

2019

ЗАДАНИЕ
на курсовой проект по дисциплине
«Программная инженерия»

- 1 Тема проекта:«Система моделирования работы автозаправочной станции»
- 2 Исходные данные к проекту: см. приложение к заданию
- 3 Перечень вопросов, подлежащих разработке:
 - 3.1 Произвести анализ предметной области: изучить основные принципы построения автозаправочных станций (АЗС), изучить стандарты для организации АЗС
 - 3.2 Выполнить обзор существующих систем-аналогов
 - 3.3 Разработать информационно-логический проект системы по методологии UML
 - 3.4 Разработать и реализовать программное и информационное обеспечение, провести его тестирование и отладку.
 - 3.5 Оформить документацию курсового проекта
 - 3.6 Подготовить презентацию по разработанной системе
- 4 Перечень графических разработок:
 - 4.1 Структурная схема системы
 - 4.2 Канонические диаграммыUML
 - 4.3 Схемы основных алгоритмов

5 Календарный план выполнения работ

№ п/ п	Содержание работы по этапам	Объем этапа в % к общему объему проекта	Срок окончания	Фактическое выполнение
1	Оформление технического задания и его утверждение	5	16.09.2019	
2	Описание и анализ предметной области	10	24.09.2019	
3	Проектирование системы	30	19.11.2019	
3.1	Разработка структурной схемы системы	5	30.09.2019	
3.2	Разработка функциональной спецификации системы и прототипа интерфейса пользователя	10	14.10.2019	
3.3	Разработка информационно-логического проекта системы и его предъявление руководителю	15	19.11.2019	
4	Реализация проекта, разработка контрольных примеров. Предъявление реализации руководителю	45	09.12.2019	
5	Корректировка проекта и оформление документации проекта. Защита проекта с представлением презентации.	10	23.12.2019	

ПРИЛОЖЕНИЕ
к заданию на курсовой проект

Тема проекта: «Система моделирования работы автозаправочной станции»

Исходные данные к проекту:

1 Характеристика объекта автоматизации:

- 1) объект автоматизации: АЗС;
- 2) виды автоматизируемой деятельности:
 - процесс составления топологии АЗС;
 - процесс настройки параметров моделирования;
 - процесс моделирования работы АЗС;
 - процесс работы с базой данных (БД);
 - процесс визуализации исполняемой модели АЗС;
- 3) минимальное количество клеток АЗС по горизонтали – 10;
- 4) минимальное количество клеток АЗС по вертикали – 7;
- 5) максимальное количество клеток АЗС по горизонтали – 35;
- 6) максимальное количество клеток АЗС по вертикали – 25;
- 7) количество шаблонов топологии – 6;
- 8) размер служебной области АЗС фиксирован и составляет 25% от общей площади АЗС;
- 9) минимальное количество топливных баков – 1;
- 10) максимальное количество топливных баков – не более размера служебной области;
- 11) минимальный объем топливных баков – 10000л;
- 12) максимальный объем топливных баков – 75000 л;

- 13) количество касс на АЗС –1;
- 14) лимит кассы –100000 руб;
- 15) критическая отметка в топливном баке для дозаправки –15%;
- 16) минимальное количество видов топлива –1;
- 17) максимальное количество видов топлива –12;
- 18) количество въездов –1;
- 19) количество выездов –1;
- 20) количество положений прилегающей части дороги к АЗС – 1;
- 21) минимальное количество топливо раздаточных колонок(ТРК)–1;
- 22) максимальное количество ТРК –6;
- 23) минимальная скорость подачи топлива в ТРК –25 л/мин;
- 24) максимальная скорость подачи топлива в ТРК –160л/мин;
- 25) количество типов транспортного потока автомобилей – 2;
- 26) количество законов распределения времени появления автомобилей
(для случайного потока) – 3;
- 27) минимальное время между появлением автомобилей – 2с;
- 28) максимальное время между появлением автомобилей – 10 с;
- 29) максимальное значение интенсивности для экспоненциального закона распределения – 5;
- 30) максимальное значение отрезков времени для равномерного закона распределения – 10 с;
- 31) максимальное значение математического ожидания для нормального закона распределения – 5 с;
- 32) максимальное значение дисперсии для нормального закона распределения – 2 с;
- 33) минимальная цена за литр топлива – 10 руб;
- 34) максимальная цена за литр топлива – 100 руб;
- 35) минимальный объем бака легкового автомобиля – 50 л;
- 36) максимальный объем бака легкового автомобиля – 100 л;
- 37) минимальный объем бака грузового автомобиля – 125 л;

- 38) максимальный объем бака грузового автомобиля – 600 л;
- 39) количество видов транспорта – 2.

2 Требования к информационному обеспечению:

- 1) информационное обеспечение разрабатывается на основе следующих источников:
 - описание АЗС [Электронный ресурс]. URL: en.wikipedia.org/wiki/Filling_station (дата обращения: 13.09.2019);
 - классификация и общая характеристика автозаправочной станции [Электронный ресурс]. URL: www.personalazs.ru/documentation/safety/characteristic_azs (дата обращения: 15.09.2019);
 - структура построения автозаправочной станции [Электронный ресурс]. URL: otherreferats.allbest.ru/manufacure/00675450_0.html (дата обращения: 15.09.2019);
- 2) шаблоны автозаправочных станций хранятся в файлах, структура файла определяется в процессе проектирования;
- 3) структура базы данных разрабатывается на основании следующих сведений:
 - о резервуарах (объем, тип топлива);
 - о ТРК (скорость подачи топлива, виды топлива);
 - о видах топлива (название, цена);
 - о автомобилях (название, объем бака, вид топлива, класс);
- 4) должна быть обеспечена целостность базы данных и защита от несанкционированного доступа.

3 Требования к техническому обеспечению:

- 1) тип ЭВМ –IBMPC совместимый;
- 2) монитор с разрешающей способностью не ниже 800 x 600;
- 3) манипулятор – мышь;
- 4) технические характеристики определяются в процессе выполнения проекта.

4 Требования к программному обеспечению:

- 1) тип операционной системы – Windows 8 и выше;
- 2) язык программирования – C#;
- 3) среда программирования – Visual Studio 2019;
- 4) СУБД – MS SQL Server2017;
- 5) среда проектирования – StarUML 3.1.0.

5 Общие требования к проектируемой системе:

5.1 Функции, реализуемые системой:

- 1) конструирование топологии АЗС:
 - задание размеров АЗС;
 - построение и исполнение модели АЗС по заданному шаблону;
 - выдача информации о свойствах элементов исполняемой модели;
- 2) моделирование работы АЗС:
 - 1 конфигурация транспортного потока;
 - 2 расчет маршрута;
 - 3 проверка корректности топологии АЗС;
- 3) настройки параметров топологии:
 - 1 выбор шаблона объекта;
 - 2 добавление объекта;
 - 3 удаление объекта;
- 4) визуализация конструирования топологии АЗС;
- 5) визуализация процессов работы исполняемой модели;
- 6) процесс работы с БД:
 - 1 добавление записи;
 - 2 удаление записи;
 - 3 редактирование записи;
- 7) сохранение топологии АЗС в файл;
- 8) загрузка топологии АЗС из файла;
- 9) визуализация процессов работы АЗС;
- 10) выдача справочной информации о системе.

5.2 Технические требования к системе:

- 1) система должна удовлетворять санитарным правилам и нормам СанПин 2.2.2./2.4.2198-07;
- 2) условия работы средств вычислительной техники (содержание вредных веществ, пыли и подвижность воздуха) должны соответствовать ГОСТ 12.1.005, 12.01.007;
- 3) температура окружающего воздуха – 15-35°C;
- 4) влажность воздуха – 45-75%.

РЕФЕРАТ

Пояснительная записка 162 с, 83 рисунков, 5 таблиц, 52 источника, 2 приложения.

Графическая часть: 28 слайдов презентации PowerPoint.

ВОЛНОВОЙ АЛГОРИТМ, МОДЕЛИРОВАНИЕ АВТОЗАПРАВОЧНОЙ СТАНЦИИ, ГЕНЕРАЦИЯ АВТОМОБИЛЕЙ, СЛУЧАЙНЫЙ ПОТОК, ДЕТЕРМИНИРОВАННЫЙ ПОТОК, ЗАКОНЫ РАСПРЕДЕЛЕНИЯ, КОНСТРУКТОР ТОПОЛОГИИ, НАСТРОЙКА ТОПОЛОГИИ.

Во время курсового проектирования разработаны алгоритмы и соответствующая им программа, позволяющая выполнять создание топологии автозаправочной станции, её настройку и моделирование её работы. Файл с топологией хранится в файле и может редактироваться внутри программы. Настройка топологии и процесса моделирования выполняется пользователем. В системе имеется возможность сохранения топологии в файл с целью последующей работы с топологией.

Программа написана на языке C# в среде Visual Studio 2019 и функционирует под управлением операционной системы Windows 8/10.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	13
1.1 Описание структуры АЗС	15
1.1.1 Классификация АЗС	16
1.1.2 Топливораздаточные колонки	18
1.1.3 Топливные баки.....	20
1.1.4 Виды топлива.....	21
1.1.5 Заправляемый транспорт.....	22
1.2 Моделирование транспортных потоков	24
1.2.1 Методы моделирования случайных величин.....	25
1.2.2 Равномерный закон распределения.....	28
1.2.3 Нормальный закон распределения	29
1.2.4 Показательный закон распределения.....	30
1.3 Описание систем-аналогов.....	30
1.3.1 Имитационное моделирование работы АЗС в среде AnyLogic .	31
1.3.2 Моделирование систем в программе VisSim	33
1.4 Диаграмма объектов предметной области	36
1.5 Постановка задачи	37
2 Проектирование системы	42
2.1 Выбор архитектуры системы	42
2.2 Структурная схема системы	45
2.3 Разработка спецификации требований	46
2.3.1 Функциональная спецификация	47
2.3.2 Перечень исключительных ситуаций	49
2.4 Разработка прототипа интерфейса пользователя системы	49
2.5 Разработка информационно-логического проекта системы.....	66
2.5.1 Язык UML	66
2.5.2 Диаграмма вариантов использования	66
2.5.3 Сценарии	68
2.5.4 Диаграмма классов.....	78

2.5.5	Диаграмма состояний	78
2.5.6	Диаграмма деятельности	84
2.5.7	Диаграмма последовательности	89
2.6	Разработка и описание волнового алгоритма для определения кратчайшего маршрута на плоскости	91
2.6.1	Постановка задачи	102
2.6.2	Описание метода	103
2.6.3	Реализация алгоритма метода.....	104
2.7	Выбор и обоснование комплекса программных средств.....	106
2.7.1	Выбор языка программирования и среды разработки	106
2.7.2	Выбор операционной системы	110
2.7.3	Выбор среды программирования	111
2.7.4	Выбор системы управления базами данных	111
3	Реализация системы	113
3.1	Описание интерфейса пользователя	113
3.1.1	Начало работы	113
3.1.2	Работа с конструктором.....	115
3.1.3	Настройка транспортного потока.....	118
3.1.4	Моделирование системы АЗС	121
3.2	Диаграммы реализации	121
3.2.1	Диаграмма компонентов	122
3.2.2	Диаграмма развертывания.....	122
3.2.3	Диаграмма классов.....	126
3.3	Физическая модель данных.....	126
3.4	Выбор и обоснование комплекса технических средств.....	130
3.4.1	Расчет объема внешней памяти	130
3.4.2	Расчет объема ОЗУ.....	131
3.4.3	Минимальные требования, предъявляемые к системе.....	132
	ЗАКЛЮЧЕНИЕ	133
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	134

ПРИЛОЖЕНИЕ А Руководство пользователя.....	140
A.1 Назначение системы	140
A.2 Условия работы системы	140
A.3 Установка системы	140
A.4 Работа с системой	141
A.4.1 Начало работы	141
A.4.2 Работа с конструктором.....	143
A.4.3 Настройка транспортного потока	149
A.4.4 Моделирование системы АЗС.....	152
ПРИЛОЖЕНИЕ Б Листинг модулей программы	153

ВВЕДЕНИЕ

Автозаправочная станция (АЗС) – неотъемлемая часть жизни каждого автомобилиста не только в России, но и во всем мире. АЗС является комплексом оборудования на придорожной территории, на которой автомобилисты имеют возможность заправить свой автомобиль соответствующим видом топлива.

Первые отдельно стоящие заправочные станции для автомобилистов появились в США в начале XX века. Они представляли собой одну или две цистерны, стоящие на подпорках, от каждой шли шланги, по которым бензин самотеком поступал в баки автомобилей [1].

В России первые бензоколонки появились в 1911 году, когда Императорское Автомобильное Общество заключило договор с Товариществом "Бр. Нобель" относительно "Бензиновых станций". Уже в 1914 году в крупных городах страны работало 440 таких станций [2].

АЗС является сложной системой, которая состоит из большого количества настраиваемых компонентов. Перед созданием АЗС в реальной жизни, необходимо проверить весь функционал каждой компоненты. Моделирование является одним из наиболее популярных методов исследования объектов. С её помощью появляется возможность проверить работу каждой компоненты и работу АЗС в целом.

Во время курсового проектирования необходимо разработать систему моделирования работы АЗС, которая бы позволяла пользователю конструировать топологию АЗС и моделировать ее работу по построенной топологии.

Разработка системы будет производиться по технологии быстрой разработки приложений RAD (Rapid Application Development), которая поддерживается методологией структурного проектирования, а также включает элементы объектно-ориентированного проектирования и анализа предметной области. При проектировании системы будут использованы методологии ООАП (Object-Oriented Analysis/Design), в основе которой

лежит объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов, и язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения [3].

1 Описание и анализ предметной области

Предметной областью часто называют ту часть реального мира, которая имеет непосредственное отношение к разработке той или иной программы. Предметная область состоит из тех частей, которые могут оказать влияние на создание конкретных объектов и их взаимосвязей между объектами [4].

1.1 Описание структуры АЗС

АЗС – комплекс зданий, сооружений и оборудования, ограниченный участком площадки и предназначенный для заправки транспортных средств моторным топливом и маслом [5].

На АЗС организуется продажа масел, консистентных смазок, запасных частей, принадлежностей к автомобилям и другим транспортным средствам, прием от владельцев индивидуального транспорта отработанных масел и мелкой тары из-под нефтепродуктов, техническое обслуживание, а также оказание сервисных услуг по обслуживанию автотранспорта. Пример типовой АЗС приведен на рисунке 1.



Рисунок 1 – Пример типовой АЗС

1.1.1 Классификация АЗС

Классификация АЗС проходит по следующим признакам [6]:

- по функциональному назначению (общего пользования, ведомственные);
- по способу размещения резервуаров (с подземным расположением, с наземным расположением, с расположением на транспортном средстве);
- по типу расположению на местности (дорожные, городские, сельские, речные);
- по нормативным параметрам типовых проектов (по числу топливораздаточных колонок, по числу заправок в час пик, по количеству заправляемых машин в сутки, по общей вместимости резервуаров).

На рисунках 2 и 3 изображены АЗС с подземным и надземным размещением резервуаров.



Рисунок 2 – АЗС с подземным размещением резервуара



Рисунок 3 – АЗС с надземным размещением резервуара

В зависимости от способа размещения резервуаров, АЗС бывают следующих типов [6]:

- Модульная АЗС – АЗС с надземным расположением резервуаров для хранения топлива, технологическая система, которой характеризуется разнесением ТРК и контейнера хранения топлива, выполненного как единое заводское изделие.
- Контейнерная АЗС – АЗС с надземным расположением резервуаров для хранения топлива, технологическая система которой характеризуется размещением ТРК в контейнере хранения топлива, выполненном как единое заводское изделие.
- Передвижная АЗС – предназначена для розничных продаж топлива мобильная технологическая система, которая установлена на автомобильной шасси, прицепе или полуприцепе и выполнена как единое заводское изделие. Передвижная АЗС изображена на рисунке 4.
- Блочная АЗС – это АЗС с подземным расположением резервуаров для хранения топлива, технологическая система которой характеризуется над блоком хранения топлива, выполненным как единое заводское изделие. На рисунке 5 представлено схематичное изображение блочной АЗС.

Из перечисленных видов АЗС в системе будет использоваться только блочная АЗС. Продажа топлива будет осуществляться через ТРК. На рисунках 4 и 5 изображены передвижная и блочная АЗС.



Рисунок 4 – Передвижная АЗС

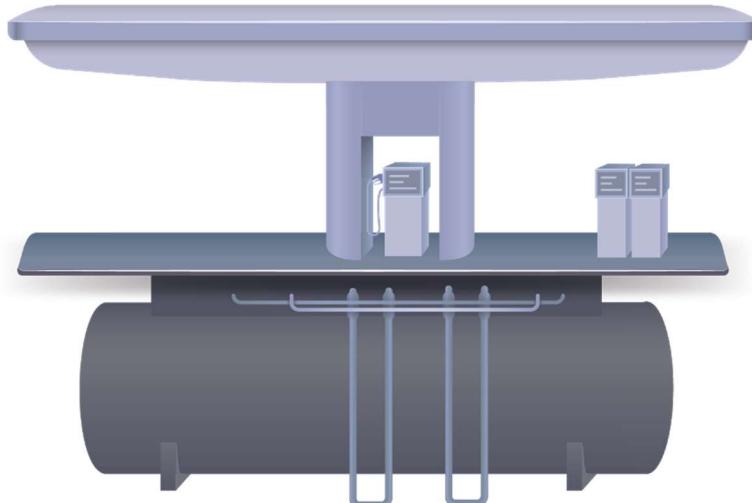


Рисунок 5 – Схематичное изображение блочной АЗС

1.1.2 Топливораздаточные колонки

Топливораздаточные колонки (ТРК) – это устройства, предназначенные для заправки автотранспортных средств топливом и учета выданного количества. Они классифицируются по следующим признакам [7]:

- по степени мобильности (переносные, стационарные);
- по виду привода (ручной, электрический, комбинированный);

- по способу управления (ручной, от местного задающего устройства, от дистанционного задающего устройства, от автоматического задающего устройства);
- по числу постов налива (однопостовые, многопостовые);
- по номинальному расходу топлива (25, 40, 50, 100, 160 л/мин);
- по основной погрешности ($\pm 0,25\text{--}0,4\%$);
- по типу отсеченного устройства (механические, электрические).

На рисунках 6 и 7 изображены переносная и стационарная ТРК соответственно.



Рисунок 6 – Переносная ТРК



Рисунок 7 – Стационарная ТРК

В разрабатываемой системе значащими признаками для ТРК являются номинал расхода топлива и число постов налива.

1.1.3 Топливные баки

Топливный бак (ТБ) – сосуд, предназначенный для хранения топлива различных видов, являющийся основным компонентом АЗС, изготовленный из стали или полимерных материалов, позволяющий осуществлять хранение запасов топлива на АЗС для последующей его выдачи. От качества стального резервуара, установленного на АЗС зависит надежность и безаварийное функционирование всей АЗС [8].

ТБ классифицируются по следующим признакам [9]:

- по конструктивным особенностям (одностенный, двустенный);
- по количеству камер (однокамерные, многокамерные);
- по объему ($3, 4, 5, 6, 8, 10, 15, 20, 25, 40, 50, 60, 75, 100 \text{ м}^3$).

При разработке системы учитывался только объем топливного бака. На рисунке 8 изображен пример надземного ТБ.



Рисунок 8 – Пример надземного ТБ

Сегодня практически все стальные резервуары изготавливаются в виде цилиндрических горизонтальных сосудов, которые имеют плоские или конические днища, технологические вырезки для трубной обвязки и технологические люки для обслуживания.

1.1.4 Виды топлива

На АЗС используются различные виды топлива. Топливо в широком смысле слова – вещество, способное выделять энергию в ходе определённых процессов, которую можно использовать для технических целей. Химическое топливо выделяет энергию в ходе экзотермических химических реакций при горении. Некоторые виды топлива способны к самостоятельному горению в отсутствие окислителя. Однако большинство видов топлива, используемых в быту и в промышленности, требует для сжигания наличия кислорода. Такие топлива также могут называться горючими материалами [10].

Наиболее распространенными горючими материалами являются органические топлива, в составе которых есть углерод и водород. Топлива подразделяются по агрегатному состоянию вещества на твердые, жидкые и газообразные, а по способу получения на природные (уголь, нефть, газ) и искусственные.

Жидкие виды топлива:

- нефтяные топлива (бензин, газолин, дизель, мазут, керосин);
- масла (сланцевое масло, растительное или животные масла);
- спирты (этанол, метанол, пропанол);
- жидкое ракетное топливо;
- эфиры;
- синтетические топлива.

Газообразные топлива:

- пропан;
- бутан;
- метан;
- водород;
- сжатый природный газ.

В России автомобильные бензины выпускаются по ГОСТ 2084-77, ГОСТ Р 51105-97 и ГОСТ Р 51866-2002, а также по ТУ 0251-001-12150839-

2015 Бензин АИ 92, 95. Бензины подразделяются на летние и зимние, в зимних бензинах содержится больше низкокипящих углеводородов. Основные марки бензинов в соответствии с ГОСТ 32513-2013 [9]:

- АИ-80 (с октановым числом по исследовательскому методу не менее 80);
- АИ-92 (с октановым числом по исследовательскому методы не менее 92, для двигателей со степенью сжатия 8,0);
- АИ-95 (с октановым числом по исследовательскому методы не менее 95, для двигателей со степенью сжатия 9,0);
- АИ-98 (с октановым числом по исследовательскому методы не менее 98);
- АИ-100, 101, 102 (АИ-95 (с октановым числом по исследовательскому методы не менее 100, 101, 102).

В соответствии с ГОСТ Р 54283-2010, бензины маркируется тремя группами знаков, распределёнными дефисом (например, «АИ-92-4»):

- «АИ», обозначающие автомобильные бензины с октановым числом по исследовательскому методу;
- цифровое обозначение октанового числа, определенного исследовательским методом (например, 80, 92, 95 или 98);
- число 2, 3, 4 или 5 обозначают класс бензина в соответствии с техническим регламентом, число совпадает с номером экологического стандарта серии «Евро», которому должен соответствовать бензин.

1.1.5 Заправляемый транспорт

На АЗС могут заправляться различные типы автомобили. Существует различные типы транспортных средств (мопеды, мотоциклы, легковые и грузовые автомобили, тракторы и т.д.). В разрабатываемой системе в качестве моделей автомобиля использованы грузовые и легковые автомобили, примеры которых изображены на рисунках 9 и 10.



Рисунок 9 – Легковой автомобиль



Рисунок 10 – Грузовой автомобиль

Автомобиль описывается следующими характеристиками: марка, название модели, год выпуска, тип кузова, вид топлива, время разгона до 100 км/ч, максимальная скорость, расход топлива, габариты, масса, объем багажника, объем ТБ, тип двигателя, тип коробки передач, размер шины и дисков [11]. В разрабатываемом проекте ключевыми характеристиками являются название модели, объем ТБ, вид топлива, категория автомобиля.

1.2 Моделирование транспортных потоков

Транспортный поток – совокупность транспортных средств, движущихся по проезжей части дороги. В зависимости от числа полос и разрешенных направлений движения, транспортный поток подразделяют на следующие виды [12]:

- однополосный односторонний;
- двухполосный односторонний или двусторонний;
- трехполосный односторонний или двусторонний;
- четырехполосный (и более) односторонний или двусторонний

В разрабатываемом проекте транспортный поток будет представлен в виде однополосного одностороннего потока. В таком случае можно транспортный поток рассматривать как поток событий.

Под потоком событий понимается последовательность событий, происходящих одно за другим в какие-то моменты времени. События, образующие поток, в общем случае могут быть различными, но мы будем рассматривать лишь поток однородных событий, различающихся только моментами появления. Такой поток можно изобразить как последовательность точек $t_1, t_2, t_3, \dots, t_k, \dots$ на числовой оси, изображенной на рисунке 11, соответствующих моментам появления событий [13]. В нашем случае под потоком событий можно понимать последовательность появлений автомобилей на дороге, где появление автомобиля в транспортном потоке является событием.



Рисунок 11 – Поток событий на числовой оси

Поток событий называется регулярным, если события следуют одно за другим через строго определенные промежутки времени [13]. Если в транспортном потоке автомобили будут появляться через заданное время, то данный поток можно считать регулярным. Если автомобиль в транспортном

потоке появляется не через одинаковое количество времени, то ее появление определяется случайной величиной (СВ).

Переменная величина называется случайной, если в результате опыта она не может принимать действительные значения с определенными вероятностями. Наиболее полной, исчерпывающей характеристикой СВ является закон распределения.

Закон распределения (ЗР) – функция, позволяющая определять вероятность того, что СВ принимает определенное значение или попадает в некоторый интервал. Если СВ имеет данный закон распределения, то говорят, что она распределена по этому закону или подчиняется этому закону распределения [13].

В разрабатываемой системе при нерегулярном появлении автомобилей в транспортном потоке СВ может иметь следующие ЗР:

- равномерный ЗР;
- нормальный ЗР;
- показательный ЗР.

1.2.1 Методы моделирования случайных величин

Моделирование СВ заключается в определении (розыгрыше) в нужный по ходу имитации момент времени конкретного значения СВ в соответствии с требуемым (заданным) законом распределения. Наибольшее распространение получили метод обратной функции и метод композиций [14].

Метод обратной функции основан на следующей теореме. Если непрерывная СВ Y имеет плотность вероятности $f(y)$, то СВ X , определяемая преобразованием:

$$x = \int_{-\infty}^y f(y)dy = F(y),$$

имеет равномерный закон распределения на интервале $[0; 1]$.

Эту теорему поясняет рисунок 12, на котором изображена функция распределения СВ Y .

Теорему доказывает цепочка рассуждений, основанная на определении понятия «функция распределения» и условии теоремы:

$$F(x) = P(X < x) = P(Y < y) = F(y) = \int_{-\infty}^y f(y)dy = x.$$

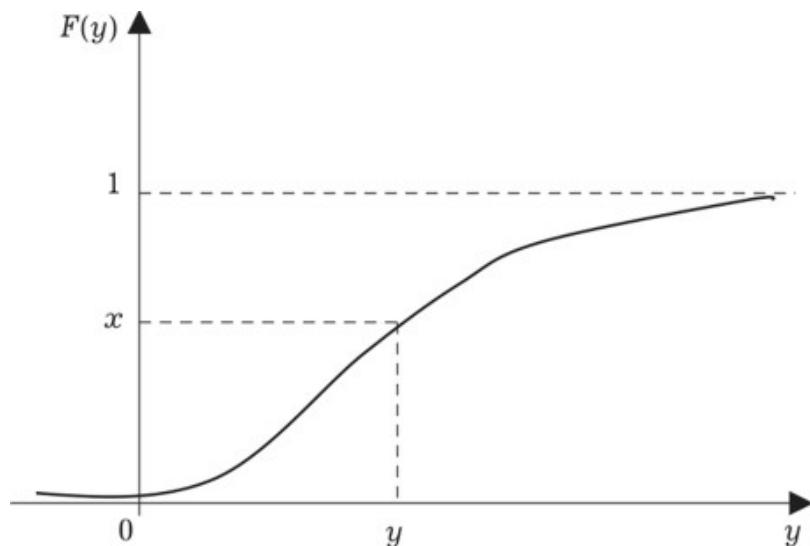


Рисунок 12 – Функция распределения СВ Y

Таким образом, получили равенство

$$F(x) = x,$$

а это и означает, что СВ X распределена равномерно на интервале $[0; 1]$.

В общем виде функция распределения равномерно распределенной на интервале $[a; b]$ СВ x имеет вид:

$$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & x > b \end{cases}$$

Теперь можно попытаться найти обратное преобразование функции распределения $F^{-1}(x)$.

Если такое преобразование существует (условием этого является наличие первой производной у функции распределения), то алгоритм метода включает всего два шага:

- моделирование псевдослучайного числа (ПСЧ), равномерно распределенного на интервале $[0; 1]$;
- подстановка этого ПСЧ в обратную функцию и вычисление значения СВ Y : $x = F(y) \Rightarrow y = F^{-1}(x)$.

При необходимости эти два шага повторяются столько раз, сколько возможных значений СВ Y требуется получить.

Применение метода композиции основано на теоремах теории вероятностей, доказывающих представимость одной СВ композицией двух или более СВ, имеющих относительно простые, более легко реализуемые законы распределения. Наиболее часто данным методом пользуются для генерации ПСЧ, имеющих нормальное распределение [15]. Согласно центральной предельной теореме распределение СВ X , задаваемой преобразованием

$$x = \frac{1}{\sqrt{\frac{k}{12}}} \cdot \left(\sum_{i=1}^k y_i - \frac{k}{2} \right),$$

где y_i – равномерно распределенные на интервале $[0; 1]$ ПСЧ, при росте k неограниченно приближается к нормальному распределению со стандартными параметрами ($\tau_y = 0$; $G_y = 1$).

Последнее обстоятельство легко подтверждается следующим образом. Введем СВ Z и найдем параметры ее распределения, используя соответствующие теоремы о математическом ожидании и дисперсии суммы СВ:

$$Z = \sum_{i=1}^k R_i,$$

$$M[Z] = m_z = M\left[\sum_{i=1}^k m_r\right] = \sum_{i=1}^k m_r = \sum_{i=1}^k \frac{1}{2} = \frac{k}{2},$$

$$\sigma_z = \sqrt{D_z} = \sqrt{\sum_{i=1}^k D_r} = \sqrt{\sum_{i=1}^k \frac{1}{12}} = \sqrt{\frac{k}{12}}.$$

При равномерном распределении в интервале $[0;1]$ СВ имеет параметры:

$$m_r = \frac{1}{2}, D_r = \frac{1}{12}.$$

Очевидно, что

$$Y = \frac{Z - m_z}{\sigma_z},$$

как любая центрированно-нормированная СВ, имеет стандартные параметры.

Как правило, берут $k = 12$ и считают, что для подавляющего числа практических задач обеспечивается должная точность вычислений.

1.2.2 Равномерный закон распределения

Непрерывная СВ X называется распределенной равномерно на отрезке $[a,b]$, если ее плотность распределения вероятность постоянна на данном отрезке [16]:

$$f(x) = \begin{cases} 0, & x \notin [a;b] \\ \frac{1}{b-a}, & x \in [a;b] \end{cases}.$$

Плотность вероятности – один из способов задания распределения СВ. Во многих практических приложениях понятия «плотность вероятность» и «плотность распределения СВ» или «функция распределения вероятностей» фактически синонимизируются и под ними подразумевается вещественная

функция, характеризующая сравнительную вероятность реализации тех или иных значений случайной переменной [17].

Функция распределения – функция, характеризующая распределение СВ; вероятность того, что СВ X примет значение, меньшее или равно x , где x – произвольное действительное число. При соблюдении известных условий плотностью определяет СВ [18].

Функция распределения в этом случае привет вид:

$$F(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x < b \\ 1, & x \geq b \end{cases}$$

В этом проекте СВ, распределенная по равномерному закону, будет получена с помощью стандартных средств языка программирования.

1.2.3 Нормальный закон распределения

Формула распределения вероятности значений случайной величины x по нормальному закону имеет вид:

$$f(x) = \frac{1}{\sigma_x \cdot \sqrt{2\pi}} \cdot e^{-\frac{(x-m_x)^2}{2\sigma_x^2}},$$

где параметр x –случайная величина, $f(x)$ – вероятность принятия случайной величиной значения x , m_x – математическое ожидание, σ_x – среднее квадратичное отклонение[19].

Нормализованным нормальным распределением называется такое нормальное распределение, у которого $m_x = 0$ и $\sigma_x = 1$. Из нормализованного распределения можно получить любое другое нормальное распределение с заданными m_x и σ_x по формуле: $z = z = m_x + x \cdot \sigma_x$.[19].

Будем формировать выборки случайных чисел, соответствующей нормальному распределению, с помощью метода композиции:

$$x = \sum_{i=1}^{12} (y_i - 6),$$

где y – равномерно распределенные на интервале $[0; 1]$ ПСЧ.

1.2.4 Показательный закон распределения

Получим в соответствии с методом обратной функции преобразование, позволяющее вычислить значения СВ, распределённой по показательному закону.

Функция распределения экспоненциального закона распределения:

$$F(x) = \begin{cases} 0, & q < 0 \\ 1 - e^{-qx}, & q \geq 0 \end{cases}$$

Смоделируем случайную величину:

$$\begin{aligned} F(x) &= 1 - e^{-qx} \\ y &= 1 - e^{-qx} \\ (1 - y) &= e^{-qx} \\ -qx &= \ln(1 - y) \quad , \\ x &= -\frac{1}{q} \ln(1 - y) = -\frac{1}{q} \ln(y) \end{aligned}$$

где x – СВ распределённая экспоненциально (интервал времени между моментами прибытия автомобилей), q – средняя интенсивность потока, а y – СВ, равномерно распределённая на $[0; 1]$.

1.3 Описание систем-аналогов

В настоящее время существует большое количество систем или сред моделирования, которые могут моделировать различные процессы, начиная от финансовой динамики экономических процессов, заканчивая моделирование транспортных потоков. Каждая модель решает определенную задачу моделирования. Рассмотрим популярные системы-аналоги.

1.3.1 Имитационное моделирование работы АЗС в среде AnyLogic

AnyLogic – программное обеспечение для имитационного моделирования, разработанное российской компанией The AnyLogic Company (бывшая «Экс Джей Текнолоджис», англ. XJ Technologies). Инструмент обладает современным графическим интерфейсом и позволяет использовать язык Java для разработки моделей [20].

