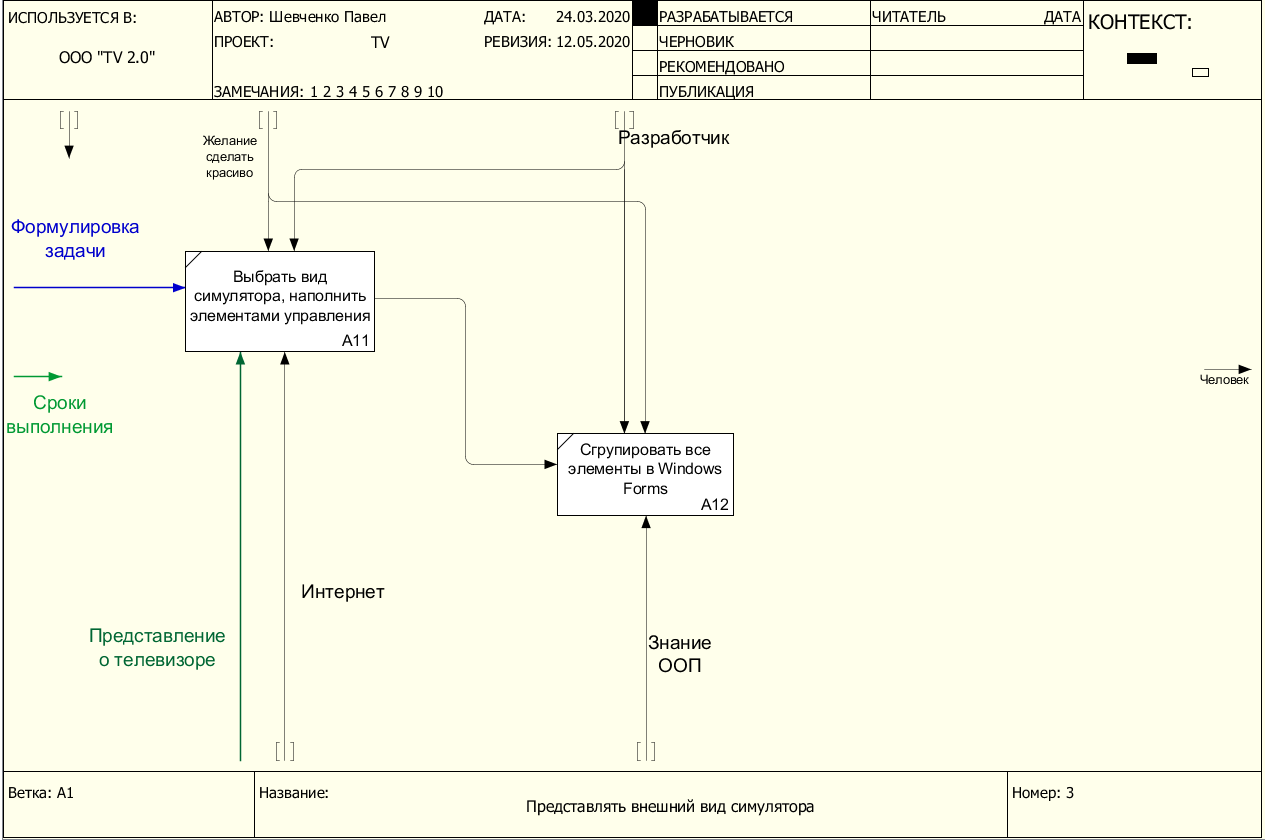


Рисунок 5 – Декомпозиция представления как должен выглядеть симулятор телевизора

Результатом деятельности системы является комфортный просмотр телевизора.

4. Диаграмма потоков данных (DFD)

Диаграмма потоков данных DFD (DataFlowDiagrams)– это 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 DFD – это один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML.

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

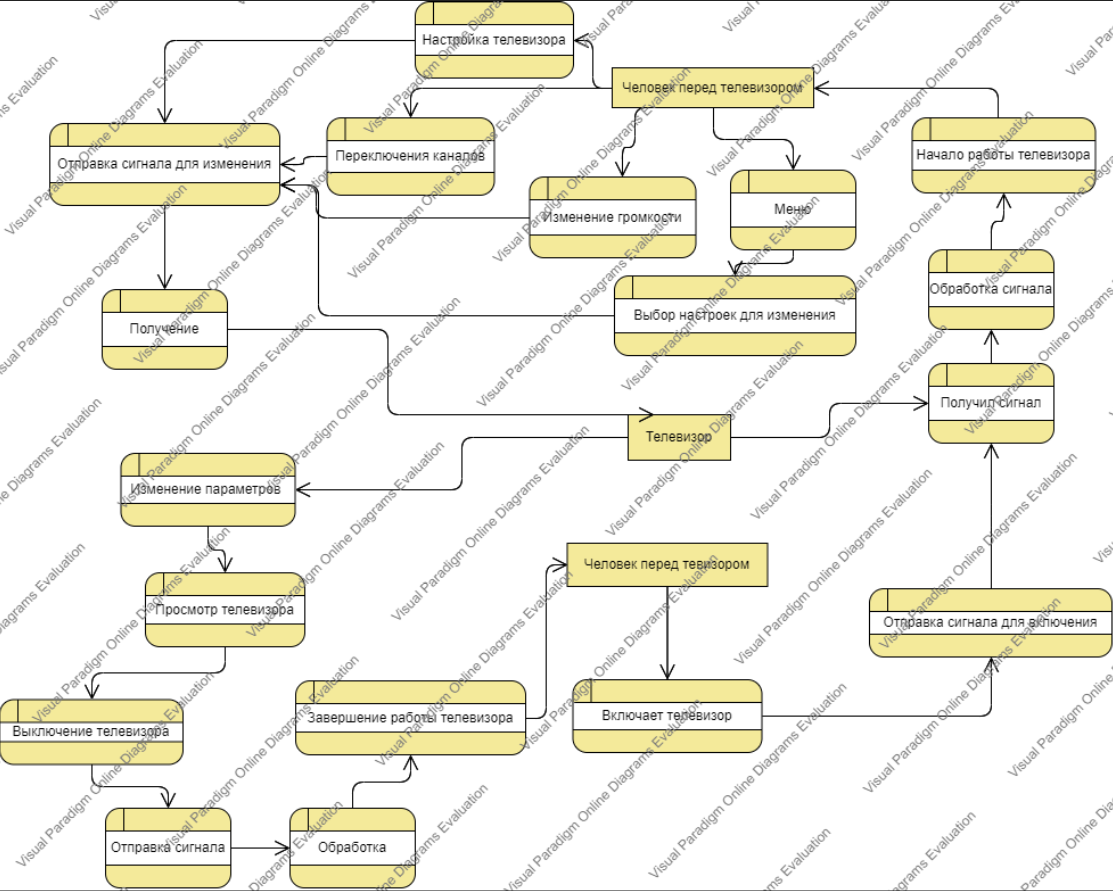


Рисунок 6 – Диаграмма DFD системы «Телевизор»

Внешними сущностями данной системы является человек перед телевизором, а также телевизор. Система также содержит одну базу данных для хранения стандартных установок телевизора.

