

## Реферат

Пояснительная записка курсового проекта содержит: 29 с., 18 рис., 1 табл., 7 источников, 2 прил.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, МОДЕЛЬ, КЛАСС, СИСТЕМА, БУДИЛЬНИК, ТАЙМЕР, UML, BPMN, EPC, FURPS+, IDEF0, IDEF1X, IDEF3 DFD, ДИАГРАММЫ.

Объектом исследования является программное обеспечение способное работать в качестве будильника имеющем систему взвода и сброса.

Целью данной курсовой работы состоит в разработке программного обеспечения “Будильник”

В процессе создания курсовой работы было создано два приложения WinForm, написанных на языке программирования C# в формате исполняемых файлов (.exe). Создавались данные программы в среде разработки Microsoft Visual Studio 2017.

При выполнении курсовой работы были выполнены все этапы создания программного продукта: от постановки задачи до практической реализации. В ходе выполнения курсовой работы были приобретены навыки работы со специализированной литературой, справочниками, стандартами.

## Содержание

Введение .....	6
1 Основная часть .....	7
1.1    Формулировка задачи .....	7
1.2    Диаграмма Ганта.....	7
1.3    Сетевая диаграмма.....	8
1.4    Создание модели As-Is в стандарте IDEF0 .....	10
1.5    Описание методологии IDEF3 .....	14
1.6    Методология IDEF1X.....	15
1.7    Расчёт трудоемкости методом подсчёта функциональных точек .....	15
1.8    UML диаграммы .....	17
1.8.1    Диаграмма вариантов использования .....	17
1.8.2    Диаграмма последовательностей.....	18
1.8.3    Диаграмма состояний .....	19
1.8.4    Диаграмма компонентов.....	21
1.9    BRMN спецификация .....	22
1.10    Результат машинного тестирования программы.....	23
1.11    Системные требования.....	25
1.12    Руководство пользователя .....	25
Заключение .....	26
Список использованных источников .....	27
Приложение А .....	28
Приложение Б .....	29

## **Нормативные ссылки**

В настоящей пояснительной записке используются ссылки на следующие нормативные документы:

ГОСТ 19.701-80 – ЕСПД. Схемы алгоритмов, программ, данных и систем.

ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления».

ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления».

ГОСТ 7.80-2000. «Библиографическая запись. Заголовок. Общие требования и правила составления».

ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления».

ГОСТ 19.701-80 – ЕСПД. Схемы алгоритмов, программ, данных и систем.

## **Введение**

Microsoft Visual C# представляет собой весьма эффективный и в то же время простой язык, предназначенный преимущественно для разработчиков, создающих сборки приложений в среде Microsoft .NET Framework. Visual C# унаследовал множество лучших свойств от C++ и Microsoft Visual Basic, но при этом его разработчики постарались избавиться от различных несоответствий и анахронизмов, в результате чего появился более понятный и логичный язык, который удобен в изучении как опытным программистам, работавшим на любом другом языке программирования высокого уровня, так и новичкам в этом деле. Именно поэтому C# позволяет студентам с легкостью начать изучать азы программирования.

Целью исследования, проводимого в рамках настоящей курсовой работы, является разработка и реализация на языках высокого уровня алгоритмов решения задач, представленных в задании курсовой работы.

Объектами исследования настоящей курсовой работы являются методы и технологии разработки программных продуктов.

Предметами исследования настоящей курсовой работы являются методы, алгоритмы и приёмы разработки программ обработки файлов и строк.

Информационной базой исследования является учебная литература по информатике и программированию, техническая документация по языку C# инструментальной среды MS Visual Studio 2017.

## **1 Основная часть**

### **1.1 Формулировка задачи**

Программное обеспечение встроенного микропроцессора для будильника. Будильник постоянно отображает текущее время (часы, минуты, например: 12: 00). Управление будильником осуществляется следующими кнопками:

- кнопкой режима установки времени, –кнопкой режима установки времени срабатывания.

- двумя отдельными кнопками для установки часов и минут, –кнопкой сброса сигнала «СБРОС». На будильнике имеется переключатель режима работы со следующими положениями: «ВЫКЛ», «ВКЛ» и «ТАЙМЕР».

Для установки текущего времени нужно нажать на кнопку режима установки и, при нажатой кнопке, нажимать на кнопки установки часов и минут. При каждом нажатии на кнопки, устанавливаемое значение увеличивается на одну единицу (один час или одну минуту соответственно). При достижении максимального значения производится сброс. Для установки времени срабатывания будильника нужно нажать на кнопку режима установки времени срабатывания и, держа кнопку нажатой, нажимать на кнопки установки часов и минут. Когда переключатель режима работы находится в положении «ВКЛ», при достижении времени срабатывания происходит подача звукового сигнала в течение одной минуты . Сигнал можно прервать, нажав на кнопку «СБРОС». При этом сигнал должен быть возобновлен через пять минут. При установке переключателя в положение «ВЫКЛ» звуковой сигнал не подается.

При переводе переключателя в положение «ТАЙМЕР» включается радиоприемник на тридцать минут, а затем часы переходят в состояние будильника (аналогично положению «ВКЛ»). При нажатии на кнопку режима установки времени, будильник должен отображать время срабатывания.

### **1.2 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.»

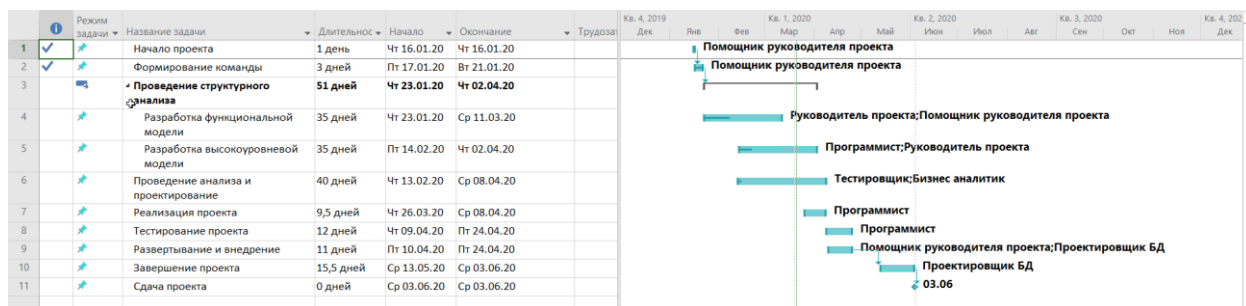


Рисунок 1 – Диаграмма Ганта для проекта «Будильник»

### 1.3 Сетевая диаграмма

Сетевая диаграмма отображает зависимости между различными этапами проекта. Основной целью её использования является эффективное планирование и управление работами и ресурсами 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.