AnyLogic содержит в себе готовые библиотеки моделирования процессов и потоков, в которых уже существуют готовые шаблоны или объекты, которые можно настраивать, а также определять взаимоотношения между ними.

Библиотека моделирования потоков позволяет пользователям моделировать процессы перевозки и хранения сыпучих материалов, жидкостей и газа, в том числе, трубопроводы, процессы добычи и транспортировки газа и топлива. Благодаря элементам библиотеки пользователи могут точно моделировать работу резервуаров, труб, конвейеров и их сетей, отслеживать партии грузов. С помощью библиотеки можно легко отобразить в модели такие характеристики потоков, как скорость и пропускная способность. Эта возможность позволяет находить узкие места и прогнозировать простои, а также оптимизировать операционные процессы [20].

Для точного моделирования поведения потоков в библиотеке используется подход, при котором скорость потока остается постоянной между двумя любыми дискретными событиями, влияющими на систему. Это делает процесс моделирования более простым и позволяет отслеживать изменения потоков.

Библиотека моделирования потоков AnyLogic была разработана как набор инструментов для моделирования хранения и транспортировки сыпучих и жидких материалов, газов и масляных веществ. С помощью библиотеки можно объединить такие непрерывные процессы, как хранение и транспортировка, в дискретную модель и наблюдать как они

взаимодействуют. Элементы библиотеку могут быть полезны для моделирования следующих задач:

- проектирование, эксплуатация и обслуживание трубопроводной сети;
- повышение эффективности объектов хранения и транспортировки нефти и газа;
- транспортировка сыпучих грузов, включая процессы погрузки и разгрузки;
- переработка полезных ископаемых и других сыпучих материалов.

Возможности библиотеки моделирования потоков AnyLogic позволяют:

- измерить общую производительность на объекте;
- оценить загруженность и пропускную способность сети и её элементов;
- провести календарное планирование отправки партий;
- спрогнозировать и предотвратить сбои в сети;
- снизить затраты на переоборудование предприятия, на техническое обслуживание и ремонт техники.

На рисунках 13 и 14 приведен пример рабочей модели АЗС и её 3D модель.

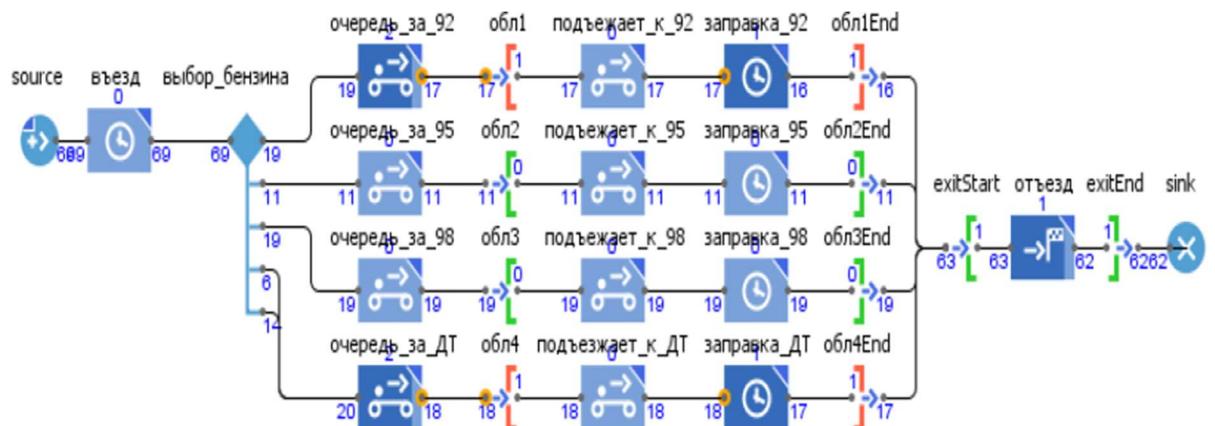


Рисунок 13 – Пример рабочей модели АЗС в среде AnyLogic

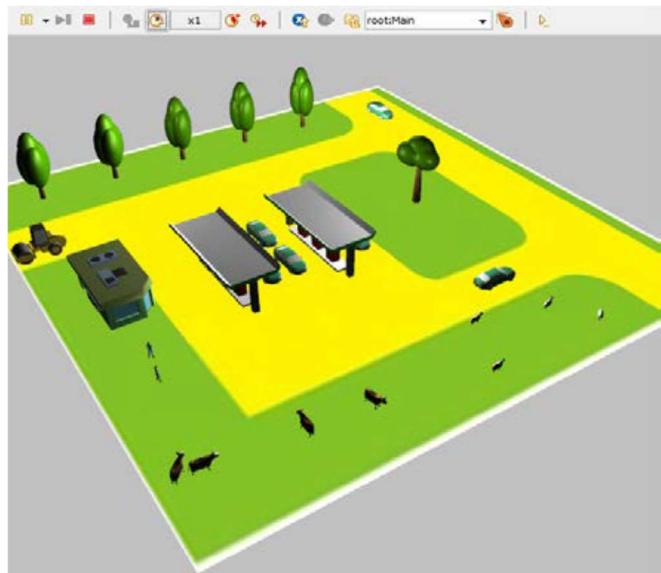


Рисунок 14 – 3D модель АЗС

Главные достоинства системы:

- наличие библиотек с набором инструментов;
- большое количество материалов для ознакомления с системой;
- возможность 3D моделирования;
- широкий спектр решения различных задач.

Основные недостатки системы:

- отсутствие пользовательского интерфейса;
- трудность в освоении необходимого материала;
- избыточный функционал для решения поставленной задачи.

1.3.2 Моделирование систем в программе VisSim

VisSim – виртуальный язык программирования, предназначенный для моделирования динамических систем, а также проектирования, базирующегося на моделях, для встроенных микропроцессоров [21].

Прикладной программный пакет VisSim – мощное, удобное в использовании, компактное и эффективное средство, предназначенное для построения, исследования и оптимизации виртуальных моделей физических и технических объектов, в том числе и систем управления. VisSim это аббревиатура выражения VisualSimulator – визуальная, воспринимаемая зренiem, среда и средство моделирования.

Программа VisSim разработана и развивается компанией VisualSolution. Предоставляет человеку развиты графический интерфейс, используя который, исследователь создает модели из виртуальных элементов с некоторой степенью условности так же, как если бы он строил реальную систему из настоящих элементов. Это позволяет создавать, а затем и исследовать и оптимизировать модели систем широкого диапазона сложности [22].

При описании и последующем построении модели в среде VisSim нет необходимости записывать и решать дифференциальные уравнения, программа это сделает сама по предложенной ей исследователем структуре системы и параметрами её элементов. Результаты решения выводятся в наглядной графической форме. Поэтому программой могут пользоваться те, кто не имеет глубоких познаний в математике и программировании.

При использовании VisSim не требуется владеть программированием на языках высокого уровня или ассемблере. В то же время специалисты, владеющие программированием, могут создавать собственные блоки, дополняя ими богатую библиотеку стандартных блоков VisSim.

Блоки в VisSim можно условно разделить на три основных категории и одну дополнительную:

- генераторы – блоки, имеющие только выход;
- преобразователи – блоки, имеющие вход и выход;
- индикаторы – блоки, имеющие только вход;
- надписи и комментарии – блоки без выходов и входов.

Программный комплекс VisSim использует развитый графический интерфейс, позволяющий основную часть создания модели выполнить с помощью мыши, а параметры элементов ввести с клавиатуры. Интерфейс VisSim состоит из основной панели (главного окна), изображенной на рисунке 15, имеющей меню и ряд кнопок управления, воспринимающих щелчки кнопок мыши, и совокупности окон, в которых строится модели и наблюдаются результаты её работы.

С точки зрения исследователя, интерфейс программы VisSim представляет собой интерактивный виртуальный лабораторный стенд,

обеспечивающий построение моделей из отдельных блоков, запуск процесса моделирования, управление им и контроль результатов.

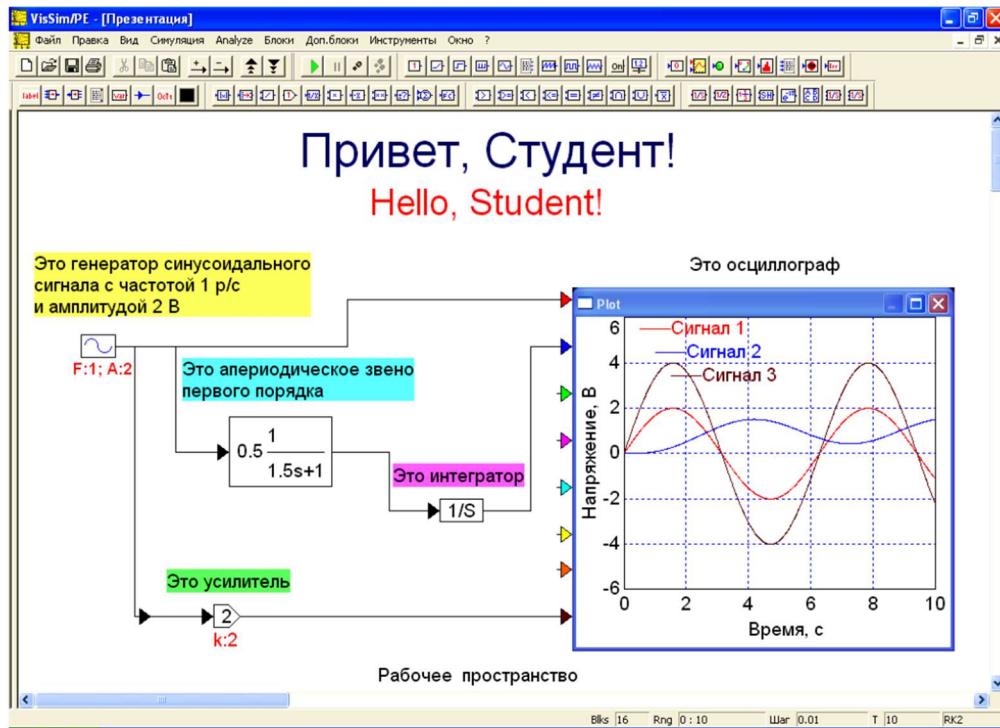


Рисунок 15 – Главное окно с основной панелью

Принцип моделирования в VisSim состоит в создании и исследовании виртуального аналога реальной системы – модели. Модель функционирует в соответствии с теми же уравнениями, что и моделируемая система.

Исходными данными для построения модели в VisSim являются структурно-алгоритмическая схема моделируемой системы, процесса или объекта и описывающие их дифференциально-алгебраические уравнения. На рисунке 16 изображен пример структурно-алгоритмической схемы. Вместо таких уравнений могут быть заданы операторы или функции, характеризующие отдельные элементы моделируемой системы.

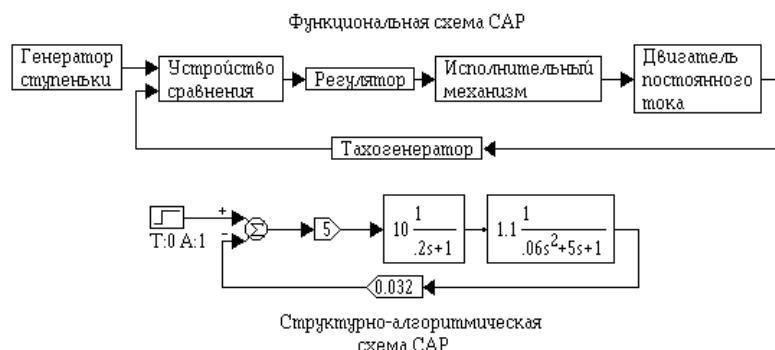


Рисунок 16 – Пример структурно-алгоритмической схемы

К достоинствам данной системы относятся:

- отсутствие необходимости записывать и решать дифференциальные уравнения;
- вывод результатов в простой графической форме;
- не требует глубоких познаний в математике и программировании.

К недостаткам системы относятся:

- не реализует полный функционал необходимой системы;
- наличие только стандартных блоков;
- отсутствие пользовательского интерфейса (интерфейс только для исследователя).

1.4 Диаграмма объектов предметной области

Анализ предметной области, позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации. Для документирования могут быть использованы различные графические средства [23].

С помощью анализа предметной области можно выделить основные объекты системы и установить взаимосвязи между ними. Большое количество автоматизируемых являются сложными, из-за чего их описание приводит к некоторым трудностям. Чтобы трудностей не возникало по ходу разработки системы, можно применить метод декомпозиции – разделения целого на части.

Декомпозиция – это научный метод исследования, в следствии которого определяется структура задачи, и одна большая задача заменяется несколькими небольшими задачами, которые взаимосвязаны друг с другом. С помощью декомпозиции любая подвергающая исследованию система может быть условно разделена на несколько более мелких, но взаимосвязанных

систем, которые, при необходимости, и далее могут подвергаться разделению [24].

На рисунке 17 приведена диаграмма объектов предметной области. АЗС состоит из кассы, въезда и выезда, через который автомобили будут заезжать и выезжать, проезжая часть, ТРК, ТБ, транспортный поток и шоссе.

ТРК характеризуется скоростью подачи топлива и видом топлива, которое поступает из ТБ.

Для ТБ главными параметрами являются его объем и вид хранимого топлива. При достижении критической отметки будет приезжать дозаправщик, который будет пополнять ТБ конкретным видом топлива.

Основной характеристикой кассы является её лимит. При достижении лимита, будет приезжать инкасация.

Автомобили, входящие в транспортный поток, с какой-то вероятностью будут заезжать на территорию АЗС и заправляться топливом. В системе будут предусмотрены грузовые и легковые типы транспорта.

Шоссе – часть дороги, к которой будет прилегать АЗС. На шоссе моделируется поток транспортных средств, который будет задаваться через параметры.

Все данные будут храниться в базе данных.

1.5 Постановка задачи

Во время выполнения курсового проекта необходимо разработать систему моделирования работы АЗС, с помощью которой можно создать рабочую модель АЗС в соответствии с заданными параметрами.

В состав разрабатываемой системы должен войти конструктор, с помощью которого можно было бы создать топологию АЗС. Перед созданием топологии пользователь будет должен указать ее размер по горизонтали (от 10 до 35 клеток) и по вертикали (от 7 до 25 клеток). Топология будет представлять собой прямоугольную область, состоящую из клеток – шаблонов, шоссе будет примыкать к топологии снизу.



Рисунок 17 – Диаграмма объектов предметной области

В системе должны быть предусмотрены следующие шаблоны:

- проезжая часть;
- ТРК (не больше 6);
- топливный бак (максимальное количество –не более размера служебной области);
- касса (в единичном экземпляре на топологии);
- въезд (в единичном экземпляре на топологии);
- выезд (в единичном экземпляре на топологии).

Работа конструктора будет осуществляться аналогично графическим редакторам с помощью выбора шаблона и размещения его на топологии. В процессе конструирования пользователь будет иметь возможность добавлять, удалять шаблоны, а также добавлять, изменять и удалять виды топлива в топливных баках (не более 12 видов топлива) и их объем. Цена за литр топлива может быть установлена от 10 рублей до 100 рублей. Объем топливных баков может быть изменен от 10000 л до 750000 л. Будет возможность изменять скорость подачи топлива ТРК от 25 л/мин до 160 л/мин. Все ТРК, расположенные на одной топологии, будут работать с одинаковой скоростью подачи топлива. Топология будет сохраняться в файл, структура файла определяется в процессе проектирования.

Для того чтобы перейти к моделированию пользователь должен создать топологию или загрузить ее из файла. После чего будет проведена проверка на корректность топологии. В ходе, которой должны будут проверяться:

- наличие въезда;
- наличие выезда;
- возможность проложить маршрут от въезда до ТРК;
- возможность проложить маршрут от ТРК до выезда;
- наличие кассы;
- наличие хотя бы одного бака с одним видом топлива;

- наличие хотя бы одной ТРК.

При успешном прохождении проверки топологии на корректность пользователю будет предложено выбрать один из трех закон распределения в соответствии с которым будут появляться автомобили и значение соответствующей ему характеристики. Будут доступны следующие варианты:

- экспоненциальный закон и его интенсивность (больше 0);
- равномерный закон и время между появлением автомобилей (не более 10 с);
- нормальный закон, его дисперсия (не больше 2 с) и математическое ожидание (не больше 5 с).

После выбора закона распределения и корректного указания значения его характеристики модель будет запущена. Генерируемый транспортный поток будет состоять двух типов автомобилей – легкового и грузового, помимо которых будут так же присутствовать автомобиль инкассации и дозаправщик. Автомобиль инкассации появляется при достижении 100000 рублей в кассе. Дозаправщик появляется при достижении любым баком критической отметки в 15% заполненности. У легковых автомобилей объем бака может быть от 50 до 100 л. У грузовых автомобилей объем бака может быть от 125 до 600 л. В процессе работы модели пользователь может получать информацию о системе: данные о заполненности топливных баков и состоянии кассы.

Таким образом, разрабатываемая система должна решать следующие задачи:

- 1) конструирование топологии АЗС:
 - задание размеров АЗС;
 - построение и исполнение модели АЗС по заданному шаблону;
 - выдача информации о свойствах элементов исполняемой модели;
- 2) моделирование работы АЗС:
 - 1 конфигурация транспортного потока;

- 2 расчет маршрута;
 - 3 проверка корректности топологии АЗС;
- 3) настройки параметров топологии:
- 1 выбор шаблона объекта;
 - 2 добавление объекта;
 - 3 удаление объекта;
- 4) визуализация конструирования топологии АЗС;
- 5) визуализация процессов работы исполняемой модели;
- 6) процесс работы с БД:
- 1 добавление записи;
 - 2 удаление записи;
 - 3 редактирование записи;
- 7) сохранение топологии АЗС в файл;
- 8) загрузка топологии АЗС из файла;
- 9) визуализация процессов работы АЗС;
- 10) выдача справочной информации о системе.

2 Проектирование системы

2.1 Выбор архитектуры системы

Архитектурой информационной системы называется концепция, согласно которой взаимодействуют компоненты информационной системы. Существуют следующие виды архитектур [25]:

- локальная;
- файл-серверная;
- клиент-серверная;
- трехслойная.

Локальные информационные системы широко использовались до появления компьютерных сетей. В этом случае все компоненты системы располагаются на одном компьютере. Очевидным недостатком этой архитектуры является возможность работать в информационной системе только одному пользователю. Другие пользователи не имеют возможности получить доступ к данным даже для чтения.

С появлением компьютерных сетей возникла возможность хранить данные в файлах на выделенном специально для этой цели компьютере. Такой компьютер называется файловым сервером или просто сервером. Компьютеры пользователей соединены с сервером сетью, поэтому доступ к данным могут получить несколько пользователей одновременно. Однако, кроме функции хранения данных и обеспечения доступа к ним, сервер никаких функций не выполняет. Приложения, обрабатывающие данные, находятся на пользовательских компьютерах.

Архитектура клиент-сервера предназначена для разрешения проблем файл-серверных приложений путем разделения компонентов приложения и размещения их там, где они будут функционировать более эффективно.

Особенностью архитектуры клиент-сервер является использование выделенных серверов баз данных, понимающих запросы на языке

структурированных запросов SQL (Structured Query Language) и выполняющих поиски, сортировку и агрегирование информации [26].

Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами [27]. Пример клиент-серверной архитектуры изображен на рисунке 18.

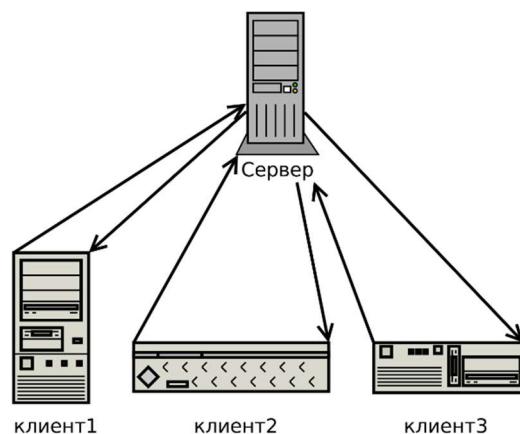


Рисунок 18 – Пример клиент-серверной архитектура

Преимущества клиент-серверной архитектуры:

- отсутствие дублирования кода сервера программами-клиентами.
- так как все вычисления выполняются на сервере.
- сервер защищён гораздо лучше большинства клиентов.

Недостатки:

- неработоспособность сервера может сделать неработоспособной всю вычислительную сеть.
- поддержка работы системы требует навыков системного администратора.

Существует два типа клиентов в архитектуре клиент-сервер: «тонкий» и «толстый» клиент.

Трехуровневая архитектура позволяет еще больше сбалансировать нагрузку на разные узлы и сеть, а также способствует специализации инструментов для разработки приложений и устраняет недостатки двухуровневой модели клиент-сервер. Нижний уровень данной архитектуры представляет собой приложения клиентов, имеющие программный интерфейс для вызова приложения на среднем уровне. Средний уровень представляет собой сервер приложений. Верхний уровень представляет собой удаленный специализированный сервер базы данных [28]. Пример трехуровневой архитектуры изображен на рисунке 19.

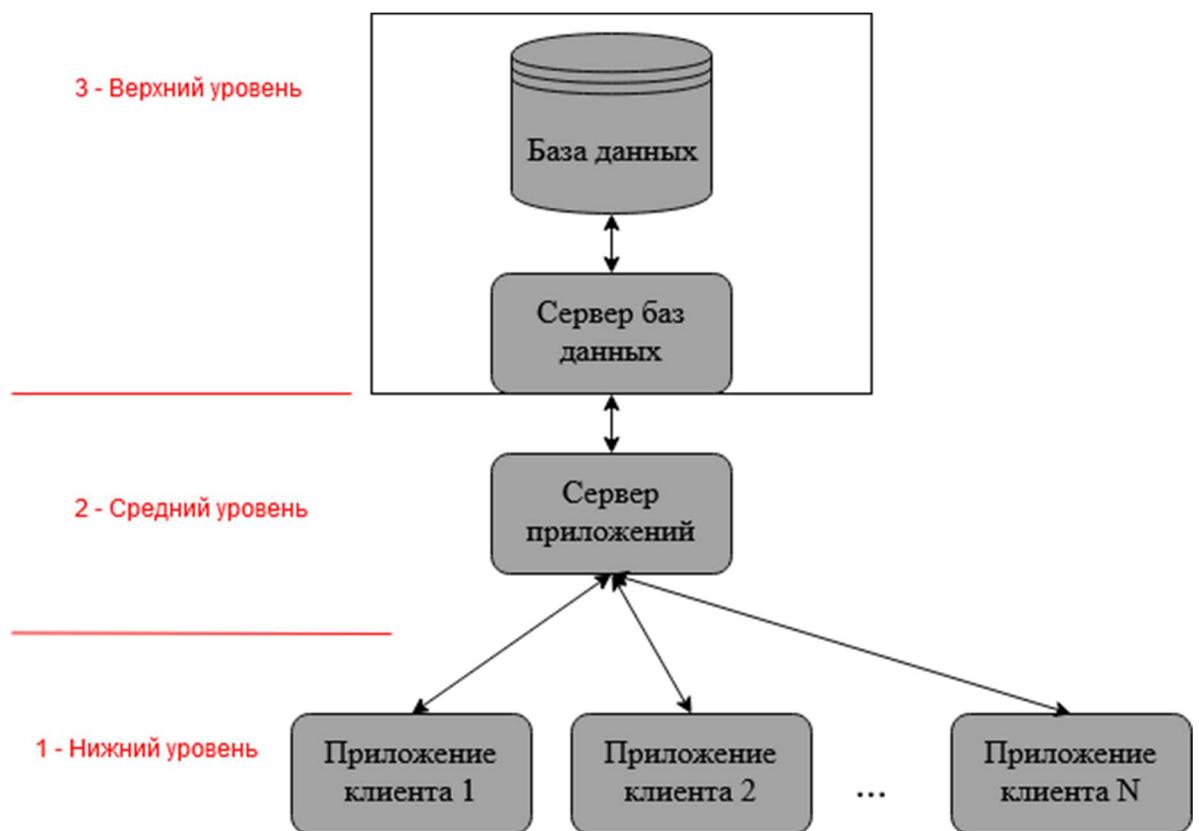


Рисунок 19 – Трехуровневая архитектура

В разрабатываемой системе будет реализована локальная архитектура. Выбор данной архитектуры обусловлен следующими факторами:

- невысокая требовательность системы;
- отсутствие ролей пользователей и многопользовательского режима;
- нет необходимости хранить большие объемы данных.

2.2 Структурная схема системы

При построении структурной схемы программной системы она по функциональному признаку разделяется на основные подсистемы, между ними указываются информационные связи и/или связи по управлению, описывается основное назначение подсистем.

Система – множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство [29].

Система должна представлять собой совокупность элементов (объектов, субъектов), находящихся между собой в определенной зависимости и составляющих некоторое единство (целостность), направленное на достижение определенной цели. Система может являться элементом другой системы более высокого порядка (надсистема) и включать в себя системы более низкого порядка (подсистемы). То есть систему можно рассматривать как набор подсистем, организованных для достижения определенной цели и описанных с помощью набора моделей (возможно, с различных точек зрения), а подсистему – как группу элементов, часть которых составляет спецификацию поведения, представленного другими ее составляющими [30].

В состав структурной схемы разрабатываемой системы входят следующие подсистемы:

1 Подсистема работы с топологией, в её состав входят следующие подсистемы:

- Подсистема настройки параметров, которая отвечает за задание название модели, размеров границы по вертикали и горизонтали и её расположение.
- Подсистема конструирования топологии, которая отвечает за размещение элементов с помощью шаблонов, в её состав входят следующие подсистемы:

- Подсистема настройки параметров объектов топологии, которая отвечает за настройку объекта, установленного на размеченной области.
- Подсистема валидации топологии, которая отвечает за проверку расстановки и настройки установленных объектов.
- Файловая подсистема, которая отвечает за загрузку и выгрузку модели АЗС в систему или в файл.

2 Подсистема моделирования работы АЗС, которая отвечает за работу модели АЗС во времени, в её состав входят следующие подсистемы:

- Подсистема настройки параметров моделирования, которая отвечает за выбор типа транспортного потока и задание параметров ЗР и СВ.
- Подсистема моделирования транспортного потока, которая отвечает за генерацию транспортного средства в транспортном потоке.
- Подсистема построения маршрутов, которая отвечает за составления маршрута автомобиля к необходимым объектам.

3 Подсистема работы с базой данных, которая отвечает заведение справочников и обеспечивает доступ ко всей информации, необходимой для работы системы.

4 Подсистема визуализации, которая отвечает за создание визуального представления во время работы модели АЗС.

5 Справочная подсистема, которая отвечает за предоставление руководства пользования системы и информации о её создателях.

На рисунке 20 приведена структурная схема разрабатываемой системы.

2.3 Разработка спецификации требований

Разработка спецификации программного обеспечения является одним из фундаментальных процессов технологии разработки ПО. Этот процесс анализа, формирования, документирования и проверки функциональных возможностей и ограничений системы называется в терминологии

программной инженерии «разработка требований» (спецификация требований). Он является критическим этапом в создании всех видов программных систем, что обусловлено тем, что ошибки, допущенные на этой стадии, ведут к возникновению серьезных проблем на этапах проектирования и разработки.

Требования – это свойства, которыми должно обладать ПО для адекватного определения функций, условий и ограничений выполнения ПО, а также объемов данных, технического обеспечения и среды функционирования [31].

Спецификация требований к ПО (SRS) – процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 9126-94, которые будут отрабатываться на этапах ЖЦ ПО [32].

В спецификации требований отражается:

- структура ПО;
- требования к функциям, качеству и документации;
- задается в общих чертах архитектура системы и ПО, алгоритмы, логика управления и структуры данных.

2.3.1 Функциональная спецификация

Функциональные требования задают «что» система должна делать; нефункциональные – с соблюдением «каких условий» (например, скорость отклика при выполнении заданной операции). При разработке этих требований в первую очередь необходимо учитывать потребности пользователя (заказчика) [33].

Пользовательские требования (User Requirements) – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы [34].

Функциональная спецификация системы приведена в таблице 1.

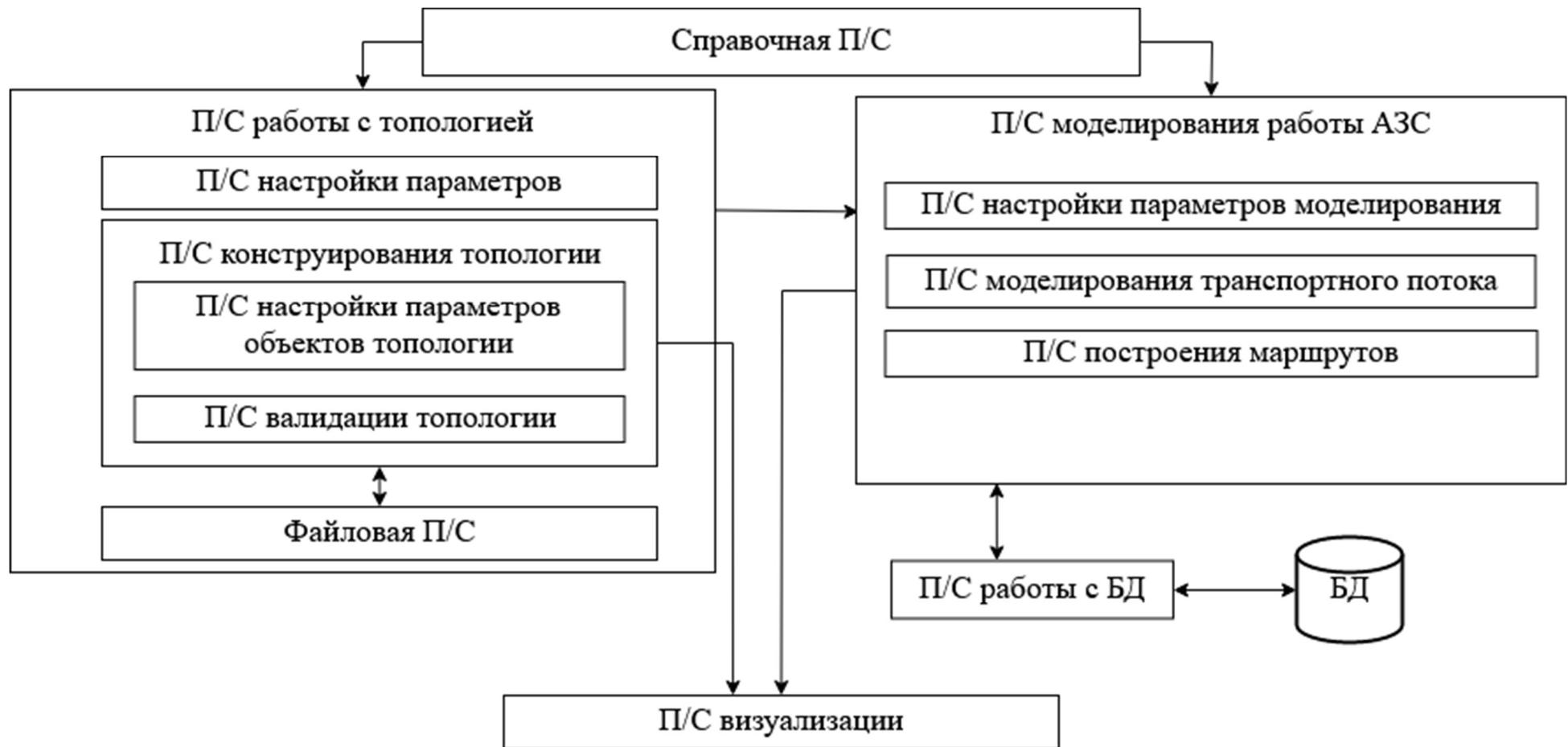


Рисунок 20 – Структурная схема системы

2.3.2 Перечень исключительных ситуаций

Исключительная ситуация – это ситуация, при которой система не может выполнить возложенных на нее функций или которая может привести к денормализации работы системы.

В таблице 2 приведен перечень исключительных ситуаций для разрабатываемой системы и описаны реакции системы на их возникновение.

2.4 Разработка прототипа интерфейса пользователя системы

Интерфейс пользователя является одним из важнейших элементов программы, это та часть программы, которая находится у всех на виду. Недочеты в пользовательском интерфейсе могут серьезно испортить впечатление даже о самых многофункциональных программах. Именно поэтому разработке и проектирования пользовательского интерфейса нужно уделить особое внимание.

Пользовательский интерфейс – совокупность средств и методов, при помощи которых пользователь взаимодействует с машинами, устройствами и аппаратурой [35].

Средства интерфейса пользователя включают средства вывода информации из устройства к пользователю экраны (дисплеи, проекторы) и средства ввода информации/команд пользователем в устройство (кнопки, переключатели, датчики положения и движения) [35].

Под методами интерфейса пользователя понимают набор правил, заложенных разработчиком устройства, согласно которым совокупность действий пользователя должна привести к необходимой реакции устройства и выполнения требуемой задачи (т. н. логический интерфейс) [35].