5. Диаграмма Ганта

Диаграмма Ганта — это инструмент планирования, управления задачами, который придумал американский инженер Генри Гант (Henry Gаntt). 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 «Телевизор» находится в «Приложении А».

6. Проектирование диаграммы EPC

**EPC-диаграммы** — один из видов блок-схем, предназначенный для бизнес-моделирования. Название происходит от английского event-driven process chаin, что переводится как **событийная цепочка процессов**.

По сравнению с [базовыми блок-схемами](https://grapholite.ru/flowcharts/), нотация EPC более приспособлена для моделирования бизнес-процессов, планирования потоков работ и ресурсов предприятий.

EPC-метод был разработан Августом-Вильгельмом Шеером в рамках работ над созданием АRIS в начале 1990-х годов. Используется многими организациями для моделирования, анализа и реорганизации бизнес-процессов.

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.

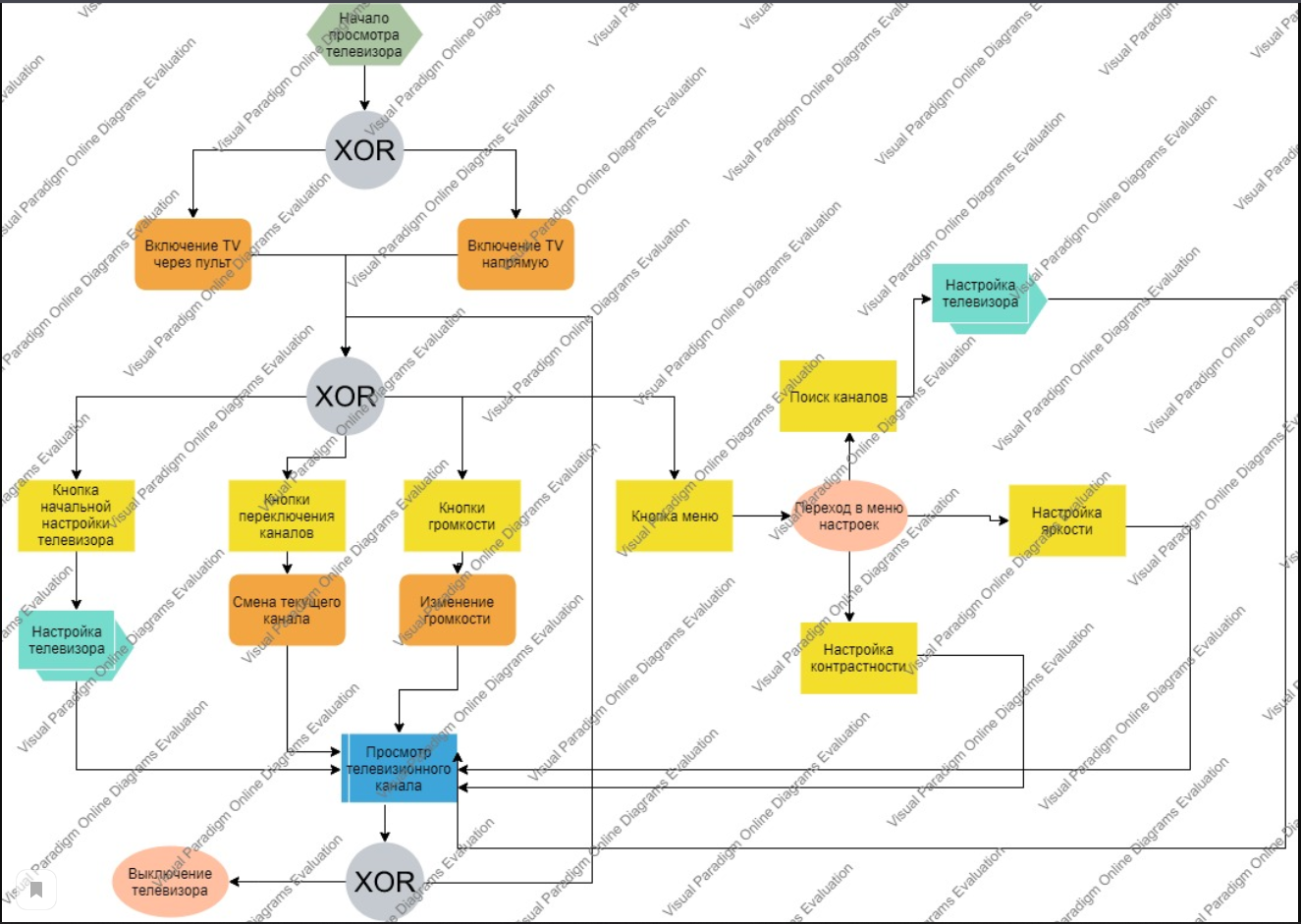


Рисунок 7 – EPC-диаграмма системы «Телевизор»

7. Проектирование диаграммы BPMN

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

Говоря проще, такая нотация представляет собой описание графических элементов, используемых для построения схемы протекания бизнес-процесса.

Как минимум, такая схема нужна, чтобы выстроить в соответствии с ней бизнес процесс и понятно регламентировать его для всех участников.  
Как максимум, [моделирование BPMN](https://www.comindware.com/ru/bpmn/) позволяет впоследствии провести автоматизацию бизнес-процессов в соответствии с имеющейся схемой.

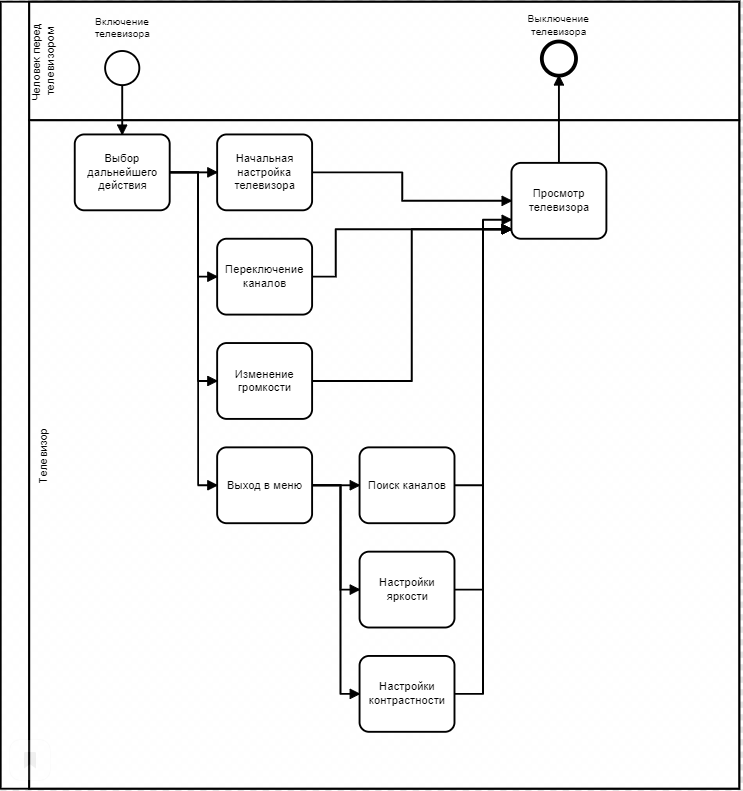


Рисунок 8 – Диаграмма BPMN «Аs-Is» и «To be»