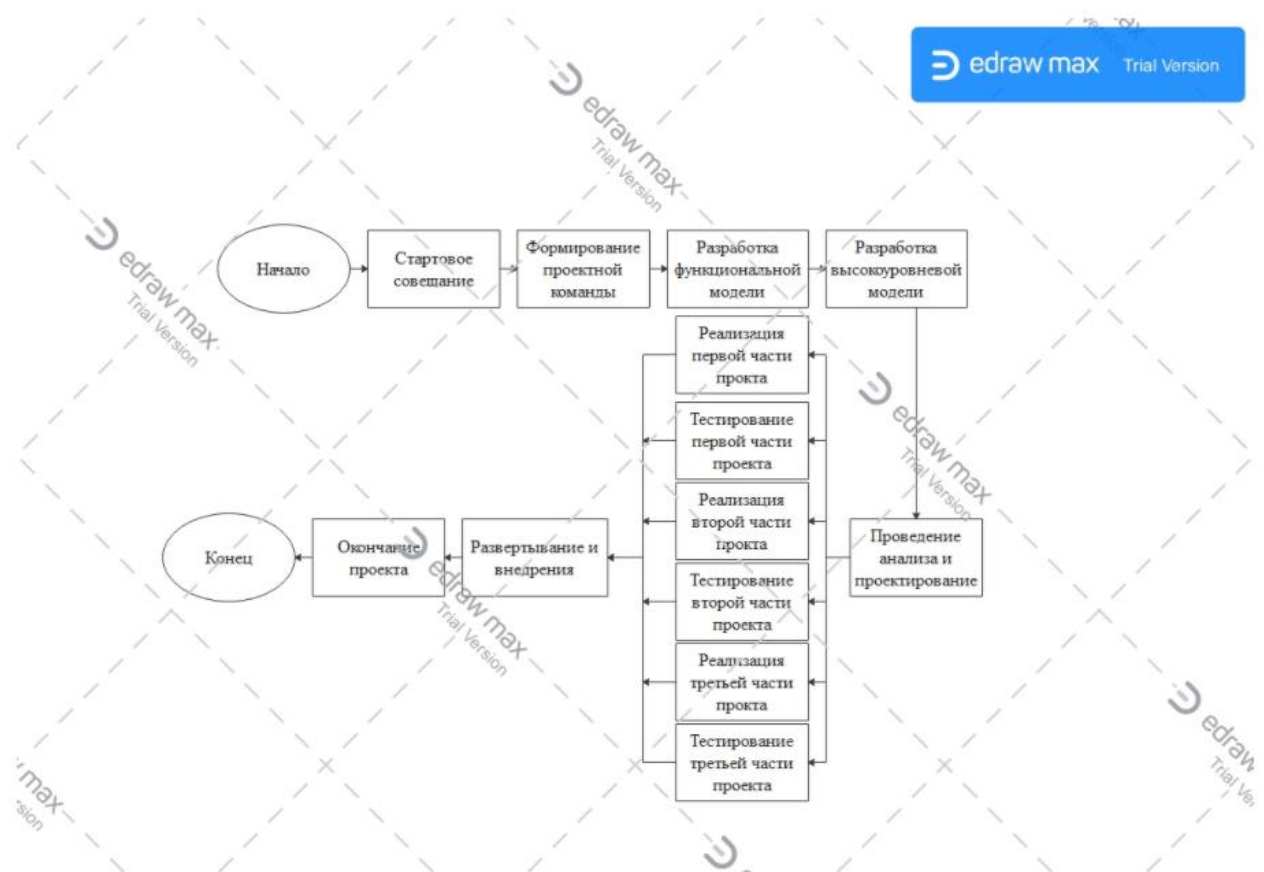


Рисунок 2 – Сетевая диаграмма

## 1.4 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. 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 1 была составлена модель As-Is проекта «Банкомат». Полученная модель может быть представлена в более подробном виде путём разбиения на большее количество составных элементов.



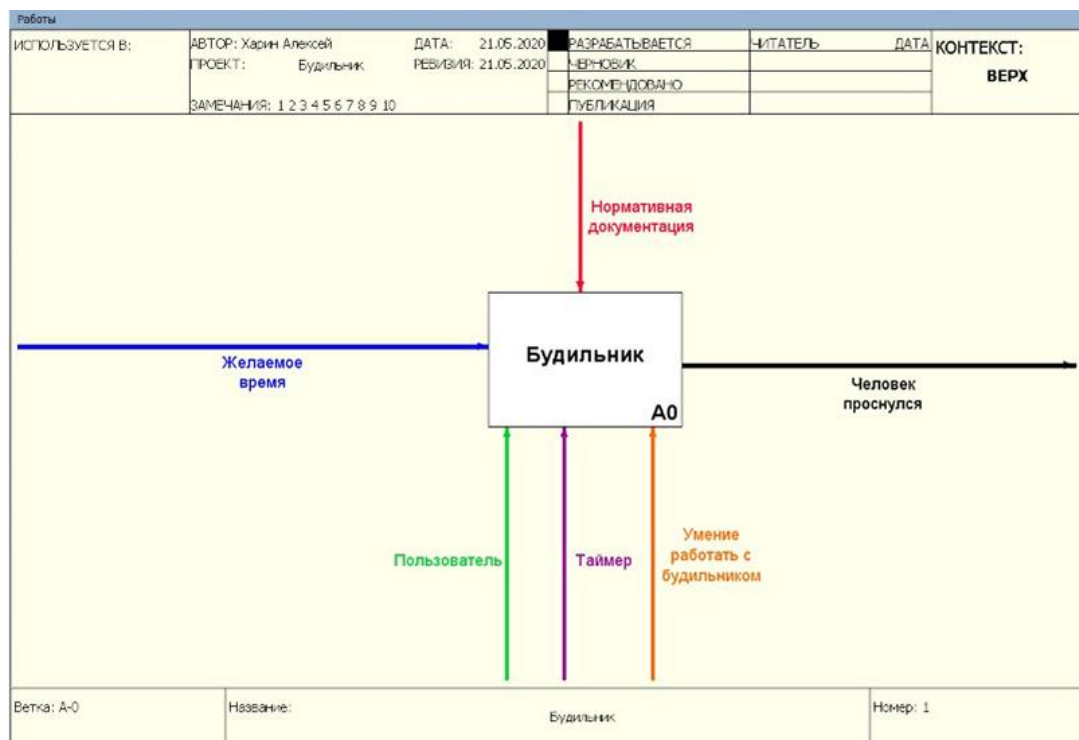


Рисунок 3 – Модель As-Is проекта «Будильник»