Таблица 1 – Перечень функций, выполняемых системой

Название подсистемы	Название функции	Информационная среда			
		Входные данные		Выходные данные	
		Назначение (наименование)	Тип, ограничения	Назначение (наименование)	Тип, ограничения
1	2	3	4	5	6
1 Справочная	1.1 Выдать сведения о разработчиках	Сведения о разработчиках системы (ФИО, номер группы)	Текстовый (*.HTML)	Визуальное отображение информации	-
	1.2 Выдать руководство пользователя	Руководство пользователя		Код ошибки	Целое
	1.3 Выдать правила конструирования топологии АЗС	Правила	Текст (строка)	Визуальное отображение информации	-

Продолжение таблицы 1

2 Настройка параметров	2.1 Задать название модели АЗС	Набор символов	Английские буквы, русские буквы, цифры	Название модели	Строка [30]
		Максимальная длина	Целое, 30		
	2.2 Выбрать расположение файла	-	-	Адрес файла	Строка
	2.3 Выбрать тип топологии	Список размеров	Маленькая (10x7)	Размеры топологии АЗС	10x7
			Средняя (20x15)		20x15
			Большая (35x25)		35x25
	2.4 Задать размер топологии по горизонтали	Допустимый диапазон значений	Целое, 10...35	Размер по горизонтали	Целое
	2.5 Задать размер топологии по вертикали		Целое, 7...25	Размер по вертикали	Целое
	2.6 Создать шаблон топологии	-	-	Размер топологии АЗС	Объект «Топология»

Продолжение таблицы 1

3 Конструирование топологии	3.1 Выбрать шаблонный элемент	Список шаблонных элементов (ШЭ)	TPK, въезд, выезд, ТБ, касса	Идентификатор ШЭ	Определяется в ходе проектирования
	3.2 Разместить ШЭ	Координата X	Целое	Модель АЗС	Объект «Топология»
		Координата Y			
		Идентификатор ШЭ	Строка		
	3.3 Выбрать ШЭ на карте	Топология	Объект «Топология»	Идентификатор ШЭ	Определяется в ходе проектирования
				Координата ШЭ по горизонтали	Целое
				Координата ШЭ по вертикали	Целое
	3.4 Переместить ШЭ на карте	Идентификатор ШЭ	Определяется в ходе проектирования	Модель АЗС	Объект «Топология»
		Старая координата ШЭ по горизонтали	Целое		

Продолжение таблицы 1

3 Конструирование топологии	3.4 Переместить ШЭ на карте	Старая координата ШЭ по вертикали	Целое	Модель АЗС	Объект «Топология»
		Новая координата ШЭ по горизонтали			
		Новая координата ШЭ по вертикали			
	3.5 Удалить ШЭ на карте	Топология	Объект «Топология»	Модель АЗС	Объект «Топология»
		Идентификатор ШЭ			
		Координата ШЭ по горизонтали			
		Координата ШЭ по вертикали	Целое	Модель АЗС	Объект «Топология»

Продолжение таблицы 1

4 Настройка параметров объектов топологии	4.1 Задать объема ТБ	Идентификатор ТБ	Определяется в ходе проектирования	Объем ТБ	Целое
		Допустимый диапазон значений	Целое, 10000...75000		
	4.2 Выбрать вид топлива для ТБ	Список видов топлива	Сущность БД "Топливо"	Вид топлива	Строка
		Идентификатор ТБ	Определяется в ходе проектирования	Вид топлива	Строка
	4.3 Задать скорость подачи топлива в ТРК	Идентификатор ТРК	Определяется в ходе проектирования	Скорость подачи топлива	Целое
		Допустимый диапазон значений	Целое, 25...160		
5 Валидация топологии	5.1 Проверить корректность модели АЗС	Модель АЗС	Объект «Топология»	Код ошибки	Целое

Продолжение таблицы 1

6 Файловая	6.1 Сохранить модель АЗС в файл	Модель АЗС	Объект «Топология»	Файл топологии	Определяется в ходе проектирования
		Полное имя файла	Строка		
	6.2 Загрузить модель	Полное имя файла	Строка	Модель АЗС	Объект «Топология»
7 Настройка параметров моделирования	7.1 Выбор типа потока	Список типов потока	Детерминированный, случайный	Тип потока	Логический
	7.2 Задать интервал времени появления ТС (для детерминированного потока)	Допустимый диапазон значений	Целое, 2...10	Интервал	Целое
	7.3 Выбрать ЗР (для случайного потока)	Список ЗР	Нормальный, равномерный, показательный	Текущий ЗР	Перечисляемый тип
	7.4 Задать параметры для равномерного ЗР	Допустимый диапазон значений	Целое, 0...10	Значение левой границы	Целое
				Значение правой границы	Целое, большее значения левой границы

Продолжение таблицы 1

7 Настройка параметров моделирования	7.5 Задать параметры для нормального ЗР	Допустимый диапазон значений	Целое, 0...5	Значение математического ожидания	Целое
			Целое, 0...2	Значение дисперсии	Целое
	7.6 Задать параметры для показательного ЗР	Допустимый диапазон значений	Целое, 0...N	Значение интенсивности	Целое
	7.7 Задать вероятность заезда ТС на АЗС	Допустимый диапазон значений	Вещественное, 0...1	Вероятность заезда	Вещественное
8 Моделирование транспортного потока	8.1 Сгенерировать время появления ТС	Параметры ЗР	Зависит от выбора ЗР	Время между появлениями ТС	Вещественное
9 Построение маршрута	9.1 Построить маршрут ТС, дозаправщика и инкассации	Модель АЗС	Объект «Топология»	Маршрут	Массив координат по горизонтали и вертикали
10 Работа с БД	10.1 Добавить вид топлива	Вид топлива	Строка	Список видов топлива	Сущность БД «Вид топлива»
		Цена топлива	Целое		
				Код ошибки	Целое

Продолжение таблицы 1

10 Работа с БД	10.2 Удалить тип топлива	ID Вида топлива	Целое число	Список видов топлива	Сущность БД «Вид топлива»
				Код ошибки	Целое
11 Визуализация	11.1 Отобразить топологию со всей инфраструктурой	Модель АЗС	Объект «Топология»	Визуальное отображение модели	-
	11.2 Отобразить текущее положение ТС, дозаправщика и инкасации на АЗС	Модель АЗС	Объект «Топология»		
		Координаты по горизонтали и вертинали	Целое, целое		

Таблица 2 – Перечень исключительных ситуаций

Название подсистемы	Название исключительной ситуации	Реакция системы
1 Справочная	1.1 Не возможно открыть файл справки	Выдача сообщения «Файл справки поврежден»
	1.2 Не возможно найти файл справки	Выдача сообщения «Отсутствует файл справки»
2 Настройка параметров	2.1 Не задано название модели	Выдача сообщения «Введите название модели»
	2.2 Не задано расположение файла	Выдача сообщения «Не задано расположение файла»
3 Валидация топологии	3.1 Отсутствует въезд	Выдача сообщения «Отсутствует въезд на карте»
	3.2 Отсутствует выезд	Выдача сообщения «Отсутствует выезд на карте»
	3.3 Отсутствует ТБ	Выдача сообщения «Отсутствует ТБ на карте»
	3.4 Отсутствует ТРК	Выдача сообщения «Отсутствует ТРК на карте»
	3.5 Отсутствует касса	Выдача сообщения «Отсутствует касса на карте»
4 Файловая	4.1 Невозможно загрузить файл	Выдача сообщения «Файл с таким именем не существует»
	4.2 Файл поврежден	Выдача сообщения «Файл Поврежден»

Продолжение таблицы 2

5 Настройка параметров моделирования	5.1 Правое значение отрезка меньше левого	Выдача сообщения "Правая граница должна быть больше левой"
6 Работа с БД	6.1 Добавить существующий вид топлива	Выдача сообщения "Указанный вид топлива уже есть в списке"
	6.2 Удалить несуществующий вид топлива	Выдача сообщения "Указанный вид топлива отсутствует в списке"

Основу взаимодействие пользователя с компьютером составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода/вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

Современные виды интерфейсов [36]:

1 WIMP-интерфейс (WIMP от: Window – окно; Image – образ; Menu – меню; Pointer – указатель) – диалог пользователя с компьютером ведется при помощи графических образов: меню, окон и других элементов. Интерфейс реализован на двух уровнях технологий: простой графический интерфейс и WIMP-интерфейс.

2 Командный интерфейс – пользователь дает команды компьютеру, который их выполняет и выдает результат пользователю. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки.

3 *Serial Digital Interface (SDI)* – последовательный цифровой интерфейс. семейство профессиональных цифровых видео-интерфейсов, стандартизованным Обществом инженеров кино и телевидения;

4 *Multiple document interface (MDI)* – способ организации графического интерфейса пользователя, предполагающий использование оконного интерфейса, в котором большинство окон (исключая, как правило, только модальные окна) расположены внутри одного общего окна.

В разрабатываемой системе взаимодействие пользователя с системой будет осуществляться с помощью WIMP-интерфейса.

Работа приложения начинается с открытия начального окна, изображенного на рисунке 21, в котором будет предложено создать модель АЗС или загрузить существующую модель, а также возможна выдача руководства пользователя.

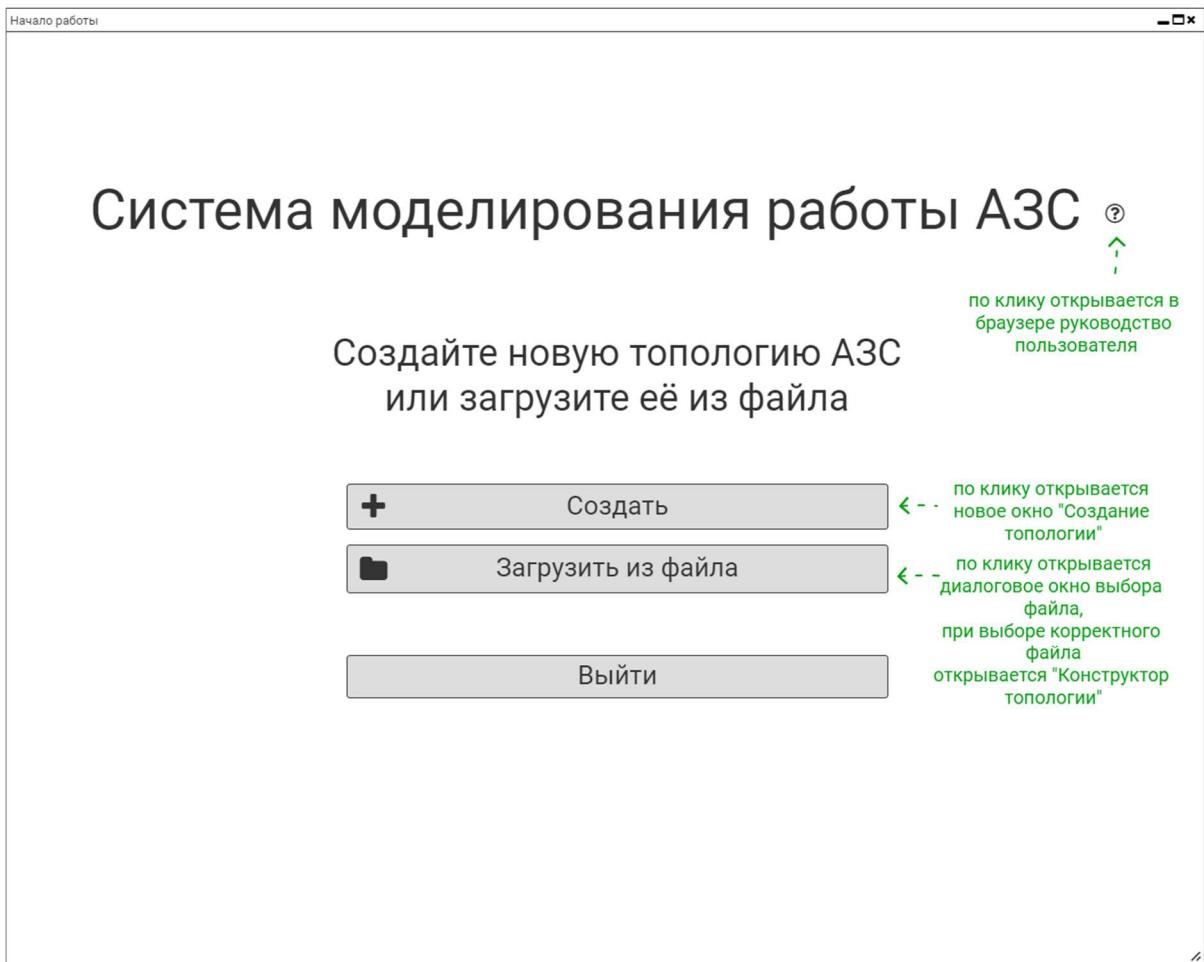


Рисунок 21 –Прототип экранной формы «Начало работы»

Если пользователь создает новую модель АЗС, то осуществляется переход к окну «Создание топологии», изображенной на рисунке 22. Пользователю необходимо задать название модели АЗС, указать путь, где будет храниться созданная модель, и выбрать её размеры из заданного списка или задать свои собственные значения по горизонтали и вертикали.

Если пользователь решил загрузить модель из файла, то ему будет предложено системное окно, в котором необходимо выбрать желаемый файл для загрузки.

После создания модели или загрузки из файла, пользователь перейдет к окну «Конструктор топологии», изображенном на рисунке 23.

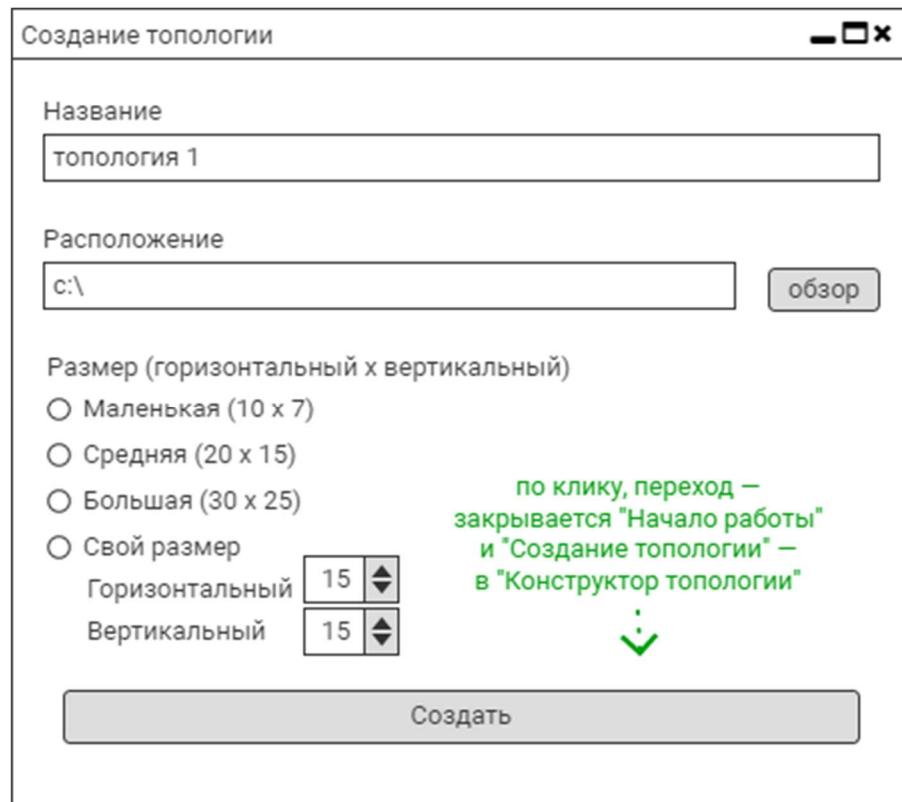


Рисунок 22 – Прототип экранной формы «Создание топологии»

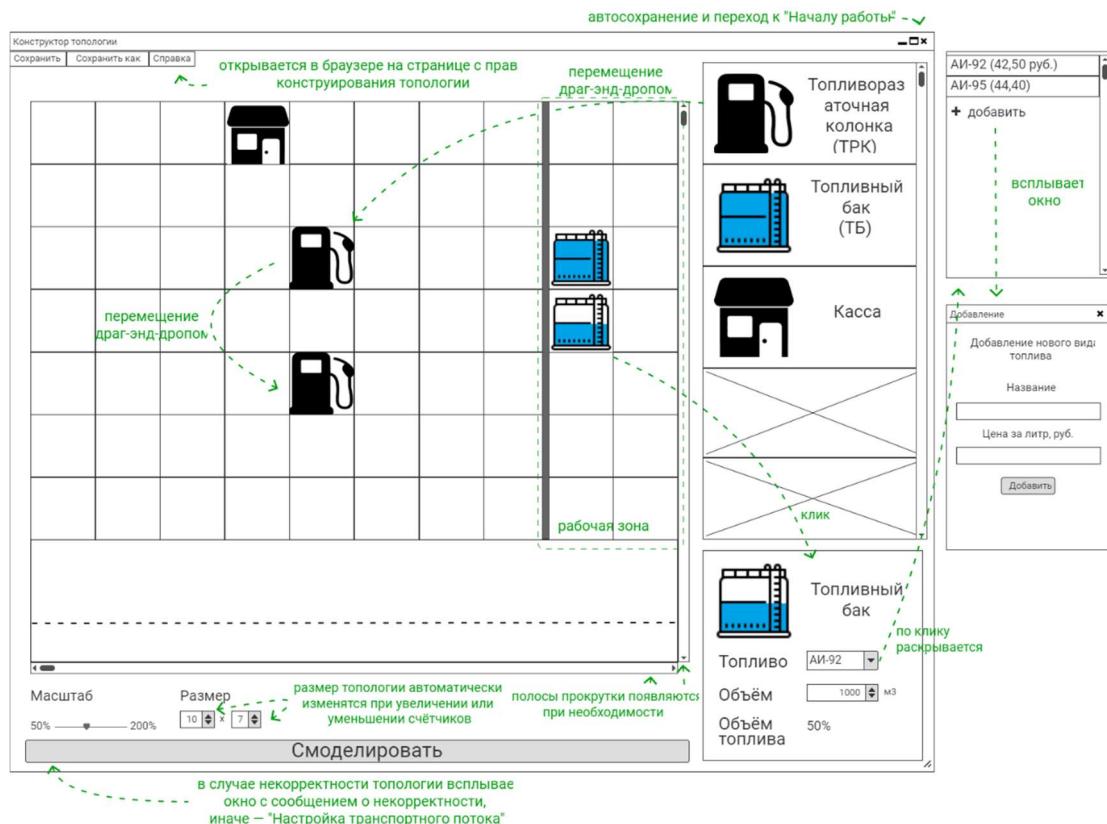


Рисунок 23 – Прототип экранной формы «Конструктор топологии»

В окне «Конструктор топологии» пользователю будет представлена возможность размещать ШЭ на топологию с помощью переноса ШЭ на топологию (DragandDrop). После того как ШЭ был размещен, пользователь должен настроить его: задать объем и вид топлива для ТБ, задать скорость подачи топлива для ТРК. Для корректной работы АЗС, пользователь должен разместить хотя бы 1 ТРК, 1 ТБ, кассу, въезд и выезд.

При установке ТБ пользователь должен выбрать существующий вид топлива, создать свой собственный вид топлива, указав название и цену за литр, или удалить уже существующий вид топлива.

После расположения ШЭ и их настройки, пользователь должен нажать на кнопку «Смоделировать», после этого осуществляется валидация топологии. Если валидация прошла успешно, то осуществляется переход к окну «Настройка транспортного потока». Прототип экранной формы изображен на рисунке 24.

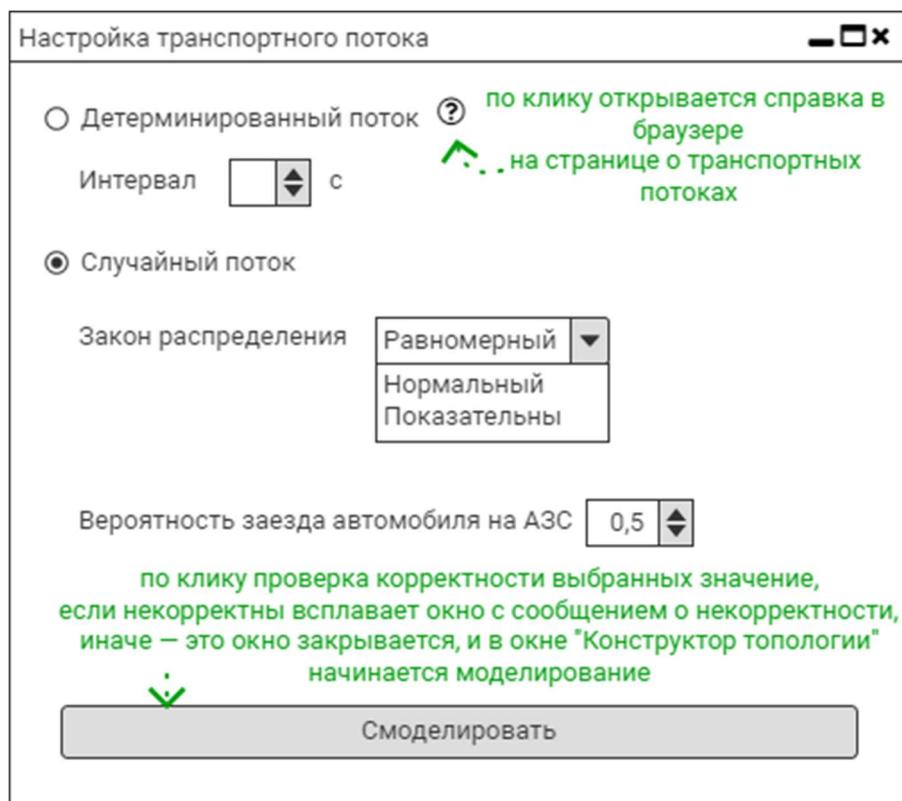


Рисунок 24 – Прототип экранной формы «Настройка транспортного потока»

В настройках транспортного потока пользователю предложат на выбор детерминированный поток, в котором пользователь должен указать интервал

появления машин в транспортном потоке, или случайный поток, где пользователь должен выбрать ЗР и настроить соответствующие параметры. Также необходимо указать вероятность заезда ТС на АЗС. Помимо этого, пользователю доступна справочная информацию о транспортных потоках, которая открывается в браузере.

Если все параметры были заданы верно, пользователь должен нажать на кнопку «Смоделировать», осуществляется переход к окну «Моделирование», на котором можно увидеть модель работы АЗС. Прототип окна «Моделирование» изображено на рисунке 25.

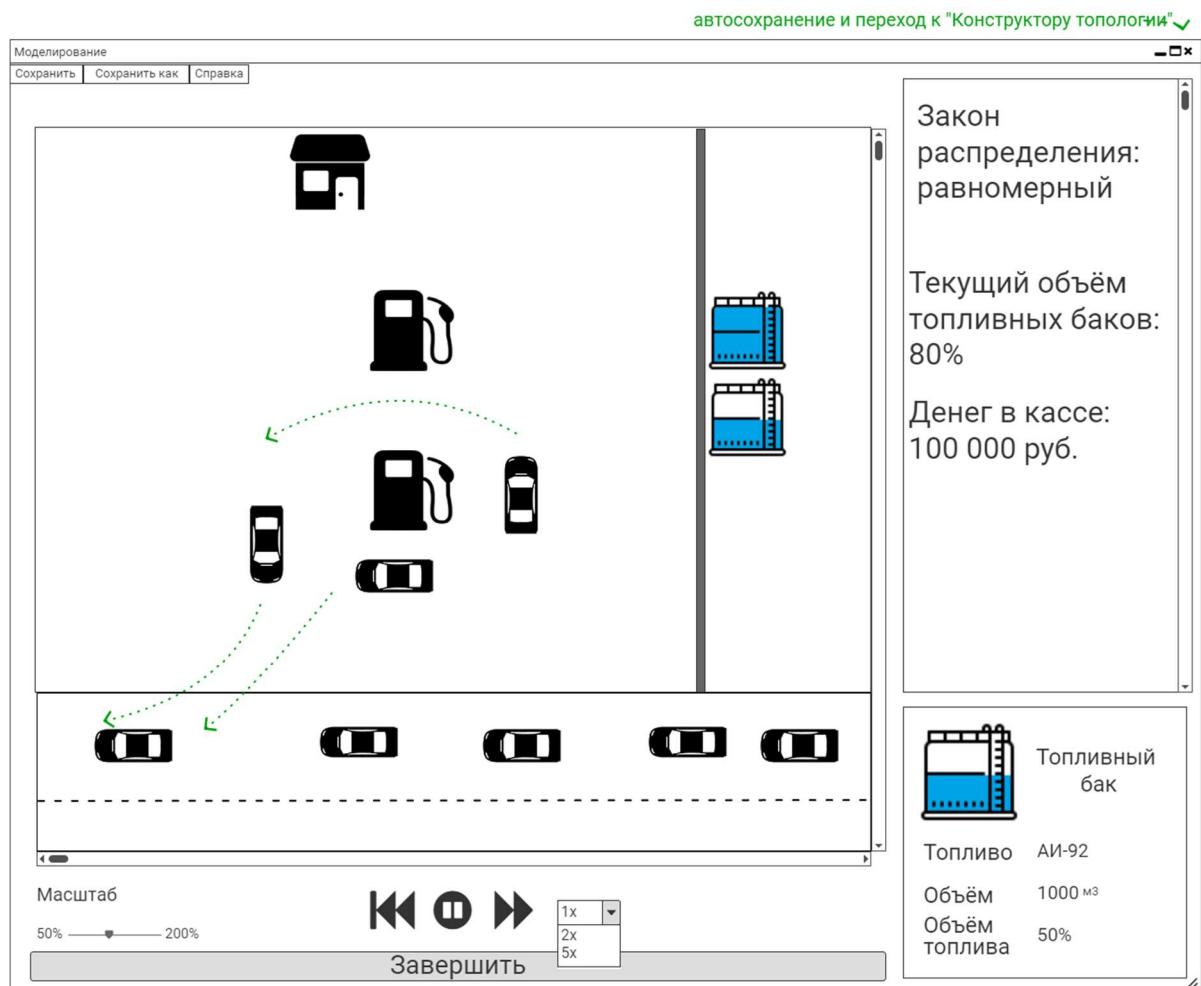


Рисунок 25 – Прототип экранной формы «Моделирование»

На рисунке 26 приведена навигационная модель разрабатываемого приложения.

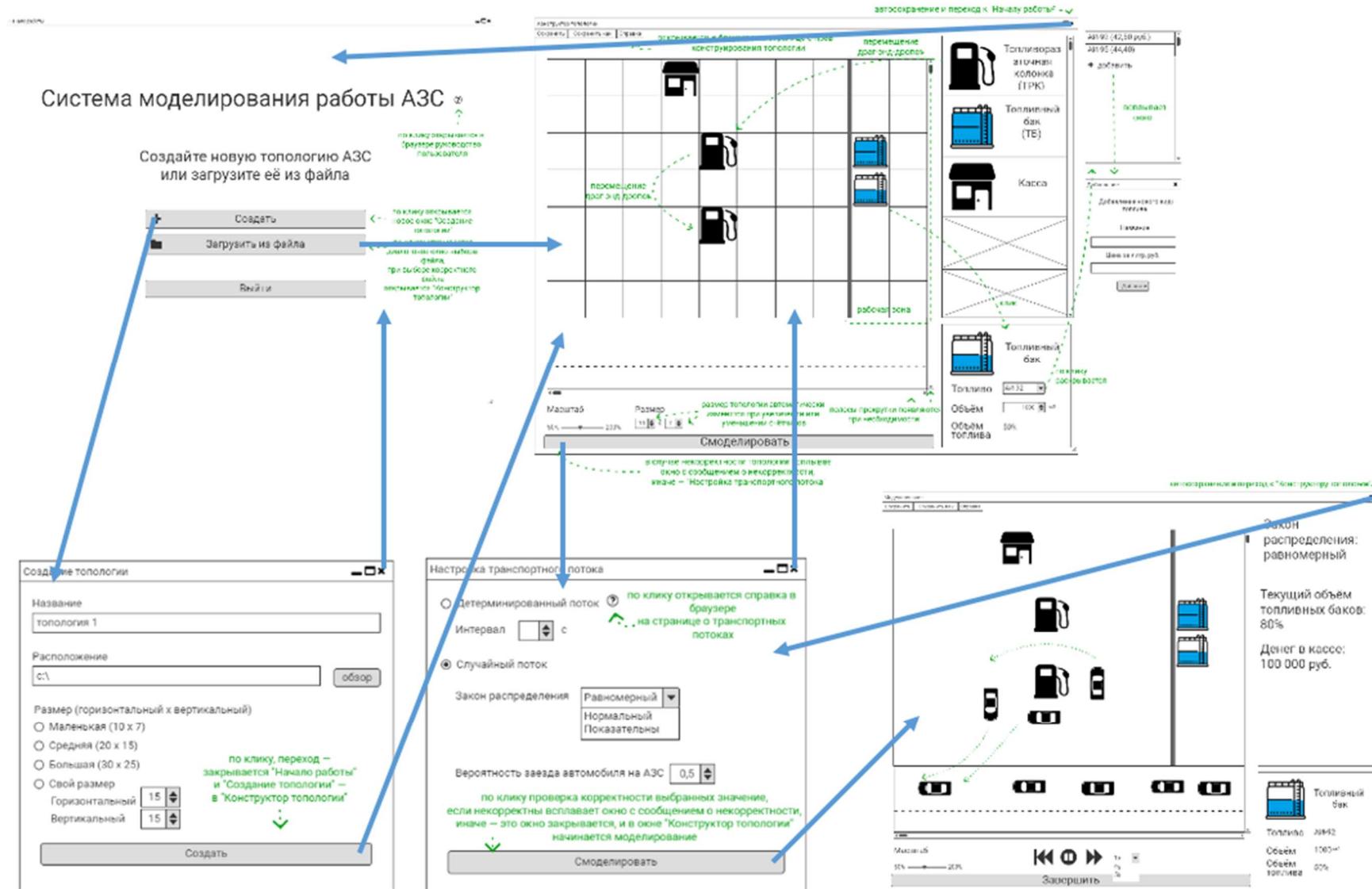


Рисунок 26 – Навигационная модель приложения

2.5 Разработка информационно-логического проекта системы

2.5.1 Язык UML

Для специфирования (построения точных, недвусмысленных и полных моделей) системы и ее документирования используется унифицированный язык моделирования UML.

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем [37].

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Этот язык позволяет рассмотреть систему со всех точек зрения, имеющих отношение к её разработке и последующему развертыванию.

Будем использовать UML для специфирования системы и ее документирования. Язык UML предоставляет стандартный способ написания проектной документации на системы, включая концептуальные аспекты, такие как бизнес-процессы и функции системы, а также конкретные аспекты, такие как выражения языков программирования, схемы баз данных и повторно используемые компоненты ПО.

2.5.2 Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актерами и вариантами использования.

Визуальное моделирование с использованием нотации UML можно представить, как процесс спуска от наиболее общей и абстрактной концептуальной модели исходной бизнес-системы к логической и

физической модели, соответствующей ПС. Вначале строится модель в форме, так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы и является исходным концептуальным представлением в процессе ее проектирования и разработки. На ней изображаются отношения между актерами и вариантами использования [38].

Актер (actor) – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними [38].

Вариант использования – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами [38].

Во время конструирования пользователь может создать свою топологию с заранее выбранными размерами или задать свои собственные, или загрузить модель топологии из файла. После того как пользователь определился с топологией, ему необходимо разместить на поле основные типы ШЭ и настроить их параметры. Для запуска процесса моделирования пользователь должен настроить параметры моделирования: выбрать поток (случайный или детерминированный), задать параметры потока (задать интервал появления автомобиля или настроить параметры выбранного ЗР), и указать вероятность заезда автомобиля на АЗС. После настройки параметров моделирования, пользователю доступен процесс моделирования, где он может менять масштаб времени, брать паузу и запускать процесс моделирования. На рисунке 27 приведена диаграмма вариантов использования системы. На диаграмме отображено, что пользователь может получать справочную информацию, создавать или загружать топологию, конструировать топологию и настраивать параметры моделирования.

2.5.3 Сценарии

Сценарий (scenario) – определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме обычного текста [39].

В контексте языка UML сценарий используется для дополнительной иллюстрации взаимодействия актеров и вариантов использования.

Рассмотрим несколько сценариев.

Вариант использования «Задать параметры равномерного ЗР»

Краткое описание. Позволяет пользователю задать параметры равномерного ЗР. Является частным случаем варианта использования «Настроить параметры ЗР».

Актант. Пользователь.

Предусловия. Компьютер пользователя включен. Пользователь запустил приложение. Пользователь создал новую или загрузил существующую топологию из файла. Система отображает форму «Конструктор топологии», на которой размещена топология с ШЭ, прошедшая валидацию. При нажатии один раз левой кнопкой мыши на кнопку «Смоделировать», поверх формы появляется модальное окно «Настройка транспортного потока».

Основной поток событий.

1 В модальном окне (см. рисунок 24) отображены две радиокнопки с выбором транспортного потока (детерминированный, случайный), полеввода для указания вероятности заезда автомобиля на АЗС. Также присутствует кнопки «Смоделировать», кнопка «» и кнопка «» со справочной информацией.

2 Пользователь с помощью нажатия левой кнопкой мыши нажимает на радиокнопку «Случайный поток».

3 При выборе радиокнопки «Случайный поток», становится доступным список ЗР («Равномерный», «Нормальный», «Показательный»).