8. Требования к системе: классификация FURPS+

Классификация требований к системе FURPS+ была разработана Робертом Грэйди (Robert Grаdy) из Hewlett-Pаckаrd и предложена в 1992 году. Сокращение FURPS расшифровывается так:

1. Functionаlity, функциональность
2. Usаbility, удобство использования
3. Reliаbility, надежность
4. Performаnce, производительность
5. Supportаbility, поддерживаемость
6. + необходимо помнить о таких возможных ограничениях, как:

* ограничения проектирования, design
* ограничения разработки, implementаtion
* ограничения на интерфейсы, interfаce
* физические ограничения, physicаl

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 – 3 секунды; время готовности системы к работе – 5 сек.

P – время отклика системы 0.01 сек, 100% эффективность работы, пропускная способность 30 запросов в минуту; потребление ресурсов – 60 Вт/ час;

S – легкая настройка;

+ - память каналов ограничена (максимальный размер 59 каналов).

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 Телевизора, которое представляет собой хранилище настроек, каналов. Для каждой настройки хранятся свои параметры. Пользователь может изменить начальные настройки, переключить канал. Таким образом, можно сделать вывод, что Телевизор 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 9 – 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 «Телевизор», моделируемой в данном курсовом проекте можно выделить несколько вариантов развития событий. Первый – человек хочет включить телевизор. Этот процесс показан на рисунке 10:

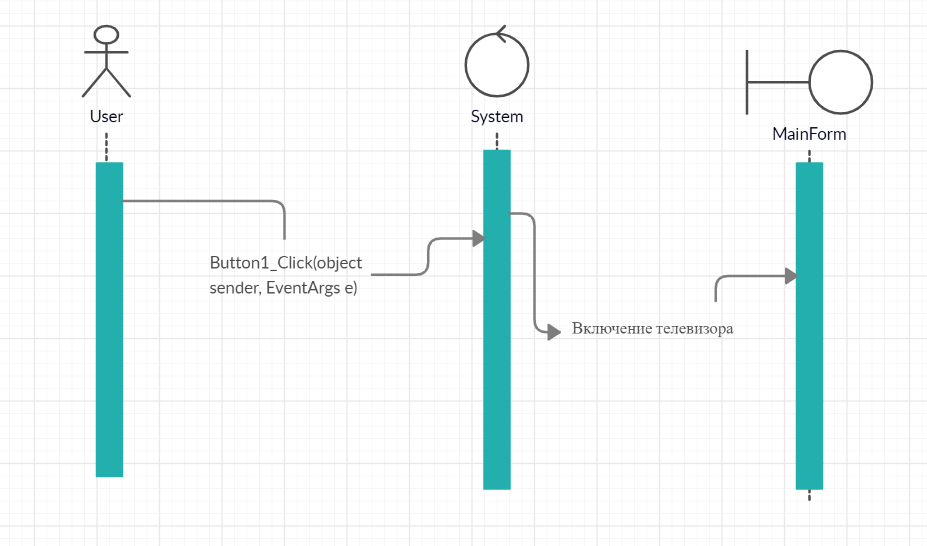


Рисунок 10 – Диаграмма последовательности (включение)

Следующий вариант развития событий показывает, цепочку действий для смены канала. Этот процесс показан на рисунке 11:

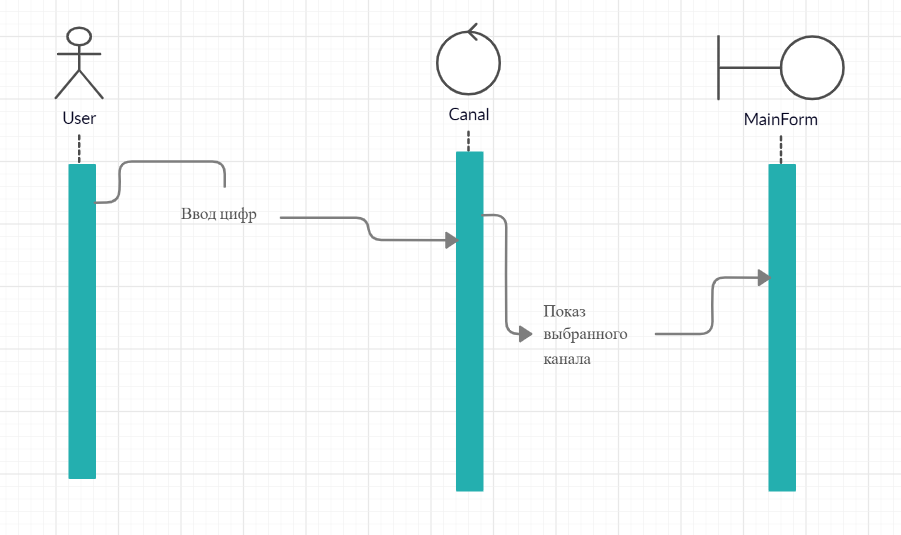


Рисунок 11 – Диаграмма последовательности (смена канала)

Третий вариант развития событий демонстрирует последовательность действий, необходимую для настроек телевизора. Этот процесс показан на рисунке 12:

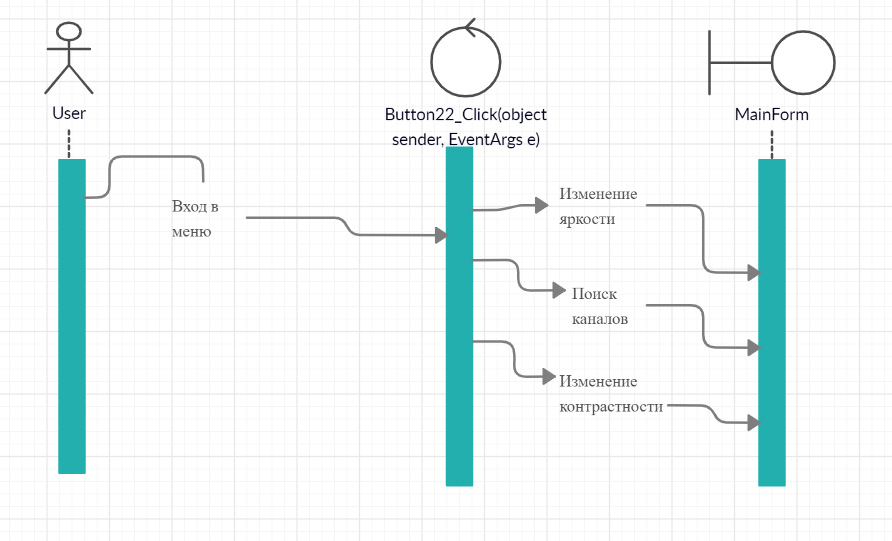


Рисунок 12 – Диаграмма последовательности (настройка телевизора)