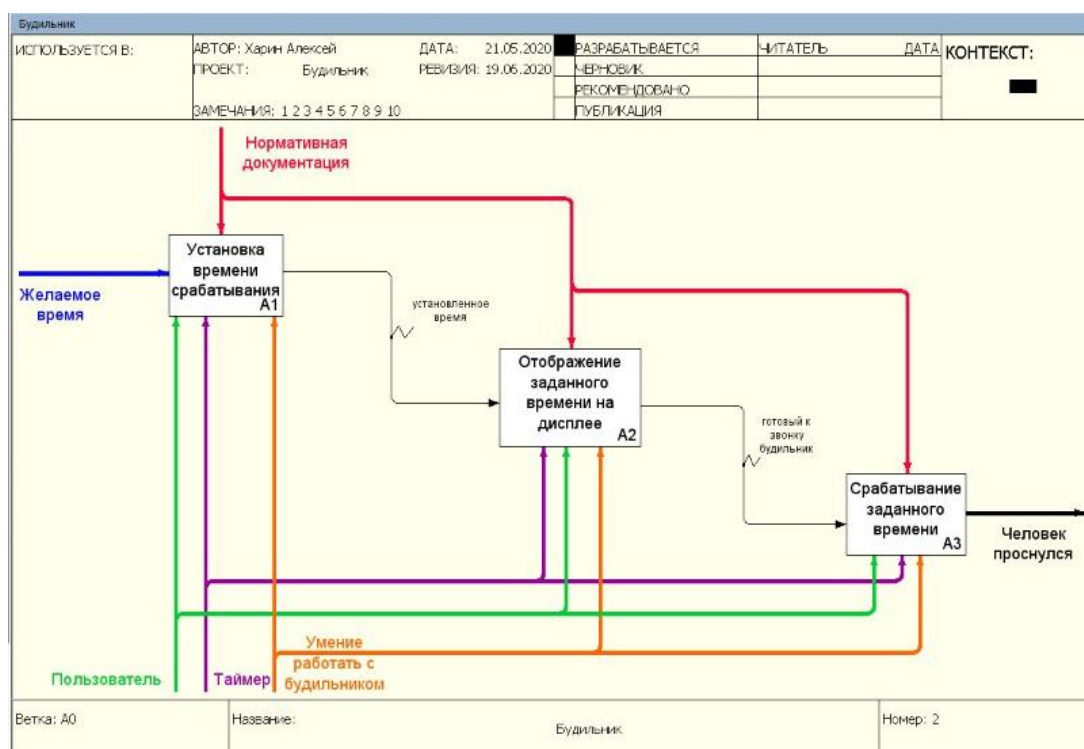


Рисунок 4 – Декомпозиция проекта «Будильник»

Ниже представлены декомпозиции составляющих элементов разработки конечного продукта.