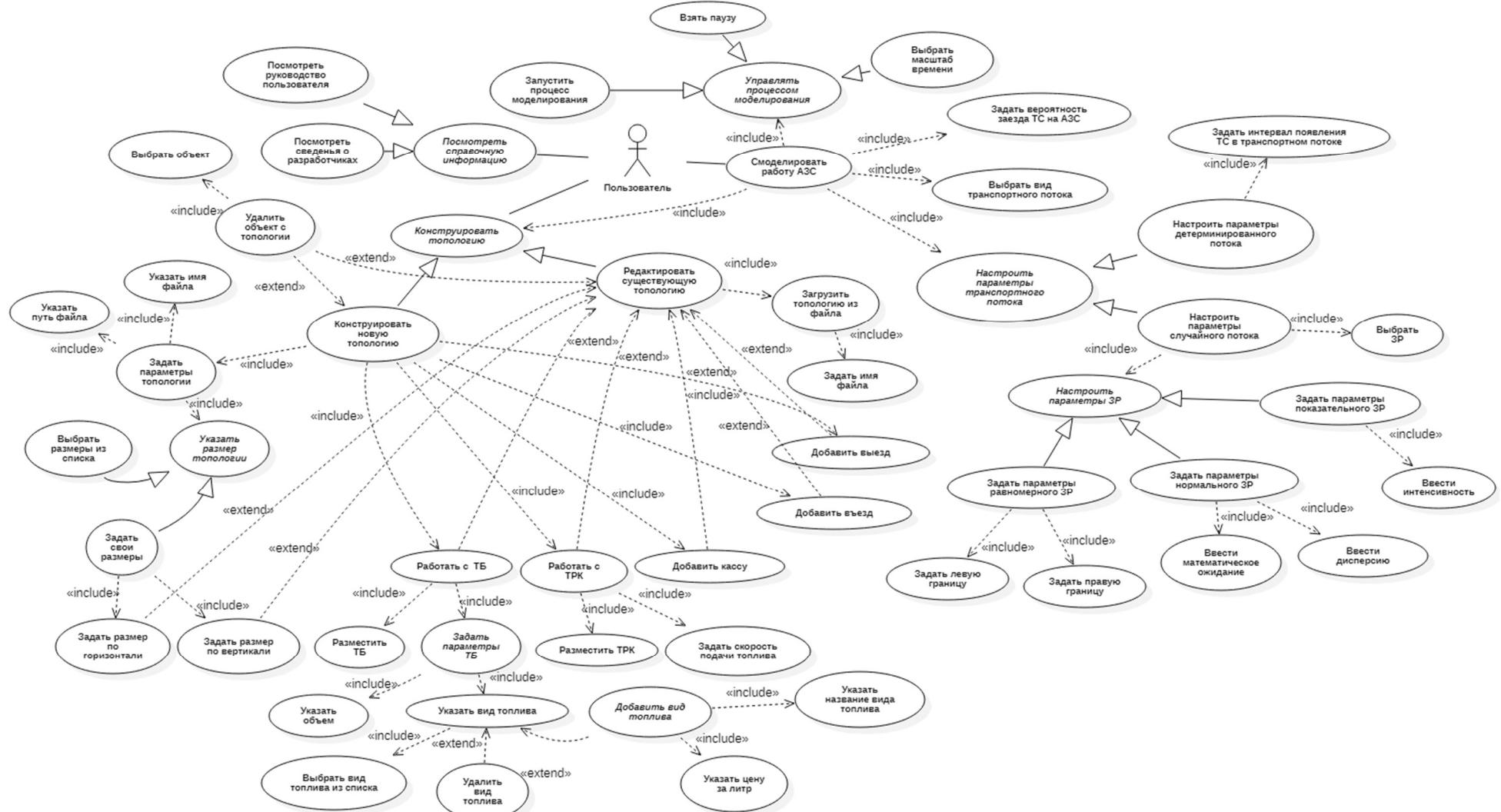


Рисунок 27 – Диаграмма вариантов использования системы

- 4 Пользователь выбрал из появившегося списков ЗР «Равномерный».
- 5 При выборе ЗР «Равномерный» в окне появляются два поля ввода «Левая граница» и «Правая граница», в которые нужно ввести левую и правую границы для Равномерного ЗР.
- 6 Пользователь вводит левую границу в соответствующее поле.
- 7 Пользователь вводит правую границу в соответствующее поле.
- 8 Пользователь нажимает на кнопку «Смоделировать».
 - A1: Пользователь щелкнул левой кнопкой мыши по кнопке «».
- 9 Система проверяет левую и правую границу на валидность (набор допустимых символов; правая граница имеет значение больше чем левая граница).
 - A2: Левая и правая граница не является валидной.
 - A3: Левая граница не является валидной.
 - A4: Правая граница не является валидной.
- 10 Система закрывает модальное окно «Настройка транспортного потока».

Вариант использования завершается успешно.

Альтернативы.

- A1: Пользователь щелкнул левой кнопкой мыши по кнопке «».
 - A1.1 Система закрывает окно и выводит на экран форму «Конструктор топологии».
 - A2: Левая и правая граница не является валидной.
 - A2.A1 Система выводит под соответствующими полями сообщение «Левая и правая граница введены неверно. Повторите ввод!».
 - A2.A1.1 Поля для ввода левой и правой границы очищаются.
 - A2.A1.2 Границы полей ввода левой и правой границы подсвечиваются красным цветом.
 - A2.A1.3 Осуществляется переход к п.6 основного потока событий.
 - A2.A2 Система выводит под соответствующими полями сообщение «Левая граница должна быть меньше правой. Повторите ввод левой и правой границы!»

A2.A2.1 Поля для ввода левой и правой границы очищаются.

A2.A2.2 Границы полей ввода левой и правой границы подсвечиваются красным цветом.

A2.A2.3 Осуществляется переход к п.6 основного потока событий.

A3: Левая граница не является валидной.

A3.1 Система выводит под соответствующим полем сообщение «Левая граница введена неверно. Повторите ввод!».

A3.2 Поле для ввода левой границы очищается.

A3.3 Граница поля ввода левой границы подсвечивается красным цветом.

A3.4 Осуществляется переход к п.6 основного потока и п.7 основного потока не выполняется.

A4: Правая граница не является валидной.

A4.1 Система выводит под соответствующим полем сообщение «Правая граница введена неверно. Повторите ввод!».

A4.2 Поле для ввода правой границы очищается.

A4.3 Граница поля ввода правой границы подсвечивается красным цветом.

A4.4 Осуществляется переход к п.7 основного потока событий.

Постусловия. При успешном завершении на экране – форма «Моделирование», на которой будет отображаться меню управления временем в моделируемой системе, справочная информация о системе и визуальное отображение работы системы.

Вариант использования «Разместить ТРК»

Краткое описание. Даёт возможность пользователю расположить ШЭ ТРК на топологии. Включается в вариант использования «Работа с ТРК».

Актант. Пользователь.

Предусловия. Компьютер пользователя включен. Пользователь запустил приложение. Пользователь создал новую или загрузил

существующую топологию из файла. На форме «Конструктор топологии» основное пространство экрана занимает поле редактора. Внизу экрана кнопка «Смоделировать». В правой части страницы находится панель инструментов с ШЭ. В верхней левой части экрана кнопки «Сохранить», «Сохранить как», «Справка».

Основной поток событий.

1 Система отображает экранную форму «Конструктор топологии», справа находится панель инструментов с ШЭ «Топливный бак», «TPK», «Касса», «Въезд», «Выезд», по центру находится поле редактора топологии.

2 Пользователь выбирает ШЭ «TPK», щелкая по нему левой кнопкой мыши (ЛКМ.).

3 Пользователь кликает мышкой на свободную клетку на поле редактирования.

A1: Пользователь, удерживая левую кнопку мыши, «перетягивает» ШЭ «TPK» на клетку на поле редактора.

A2: Пользователь нажимает на клетку, которая занята.

4 Система проверяет количество размещенных ШЭ «TPK» на поле топологии.

A3: Достигнуто максимальное количество ШЭ «TPK» на поле топологии.

5 В правой нижней части экранной формы «Конструктор топологии» появляется информация о размещенном элементе, где предлагается ввести скорость подачи топлива выбранного элемента.

6 Пользователь выполняет пункты 2 и 3 основного потока событий, пока не разместит необходимое количество TPK (не более 6). Вариант использования завершается успешно.

A4: Пользователь нажимает на кнопку «».

Альтернативы.

A1: Пользователь, удерживая левую кнопку мыши, «перетягивает» элемент на клетку на поле редактора.

A1.A1 Пользователь, удерживая левую кнопку мыши «перетягивает» элемент на свободную клетку на поле редактора.

A1.A1.1 Система отображает ШЭ в поле редактора в заданной клетке.

A1.A2 Пользователь, удерживая левую кнопку мыши «перетягивает» элемент на клетку, которая занята.

A1.A2.1 ШЭ не размещается в поле редактора в заданной клетке.

A2: Пользователь нажимает на клетку, которая занята.

A2.1 Выбранный ШЭ не размещается на клетке. Осуществляется переход к п.б основному потоку событий.

A3: Достигнуто максимальное количество ШЭ «TPK» на поле топологии.

A3.1 ШЭ не размещается на поле редактора в заданной клетке.

A4: Пользователь нажимает на кнопку «».

A4.1 Система сохраняет топологию и настроенные в ней параметры в файл и завершает работы программы. Вариант использования завершается.

Постусловия. При успешном завершении на экране – форма конструирования топологии.

Вариант использования «Добавить вид топлива»

Краткое описание. Даёт возможность пользователю добавить вид топлива к уже имеющимся. Является частным случаем варианта использования «Указать вид топлива».

Актант. Пользователь.

Предусловия. Компьютер пользователя включен. Пользователь запустил приложение. Пользователь создал новую или загрузил существующую топологию из файла. На топологии размещен хотя бы один элемент «Топливный бак». Система отображает экранную форму «Конструктор топологии», справа находится панель инструментов с ШЭ «Топливный бак», «TPK», «Касса», «Въезд», «Выезд», по центру находится

поле редактора топологии. На топологии размещен элемент «Топливный бак». Пользователь нажимает левой кнопкой мыши на ШЭ «Топливный бак». В правой нижней части формы появляются характеристики выделенного элемента: «Топливо», «Объем», «Объем топлива». Пользователь нажимает на название топлива, после чего появляется выпадающий список с видами топлива. Пользователь нажимает левой кнопкой мыши на последний элемент списка «Добавить».

Основной поток событий.

- 1 Открывается модальное окно «Добавление», в котором размещены поля ввода «Название» и «Цена за литр, руб.», кнопка «Добавить» и кнопка «».
- 2 Пользователь вводит название топлива в соответствующее поле.
- 3 Пользователь вводит цену за литр топлива в соответствующее поле.
- 4 Пользователь щелкает на кнопку «Добавить».
A1: Пользователь щелкнул левой кнопкой мыши по кнопке «».
- 5 Система проверяет название и цену за литр на валидность (набор допустимых символов, наличие значения в полях).

A2: Название и цена за литр не является валидной.

A3: Название не является валидным.

A4: Цена за литр не является валидным.

- 6 Система закрывает окно «Добавление». В списках видах топлива появляется новый вид топлива. Добавленный вид топлива становится выбранным видом топлива для выбранного ТБ. Вариант использования завершается успешно.

Альтернативы.

A1: Пользователь щелкнул левой кнопкой мыши по кнопке «».

A1.1 Система закрывает модальное окно «Добавление», не обновляя список видов топлива

A2: Название и цена за литр не является валидной.

A2.1 Система выводит под соответствующими полями сообщение «Название и цена за литр заполнены неверно. Повторите ввод!».

A2.2 Поля для ввода названия и цены за литр очищаются.

A2.3 Осуществляется переход к п.2 основного потока событий.

A3: Название не является валидным.

A3.1 Система выводит под соответствующим полем сообщение «Название указано неверно. Повторите ввод!».

A3.2 Поле ввода «Название» очищается.

A3.3 Осуществляется переход к п.2 основного потока событий и п.3 основного потока событий не выполняется.

A4: Цена за литр не является валидным.

A4.1 Система выводит под соответствующем полем сообщение «Цена за литр указана неверно. Повторите ввод!».

A4.2 Поле ввода «Цена за литр руб.» очищается.

A4.3 Осуществляется переход к п.3 основного потока событий.

Постусловия. При успешном завершении на экране – форма «Конструктор топологии», с активным блоком настройки параметров ТБ. Возможен переход к варианту использования «Задать объем».

Вариант использования «Указать размер топологии»

Краткое описание. Позволяет пользователю задать параметры для конструирования новой топологии. Включается в вариант использования «Задать параметры топологии».

Актант. Пользователь.

Предусловие. Компьютер пользователя включен. Пользователь запустил приложение. На экране отображена форма «Создание топологии». Пользователь создает новую топологию, поля «Название» и «Расположение» уже заполнены корректно.

Основной поток событий.

1 Система отображает форму «Создание топологии», на которой размещены поля ввода для названия топологии, расположения файла, системная кнопка «Обзор» и кнопка «Создать», радиокнопки с заранее заданным размерами топологии («Маленькая (10 x 7)», «Средняя (20 x 15)», «Большая (30 x 25)»), а также

радиокнопка «Свой размер» с неактивными полями ввода вертикального и горизонтального размера топологии. Активна радиокнопка «Средняя (20 x 15)».

2 Пользователь нажимает на кнопку «Создать».

А1: Пользователь ЛКМ выбирает радиокнопку «Свой размер».

А2: Пользователь ЛКМ выбирает радиокнопку «Маленькая (10x7)».

А3: Пользователь ЛКМ выбирает радиокнопку «Большая (30 x 25)».

А4: Пользователь щелкнул ЛКМ по кнопке «».

3 Система закрывает окно «Создание топологии» и выводит на экран форму «Конструктор топологии», на которой будет отображаться топология с заданным размером, панель инструментов с ШЭ. Вариант использования завершается успешно.

Альтернативы:

А1: Пользователь выбирает кнопку «Свой размер».

А1.1 Поля ввода размеров топологии «Горизонтальный» и «Вертикальный» становятся активными, в них указаны размеры для средних размеров топологии.

А1.1.А1 Пользователь вводит размер по горизонтали в соответствующее поле.

А1.1.А2 Пользователь нажимает на  для поля «Горизонтальный».

А1.1.А2.1 В поле ввода «Горизонтальный» значение увеличивается на 1.

А1.1.А2.2 Поле ввода «Горизонтальный» не изменяется, так как в поле указан максимальный размер по горизонтали.

А1.1.А3 Пользователь нажимает на  для поля «Горизонтальный».

А1.1.А3.1 В поле ввода «Горизонтальный» значение уменьшается на 1.

А1.1.А3.2 Поле ввода «Горизонтальный» не изменяется, так как в поле указан минимальный размер по горизонтали.

A1.1.A4 Пользователь вводит размер по вертикали в соответствующее поле.

A1.1.A5 Пользователь нажимает на «» для поля «Вертикальный».

A1.1.A5.1 В поле ввода «Вертикальный» значение увеличивается на 1.

A1.1.A5.2 Поле ввода «Вертикальный» не изменяется, так как в поле указан максимальный размер по вертикали.

A1.1.A6 Пользователь нажимает на «» для поля «Вертикальный».

A1.1.A6.1 В поле ввода «Вертикальный» значение уменьшается на 1.

A1.1.A6.2 Поле ввода «Вертикальный» не изменяется, так как в поле указан минимальный размер по вертикали.

A1.2 Пользователь может выполнять пункты A1.1.A2, A1.1.A3, A1.1.A5, A1.1.A6, пока не установит необходимое значение соответствующих границ.

A1.3 Пользователь нажимает кнопку «Создать».

A1.4 Осуществляется переход к п.3 основного потока событий.

A2: Пользователь выбирает радиокнопку «Маленькая (10 x 7)».

A2.1 Становится активной радиокнопка «Маленькая (10 x 7)».

A2.2 Осуществляется переход к п.3 основного потока событий.

A3: Пользователь выбирает радиокнопку «Большая (30 x 25)».

A3.1 Становится активной радиокнопка «Большая (30 x 25)».

A3.2 Осуществляется переход к п.3 основного потока событий.

A4: Пользователь щелкнул левой кнопкой мыши по кнопке «».

A4.1 Система закрывает окно «Создание топологии» и выводит на экран форму «Начало работы».

Постусловия. При успешном завершении на экране – форма «Конструктор топологии», на которой будет отображаться топология с заданным размером, панель инструментов с ШЭ.

2.5.4 Диаграмма классов

Диаграммы классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем, они показывают набор классов, интерфейсов и коопераций, а также их связи. На практике диаграммы классов применяют для моделирования статического представления системы, они служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения [40].

На рисунке 28 приведена диаграмма классов системы (этап проектирования). В таблице 3 приведено описание классов.

2.5.5 Диаграмма состояний

Для моделирования поведения на логическом уровне в языке UML могут использоваться сразу несколько канонических диаграмм: состояний, деятельности, последовательности и кооперации, каждая из которых фиксирует внимание на отдельном аспекте функционирования системы. В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта [41].

Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

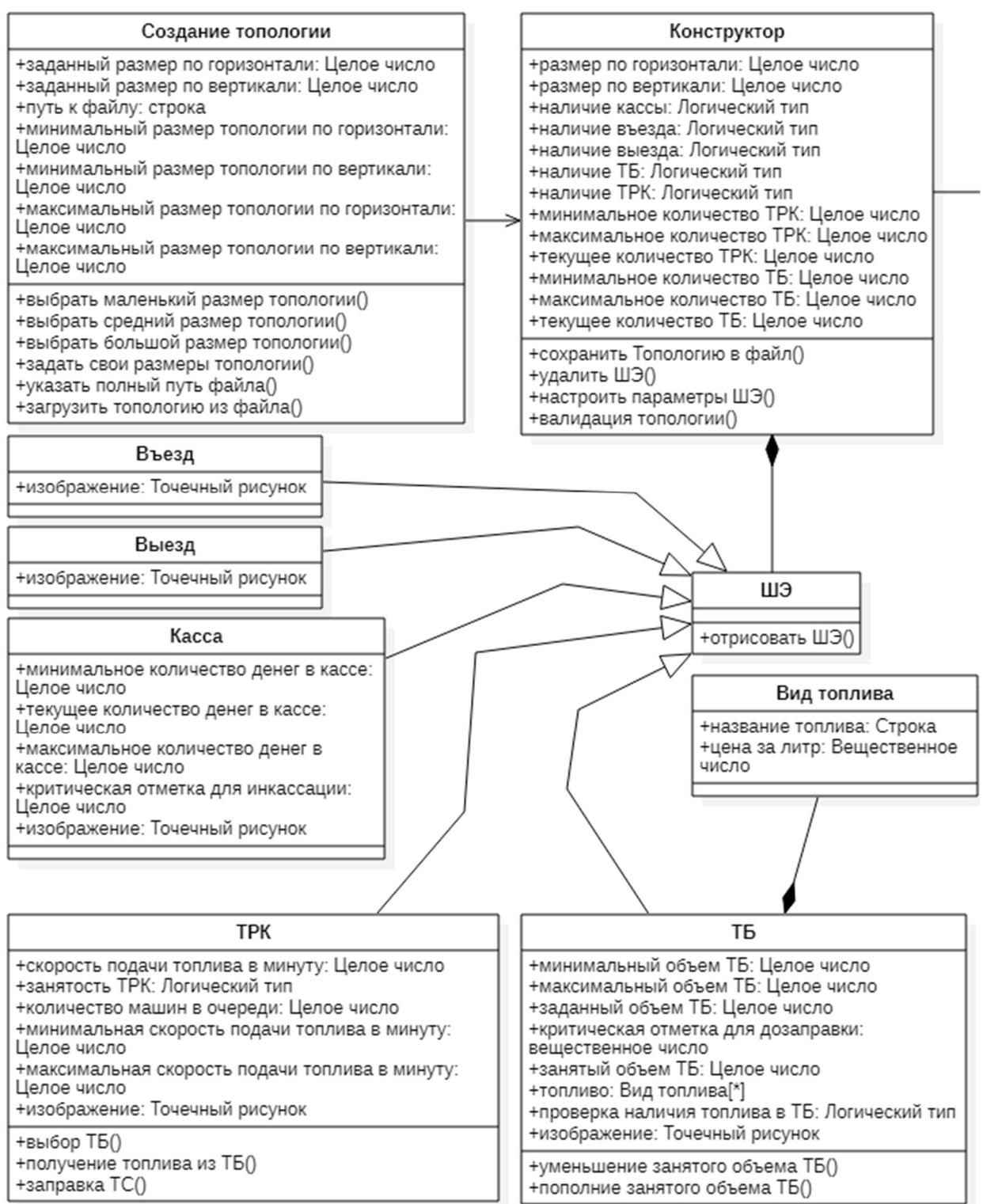


Рисунок 28 – Диаграмма классов системы (начало)

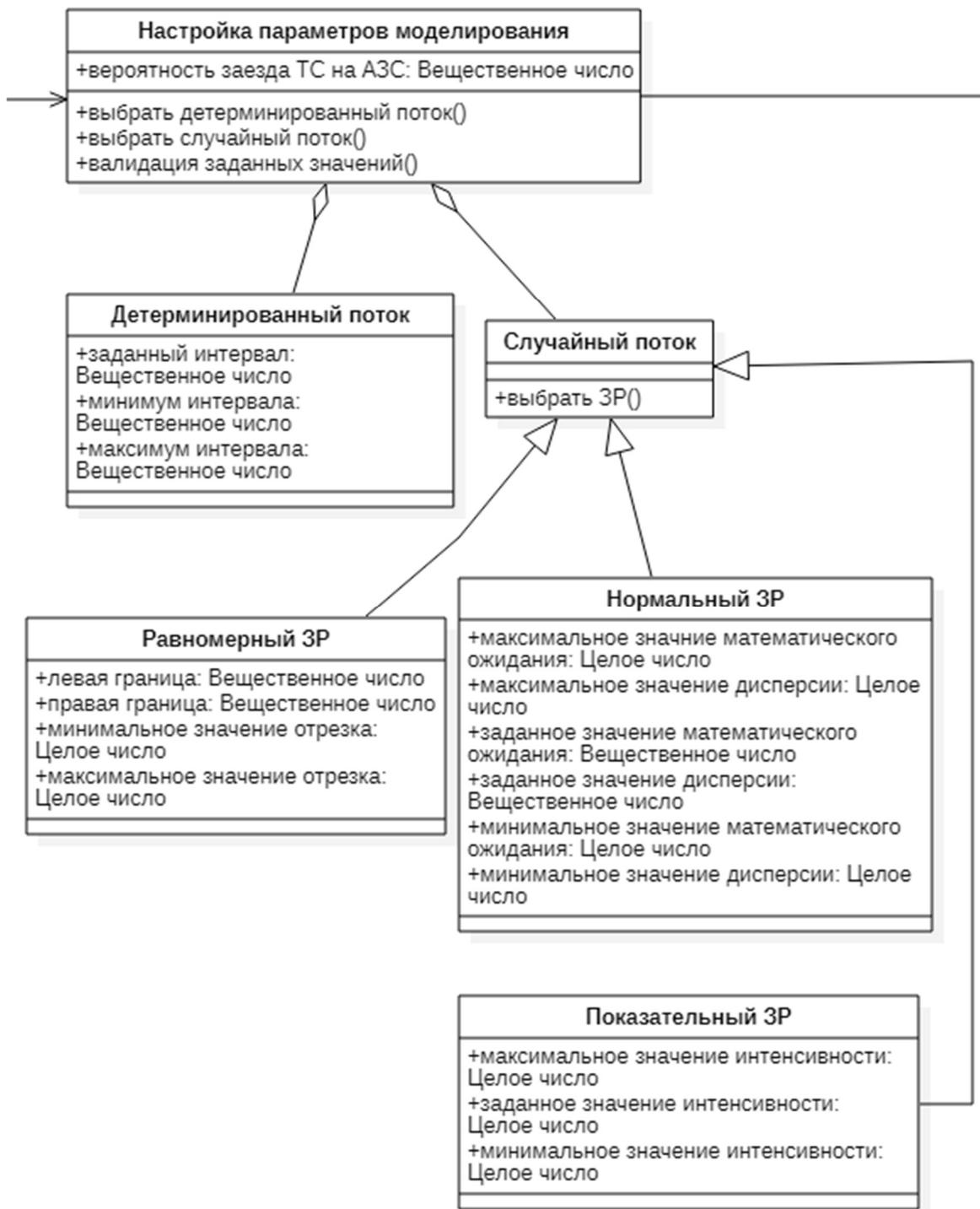


Рисунок 28 – Диаграмма классов системы (продолжение)

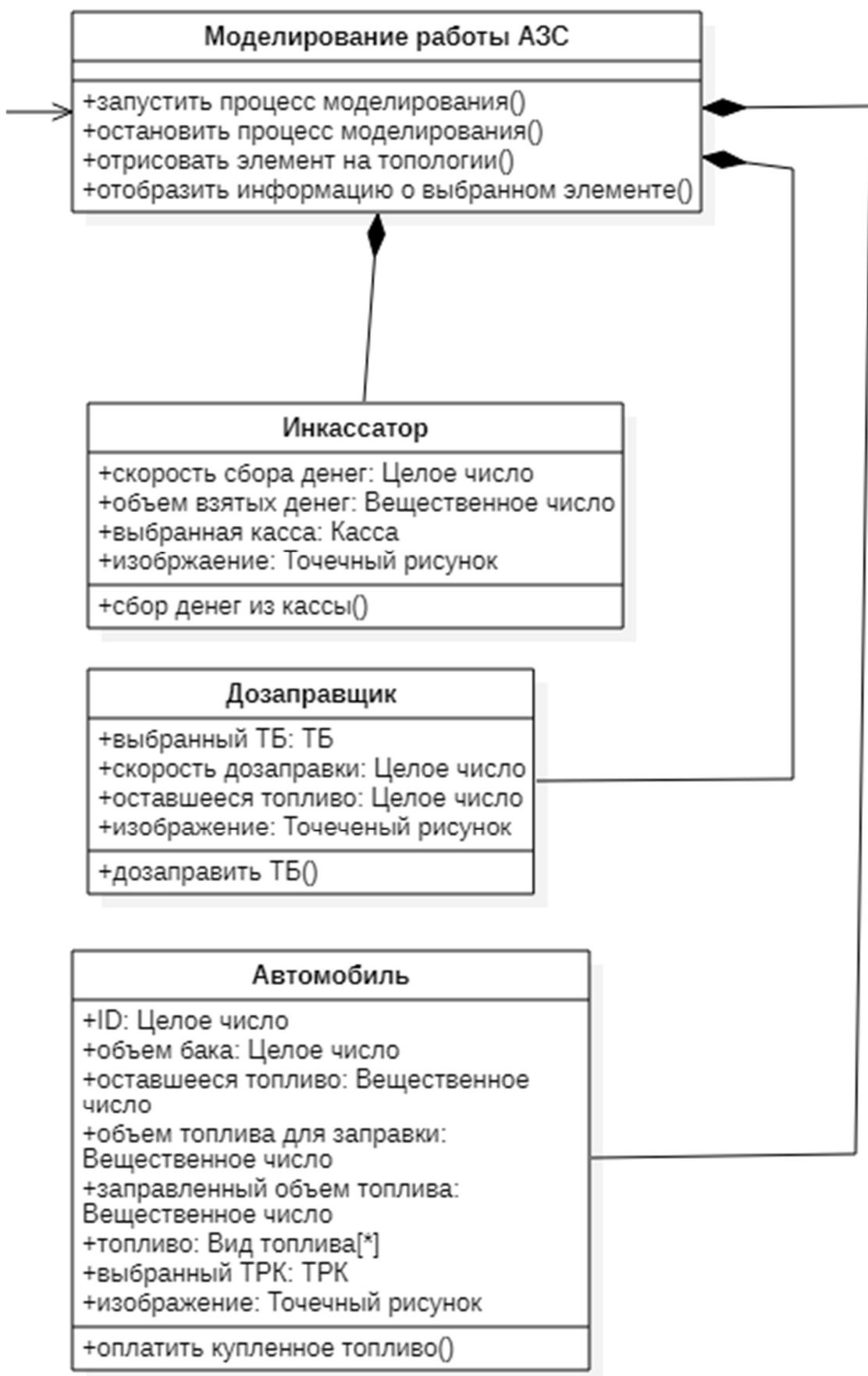


Рисунок 28 – Диаграмма классов системы (окончание)

Таблица 3 – Описание классов системы

Название класса	Назначение
1	2
Создание топологии	Класс необходим для задания размеров топологии и расположения файла с топологией.
Конструктор	Отвечает за создание топологии с помощью ШЭ.
ШЭ	Абстрактный класс, который содержит в себе различные типы ШЭ и их отрисовку.
Выезд	Класс, который содержит в себе координаты выезда.
Въезд	Класс, который содержит в себе координаты въезда.
Касса	Класс, который содержит лимит кассы, количество денег в кассе, ограничения на цены топлива и точечное изображение кассы.
TPK	Класс, который содержит ограничения на скорость подачи топлива, заданную скорость, взаимодействия в процессе моделирования и точечное изображение ТPK.
ТБ	Класс, который содержит ограничения на объем ТБ, выбранный вид топлива, остаток топлива в ТБ и точечное изображение ТБ.
Вид топлива	Экземпляр данного класса представляет из себя виды топлива.

Продолжение таблицы 3

1	2
Настройка параметров моделирования	Ключевой класс системы. Отвечает на настройку транспортного потока.
Детерминированный поток	Класс для задания параметров детерминированного потока.
Случайный поток	Абстрактный класс, который является одним из выбранных потоков. Содержит в себе различные ЗР.
Равномерный ЗР	Отвечает за задание параметров равномерного ЗР.
Нормальный ЗР	Отвечает за задание параметров нормального ЗР.
Показательный ЗР	Отвечает за задание параметров показательного ЗР.
Моделирование работы АЗС	Ключевой класс системы. Отвечает за процессы моделирования и отображение элементов.
Автомобиль	Класс для задания параметров автомобиля.
Дозаправщик	Класс для задания параметров дозаправщика.
Инкассатор	Класс для задания параметров инкасации.

Диаграмма состояний по существу является ориентированным графом специального вида, который представляет некоторый автомат. Понятие автомата в контексте UML обладает довольно специфической семантикой, основанной на теории автоматов. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния),

которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние [41].

На рисунках 29 и 30 приведены диаграммы состояний приложения. При запуске приложения система отображает окно «Начало работы». После выбора топологии, пользователь может осуществлять работу с ШЭ и настраивать их. После настройки топологии и ШЭ, пользователь настраивает параметры моделирования. После настроек параметром моделирования, система отображает окно «Моделирование». Система ожидает действий от пользователя: при нажатии на панель управления, на соответствующую кнопку, произойдет запуск системы моделирования или система возьмет паузу, ускорение или замедление системы работы моделирования АЗС; при нажатии на ШЭ на топологии, система покажет информацию о выбранном ШЭ. При нажатии на кнопку «Завершить» происходит завершение работы приложения.

2.5.6 Диаграмма деятельности

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на этих диаграммах также присутствуют обозначения состояний и переходов.

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние выполняется только при завершении этой операции [42].

На рисунках 31 и 32 приведена диаграмма деятельности конструирования топологии. Система загружает выбранную топологию, чтобы пользователь мог осуществлять работу в ней. Пользователю доступны следующие функции: изменение размеров топологии, размещение и настройках ШЭ.