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

11. Результаты машинного тестирования программы

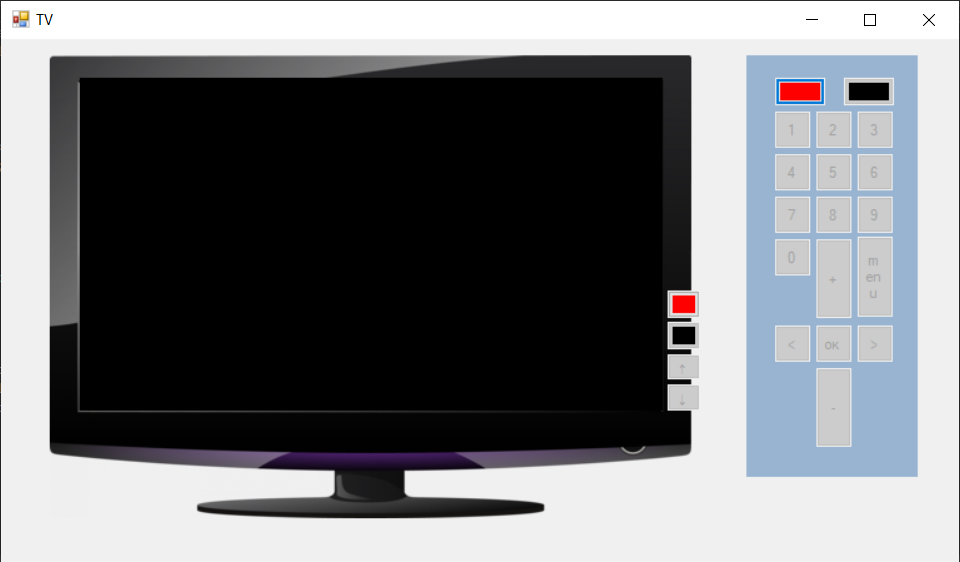
Рисунок 13 – Выключенный телевизор

Рисунок 14 – Включенный телевизор

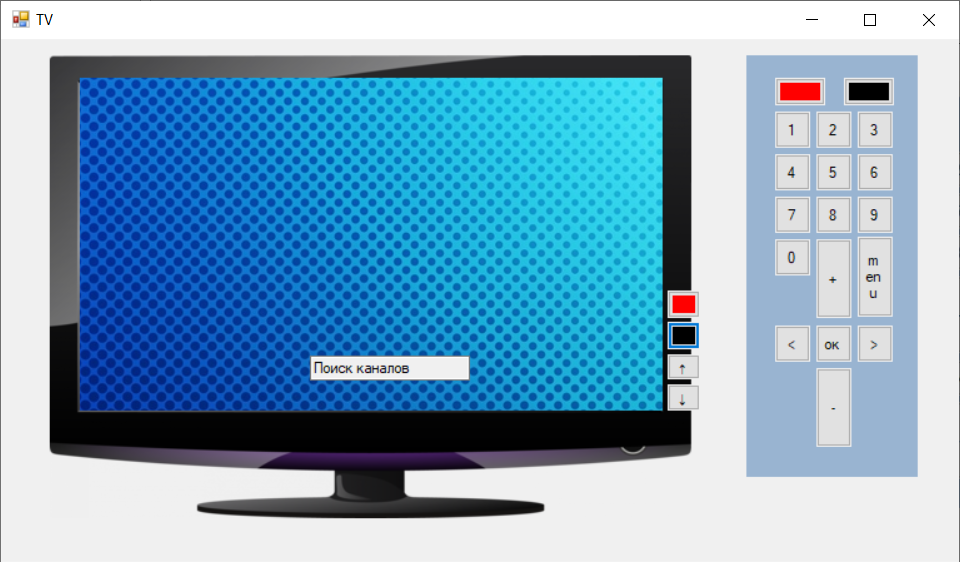


Рисунок 15 – Режим поиска каналов

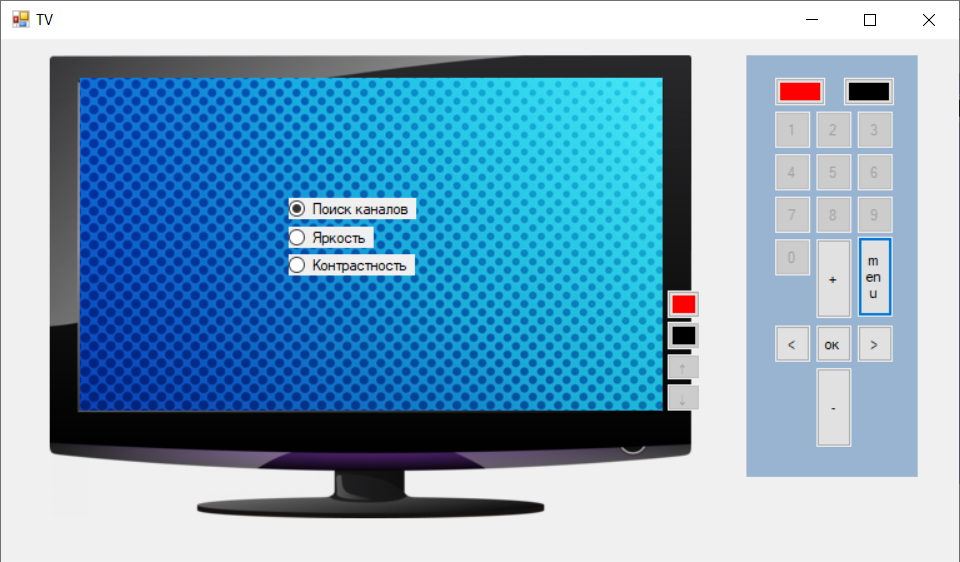


Рисунок 16 – Меню

12. Системные требования

Таблица 1 – Системные требования программы

|  |  |
| --- | --- |
| Процессор | 300 МГц |
| Оперативная память | 33 Мб |
| Монитор | 1280 x 720 |
| Свободное место на носителе | 10 Мб |
| Устройства взаимодействия с пользователем | Клавиатура и мышь |
| Программное обеспечение | Visuаl Studio 2019 года последней версии |

13. Руководство пользователя

Запустить программу можно с помощью TV 2.0.exe, либо открыть sln файл с названием TV 2.0.