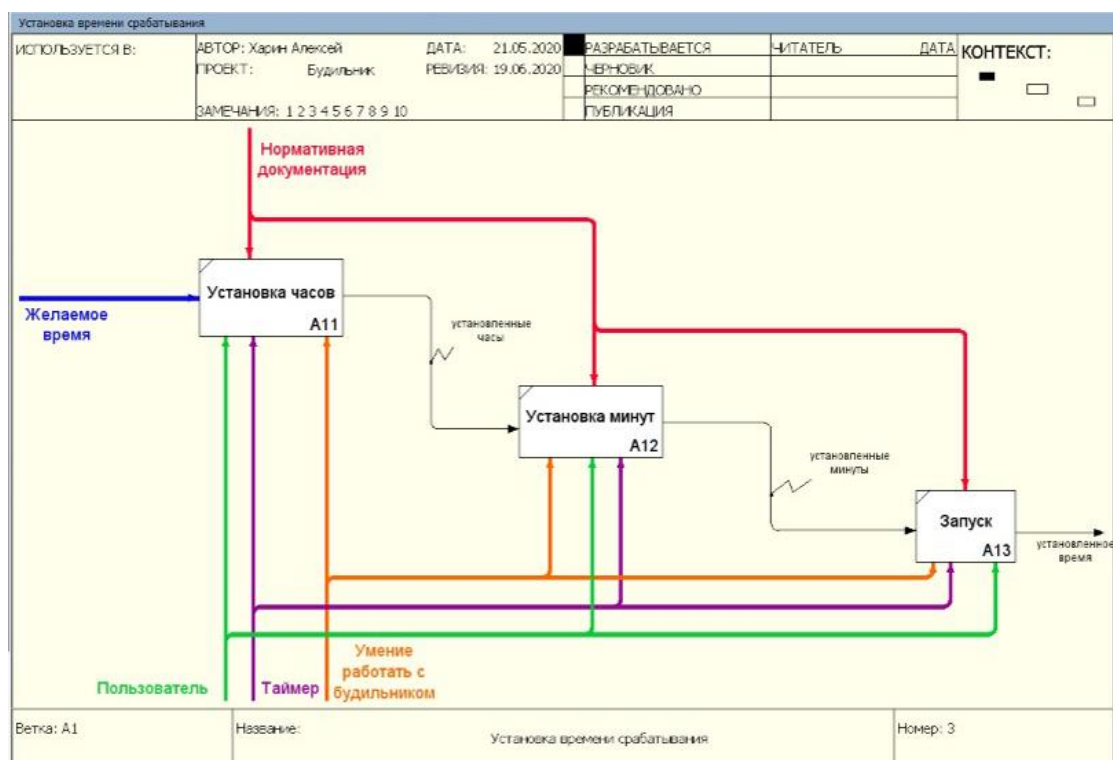


Рисунок 5 – Установка времени срабатывания

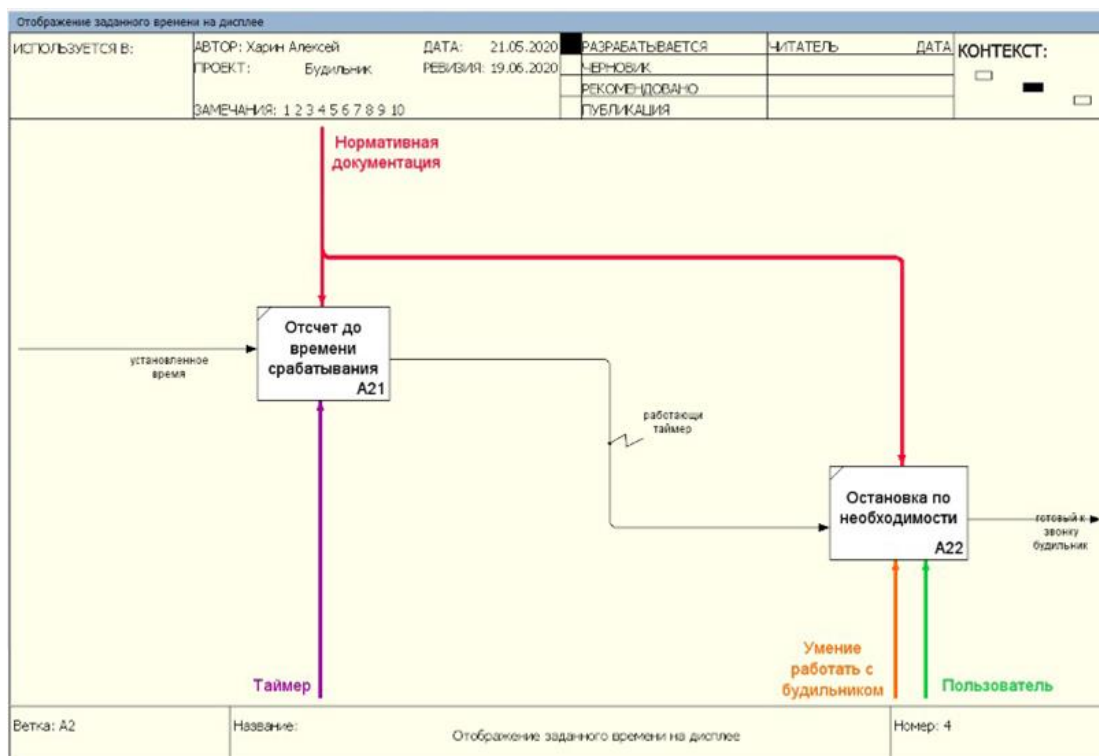


Рисунок 6 – Отображение заданного времени на дисплее

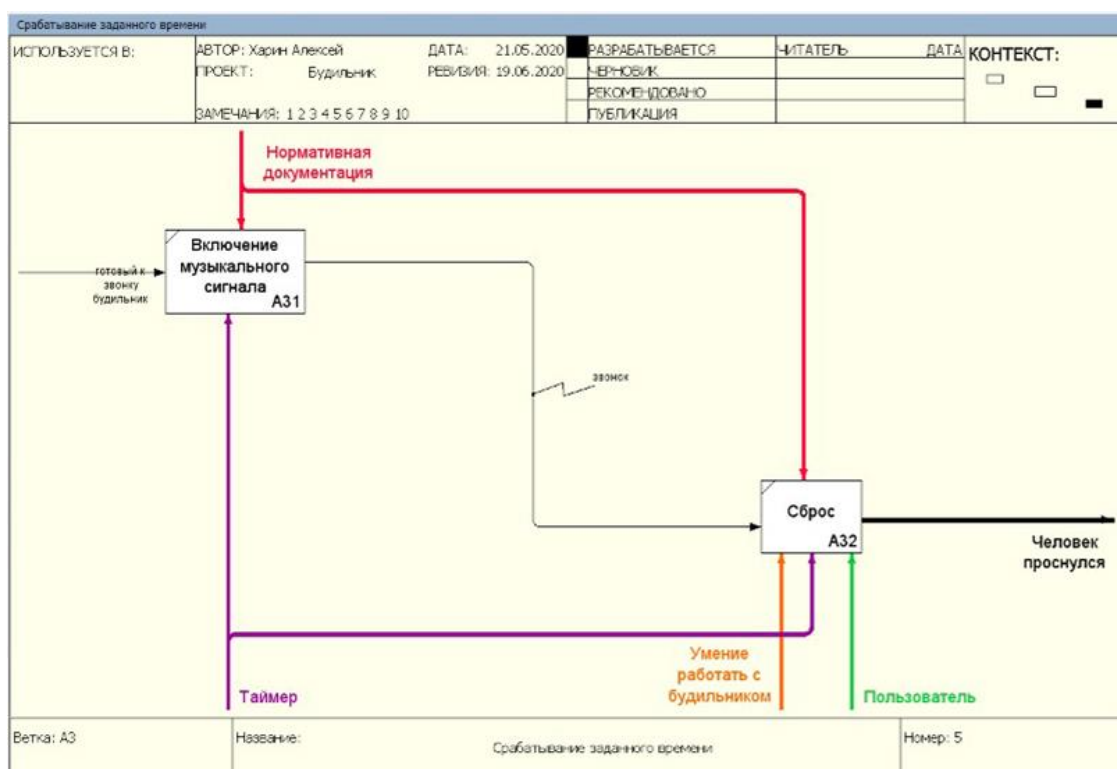


Рисунок 7 – Срабатывание заданного времени

## 1.5 Описание методологии 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 окончания 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.