Рисунок 29 – Диаграмма состояний системы (общая часть)

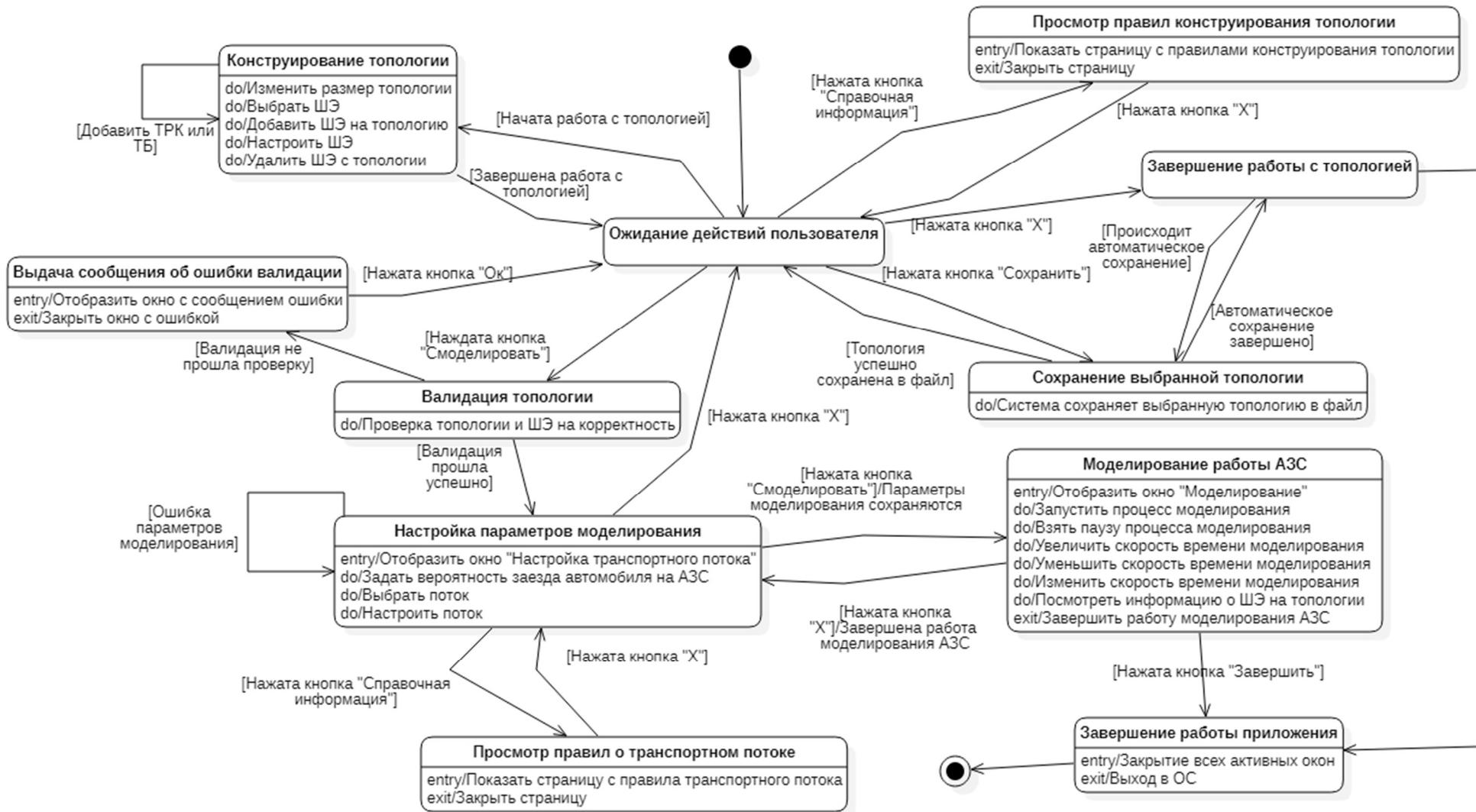


Рисунок 30 – Диаграмма состояний «Работа с выбранной топологией»

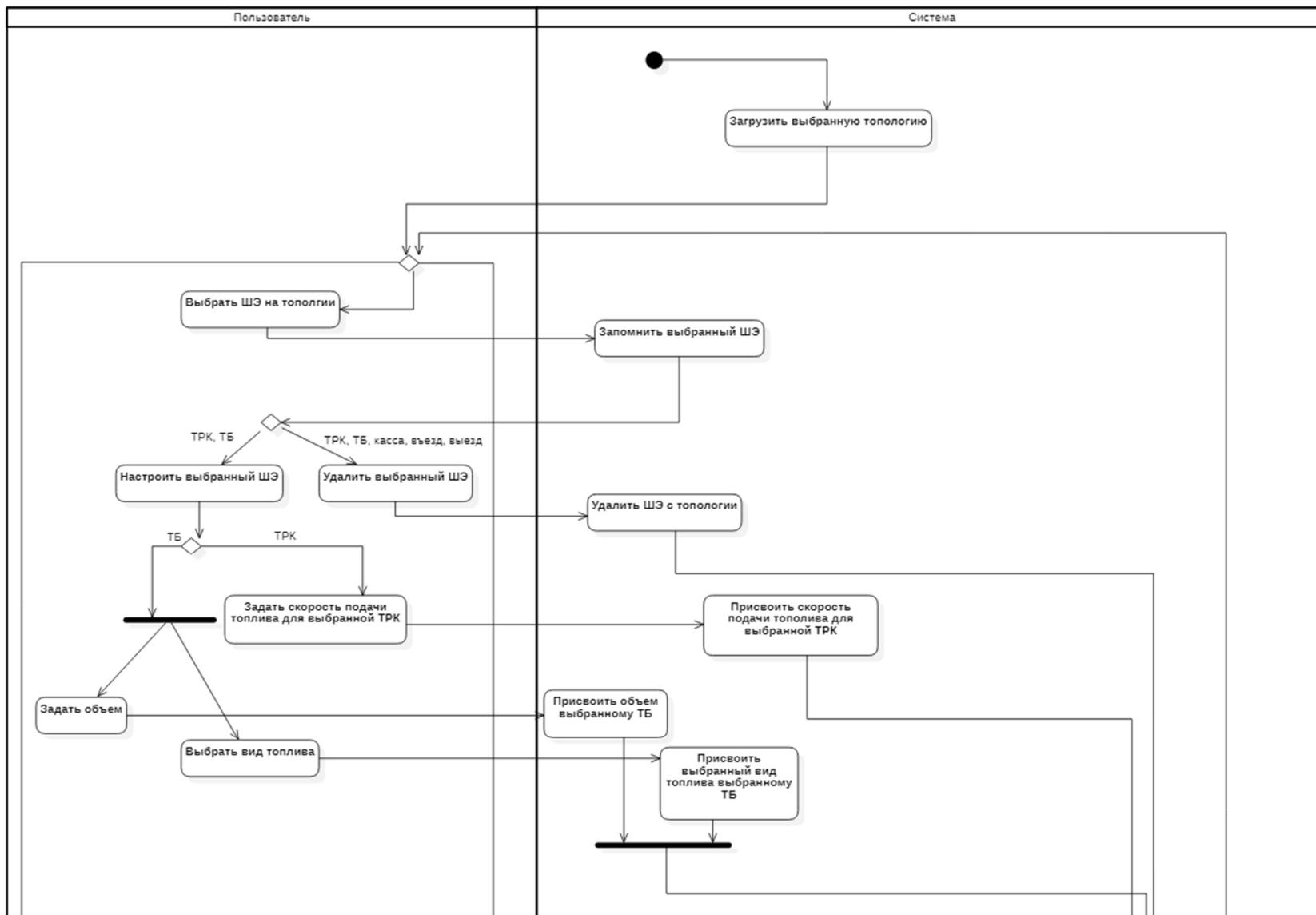


Рисунок 31 – Диаграмма деятельности для конструирования топологии (начало)

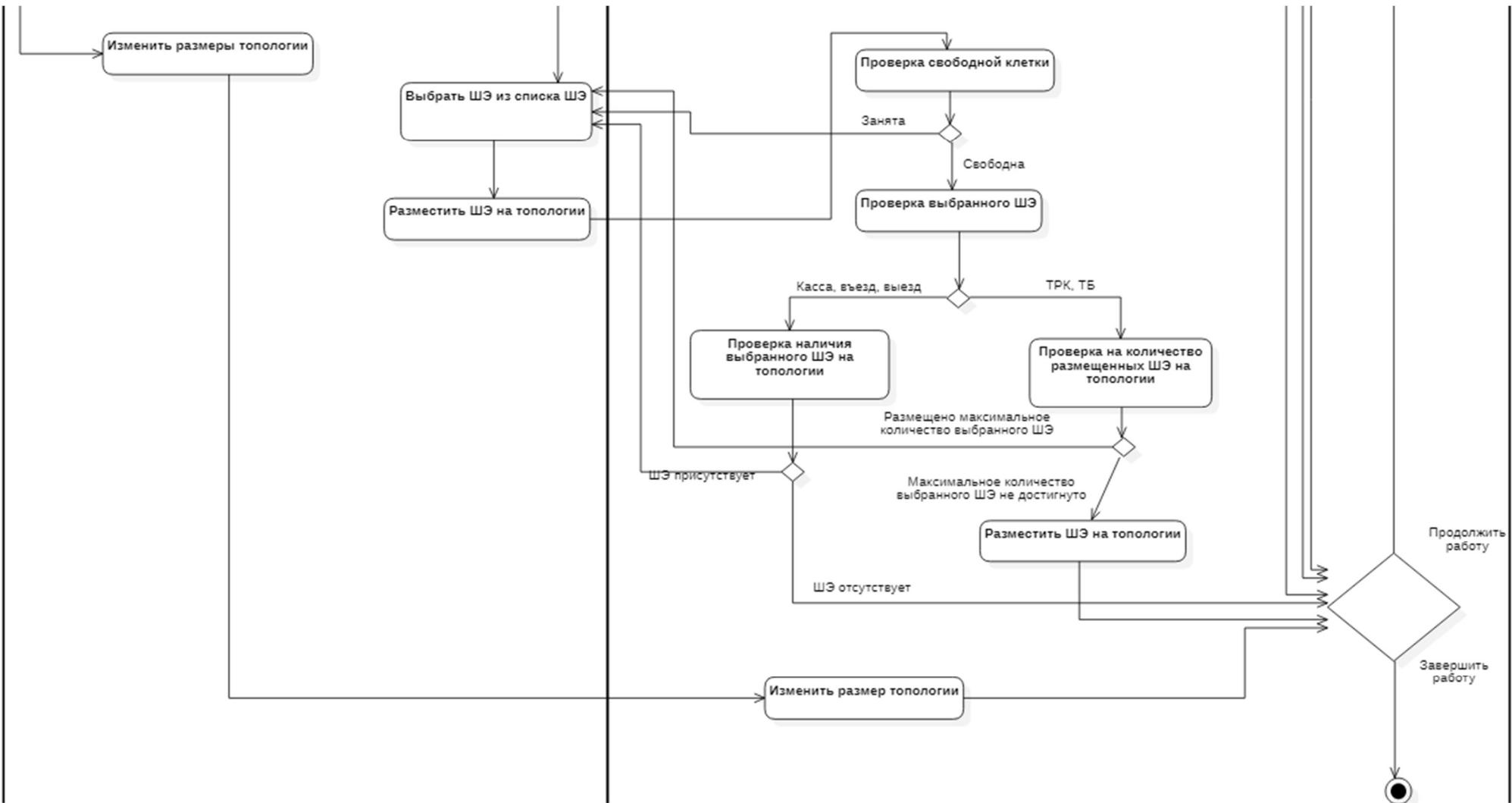


Рисунок 31 – Диаграмма деятельности для конструирования топологии (окончание)

Для изменений размеров топологии, пользователь может менять размер по горизонтали и вертикали в соответствующих полях.

Для размещения ШЭ на топологии, пользователь выбирает из списка ШЭ желаемый ШЭ и размещает его на топологии. Существуют ограничения на количество размещённых ШЭ на топологии (ТРК, ТБ, касса, въезд, выезд), система следит за тем, чтобы на топологии размещалось их допустимое количество.

Для настройки ШЭ, пользователь выбирает желаемый ШЭ. В зависимости от выбранного ШЭ, система отображает на экране область, где пользователь может настроить параметры ШЭ.

В завершении процесса пользователь нажимает на кнопку «Смоделировать», после чего осуществляется переход в другое состояние.

На рисунке 32 приведена диаграмма деятельности для процесса моделирования АЗС. Система загружает заданные ранее параметры моделирования. Пользователь запускает процесс моделирования АЗС. Также он имеет следующие возможности: изменить процесс моделирования (взять паузу, изменить масштаб времени, запустить процесс моделирования) и посмотреть информацию о выбранном элементе на топологии в момент моделирования.

В завершении процесса пользователь нажимает на кнопку «Завершить», после чего система останавливает процесс моделирования и осуществляет переход в следующее состояние.

2.5.7 Диаграмма последовательности

Диаграмма последовательности – диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

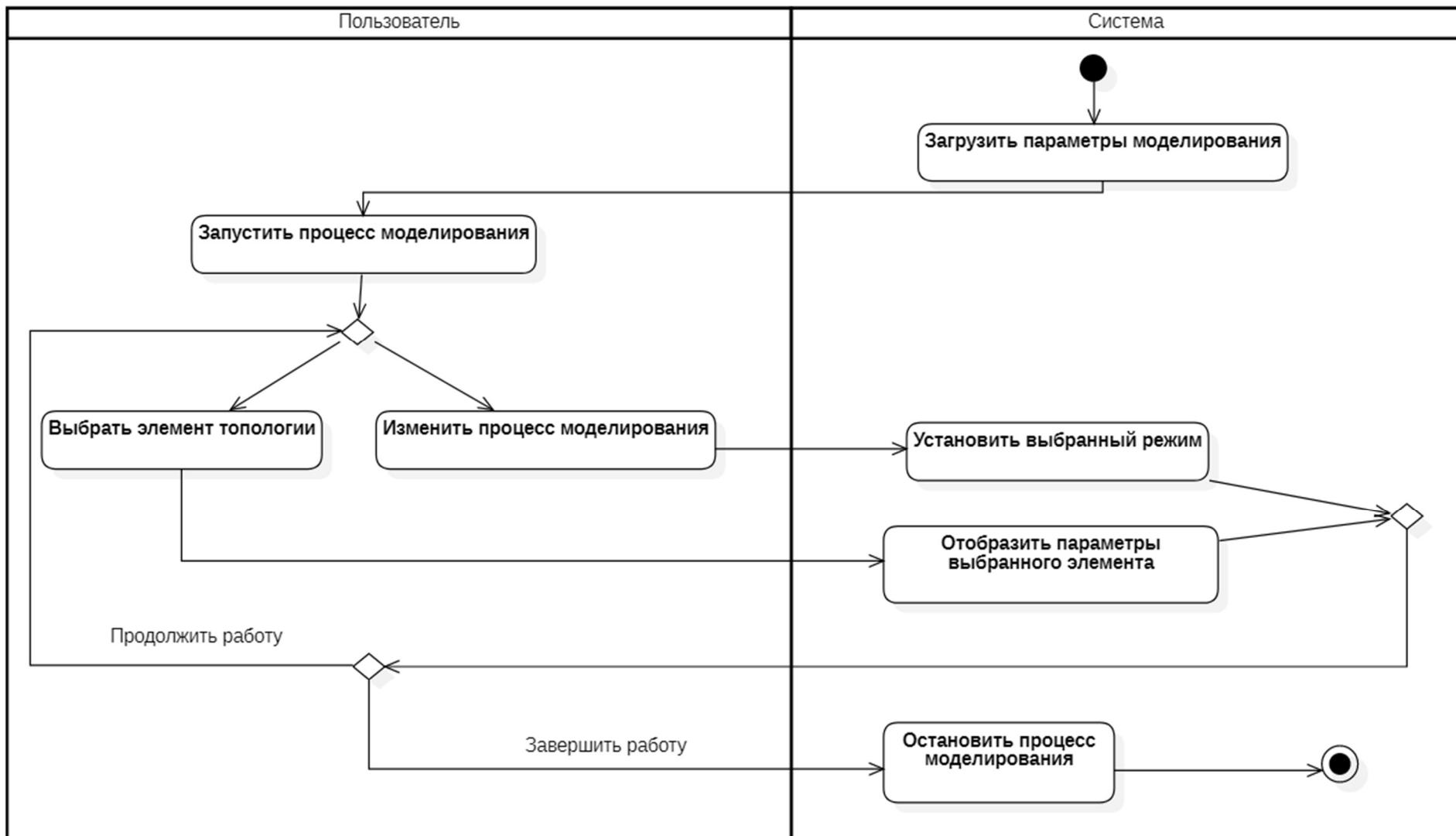


Рисунок 32 – Диаграмма деятельности для процесса моделирования АЗС

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами [43].

На рисунке 33 представлена диаграмма последовательности для сценария «Задать параметры равномерного ЗР».

На рисунке 34 представлена диаграмма последовательности для сценария «Разместить ТРК».

На рисунке 35 представлена диаграмма последовательности для сценария «Добавить вид топлива».

На рисунке 36 представлена диаграмма последовательности для сценария «Указать размер топологии».

2.6 Разработка и описание волнового алгоритма для определения кратчайшего маршрута на плоскости

Задача моделирования маршрутов (трассировки) – одна из наиболее трудоемких задач в общей проблеме автоматизации проектирования. Это связано с несколькими факторами, в частности с многообразием способов конструктивно-технологической реализации различных трасс, для каждого из которых при алгоритмическом решении задачи применяются специфические критерии оптимизации и ограничения. С математической точки зрения трассировка – наиложнейшая задача выбора оптимального решения из огромного числа вариантов.

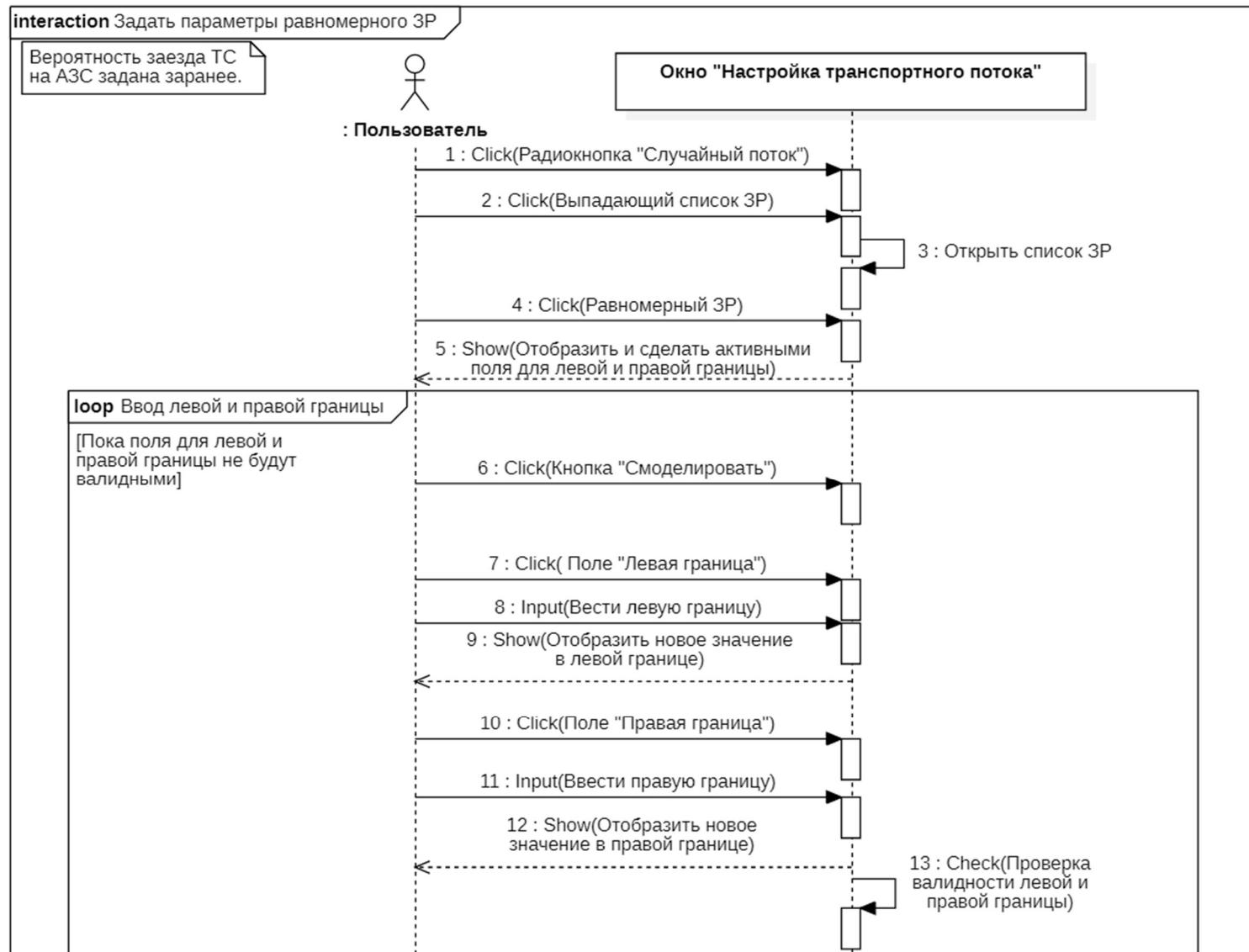


Рисунок 33 – Диаграмма последовательности «Задать параметры равномерного ЗР» (начало)

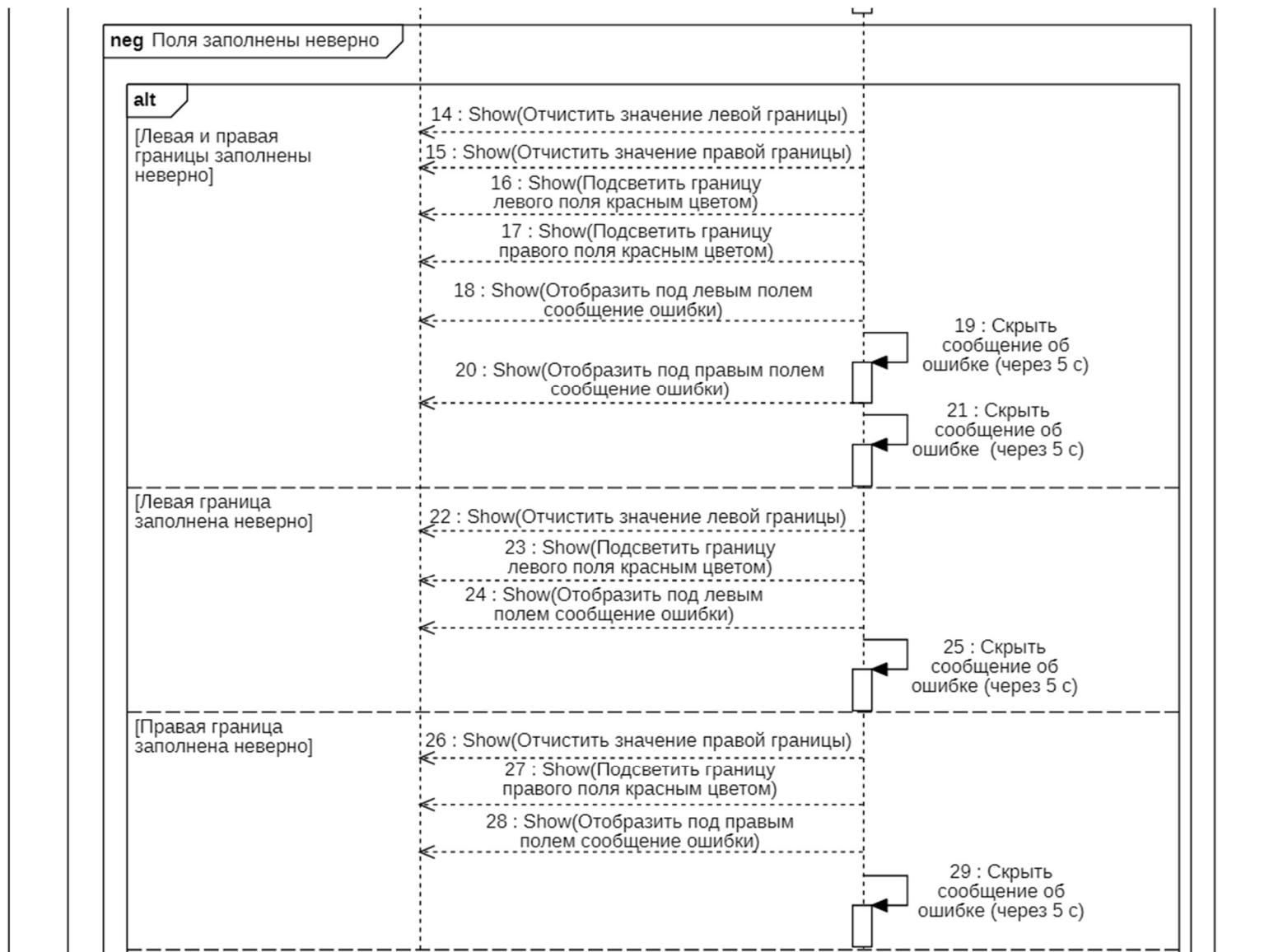


Рисунок 33 – Диаграмма последовательности «Задать параметры равномерного ЗР» (продолжение)

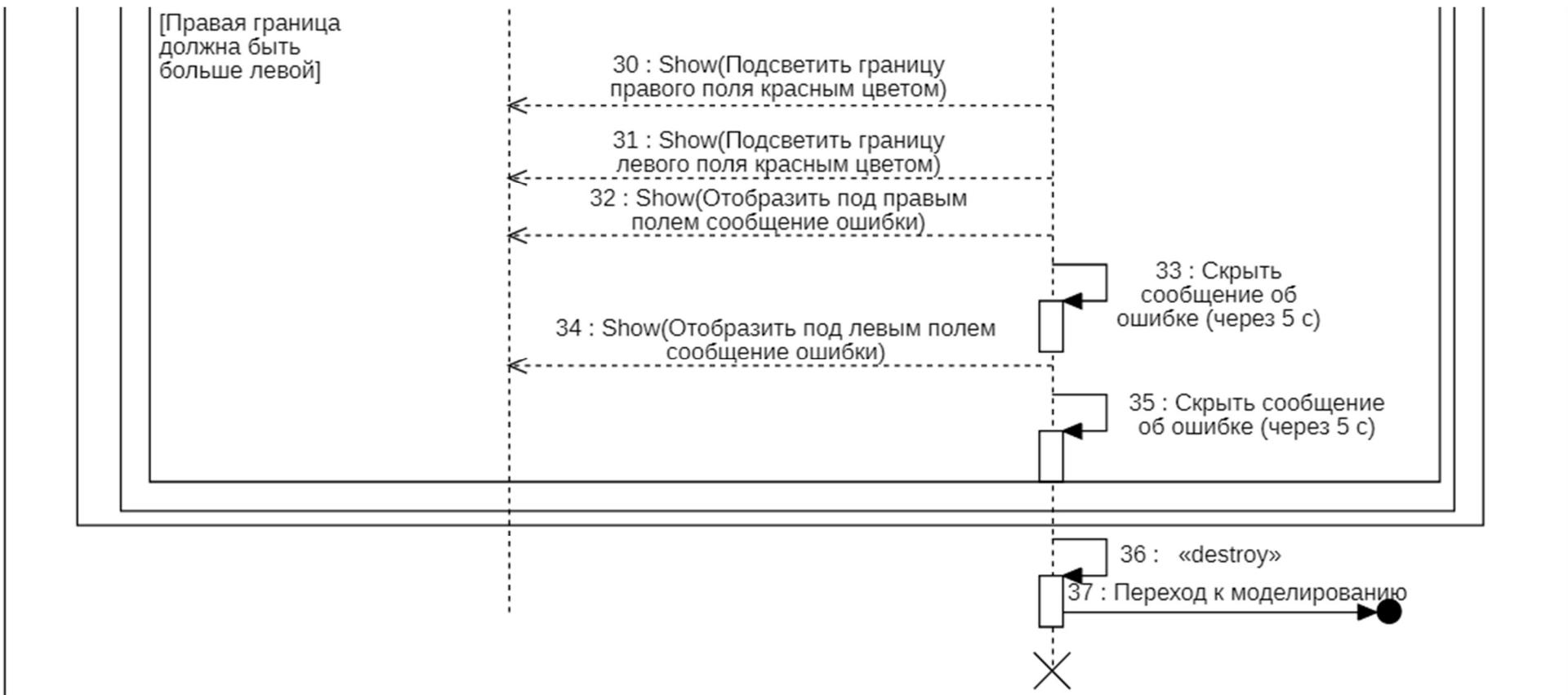


Рисунок 33 – Диаграмма последовательности «Задать параметры равномерного ЗР» (окончание)

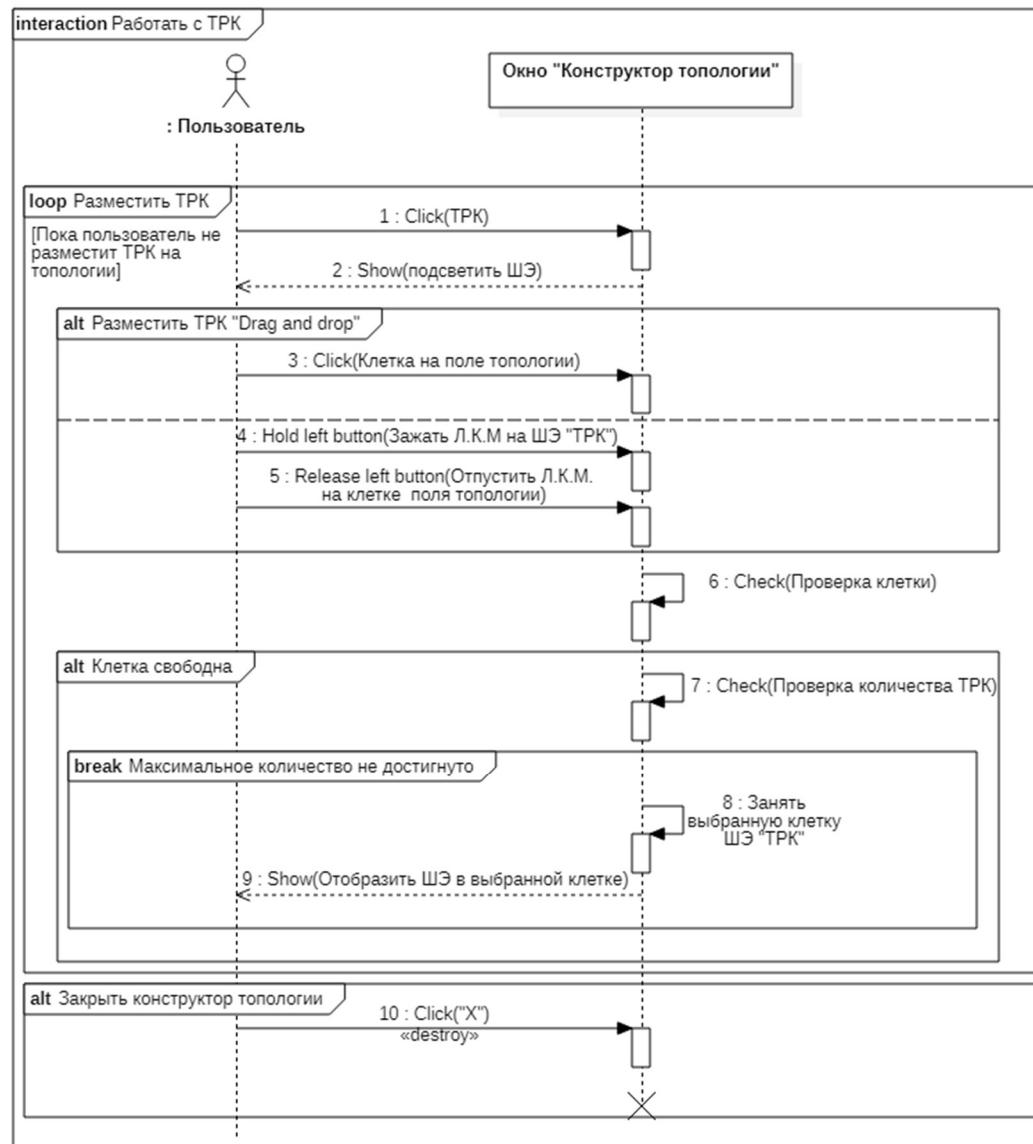


Рисунок 34 – Диаграмма последовательности «Разместить ТРК»

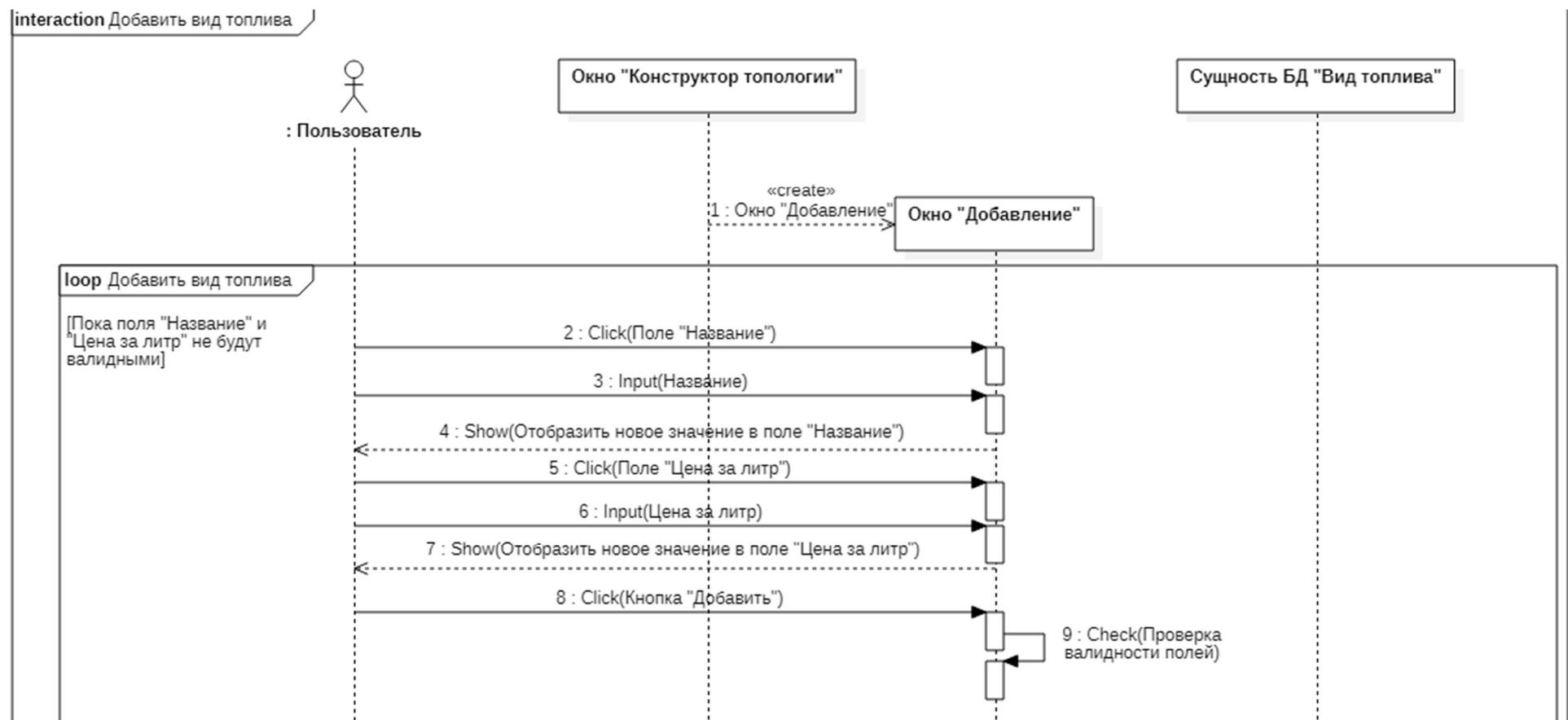


Рисунок 35 – Диаграмма последовательности «Добавить вид топлива» (начало)