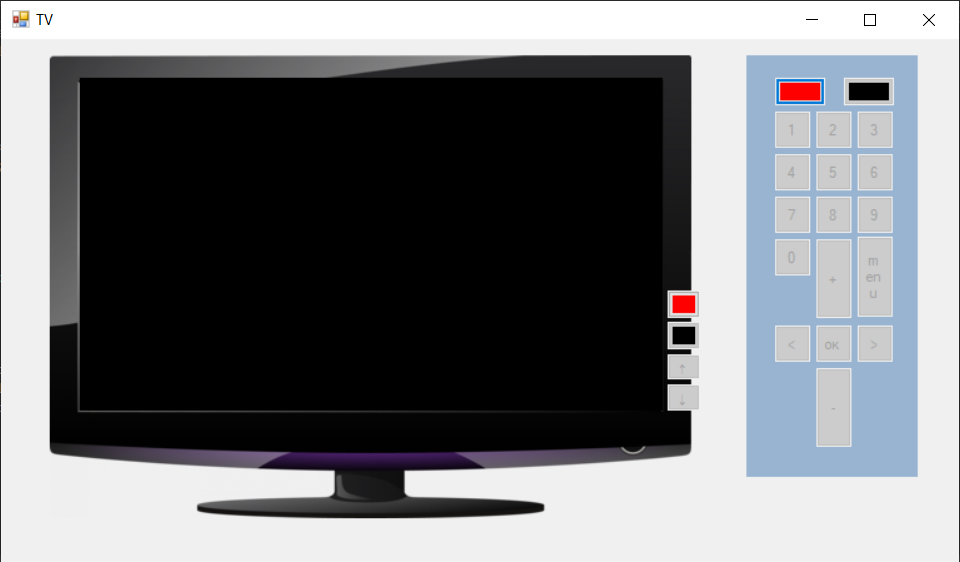
Запуск программы через exe файл:

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 Visuаl Studio;
4. Далее попадаем в проект и запускаем его.

Теперь перед нами интерфейс программы: симулятор телевизора и пульт от него.

Рисунок 17 – Начальный интерфейс программы

Для начала работы программы пользователь должен включить телевизор. Выполнить это можно нажав кнопку на корпусе или на пульте. Далее следует выбрать нужную функцию. Для смены канала можно воспользоваться кнопками «+» и «-» или использовать цифровой блок пульта. Изменение громкости можно провести с помощью кнопок «<» и «>». Чтобы изменить настройки телевизора следует перейти в меню, нажав соответствующую кнопку. Далее выбрать из представленных функций необходимую, подтвердив свой выбор кнопкой «ок», переход между позициями осуществляется кнопками «+» и «-».

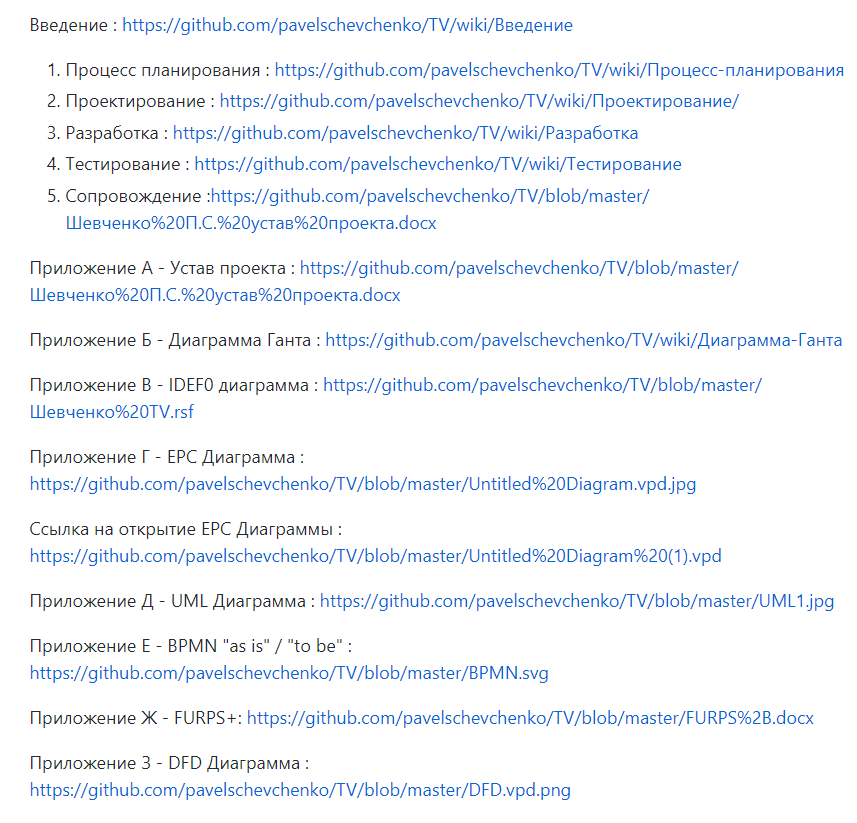
В любой момент использования программы телевизор можно выключить кнопкой «ВКЛ/ВЫКЛ», после этого все кнопки будут недоступны для выполнения всех функций.

14. Сопровождение ПО

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

https://github.com/pаvelschevchenko/TV/wiki/Оглавление

На рисунке 18 показано оглавление:



Заключение

В результате выполнения данного курсового проекта было разработано ПО для симулятора телевизора. Программирование осуществлялось на языке высокого уровня С#, использование данного языка позволило удобно разработать симулятор и наглядно показать результат работы при помощи встроенного в Visuаl 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.

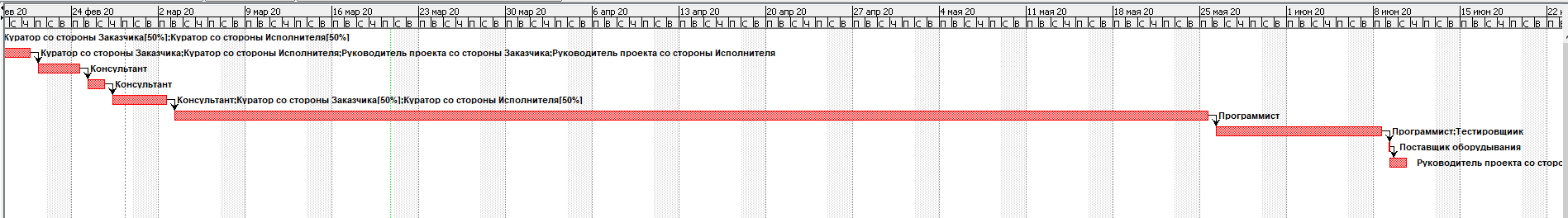
Благодаря детальному разбору проекта при помощи диаграмм проектирования, полученных в процессе разработки, можно с уверенностью сказать, что разработанное ПО удовлетворяет все потребности потребителя, позволяет комфортно использовать телевизор.

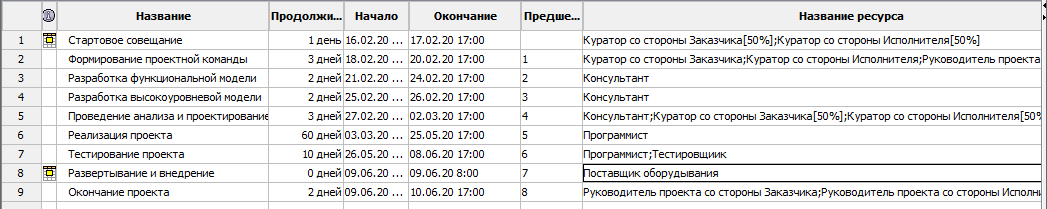
Были получены важные знания и практические навыки как в области использования объектно-ориентированных языков программирования в целом, так и в области построения диаграмм проектирования, отображающих поведение различных организационных структур.