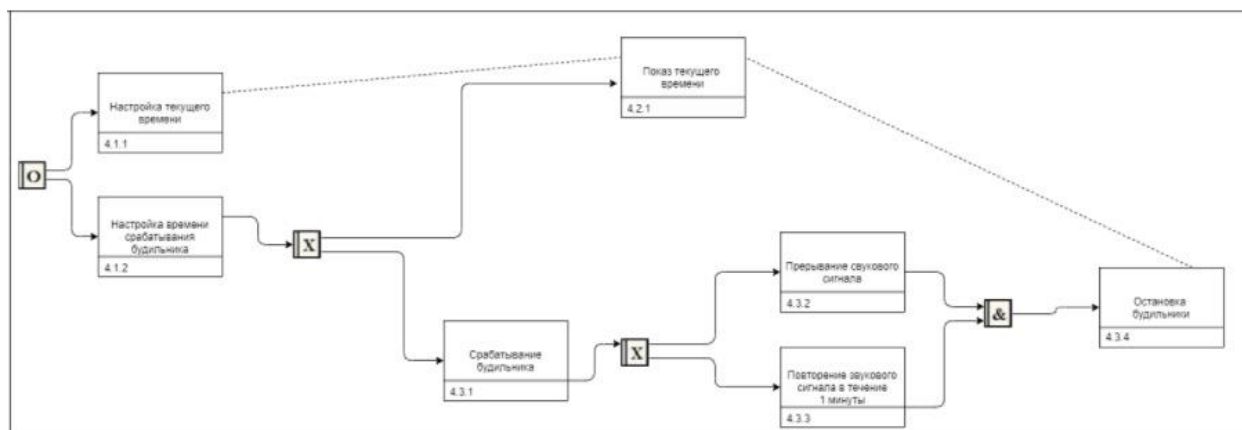


Рисунок 8 – Диаграмма IDEF3

## 1.6 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.

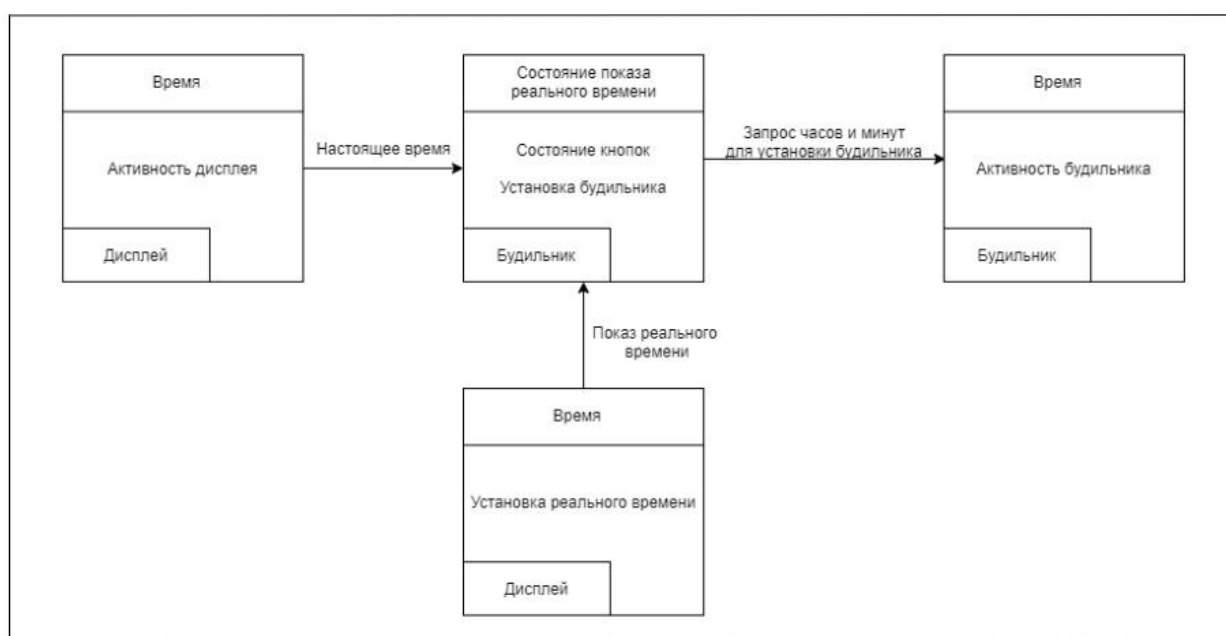


Рисунок 9 – Диаграмма IDEF1X

## 1.7 Расчёт трудоемкости методом подсчёта функциональных точек

Одним из важнейших пунктов планирования выполнения любого программного проекта является расчёт трудоёмкости исполнителей при разработке программного обеспечения. Модели и методы оценки

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

Подсчёт функциональных точек – стандартный метод измерения размера программного продукта с точки зрения пользователей системы.

Метод предназначен для оценки на основе логической модели объёма программного продукта, количеством функционала, требуемого заказчиком и предоставляемого разработчиком.

Общее количество функциональных точек (далее FP) рассчитывается по формуле  $FP = X * (0,65 + 0,01 * \sum_{i=1}^{14} Fi)$  где:

$X$  – это суммарное количество функциональных точек для каждого бизнес-процесса;

$Fi$  – это коэффициент регулировки сложности проекта;

Уточним количество функциональных точек:

Рассчитаем коэффициент регулировки сложности  $W = 0.65 + (0.01 * 30) = 0.95$

Подставим полученный коэффициент в формулу расчета функциональных точек  $R(F) 210 * 0.95 = 199.5$

Рассчитаем зарплату для каждого участника разработки проекта «Интернет магазин»:

- Кураторы проекта = 12000 RUB
- Руководители проекта = 20000 RUB
- Системный аналитик = 8000 RUB
- Системный архитектор = 8000 RUB
- Помощник руководитель = 7000 RUB
- Тестировщик 8000 RUB
- Программист 12000 RUB

## **1.8 UML диаграммы**

UML (англ. Unified Modeling Language — «унифицированный язык моделирования») — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.»

В UML используются следующие виды диаграмм:

### **1.8.1 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.

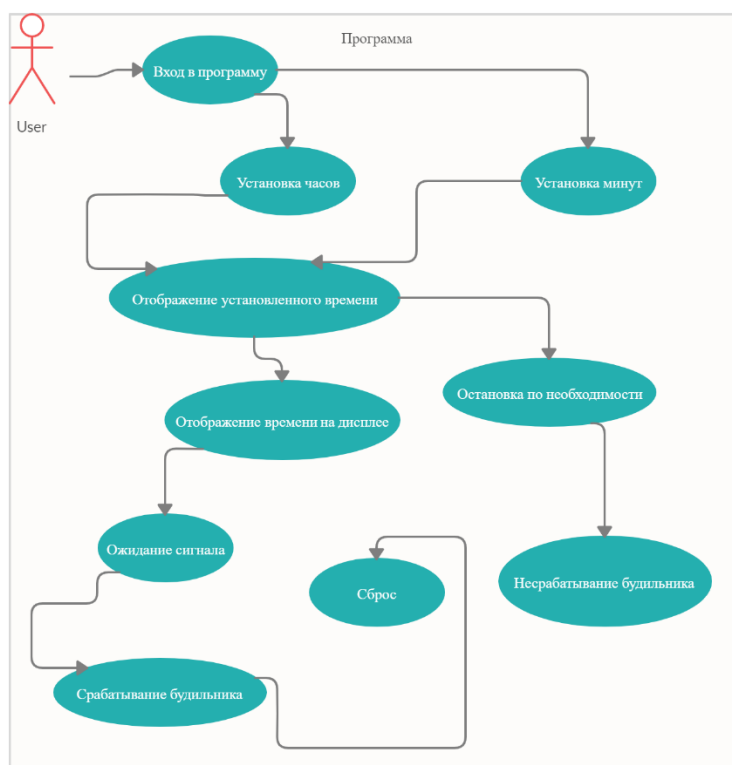


Рисунок 10 – Диаграмма вариантов использования проекта «Будильник»

### 1.8.2 Диаграмма последовательностей

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).