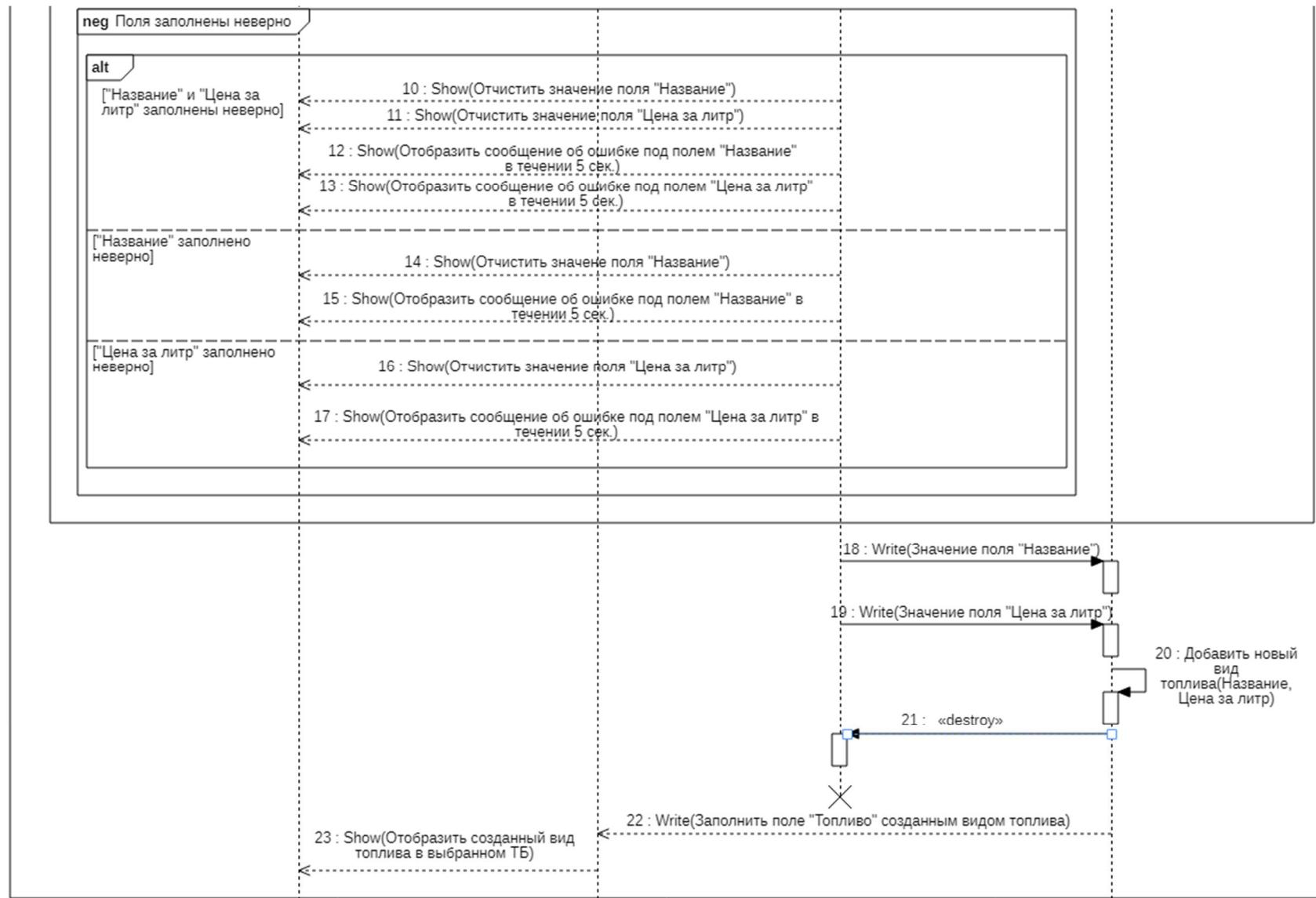


Рисунок 35 – Диаграмма последовательности «Добавить вид топлива» (окончание)

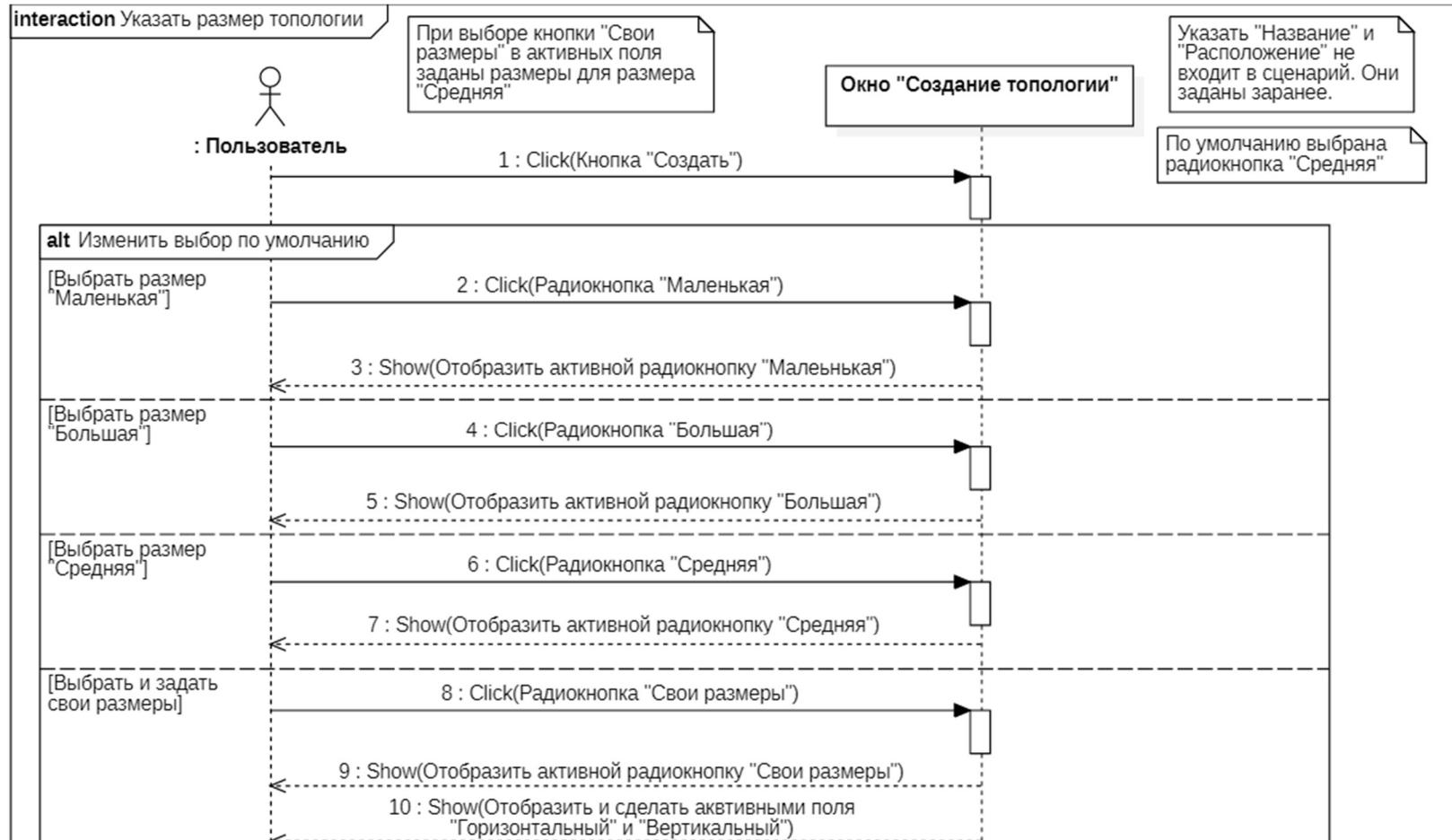


Рисунок 36 – Диаграмма последовательности «Указать размеры топологии» (начало)

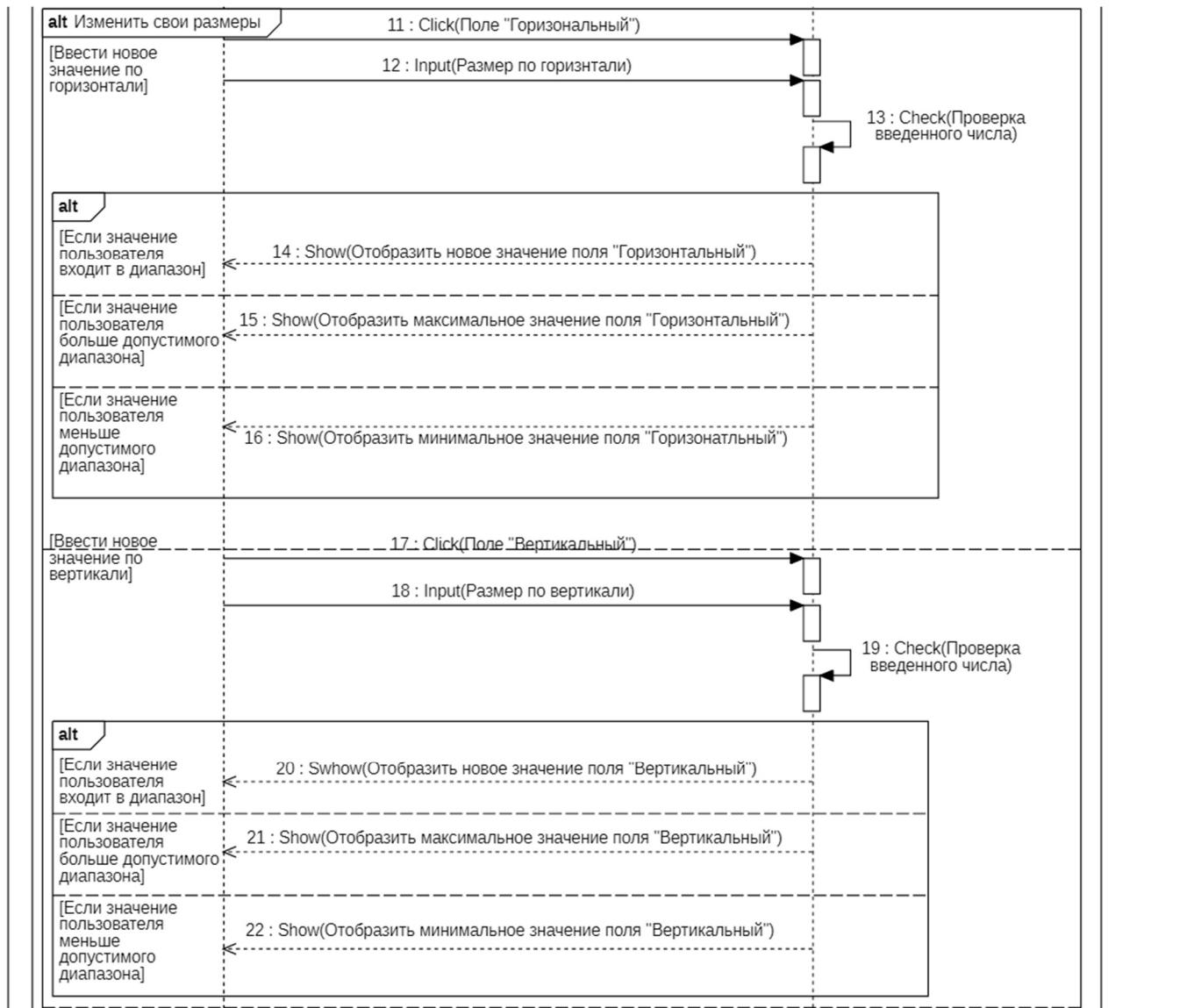


Рисунок 36 – Диаграмма последовательности «Указать размеры топологии» (продолжение)

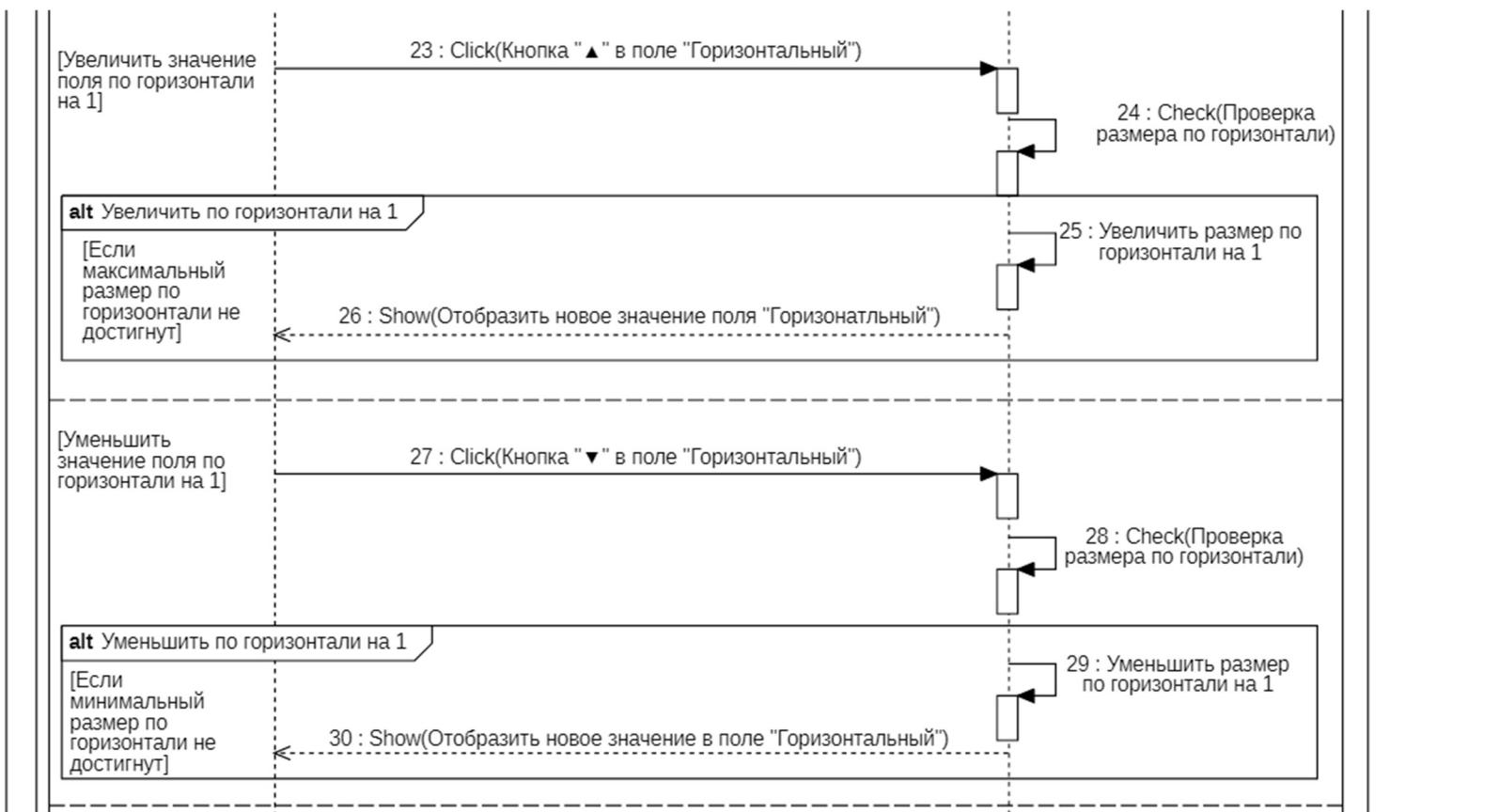


Рисунок 36 – Диаграмма последовательности «Указать размеры топологии» (продолжение)

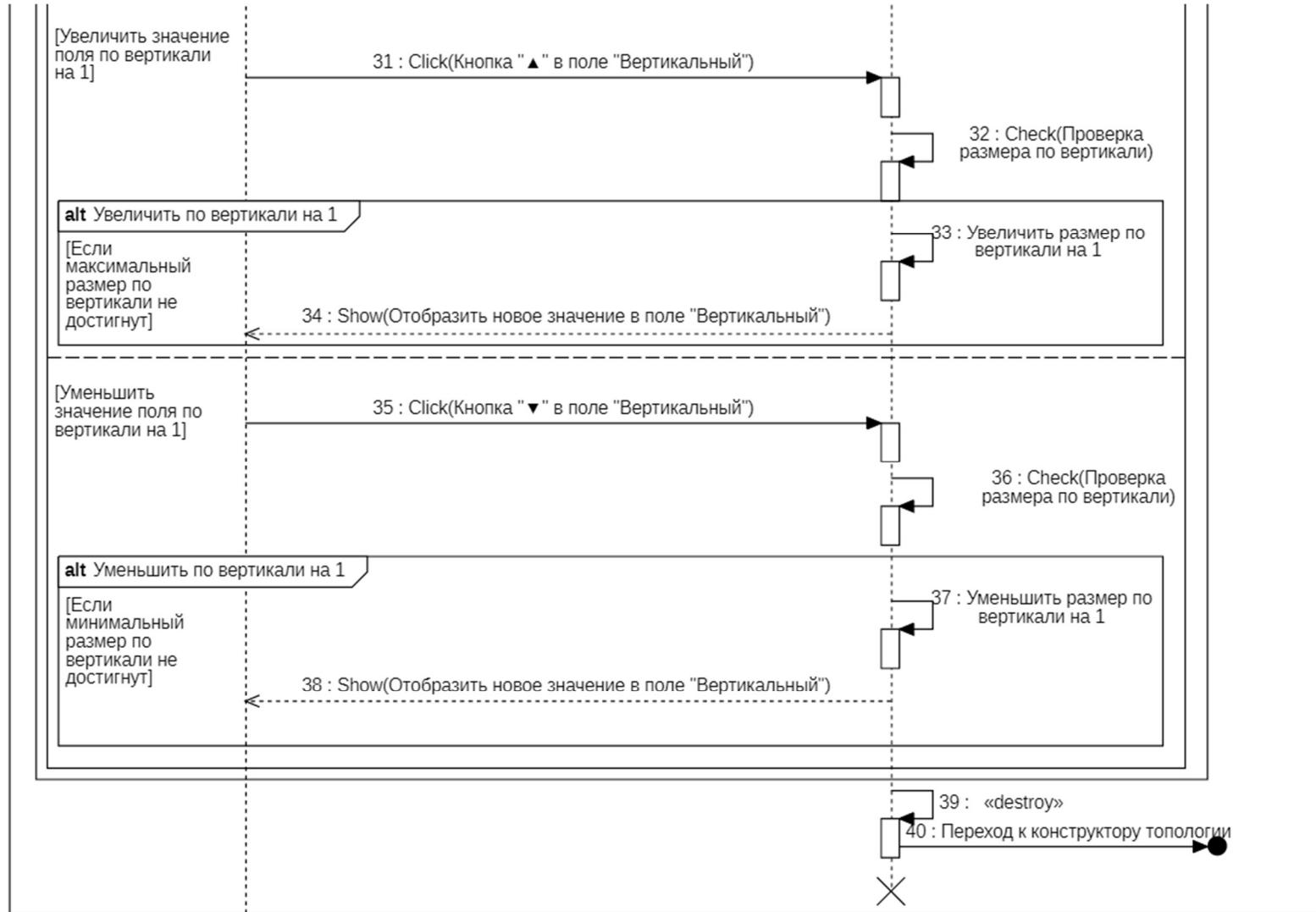


Рисунок 36 – Диаграмма последовательности «Указать размеры топологии» (окончание)

2.6.1 Постановка задачи

На двумерной клетчатой карте (матрице), состоящей из «проходимых» и «непроходимых» клеток, обозначена клетка старта и клетка финиша. Цель алгоритма – проложить кратчайший путь от клетки старта к клетке финиша, если это возможно. От старта во все направления распространяется волна, причем каждая пройденная волной клетка помечается как «пройденная». Волна, в свою очередь, не может проходить через клетки, помеченные как «пройденные» или «непроходимые». Волна движется, пока не достигнет точки финиша или пока не останется клеток, которых волна не прошла. Если волна прошла все доступные клетки, но так и не достигла клетки финиша, значит путь от старта до финиша проложить невозможно. После достижения волной финиша прокладывается путь и сохраняется в массиве (по координатам пройденных клеток на плоскости).

Волновой метод позволяет осуществлять поиск пути по четырем (алгоритм Ли) либо по восьми направлениям. Вторая разновидность позволяет найти более короткий путь с наименьшими потерями, если взять критерий минимальной длины проводника за критерий трассировки. Исходя из этого выбираем вторую разновидность метода. На рисунке 37 представлены разновидности волнового алгоритма.

Введем допущение – клетки исходной матрицы зададим квадратными. Так как путь по диагонали длиннее пути по горизонтали или вертикали, то необходимо чтобы волна задерживалась на этом отрезке дольше, для этого в алгоритм вводятся коэффициенты, показывающие, насколько дольше волна проходит участок трассы. Исходя из известного соотношения сторон квадрата и его диагонали, введем коэффициент, равный $2/3$.

	$i+2$	
$i+2$	i	$i+2$
	$i+2$	

	$i+3$	$i+2$	$i+3$
$i+2$	i	$i+2$	
$i+3$	$i+2$	$i+3$	

Рисунок 37 – Разновидности волнового алгоритма

2.6.2 Описание метода

В нашем случае из конечного элемента волна распространяется в 8 направлениях. Элемент, в который пришла волна, образует фронт волны, изображенной на рисунке 38. На рисунках цифрами обозначены номера фронтов волны.

3	2	3
2	A	2
3	2	3

Рисунок 38 – Первый фронт волны

Каждый элемент первого фронта волны является источником вторичной волны, изображенной на рисунке 39. Элементы второго фронта волны генерируют волну третьего фронта и т. д. Процесс продолжается до тех пор, пока не будет достигнут начальный элемент.

6	5	4	5	6
5	3	2	3	5
4	2	A	2	4
5	3	2	3	5
6	5	4	5	6

Рисунок 39 – Третий фронт волны

На втором этапе строится сама трасса. Её построение осуществляется в соответствии со следующими правилами:

- движение при построении трассы осуществляется в соответствии с выбранными приоритетами;
- при движении от начального элемента к конечному элементу номера фронта волны должны уменьшаться.

На рисунке 40 приведен пример использования волнового алгоритма при прохождении лабиринта.

Красным цветом отмечены запрещенные элементы. Серым цветом – трасса после действия алгоритма. А – начальная точка (старт), В – конечная точка (финиш). Приоритеты движения влево, вправо, вверх, вниз. Построение трассы ведется от начальной точки к конечной точке. Приоритетные направления показаны стрелками.

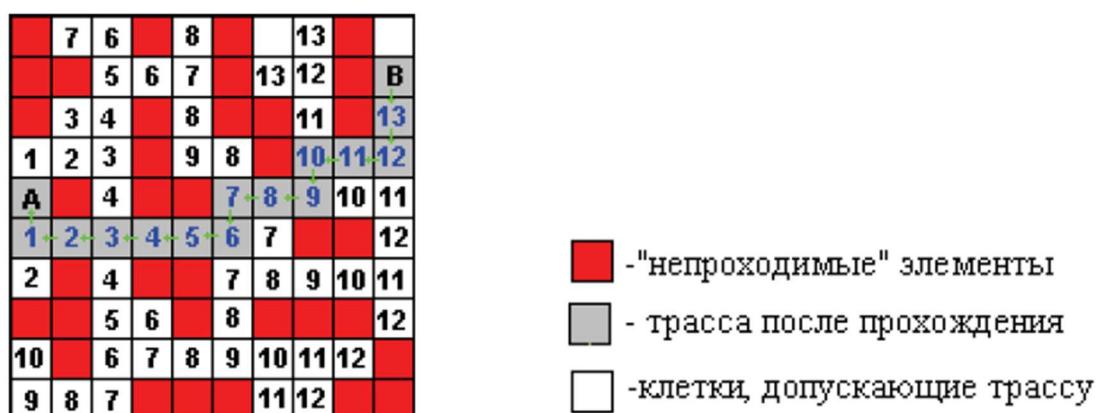


Рисунок 40 – Пример использования алгоритма

2.6.3 Реализация алгоритма метода

Исходя из сути волнового метода, был построен алгоритм, реализация которого была разбита на две процедуры, одна из которых реализует расчеты коэффициентов фронта волны и различие препятствий, другая – проведение пути по кратчайшему маршруту.

Этап расчета коэффициентов фронта волны и различие препятствий:

1. Сначала необходимо создать рабочий массив $R(M \times N)$, равный по размеру массиву игрового поля $P(M \times N)$.

2. Каждому элементу рабочего массива $R(i, j)$ присваивается некоторое значение в зависимости от свойств элемента игрового поля $P(i, j)$ по следующим правилам:

- a) если поле $P(i, j)$ непроходимо, то $R(i, j):=MxN+2;$
- b) если поле $P(i, j)$ проходимо, то $R(i, j):=MxN+1;$
- c) если поле $P(i, j)$ является целевой (финишной) позицией, то $R(i, j):=0;$
- d) если поле $P(i, j)$ является стартовой позицией, то $R(i, j):=MxN.$

3. Этап распространения волны. Вводим переменную N_i , счётчик итераций, и присваиваем ей начальное значение 0.

4. Вводим константу N_k , которую устанавливаем равной максимально возможному числу итераций.

5. Построчно просматриваем рабочий массив R .

6. Если $R(i, j)$ равен N_i , то просматриваются соседние элементы по следующему правилу (в качестве примера рассмотрим $R(i+1, j)$):

- a) если $R(i+1, j)$ непроходимо, то переходим к пункту 10;
- b) если $R(i+1, j)$ проходимо, выполняется присваивание $R(i+1, j):=N_i+1;$
- c) во всех остальных случаях $R(i+1, j)$ остается без изменений.

7. Аналогично поступаем с остальными элементами.

8. По завершению построчного просмотра всего массива увеличиваем N_i на 1.

9. Если $N_i > N_k$, то поиск маршрута признаётся неудачным. Выход из программы.

10. Переходим к пункту 5.

Этап построения маршрута.

1. Присваиваем переменным X и Y значения координат стартовой позиции.

2. В окрестности позиции $R(X, Y)$ ищем элемент с наименьшим значением (т. е. для этого просматриваем $R(X+1, Y)$, $R(X-1, Y)$, $R(X, Y+1)$, $R(X, Y-1)$). Координаты этого элемента заносим в переменные $X1$ и $Y1$.

3. Совершаем перемещение по игровому полю из позиции $[X, Y]$ в позицию $[X1, Y1]$.

4. Если $R(X1, Y1) = 0$, то переходим к пункту 5.

5. Выполняем присваивание $X:=X1$, $Y:=Y1$. Переходим к пункту 1.

На рисунке 41 представлена схема работы волнового алгоритма, где:

- массив Point – массив координат начальных и конечных точек;
- Way – процедура проведения пути;
- Nik – переменная, отвечающая за остановку алгоритма в случае, если путь невозможно продолжить при данных условиях.

Процедура «Way» отвечает в основном за графический вывод маршрута.

2.7 Выбор и обоснование комплекса программных средств

2.7.1 Выбор языка программирования и среды разработки

C# – простой, современный объектно-ориентированный и типобезопасный язык программирования. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript [44].

C# – является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности.

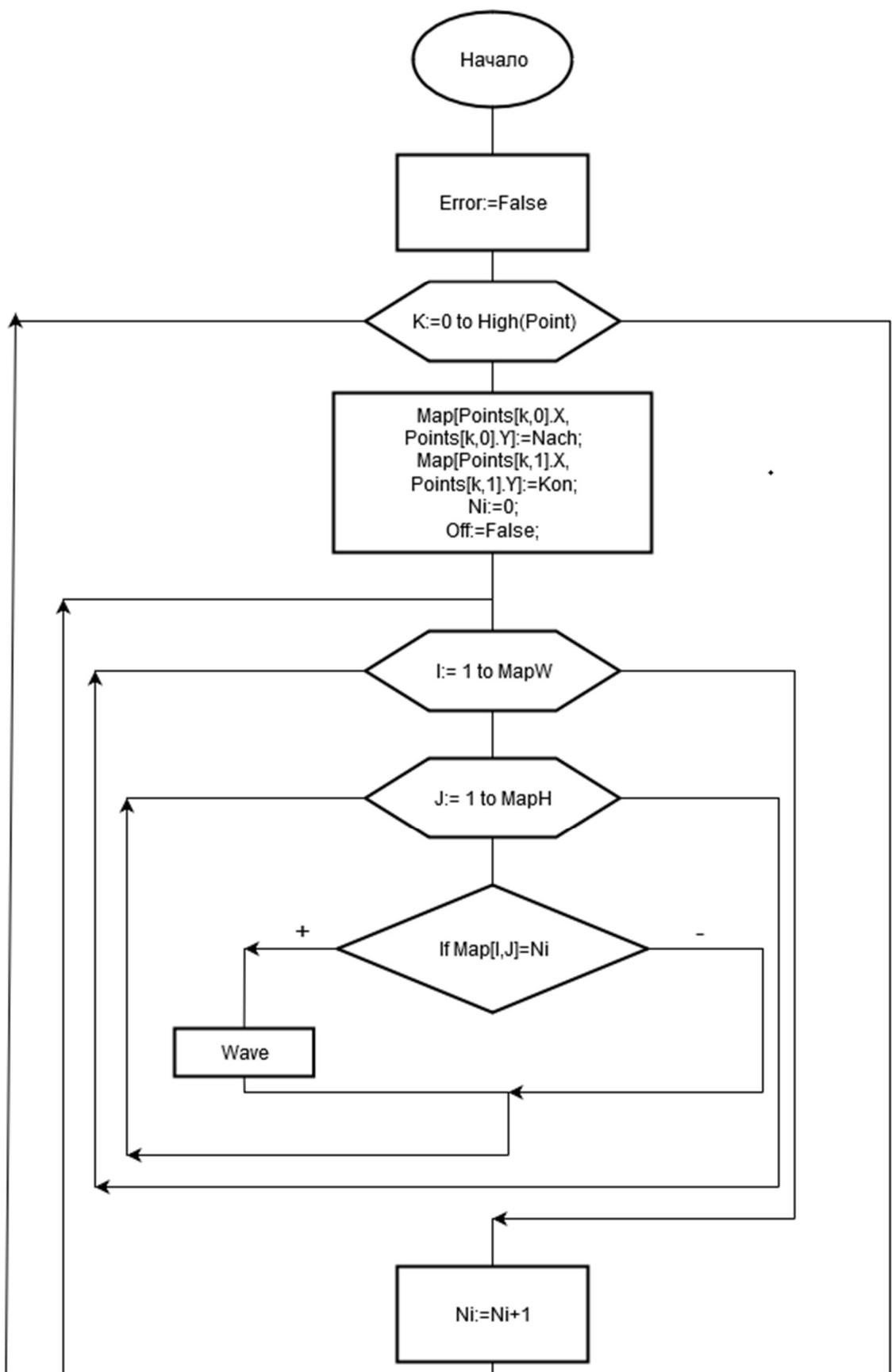


Рисунок 41 – Схема волнового алгоритма (начало)

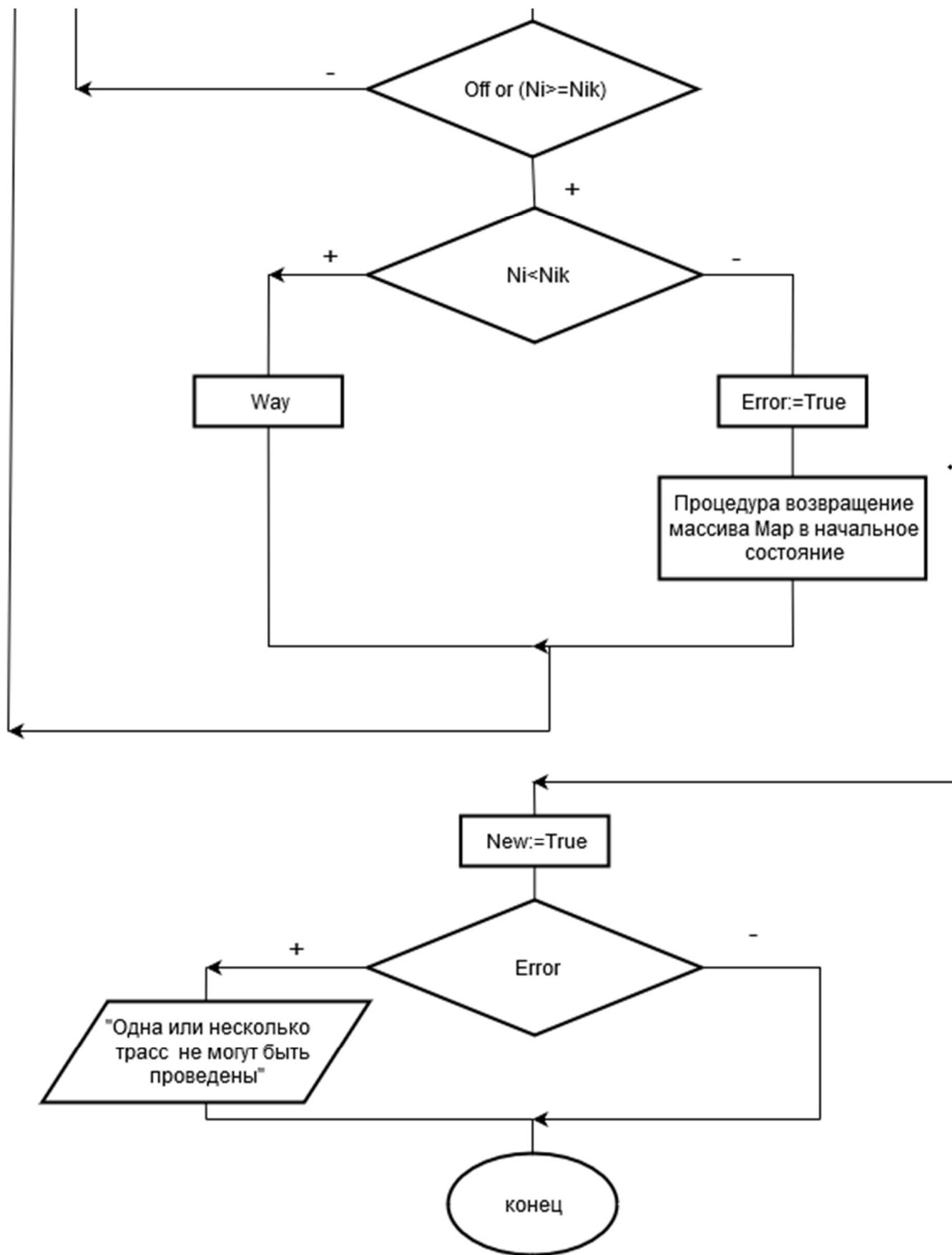


Рисунок 41 – Схема волнового алгоритма (окончание)

Важная особенность таких компонентов – это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# отлично подходит для создания и применения программных компонентов.