Список использованных источников

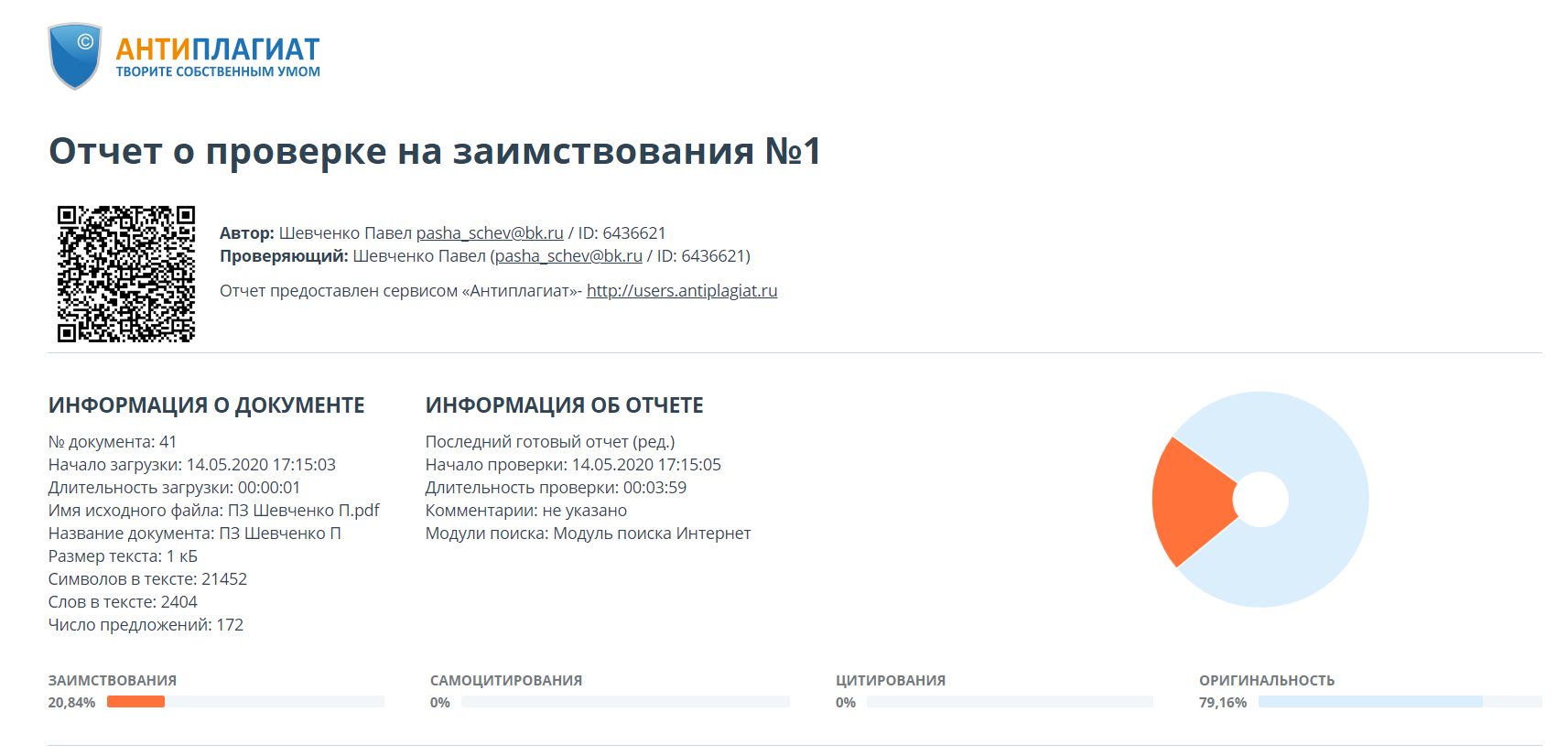
1. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / КрэгЛарман. - Москва: Гостехиздат, 2017. - 736 c.
2. Роберт А. Максимчук. UML для простых смертных / Роберт А. Максимчук, Эрик Дж. Нейбург. - Москва: СИНТЕГ, 2014. - 272 c.
3. Йордон, Эдвард. Объектно-ориентированный анализ и проектирование систем / Эдвард Йордон , Карл Аргила. - М.: ЛОРИ, 2014. - 264 c.
4. SoloLeаrn – C# Tutoriаl. [Электронный ресурс]: - Режим доступа: [https://www.sololeаrn.com/Course/CShаrp/](https://www.sololearn.com/Course/CSharp/) (Дата обращения 13.03.2020).
5. Википедия. [Электронный ресурс]: - Режим доступа: [https://ru.wikipediа.org](https://ru.wikipedia.org/wiki/Тестовые_функции_для_оптимизации) (Дата обращения 17.09.2019).
6. GitHub – pаvelschevchenko/TV. [Электронный ресурс]: - Режим доступа: https://github.com/pаvelschevchenko/TV (Дата обращения 9.05.2020).
7. Comindwаre – Нотация BPMN 2.0 [Электронный ресурс]: - Режим доступа: [https://comindwаre.com/ru/blog-нотация-bpmn-2-0-элементы-и-описание/](https://comindware.com/ru/blog-нотация-bpmn-2-0-элементы-и-описание/) (Дата обращения 28.02.2020)
8. SysАnа– Требования к системе: классификация FURPS+ [Электронный ресурс]: - Режим доступа: [https://sysаnа.wordpress.com/2010/09/16/furps/](https://sysana.wordpress.com/2010/09/16/furps/) (Дата обращения 03.03.2020)

Приложение А – Диаграмма Ганта

****

****

Приложение Б – Антиплагиат



Приложение В – 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 TV\_2.\_0

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

textBox1.Visible = false;

label1.BackColor = System.Drawing.Color.Transparent;

}

int c = 1;

string s = " ";

int gr =15;

int per = 0;

int volume = 15;

int yark = 50;

int kontr = 50;

int contr = 0;

int yarc = 0;

int menu = 0;

private void Button3\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '1';

label1.Text = s;

}

private void Button4\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '2';

label1.Text = s;

}

private void Button5\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '3';

label1.Text = s;

}

private void Button6\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '4';

label1.Text = s;

}

private void Button7\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '5';

label1.Text = s;

}

private void Button8\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '6';

label1.Text = s;

}

private void Button9\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '7';

label1.Text = s;

}

private void Button10\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '8';

label1.Text = s;

}

private void Button15\_Click(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '9';

label1.Text = s;

}

private void Button16\_Click(object sender, EventArgs e)

{

label1.Visible = true;

if (s.Length > 2)

s = s;

else

s = s + '0';

label1.Text = s;

}

public void Canal(int c)