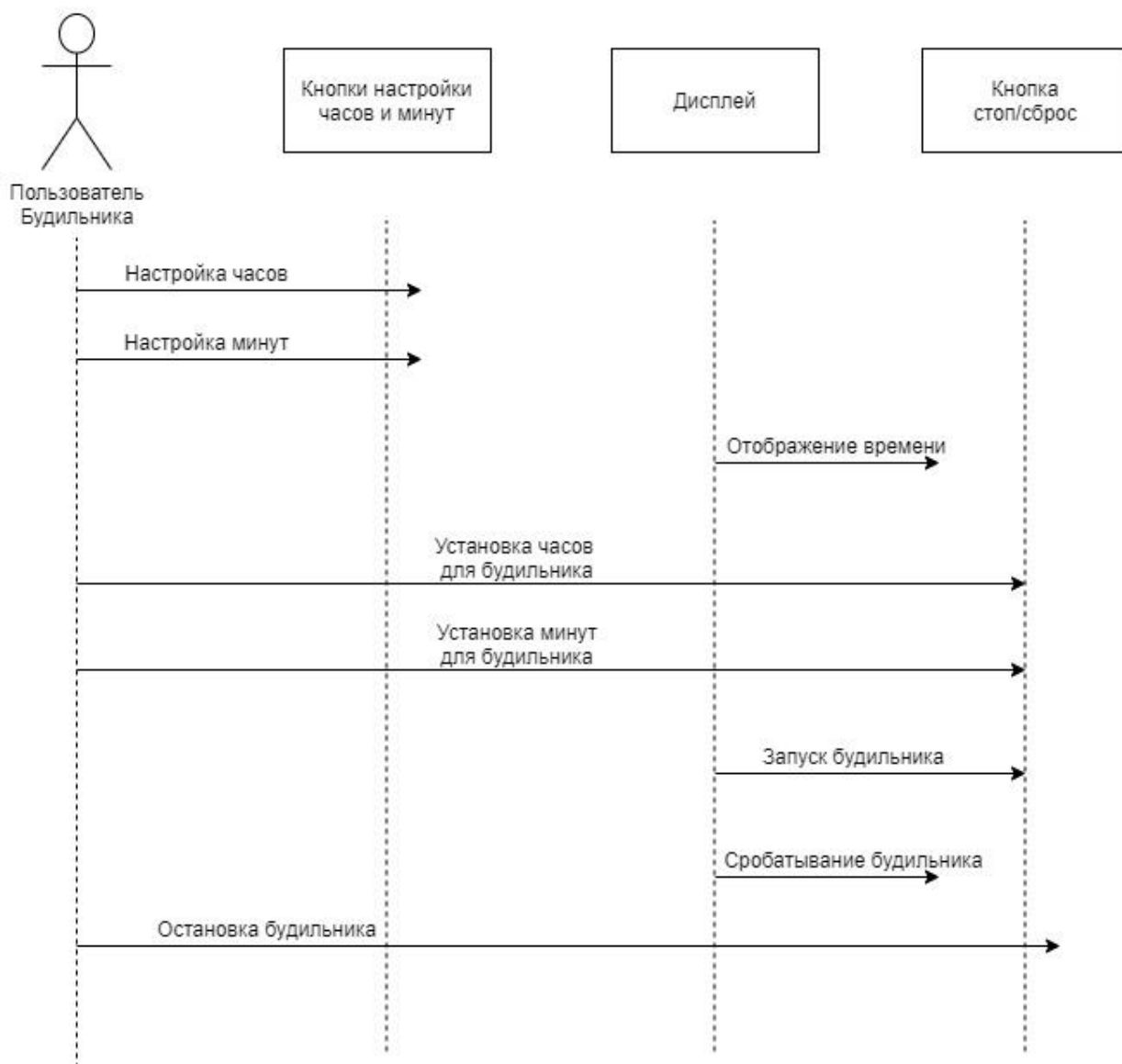


Рисунок 11 – Диаграмма последовательностей проекта «Будильник»

### 1.8.3 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 или их может быть сколько угодно, но должно быть одно и только одно начальное состояние.

Диаграмма состояний состоит из:

1. Деятельность – поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность изображают внутри самого состояния, ей должно предшествовать слово `do` (делать) и двоеточие.
2. Входное действие – поведение, которое выполняет объект при переходе в данное состояние. Входное действие также показывают внутри состояния, ему предшествует слово `entry` (вход) и двоеточие.
3. Выходное действие – поведение, которое выполняет объект при выходе из данного состояния. Выходное действие изображают внутри состояния, ему предшествует слово `exit` (выход) и двоеточие.
4. Событие – то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода.
5. Ограждающие условия определяют, когда переход может осуществиться, а когда нет. Ограждающие условия задавать необязательно. Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.
6. Действие – часть перехода. Рисуют вдоль линии перехода после имени события, ему предшествует косая черта. Действие рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

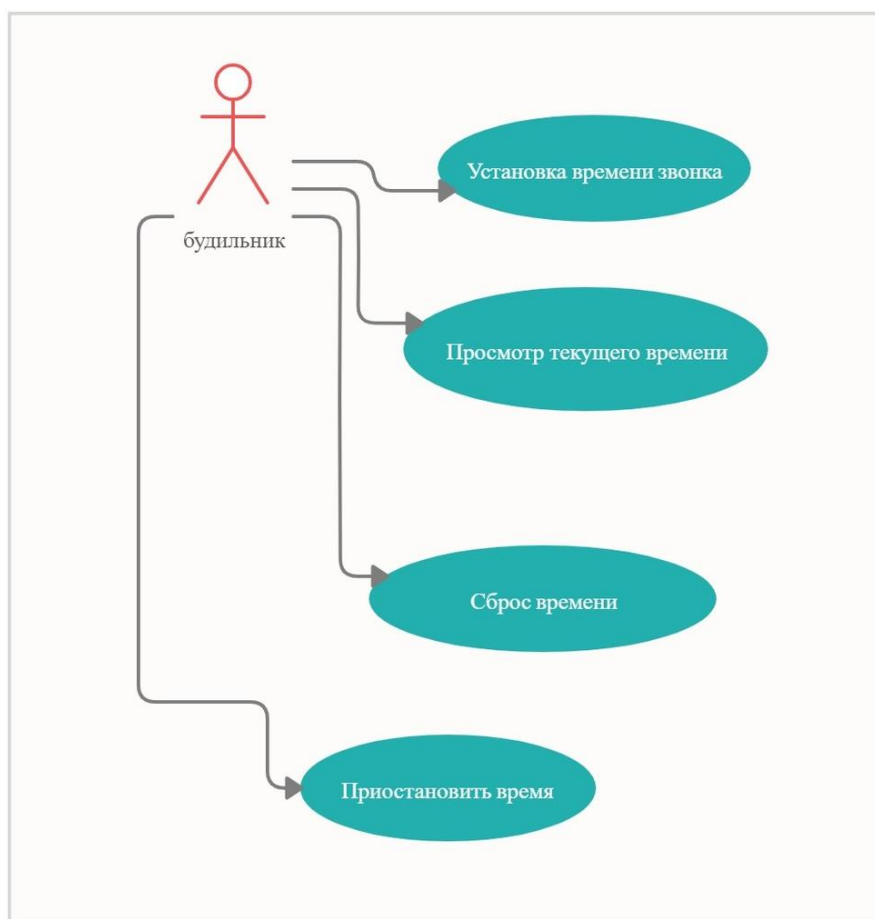


Рисунок 12 – Диаграмма состояний для проекта «Будильник»

#### 1.8.4 Диаграмма компонентов

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

- исполняемые компоненты;
- библиотеки кода.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы. Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Из нее видно, в каком порядке

надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. На такой диаграмме показано соответствие классов реализованным компонентам. Она нужна там, где начинается генерация кода.