Вот лишь несколько функций языка C#, обеспечивающих надежность и устойчивость приложений:

- сборка мусора автоматически освобождает память, занятую уничтоженными и неиспользуемыми объектами;
- обработка исключений предоставляет структурированный и расширяемый способ выявлять и обрабатывать ошибки;
- строгая типизация языка не позволяет обращаться к неинициализированным переменным, выходить за пределы индексируемых массивов или выполнять неконтролируемое приведение типов.

В C# существует единая система типов. Все типы C#, включая типы-примитивы, такие как `int` и `double`, наследуют от одного корневого типа `object`. Таким образом, все типы используют общий набор операций, и значения любого типа можно хранить, передавать и обрабатывать схожим образом. Кроме того, C# поддерживает пользовательские ссылочные типы и типы значений, позволяя как динамически выделять память для объектов, так и хранить упрощенные структуры в стеке.

Чтобы обеспечить совместимость программ и библиотек C# при дальнейшем развитии, при разработке C# много внимания было уделено управлению версиями. Многие языки программирования обходят вниманием этот вопрос, и в результате программы на этих языках ломаются чаще, чем хотелось бы, при выходе новых версий зависимых библиотек. Вопросы управления версиями существенно повлияли на такие аспекты разработки

C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса [43].

2.7.2 Выбор операционной системы

Windows – семейство проприетарных операционных систем (ОС) корпорации Microsoft, ориентированных на применение графического интерфейса при управлении. Изначально Windows была всего лишь графической надстройкой-программой для операционной системы 80-х и 90-х годов MS-DOS [45].

По результатам очередного исследования ресурс Netmarketshare опубликовал данные о глобальном распространении настольных операционных систем в августе 2019 года. Представленные данные демонстрируют, что продукция корпорации Microsoft постепенно наращивает долю присутствия. В отчёте видно, что Windows 10 удалось захватить более 50 % мирового рынка впервые с момента запуска.

Что касается операционной системы Windows 10, то она за последний месяц прибавила порядка 2 %, а суммарная доля её присутствия на рынке составляет 50,99 %. Следом за ней идёт Windows 7, которая имеет долю 30,44 %, а на третьем месте с результатом 5,95 % расположилась macOS. Несмотря на то, что доля Windows 10 постепенно увеличивается, Microsoft всё ещё приходится подталкивать более 30 % пользователей к переходу на последнюю версию программной платформы. Один из главных аргументов сделать это быстрее заключается в том, что поддержка Windows 7 закончится 14 января 2020 года [46].

Таким образом, Windows является наиболее распространенной операционной системой, а также обладает удобным пользовательским интерфейсом и поддержкой многозадачности.

2.7.3 Выбор среды программирования

Интегрированная среда разработки Visual Studio – это оригинальная среда запуска, которая позволяет редактировать, отлаживать и создавать код, а затем публиковать приложения. Интегрированная среда разработки (IDE) — это многофункциональная программа, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства выполнения кода, графические конструкторы и многие другие функции для упрощения процесса разработки программного обеспечения [47].

Ниже перечислены некоторые популярные возможности Visual Studio, которые помогли нам повысить продуктивность разработки программного обеспечения:

- волнистые линии и быстрые действия;
- отчистка кода;
- рефакторинг;
- функции IntellSense;
- поле поиска;
- иерархия вызовов;
- CodeLens.

2.7.4 Выбор системы управления базами данных

MS SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов [48].

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных

представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных. Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

MS SQL Server характеризуется такими особенностями как:

- надежность;
- производительность;
- безопасность;
- простота;
- шифрование данных.

3 Реализация системы

3.1 Описание интерфейса пользователя

Разработанная система представляет собой приложение, которое реализуется с помощью Windows Forms.

3.1.1 Начало работы

При запуске системы отобразит окно, в котором пользователю будет предложено создать новую топологию или загрузить существующую топологию. На рисунке 42 представлена начальная экранная форма системы.

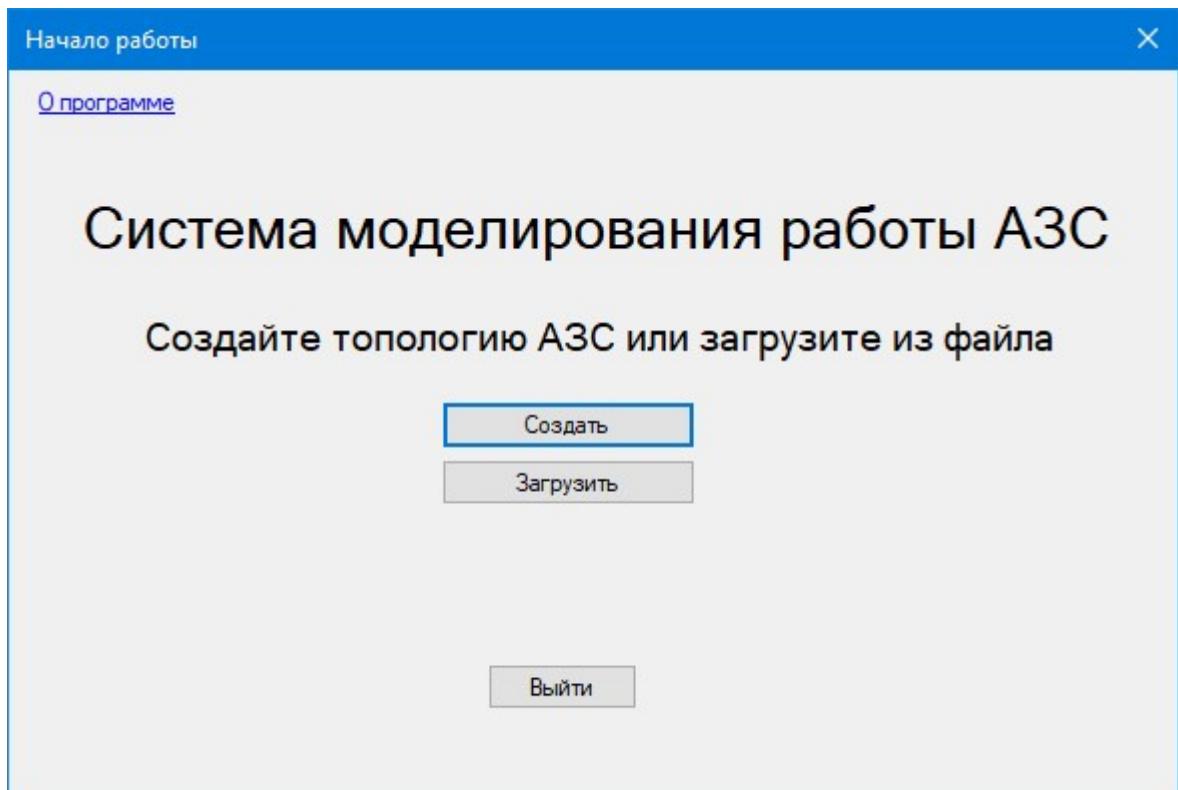


Рисунок 42 – Начальная экранная форма системы

При нажатии на гиперссылку «О программе» система откроет браузер по умолчанию, в котором отобразит информацию о возможностях программы. Информация о программе представлена в виде HTML-страницы.

При нажатии на кнопку «Загрузить» откроется диалоговое окно, в котором пользователь должен выбрать файл с существующей топологией.

Если выбранный файл будет поврежден, система сообщит об ошибке, выбранный файл не будет загружен.

При нажатии на кнопку «Создать» откроется экранная форма «Создание топологии», изображенная на рисунке 43, в которой пользователю будет предоставлена возможность создать топологию. Пользователь должен указать расположение файла топологии, нажав на кнопку «Обзор».

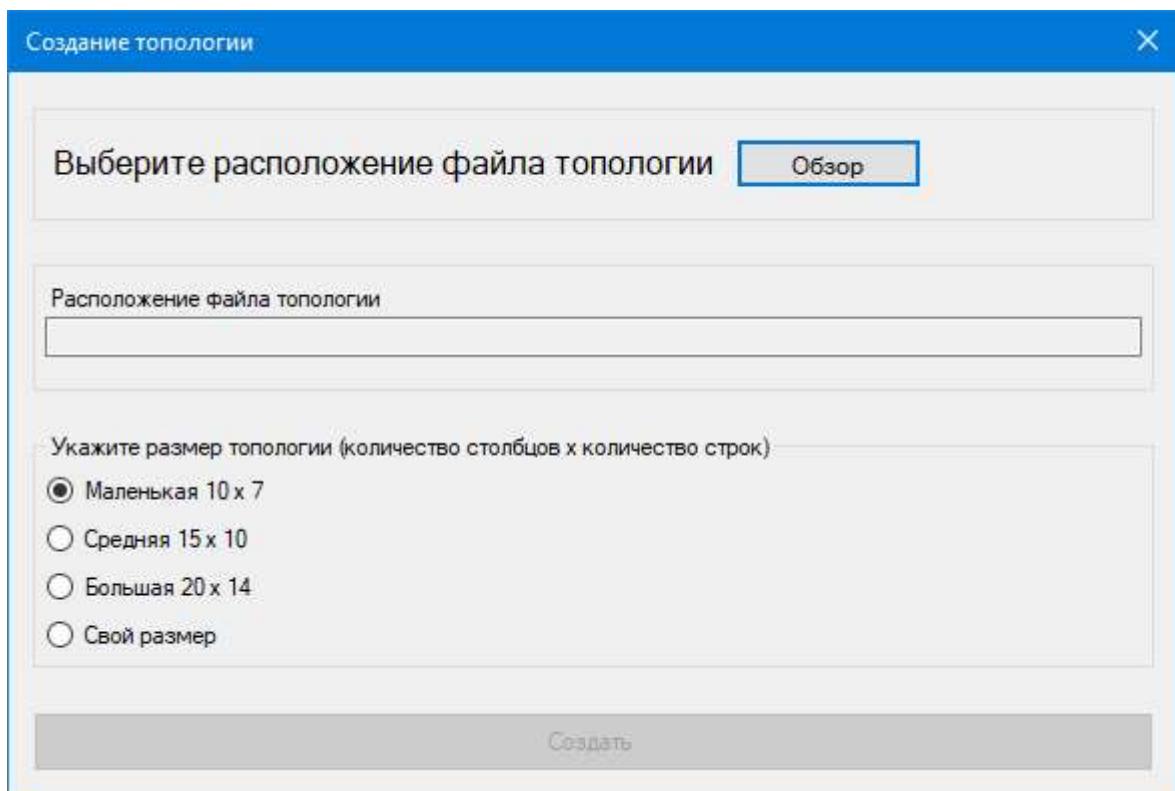


Рисунок 43 – Экранная форма «Создание топологии»

После этого, пользователь может выбрать стандартные размеры топологии (маленькая, средняя, большая) или же задать свои размеры.

Если пользователь решил задать свои размеры для топологии, то ему станут видимы и доступны поля для задания размеров по горизонтали и вертикали. На рисунке 44 изображена настройка своих размеров топологии

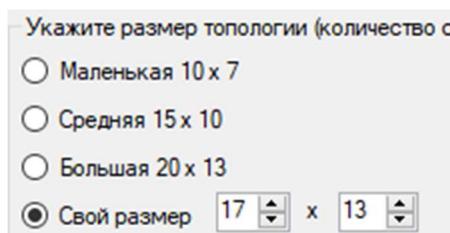


Рисунок 44 – Настройка своих размеров топологии

Кнопка «Создать» станет доступной после того, как пользователь задал все параметры.

3.1.2 Работа с конструктором

После того как была создана и настроена или загружена из файла топология, система отобразит экранную форму «Конструктор топологии», изображенную на рисунке 45, в котором пользователь может выполнять различные действия: размещать ШЭ, удалять ШЭ, настраивать ШЭ.

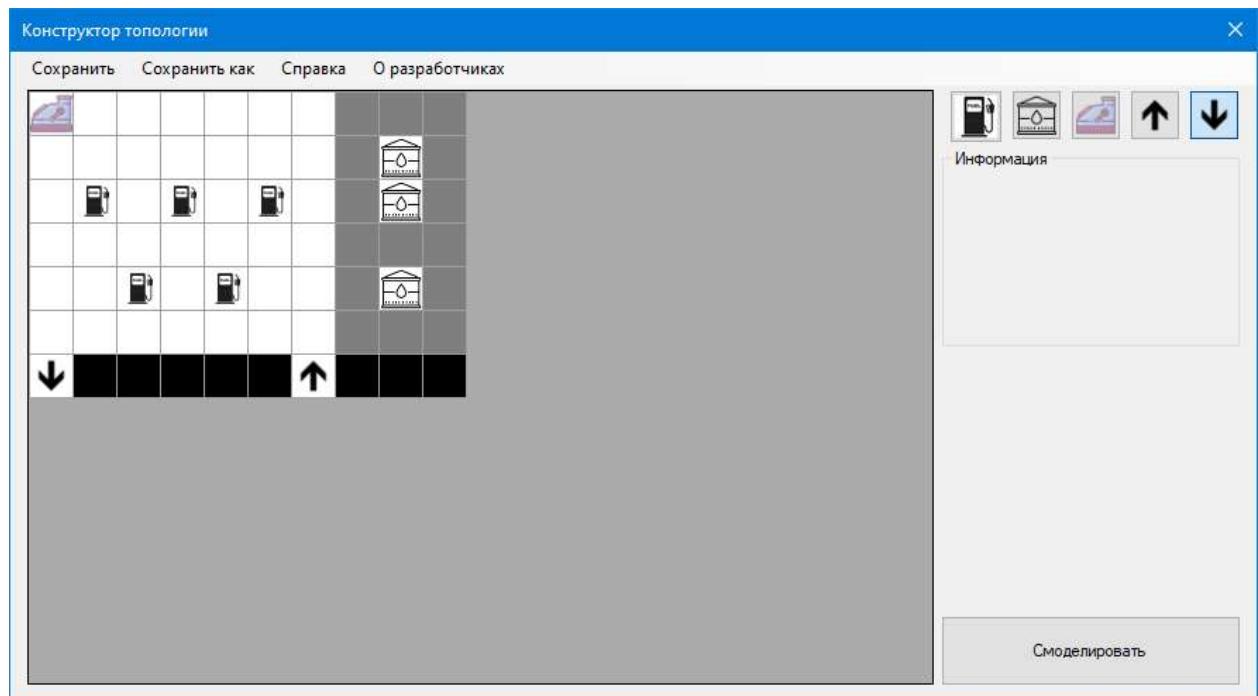


Рисунок 45 – Экранная форма «Конструктор топологии»

В левом верхнем углу располагаются кнопки «Сохранить», «Сохранить как», «Справка» и «О разработчиках». С помощью кнопок «Сохранить» и «Сохранить как» пользователь может сохранить топологию в файл, в котором осуществляется работа или же сохранить топологию в новый файл.

Нажав на кнопки «Справка» и «О разработчика», пользователь может посмотреть информацию о конструкторе и разработчиках соответственно.

В правом верхнем углу расположена панель ШЭ (ТРК, ТБ, касса, въезд, выезд). Под этой панелью располагается окно, в котором отображается

информация о выбранном ШЭ, где пользователь может настроить параметры ШЭ.

Большая часть экрана отображает пространство, в котором расположено сама топология и расположенные на ней ШЭ. Чтобы добавить ШЭ на топологию, пользователь должен выбрать ШЭ и, кликнув ЛКМ, размещать ШЭ. Пользователь может размещать ШЭ с помощью функции «Drag and drop». Чтобы удалить ШЭ с топологии, пользователь должен кликнуть правой кнопкой мыши (ПКМ) по ШЭ, который он хочет удалить.

Чтобы настроить ШЭ «ТРК», пользователь должен кликнуть ЛКМ по ТРК на топологии, после чего станет доступным параметр выбранной ТРК, а именно параметр «Скорость подачи». В данном поле пользователь может указать любое число. Если ведённое значение выходит за пределы допустимых значений, то присвоено будет минимальное или максимальное значение соответственно. На рисунке 46 изображена область, в которой происходит настройка ТРК.

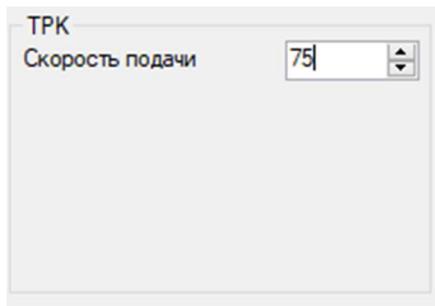


Рисунок 46 – Настройка ТРК

Чтобы настроить ШЭ «ТБ», пользователь должен кликнуть ЛКМ по ТБ на топологии, после чего станут доступны такие параметры как «Объем» и «Объем топлива» и «Топливо». На рисунке 47 изображена настройка параметров ТБ.

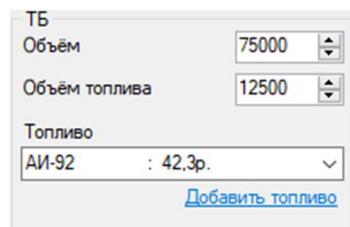


Рисунок 47 – Настройка ТБ

В поле «Объем» пользователь указывает размер ТБ. В поле «Объем топлива» пользователь указывает объем выбранного топлива в ТБ. В поле «Топливо» пользователь указывает какой вид топлива будет находиться в выбранном ТБ. Для того чтобы выбрать топливо, пользователю необходимо открыть список видов топлива, кликнув ЛКМ по стрелке в конце поля. После этого откроется список доступных видов топлива, где он может выбрать нужный ему вид топлива.

Если пользователь захочет создать новый вид топлива, то он кликает на гиперссылку «Добавить топливо». После нажатия на эту кнопку, откроется модальное окно «Добавление топлива». В данном поле пользователю необходимо заполнить поля «Название» и «Цена за литр, руб». После успешного заполнения соответствующих полей, создается новый вид топлива и выбранный ТБ выбирает новое топливо в качестве параметров. На рисунке 48 изображена экранная форма «Добавление топлива».

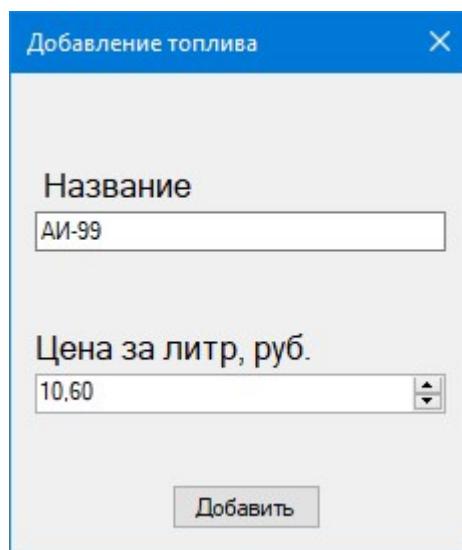


Рисунок 48 – Экранная форма «Добавление топлива»

ШЭ «касса», «въезд» и «выезд» не имеют настраиваемых параметров.

При нажатии на кнопку «Смоделировать» система выполняет валидацию. Если валидация пройдет успешно, то осуществляется переход к экранной форме «Настройка транспортного потока», иначе система выдаст сообщение об ошибке с указанием ее причины.

3.1.3 Настройка транспортного потока

В экранной форме «Настройка транспортного потока» пользователь может настроить параметры транспортного потока или перейти к процессу моделирования с помощью нажатия кнопки «Смоделировать».

Пользователь может выбрать детерминированный или случайный поток с помощью выбора соответствующих радиокнопок.

Если выбран «Детерминированный поток», то система отображает поле «Время между появлением автомобилей, с». В данном поле пользователь может указать интервал времени, по истечению которого будут появляться новые машины в транспортном потоке. На рисунке 49 изображена экранная форма настройки детерминированного потока.

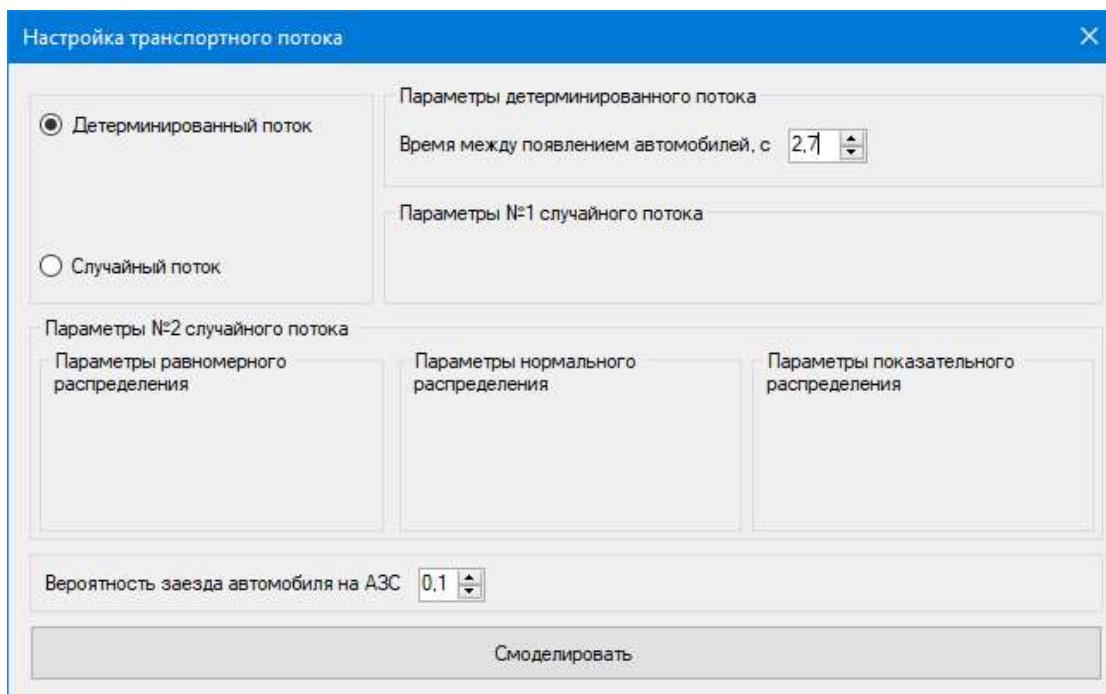


Рисунок 49 – Экранная форма настройки детерминированного потока

Если выбран «Случайный поток», то система отобразит поле, где пользователю необходимо выбрать ЗР (равномерный, нормальный и показательный) и задать его параметры. На рисунках 50 – 52 изображены экранные формы настроек параметров ЗР.

Пользователь может менять параметр поля, который отвечает за вероятность заезда автомобиля на АЗС. На рисунке 53 изображена экранная форма настройки данного параметра.

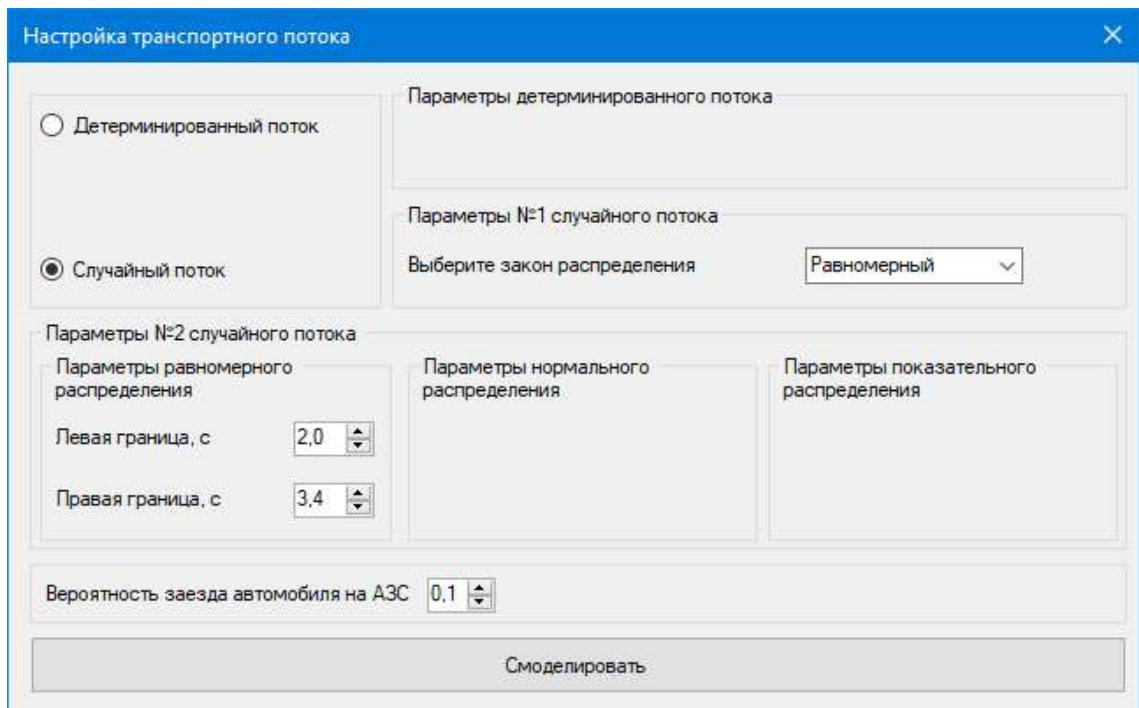


Рисунок 50 – Экранная форма настройки равномерного ЗР

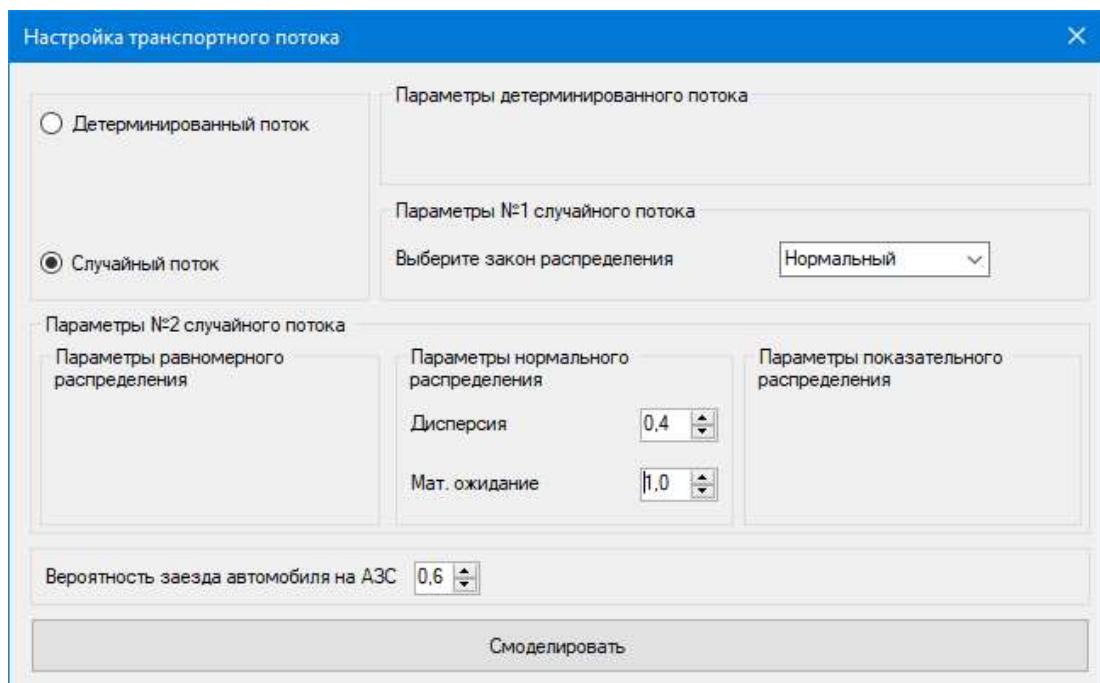


Рисунок 51 – Экранная форма настройки нормального ЗР

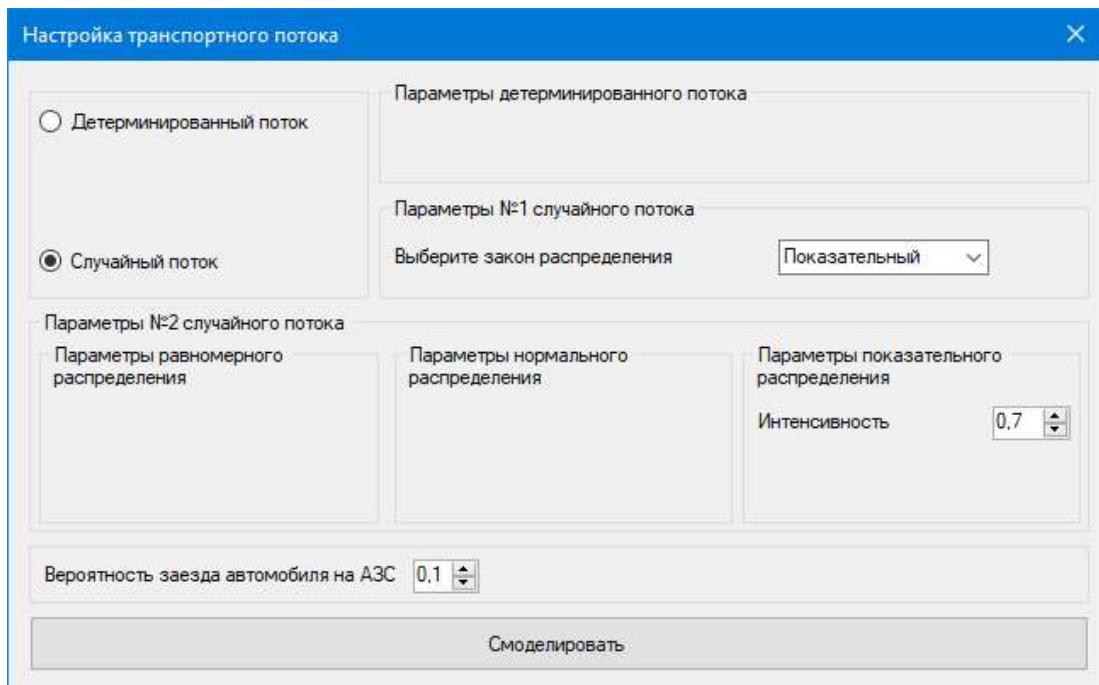


Рисунок 52 – Экранная форма настройки показательного ЗР

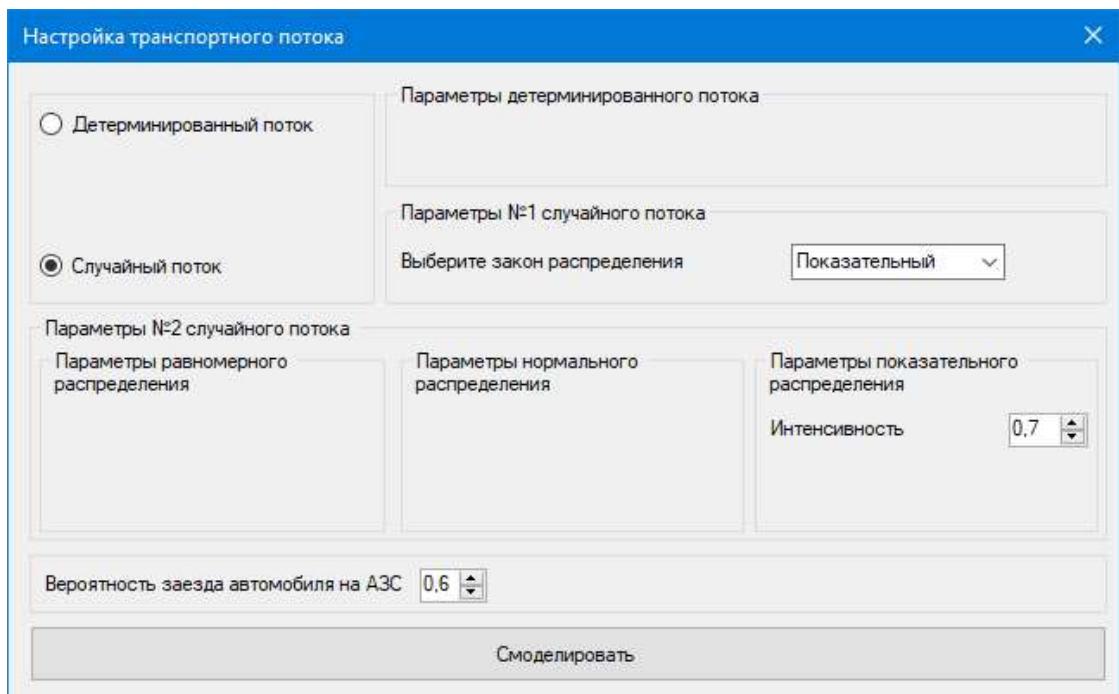


Рисунок 53 – Экранная форма настройки вероятности заезда автомобиля на АЗС

Если указанные параметры введены верно, то при нажатии на кнопку «Смоделировать» система отобразит экранную форму «Моделирование», иначе система выдаст ошибку, в которой будет указано, какие параметры были указаны неверно.