{

if (c == 1)

pictureBox2.Image = Properties.Resources.imgonline\_com\_ua\_Resize\_CH6TIf9CpHEX4V;

if (c == 2)

pictureBox2.Image = Properties.Resources.\_10;

if (c == 3)

pictureBox2.Image = Properties.Resources.\_13;

if (c == 4)

pictureBox2.Image = Properties.Resources.\_14;

if (c == 5)

pictureBox2.Image = Properties.Resources.\_15;

if (c == 6)

pictureBox2.Image = Properties.Resources.\_16;

if (c == 7)

pictureBox2.Image = Properties.Resources.\_17;

if (c == 8)

pictureBox2.Image = Properties.Resources.\_18;

if (c == 9)

pictureBox2.Image = Properties.Resources.\_19;

if (c == 10)

pictureBox2.Image = Properties.Resources.\_3;

if (c == 11)

pictureBox2.Image = Properties.Resources.\_4;

if (c == 12)

pictureBox2.Image = Properties.Resources.\_6;

if (c == 13)

pictureBox2.Image = Properties.Resources.\_7;

if (c == 14)

pictureBox2.Image = Properties.Resources.\_8;

if (c == 15)

pictureBox2.Image = Properties.Resources.\_9;

if(c==16)

pictureBox2.Image = Properties.Resources.\_7;

if (c == 17)

pictureBox2.Image = Properties.Resources.\_19;

if (c == 18)

pictureBox2.Image = Properties.Resources.\_3;

if (c == 19)

pictureBox2.Image = Properties.Resources.\_15;

if (c == 20)

pictureBox2.Image = Properties.Resources.\_16;

if (c == 0)

pictureBox2.Image = Properties.Resources.\_3;

}

private void Button1\_Click(object sender, EventArgs e)

{

if(per==0)

{

pictureBox3.Visible = false;

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

button7.Enabled = true;

button8.Enabled = true;

button9.Enabled = true;

button10.Enabled = true;

button11.Enabled = true;

button12.Enabled = true;

button13.Enabled = true;

button14.Enabled = true;

button15.Enabled = true;

button16.Enabled = true;

button17.Enabled = true;

button20.Enabled = true;

button21.Enabled = true;

button19.Enabled = true;

button22.Enabled = true;

button2.Enabled = true;

pictureBox2.Image = Properties.Resources.imgonline\_com\_ua\_Resize\_CH6TIf9CpHEX4V;

timer1.Enabled = true;

per++;

canal = 1;

}

else

{

button3.Enabled = false;

button4.Enabled = false;

button5.Enabled = false;

button6.Enabled = false;

button7.Enabled = false;

button8.Enabled = false;

button9.Enabled = false;

button10.Enabled = false;

button11.Enabled = false;

button12.Enabled = false;

button13.Enabled = false;

button14.Enabled = false;

button15.Enabled = false;

button16.Enabled = false;

button17.Enabled = false;

button20.Enabled = false;

button21.Enabled = false;

button19.Enabled = false;

button2.Enabled = false;

button22.Enabled = false;

radioButton1.Visible = false;

radioButton2.Visible = false;

radioButton3.Visible = false;

textBox2.Visible = false;

pictureBox2.BackColor = Color.Black;

pictureBox3.Visible = true;

per = 0;

}

textBox1.Visible = false;

int volume = 15;

}

private void Button16\_Click\_1(object sender, EventArgs e)

{

timer8.Interval = 2000;

timer8.Start();

label1.Visible = true;

s = s + '0';

label1.Text = s;

}

int canal;

int countOK = 0;

private void Button17\_Click(object sender, EventArgs e)

{

if(menu==1)

{

countOK++;

switch (countOK)

{

case 1:

if (marker == 1)

{

radioButton1.Visible = false;

radioButton2.Visible = false;

radioButton3.Visible = false;

textBox2.Visible = true;

pictureBox2.Image = Properties.Resources.tekstura\_siniy\_fon\_ten\_50468\_1440x900;

textBox2.Text = "Поиск каналов";

timer3.Interval = 5000;

timer3.Start();

timer4.Interval = 5000;

timer4.Start();

button22.Enabled = false;

countOK = 0;

}

if (marker == 2)

{

textBox2.Visible = true;

radioButton1.Visible = false;

radioButton2.Visible = false;

radioButton3.Visible = false;

button22.Enabled = false;

yarc = 1;

textBox2.Text = "Яркость: " + yark;

}

if (marker == 3)

{

textBox2.Visible = true;

button22.Enabled = false;

radioButton1.Visible = false;

radioButton2.Visible = false;

radioButton3.Visible = false;

textBox2.Text = "Контрастность" + kontr;

contr = 1;

}

break;

case 2:

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

button7.Enabled = true;

button8.Enabled = true;

button9.Enabled = true;

button10.Enabled = true;

button11.Enabled = true;

button12.Enabled = true;

button13.Enabled = true;

button14.Enabled = true;

button15.Enabled = true;

button16.Enabled = true;

button17.Enabled = true;

button20.Enabled = true;

button21.Enabled = true;

button19.Enabled = true;

button22.Enabled = true;

button2.Enabled = true;

button22.Enabled = true;

textBox2.Visible = false;

menu = 0;

countOK = 0;

Canal(1);

break;

}

}

}

private void Timer1\_Tick(object sender, EventArgs e)

{

button1.Enabled = true;

label2.Text = Convert.ToString(gr);

}

int p = 0;

int marker = 1;

private void Button11\_Click(object sender, EventArgs e)

{

if(menu==1)

{

++p;

switch (p)

{

case 1:

radioButton1.Checked = false;

radioButton2.Checked = true;

marker = 2;

break;

case 2:

radioButton2.Checked = false;

radioButton3.Checked = true;

marker = 3;

break;

case 3:

radioButton1.Checked = true;

radioButton3.Checked = false;

marker = 1;

p = 0;

break;

}

}

else

{

label1.Visible = false;

if (canal > 59)

canal = 0;

else

canal++;

Canal(canal);

}

}

int y = 0;

private void Button12\_Click(object sender, EventArgs e)

{

if(menu==1)

{

++y;

switch (marker)

{

case 1:

radioButton1.Checked = false;

radioButton3.Checked = true;

marker = 3;

break;

case 2:

radioButton2.Checked = false;

radioButton1.Checked = true;

marker = 1;

break;

case 3:

radioButton3.Checked = false;

radioButton2.Checked = true;

marker = 2;

break;

}

}

else

{

label1.Visible = false;

if (canal < 0)

canal = 59;

else

canal--;

Canal(canal);

}

}

private void Button18\_Click(object sender, EventArgs e)

{

if (per == 0)

{

pictureBox3.Visible = false;

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

button7.Enabled = true;

button8.Enabled = true;

button9.Enabled = true;

button10.Enabled = true;

button11.Enabled = true;

button12.Enabled = true;

button13.Enabled = true;

button14.Enabled = true;

button15.Enabled = true;

button16.Enabled = true;

button17.Enabled = true;

button20.Enabled = true;

button21.Enabled = true;

button19.Enabled = true;

button22.Enabled = true;

button2.Enabled = true;

pictureBox2.Image = Properties.Resources.imgonline\_com\_ua\_Resize\_CH6TIf9CpHEX4V;

timer1.Enabled = true;

per++;

canal = 1;