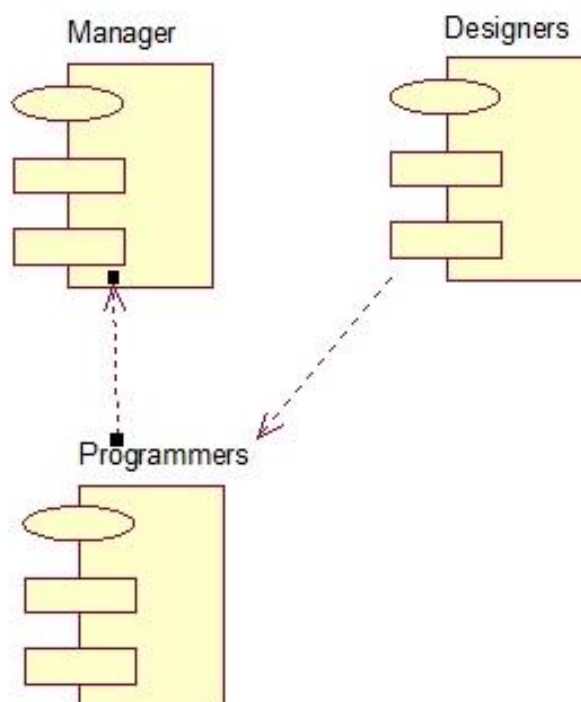


Рисунок 13 – Диаграмма компонентов проекта «Будильник»

## 1.9 BPMN спецификация

Спецификация BPMN описывает условные метки и их описание в XML для отображения бизнес-процессов в виде диаграмм бизнес-процессов. BPMN ориентируется как на технических специалистов, так и на корпоративных пользователей. Язык использует базовый набор интуитивных функций, которые позволяют определять сложные семантические предложения. Кроме того, спецификация BPMN определяет, как диаграммы, описывающие бизнес-процесс, могут быть преобразованы в исполняемые модели.

Основной задачей BPMN является создание стандартного набора условных меток, понятных всем бизнес-пользователям. Бизнес-пользователи включают в себя бизнес-аналитиков, которые создают и совершенствуют процессы, технических разработчиков, ответственных за реализацию процессов, и менеджеров, которые следят за процессами и управляются ими.

Таким образом, BPMN призван служить связующим звеном между этапом проектирования бизнес-процесса и этапом внедрения.

На рис. 10 представлена схема работы системы поддержки BPMN будильника.

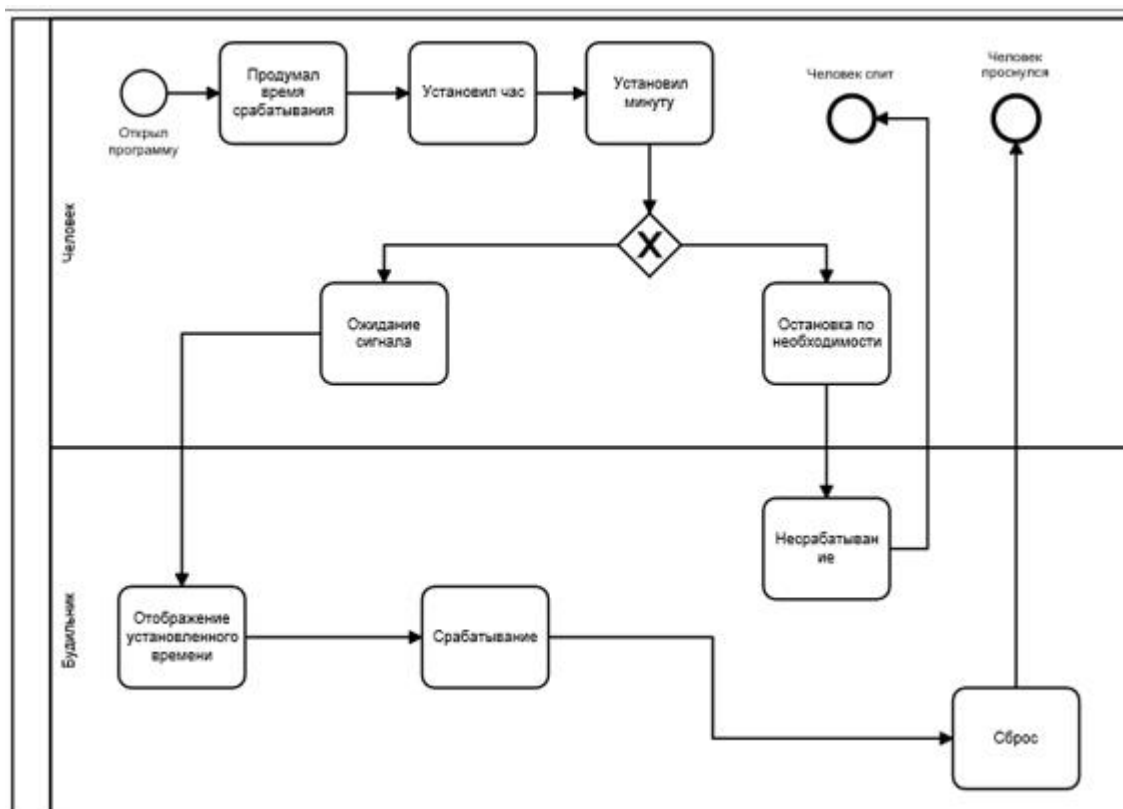


Рисунок 14 – Диаграммы BPMN «As-Is» и «To be»

### 1.10 Результат машинного тестирования программы

На рисунке 15 представлен вид системы сразу после запуска

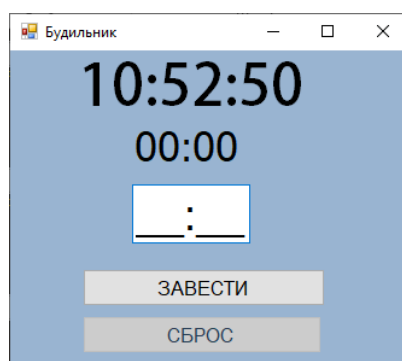


Рисунок 15 – Изначальный экран запуска

На рисунке 16 представлен интерфейс системы, где отображено реальное время и поле, где нужно указать время срабатывания будильника.