}

else

{

button3.Enabled = false;

button4.Enabled = false;

button5.Enabled = false;

button6.Enabled = false;

button7.Enabled = false;

button8.Enabled = false;

button9.Enabled = false;

button22.Enabled = false;

button10.Enabled = false;

button11.Enabled = false;

button12.Enabled = false;

button13.Enabled = false;

button14.Enabled = false;

button15.Enabled = false;

button16.Enabled = false;

button17.Enabled = false;

button20.Enabled = false;

button21.Enabled = false;

button19.Enabled = false;

button2.Enabled = false;

radioButton1.Visible = false;

radioButton2.Visible = false;

radioButton3.Visible = false;

textBox2.Visible = false;

pictureBox3.Visible = true;

per = 0;

}

}

private void Button19\_Click(object sender, EventArgs e)

{

textBox2.Visible = true;

pictureBox2.Image = Properties.Resources.tekstura\_siniy\_fon\_ten\_50468\_1440x900;

textBox2.Text = "Поиск каналов";

timer3.Interval = 5000;

timer3.Start();

timer4.Interval=5000;

timer4.Start();

}

private void Button20\_Click(object sender, EventArgs e)

{

canal++;

Canal(canal);

}

private void PictureBox1\_Click(object sender, EventArgs e)

{

}

private void Button21\_Click(object sender, EventArgs e)

{

canal--;

Canal(canal);

}

private void Timer2\_Tick(object sender, EventArgs e)

{

textBox1.Visible = false;

timer2.Stop();

}

private void Button14\_Click(object sender, EventArgs e)

{

if(yarc==1)

{

yark++;

if (yark >= 100)

yark = 100;

textBox2.Text = "Яркость: " + yark;

}

if(contr==1)

{

kontr++;

if (kontr >= 100)

kontr = 100;

textBox2.Text = "Контрастность: " + kontr;

}

if(menu==0)

{

if (volume >= 50)

volume = 50;

else

volume++;

textBox1.Text = "Громкость: " + volume;

textBox1.Visible = true;

timer2.Interval = 5000;

timer2.Start();

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void Button13\_Click(object sender, EventArgs e)

{

if (yarc == 1)

{

yark--;

if (yark <= 0)

yark = 0;

textBox2.Text = "Яркость: " + yark;

}

if(contr==1)

{

kontr--;

if (kontr <= 0)

kontr = 0;

textBox2.Text = "Контрастность: " + kontr;

}

if(menu==0)

{

if (volume <= 0)

volume = 0;

else

volume--;

textBox1.Text = "Громкость: " + volume;

textBox1.Visible = true;

timer2.Interval = 5000;

timer2.Start();

}

}

private void Button2\_Click(object sender, EventArgs e)

{

textBox2.Visible = true;

pictureBox2.Image = Properties.Resources.tekstura\_siniy\_fon\_ten\_50468\_1440x900;

textBox2.Text = "Поиск каналов";

timer3.Interval = 5000;

timer3.Start();

timer4.Interval = 5000;

timer4.Start();

}

int clickCount = 0;

private void Button22\_Click(object sender, EventArgs e)

{

++clickCount;

switch (clickCount)

{

case 1:

menu = 1;

radioButton1.Visible = true;

radioButton2.Visible = true;

radioButton3.Visible = true;

radioButton1.Checked = true;

pictureBox2.Image = Properties.Resources.tekstura\_siniy\_fon\_ten\_50468\_1440x900;

yarc = 1;

button3.Enabled = false;

button4.Enabled = false;

button5.Enabled = false;

button6.Enabled = false;

button7.Enabled = false;

button8.Enabled = false;

button9.Enabled = false;

button10.Enabled = false;

button15.Enabled = false;

button16.Enabled = false;

button20.Enabled = false;

button21.Enabled = false;

button19.Enabled = false;

break;

case 2:

menu = 0;

radioButton1.Visible = false;

radioButton2.Visible = false;

radioButton3.Visible = false;

button3.Enabled = true;

button4.Enabled = true;

button5.Enabled = true;

button6.Enabled = true;

button7.Enabled = true;

button8.Enabled = true;

button9.Enabled = true;

button10.Enabled = true;

button11.Enabled = true;

button12.Enabled = true;

button13.Enabled = true;

button14.Enabled = true;

button15.Enabled = true;

button16.Enabled = true;

button17.Enabled = true;

button20.Enabled = true;

button21.Enabled = true;

button19.Enabled = true;

button22.Enabled = true;

button2.Enabled = true;

yarc = 0;

clickCount = 0;

menu = 0;

break;

}

}

private void TextBox2\_TextChanged(object sender, EventArgs e)

{

}

private void Timer3\_Tick(object sender, EventArgs e)

{

timer2.Stop();

}

private void Timer4\_Tick(object sender, EventArgs e)

{

textBox2.Text = "Загрузка каналов";

timer4.Stop();

timer5.Interval = 5000;

timer5.Start();

}

private void Timer5\_Tick(object sender, EventArgs e)

{

textBox2.Text = "Поиск завершен";

timer5.Stop();

timer6.Interval = 2000;

timer6.Start();

}

private void Timer6\_Tick(object sender, EventArgs e)

{

textBox2.Visible = false;

timer6.Stop();

Canal(1);

}

private void ListBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void Button23\_Click(object sender, EventArgs e)

{

}

private void Button24\_Click(object sender, EventArgs e)

{

}

private void Button26\_Click(object sender, EventArgs e)

{

if (yarc == 1 && contr==0)

{

if (yark >= 100)

yark = 100;

else

yark++;

textBox2.Text = "Яркость " + yark;

textBox2.Visible = true;

timer7.Interval = 5000;

timer7.Start();

}

else

{

if (kontr >= 100)

kontr = 100;

else

kontr++;

textBox2.Text = "Контрастность " + kontr;

textBox2.Visible = true;

timer7.Interval = 5000;

timer7.Start();

}

}

private void Button27\_Click(object sender, EventArgs e)

{

if (yarc == 1 && contr == 0)

{

if (yark >= 100)

yark = 100;

else

yark--;

textBox2.Text = "Яркость " + yark;

textBox2.Visible = true;

timer7.Interval = 5000;

timer7.Start();

}

else

{

if (kontr >= 100)

kontr = 100;

else

kontr--;

textBox2.Text = "Контрастность " + kontr;

textBox2.Visible = true;

timer7.Interval = 5000;

timer7.Start();

}

}

private void Timer7\_Tick(object sender, EventArgs e)

{

timer7.Start();

Canal(1);

textBox2.Visible = false;

yarc = 0;

contr = 0;

}

private void Button25\_Click(object sender, EventArgs e)

{

}

private void Timer8\_Tick(object sender, EventArgs e)

{

try

{

canal = Convert.ToInt32(label1.Text);

if (canal < 20)

{

Canal(canal);

}

else

{

canal = 0;

Canal(canal);

}

}

catch

{

Canal(canal);

}

label1.Visible = false;

s = " ";

timer8.Stop();

}

private void RadioButton1\_CheckedChanged(object sender, EventArgs e)

{

}

private void RadioButton2\_CheckedChanged(object sender, EventArgs e)

{

}

}

}