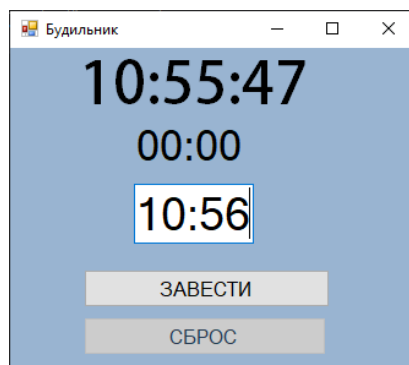


Рисунок 16 – Заведение будильника

На рисунках 17 и 18 представлено работа и момент срабатывания будильника.

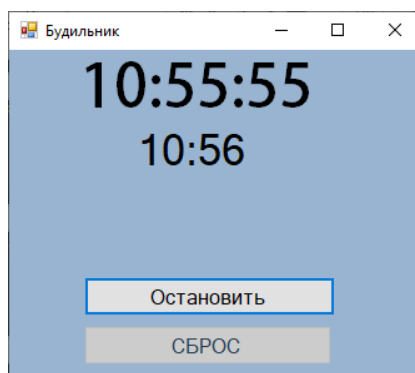


Рисунок 17 – Работа будильника

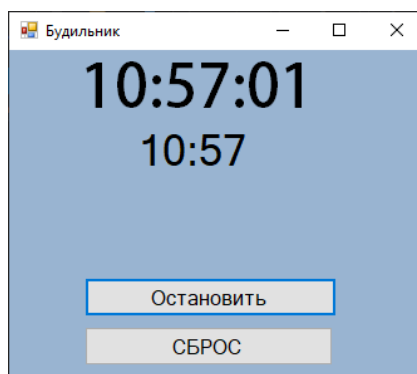


Рисунок 18 – Момент срабатывания

### 1.11 Системные требования

Таблица 1 – Системные требования программ проекта «Будильник»

Процессор	2.5 ГГц
Оперативная память	150 Мб
Монитор	1920 x 1080
Свободное место на носителе	15 Мб
Устройства взаимодействия с пользователем	Клавиатура и мышь
Программное обеспечение	Visual Studio 2019 года последней версии

### 1.12 Руководство пользователя

Запуск программы осуществляется открытием исполняемого файла .exe.

Теперь перед нами открылось стартовое и основное меню программы.

Первым делом вам в пустое поле необходимо ввести время, в которое Будильник должен прозвенеть. Далее вам нужно будет подождать этого времени и отключить сам Будильник.

Имеется возможность остановить будильник до момента его срабатывания.

## **Заключение**

В результате выполнения данного курсового проекта была спроектирована программа обеспечения будильника на языке высокого уровня C#, позволяющая наглядно продемонстрировать работу всех её компонентов. Полученные диаграммы позволяют детально изучить не только процесс машинного выполнения программы, но также и оценить процесс создания (проектирования и реализации) данного проекта.

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.

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.

При выполнении каждого отдельного пункта проектирования системы были получены важные знания и практические навыки работы в области разработки комплексных программных систем, были закреплены навыки работы с языком высокого уровня C#.




## Список использованных источников

1. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - Москва: Гостехиздат, 2017. - 736 с.
2. Роберт А. Максимчук. UML для простых смертных / Роберт А. Максимчук, Эрик Дж. Нейбург. - Москва: СИНТЕГ, 2014. - 272 с.
3. Йордон, Эдвард. Объектно-ориентированный анализ и проектирование систем / Эдвард Йордон, Карл Аргила. - М.: ЛОРИ, 2014. - 264 с.
4. Попова О.Б. Теория разработки программного обеспечения. Методические указания к выполнению Л/Р
5. Попова О.Б. Теория разработки программного обеспечения. Конспекты лекций.
6. GitHub – MeysamResan/Online\_Store. [Электронный ресурс] [https://github.com/MeysamResan/Online\\_Store](https://github.com/MeysamResan/Online_Store) (Дата обращения 15.12.2020).
7. Антиплагиат [Электронный ресурс] <https://www.antiplagiat.ru/> (Дата обращения 15.12.2020).

# Приложение А

## Проверка на Антиплагиат




**АНТИПЛАГИАТ**

ТВОРИТЕ СОБСТВЕННЫМ УМОМ

sk


оценивает


главная / кабинет / результаты проверки



ПОЛЬЗОВАТЕЛЬ


eldarklyuev@mail.com






БАЛЛОВ

0




ТАРИФ

Бесплатный доступ (0/0)



МОДУЛИ И КОЛЛЕКЦИИ

Подключено: 1 смотреть



МЕНЮ

ru

Оригинальность

61.75%

Займствованная

38.25%

Цитирования

0%

Самоцитирования

0%

Полный отчет

Краткий отчет

История отчетов

РАСПЕЧАТАТЬ

ВЫГРУЗИТЬ

СОЗДАТЬ ССЫЛКУ

Свойства документа

Имя исходного файла

Курсовой\_проект\_Клюев Э.С..pdf

Параметры проверки

Авторы документа

Не указано

Текстовые метрики

Название документа

Курсовой\_проект\_Клюев Э.С.

Статистика по документу

Тип документа

Не указано

РЕДАКТИРОВАТЬ СВОЙСТВА

28

## Приложение Б

Листинг класса Form.cs:

```
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;

using System.Media;

namespace Alarm_5

{

    public partial class Form1 : Form // добавление коммента

    {

        public Form1()

        {

            Timer timer1 = new Timer();

            InitializeComponent();

        }

        SoundPlayer sp = new SoundPlayer("D:\\\\Dram.wav");

        bool b = false;
```

```

        private void maskedTextBox1_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

            label1.Text = DateTime.Now.Hour.ToString("00") + ":" +
DateTime.Now.Minute.ToString("00") + ":" + DateTime.Now.Second.ToString("00");
        }

        private void timer1_Tick(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            button2.Enabled = false;

            timer1.Interval = 1000;

            timer1.Tick += new EventHandler(timer1_Tick);

            timer1.Start();
        }

```

```
private void timer1_Tick_1(object sender, EventArgs e)
{
    label1.Text = DateTime.Now.Hour.ToString("00") + ":" +
DateTime.Now.Minute.ToString("00") + ":" + DateTime.Now.Second.ToString("00");
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    label2.Text = maskedTextBox1.Text;

    timer2.Start();

    maskedTextBox1.Visible = false;

    button1.Text = "Остановить";

    b = true;
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    sp.Stop();

    button2.Enabled = false;

    maskedTextBox1.Visible = true;

    button1.Text = "Завести";

    b = false;
}
```

```
private void timer2_Tick(object sender, EventArgs e)
{
```

```

        if (label1.Text == label2.Text + ":00")
        {
            button2.Enabled = true;

            sp.Play();
        }
    }

    private void label2_Click(object sender, EventArgs e)
    {
        if (b == true)
        {
            label2.Text = "00:00";

            timer2.Stop();

            maskedTextBox1.Visible = true;

            button1.Text = "Завести";

            b = false;
        }
    }
}

```