3.1.4 Моделирование системы АЗС

Экранная форма «Моделирование» представляет моделирование работы АЗС с учетом указанных ранее настроек транспортного потока и выбранной топологии. В правой части экрана отображается состояние системы: «Активна» или «Пауза». Можно менять состояние процесса, с помощью нажатия на кнопки «►» или «||», которые расположены в нижней части окна. Данные кнопки меняют друг друга, в зависимости от того, в каком состоянии находится система в данный момент времени.

Под строкой состояния отображается информация о кассе, последнем выбранном ТРК и ТБ. Ниже отображается информация о выбранном пользователем элементе. В качестве элемента можно будет выбрать кассу, ТБ, ТРК, въезд, выезд, автомобиль, автомобиль инкассации, дозаправщика и сервисную зону. На рисунке 54 изображено моделирование работы АЗС.

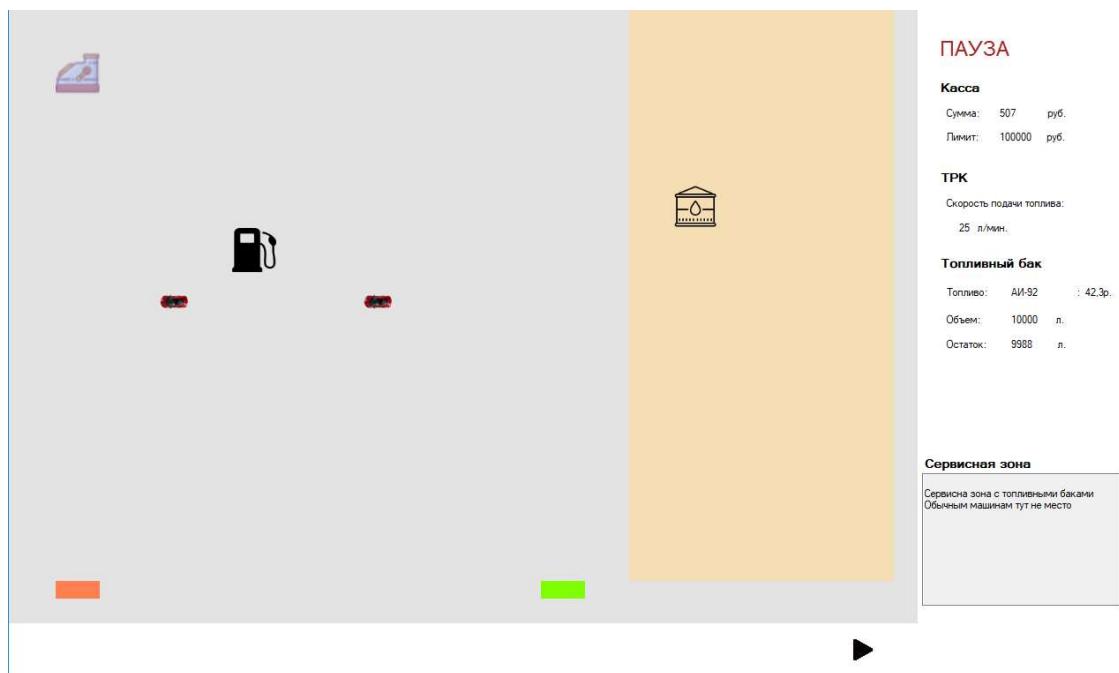


Рисунок 54 – Моделирование работы АЗС

3.2 Диаграммы реализации

Диаграммы реализации предназначены для отображения состава компилируемых и выполняемых модулей системы, а также связей между ними. Диаграммы реализации разделяются на два конкретных вида:

диаграммы компонентов (component diagrams) и диаграммы развертывания (deployment diagrams) [49].

3.2.1 Диаграмма компонентов

Диаграмма компонентов описывает особенности физической реализации приложения, определяет архитектуру приложения и устанавливает зависимость между компонентами, в роли которых выступает исполняемый код. Диаграмма компонентов отображает общие зависимости между компонентами. Основными графическими элементами диаграммы являются компоненты, интерфейсы и зависимости между ними [50].

В таблице 4 приведено описание компонентов. Диаграмма компонентов для системы изображена на рисунке 55.

3.2.2 Диаграмма развертывания

Диаграмма развертывания в UML моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Существуют два типа узлов:

- узел устройства;
- узел среды выполнения.

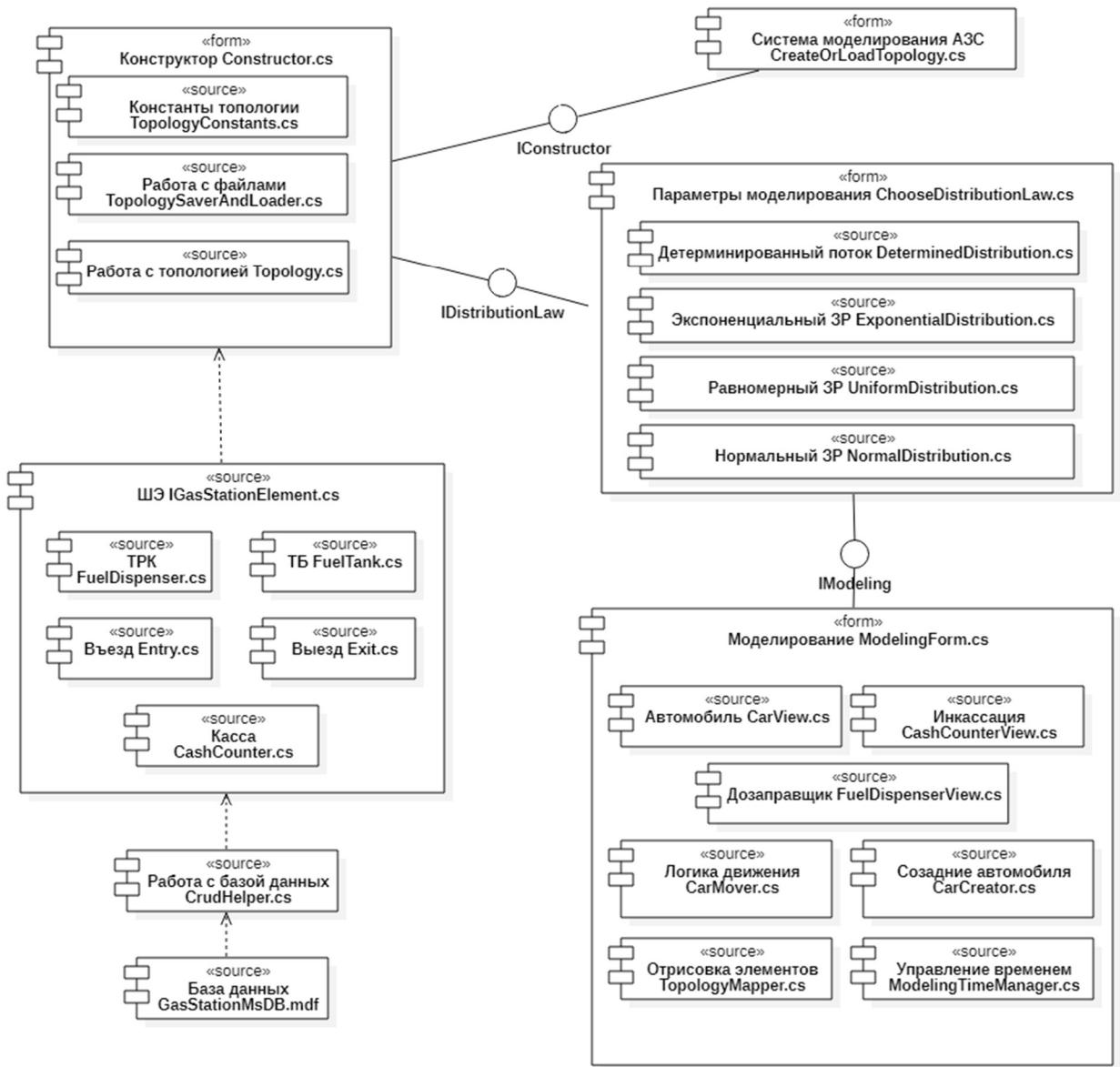


Рисунок 55 – Диаграмма компонентов

Таблица 4 – Описание компонентов системы

Название компонента	Назначение
1	2
Система моделирования АЗС	Предоставляет пользовательский интерфейс системы.
IConstructor	Программный интерфейс для абстракции работы с конструктором.
Конструктор	Отвечает за прием данных от пользователя и передачу данных системе.

Продолжение таблицы 4

1	2
Константы топологии	Структуры, необходимые для валидации системы.
Работа с файлами	Отвечает за сохранение и загрузку файлов топологии.
TPK	Отвечает за настройку параметров ТPK.
TБ	Отвечает за настройку параметров ТБ.
Въезд	Отвечает за настройку въезда.
Выезд	Отвечает за настройку выезда.
Касса	Отвечает за настройку кассы.
Работа с базой данных	Вспомогательный компонент, необходимый для взаимодействия с базой данных.
IDistributionLaw	Программный интерфейс для работы с настройками транспортного потока.
Параметры моделирования	Отвечает за прием данных от пользователя и передачу данных системе.
Детерминированный поток	Отвечает за настройка параметров детерминированного потока.
Равномерный ЗР	Отвечает за настройку параметров равномерного ЗР.
Нормальный ЗР	Отвечает за настройку параметров нормального ЗР.
Показательный ЗР	Отвечает за настройку параметром показательного ЗР.
IModeling	Программный интерфейс для работы процесса моделирования.

Продолжение таблицы 4

1	2
Автомобиль	Отвечает за работу автомобиля в процессе моделирования.
Инкасация	Отвечает за работу автомобиля инкасации в процессе моделирования.
Дозаправщик	Отвечает за работу дозаправщика в процессе моделирования.
Логика движения	Отвечает за передвижение ТС на дороге и АЗС.
Создание автомобиля	Отвечает за генерацию ТС.
Отрисовка элементов	Отвечает за отображение таких элементов как ТС, ШЭ и сервисной зоны.
Управление временем	Отвечает за время в процессе моделирования.

Узлы устройств – это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны. Узел среды выполнения – это программный вычислительный ресурс, который работает внутри внешнего узла и который предоставляет собой сервис, выполняющий другие исполняемые программные элементы [51].

На рисунке 56 изображена диаграмма развертывания. В качестве базы данных в приложении необходимо наличие MS SQL Server, с помощью которого происходит непосредственное взаимодействие с базой данных. Для отображения справочной системы необходимо наличие браузера. Сохранение и загрузка файлов происходит за счет использования файловой системы.

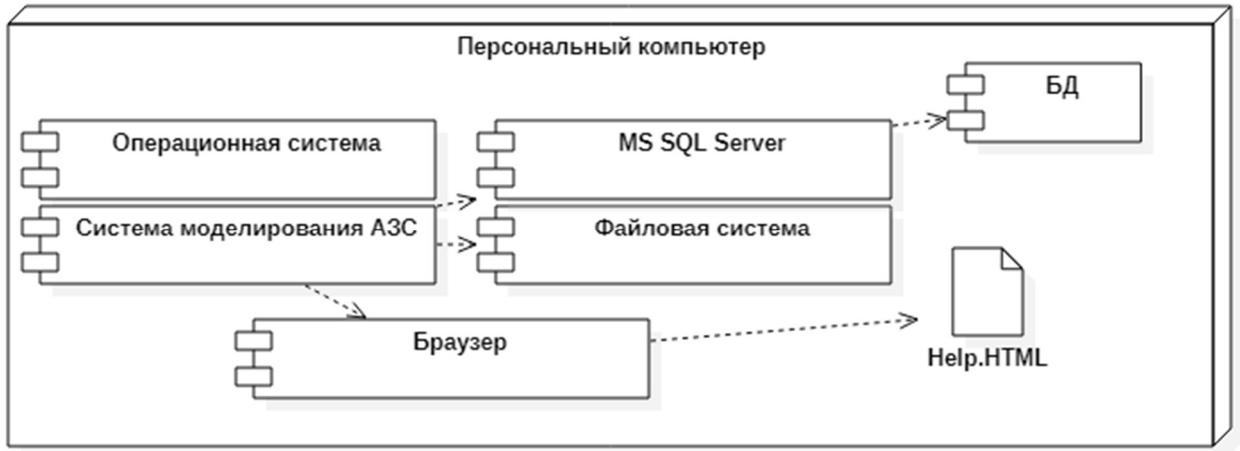


Рисунок 56 – Диаграмма развертывания

3.2.3 Диаграмма классов

Диаграммы классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем, они показывают набор классов, интерфейсов и коопераций, а также их связи. На практике диаграммы классов применяют для моделирования статического представления системы, они служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения [40].

В соответствии со спецификациями, указанными в пункте 2.3, и выбором комплекса программных средств, указанным в пункте 2.7, были созданы классы с различными параметрами и методами. На рисунке 57 приведена диаграмма классов системы (этап реализации).

3.3 Физическая модель данных

Физическая модель – модель базы данных, выраженная в терминах языка описания данных конкретной СУБД. Она содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т.п. Физическая модель строится на основе логической с учетом ограничений, накладываемых возможностями выбранной СУБД [52].

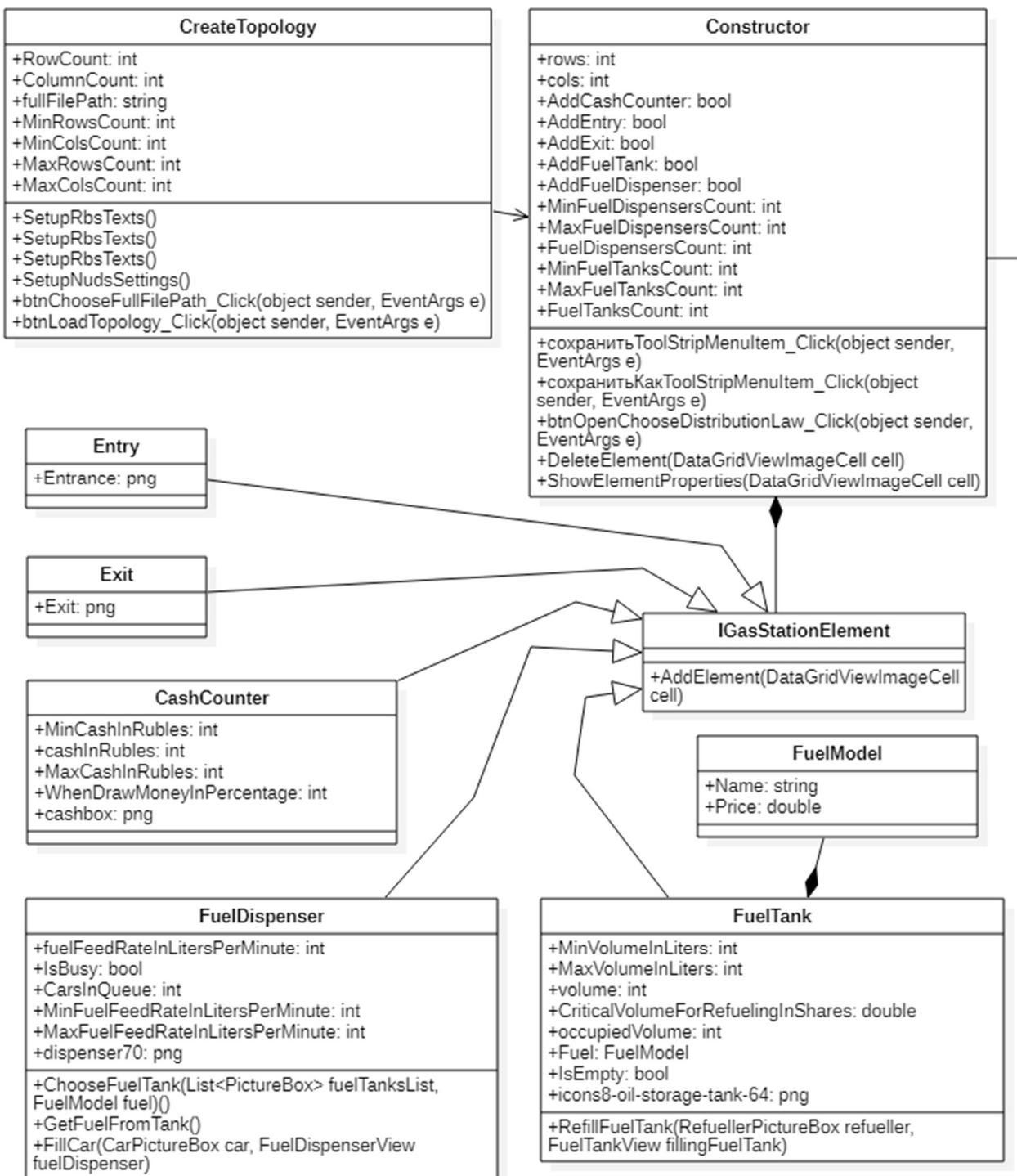


Рисунок 57 – Диаграмма классов (начало)

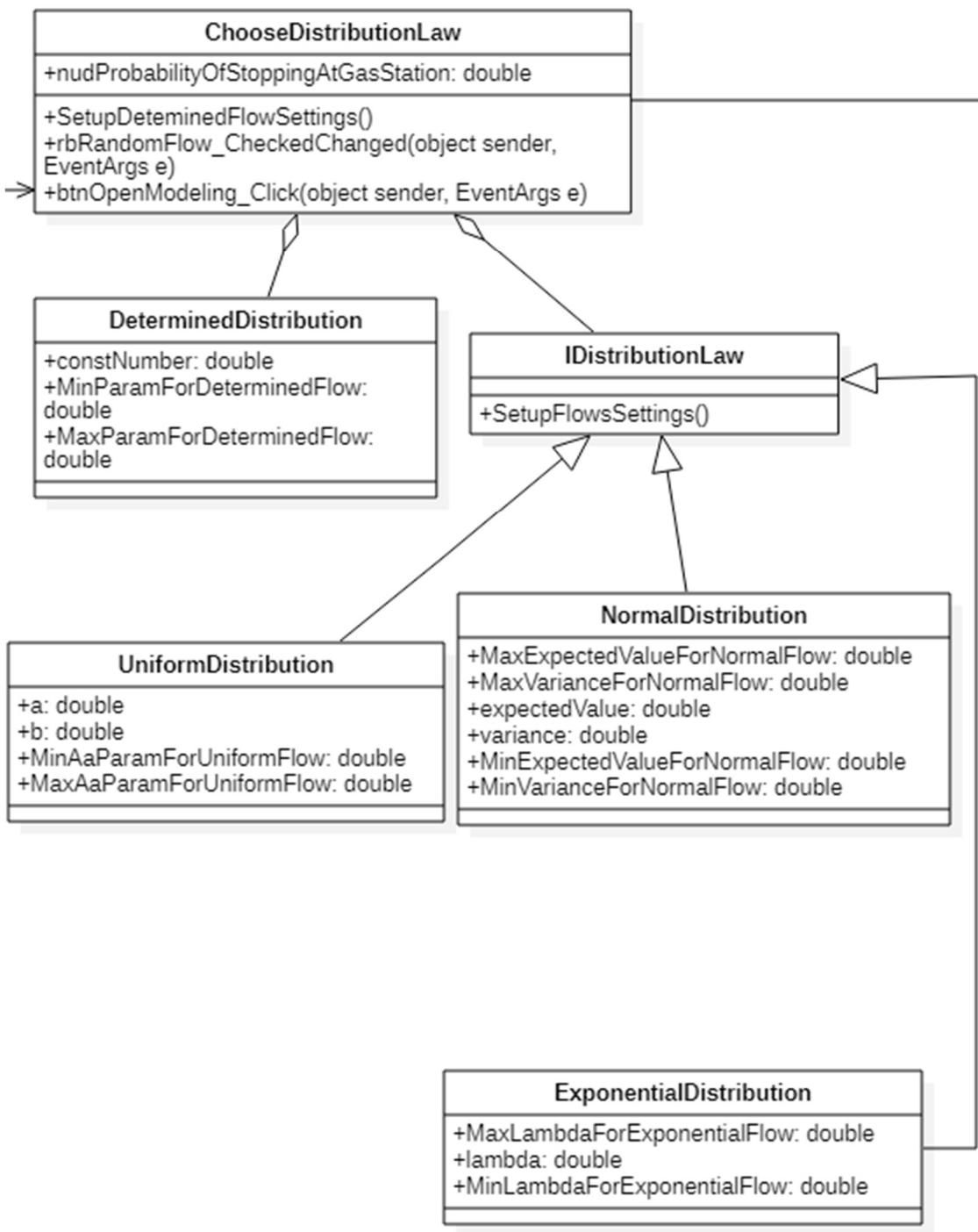


Рисунок 57 – Диаграмма классов (продолжение)

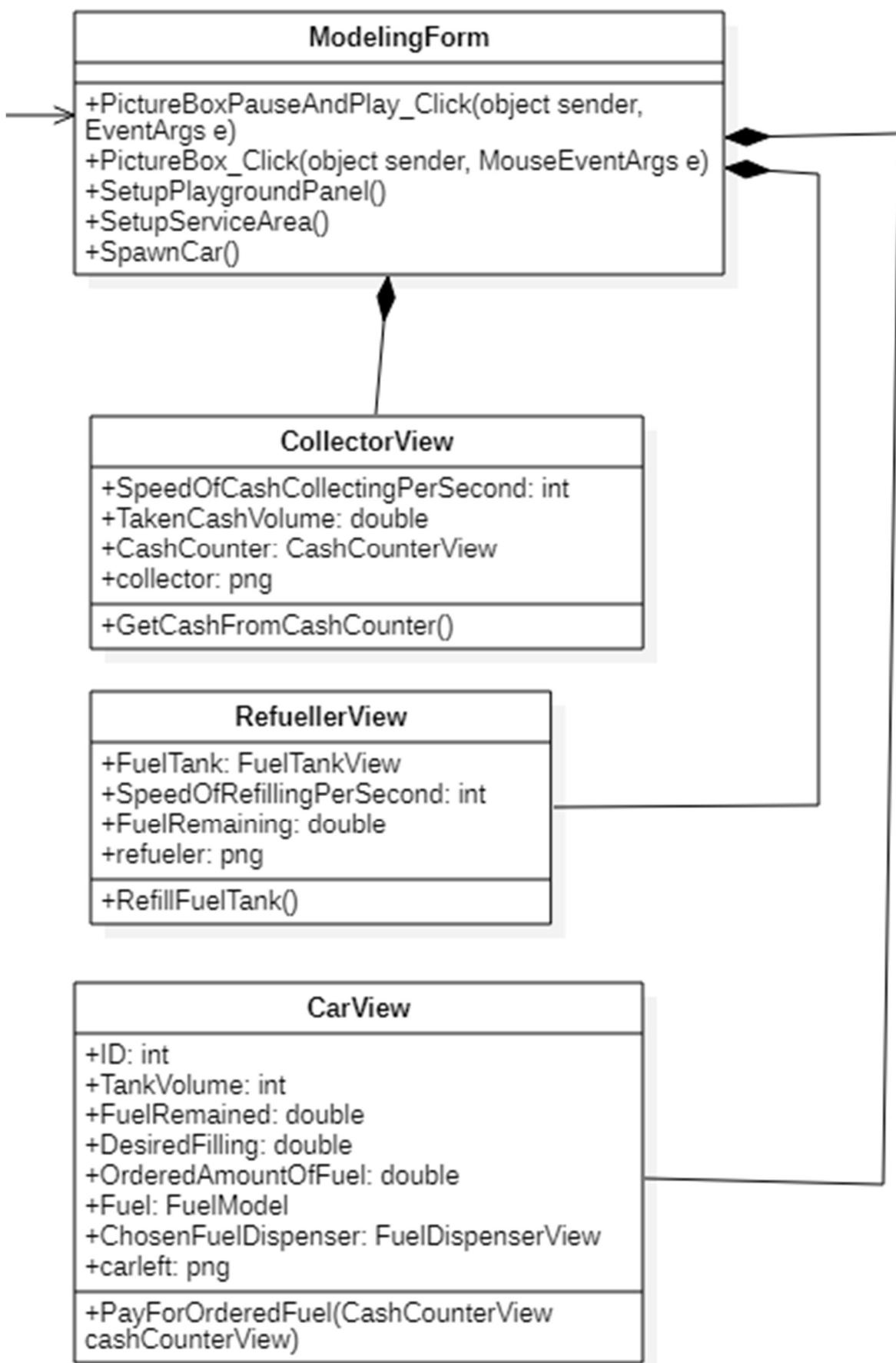


Рисунок 57 – Диаграмма классов (конец)

На рисунке 58 изображена физическая модель данных. На ней изображены таблицы «Автомобиль» и «Вид топлива».

В таблице «Вид топлива» первичным ключом является атрибут «ID». Атрибуты «Имя» и «Цена» должны быть обязательно заполнены при добавлении вида топлива.

В таблице «Автомобиль» первичным ключом является атрибут «ID». Атрибуты «Имя», «Объем бака» и «Грузовик» должны быть обязательно указаны при создании машины. Связь между таблицами осуществляется за счет внешнего ключа «FuelId».

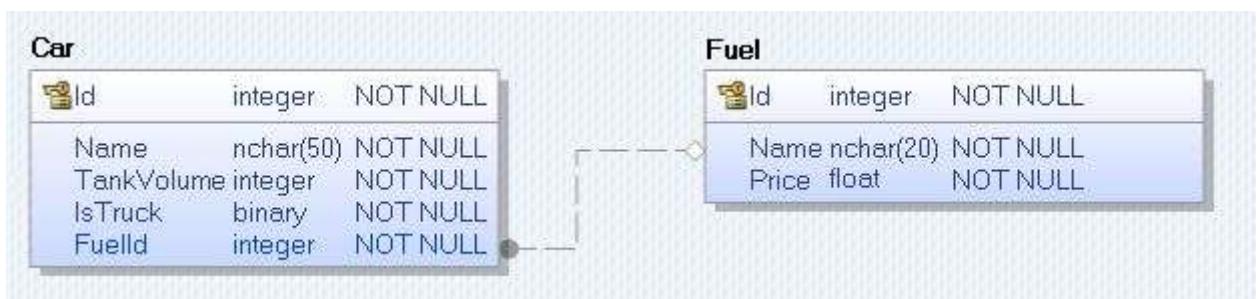


Рисунок 58 – Физическая модель данных

3.4 Выбор и обоснование комплекса технических средств

3.4.1 Расчет объема внешней памяти

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования системы, воспользуемся следующей формулой:

$$V_{ЖД} = V_{ОС} + V_{ПР} + V_{БД} + V_{СПО} + V_{Ф},$$

где $V_{ОС}$ – объем памяти, занимаемый операционной системой (операционная система Windows 10, $V_{ОС} = 30$ Гб);

$V_{ПР}$ – объем памяти, занимаемый непосредственно файлами приложения ($V_{ПР} = 8$ Мб);

$V_{БД}$ – объем памяти, занимаемый базой данных (всеми таблицами) при ее максимальном заполнении. Расчета этой составляющей приведен в таблице 5; исходные данные для расчета взяты из описания таблиц БД.

$V_{\text{СПО}}$ – объем памяти, занимаемый всем необходимым сопутствующим программным обеспечением (сюда входят СУБД MS SQL Server2017 и Microsoft Visual Studio 2019); дадим оценку сверху $V_{\text{СПО}}$ в 15 Гб);

V_{Φ} – объем памяти, необходимый для хранения файлов, необходимых для работы программы (дадим ему оценку сверху в 1 Мб)/

Таблица 5 – Расчет объема внешней памяти, необходимой для хранения БД

Таблица	Размер записи (байт)	Максимум записей	Всего (байт)
Вид топлива	56	12	672
Автомобиль	125	100	12500
Итого			13172

$$V_{\text{БД}} = 13172 \text{ байт} = 13,172 \text{ Мб.}$$

Таким образом, суммарный объем внешней памяти составит:

$$V_{\text{ЖД}} = 30 \text{ Гб} + 8 \text{ Мб} + 13,172 \text{ Мб} + 15 \text{ Гб} + 1 \text{ Мб} \approx 46 \text{ Гб.}$$

3.4.2 Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

$$V_{\text{ОЗУ}} = V_{\text{ОС}} + V_{\text{ПР}} + V_{\text{СПО}} + V_{\text{БД}},$$

где $V_{\text{ОС}}$ – ОЗУ, занимаемое операционной системой (256 Мб);

$V_{\text{ПР}}$ – ОЗУ, которое займет само приложение (не превысит 30 Мб);

$V_{\text{СПО}}$ – ОЗУ, занимаемое СУБД и другим сопутствующим ПО (оценим его сверху значением в 200 Мб);

$V_{\text{БД}}$ – объем данных из базы, который может быть одновременно загружен в оперативную память (дадим ему оценку сверху в 15 Мб).

Суммарные объемы ОЗУ составят:

$$V_{\text{ОЗУ}} = 256 \text{ Мб} + 30 \text{ Мб} + 200 \text{ Мб} + 15 \text{ Мб} = 501 \text{ Мб.}$$

Таким образом, 512 Мб оперативной памяти можно счесть минимально необходимым для функционирования системы.

3.4.3 Минимальные требования, предъявляемые к системе

Для корректного функционирования системы необходимо:

- 1) тип ЭВМ: x86-64 совместимый;
- 2) объем ОЗУ – не менее 512 Мб;
- 3) объем свободного дискового пространства – не менее 10 Гб;
- 4) клавиатура или иное устройство ввода;
- 5) мышь или иное манипулирующее устройство.
- 6) процессор – Intel или AMD с частотой не менее 1,5 ГГц;
- 7) монитор с разрешающей способностью не ниже 800 x 600;
- 8) любой браузер.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта была разработана автоматизированная система моделирования работы АЗС, позволяющая создавать территорию АЗС, настраивать её и объекты на ней и моделировать работу АЗС в реальном времени

В первом разделе описаны основные компоненты АЗС, правила моделирования транспортного потока, рассмотрены системы-аналоги, приведен их сравнительный анализ, отмечены их достоинства и недостатки. На основа проведенного анализа выполнена объектная декомпозиция, отраженная в диаграмме объектов, и сформулирована постановка задачи.

В разделе «Проектирование системы» была выбрана и обоснована архитектура системы, описана структурная схема системы, разработана функциональная спецификация и описаны прототипы экранных форм системы. Также был разработан логический проект, в который вошли все канонические диаграммы: диаграмма вариантов использования системы, текстовые сценарии, диаграмма состояний, диаграмма деятельности, диаграмма последовательности и диаграмма классов. Выбран комплекс программных средств.

В разделе «Реализация системы» приведено описание интерфейса пользователя, диаграмм реализации (классов, компонент и развертывания), выбран и обоснован основной комплекс технических средств. Также были проведены ресурсные расчеты и определены технические требования для функционирования системы.

Данная программа может использоваться для моделирования работы АЗС в реальном времени.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Автомобильная заправочная станция [Электронный ресурс] // Википедия:[сайт].URL: https://ru.wikipedia.org/wiki/Автомобильная_заправочная_станция#История_возникновения_AZC (дата обращения: 29.09.2019).
- 2 История бензовозаправок (АЗС) в мире и в СССР [Электронный ресурс] // Fishki.net:[сайт]. URL: <https://fishki.net/auto/2235886-istorija-benzozapravok-azs-v-mire-i-sssr.html> (дата обращения: 29.09.2019).
- 3 UML-диаграммы [Электронный ресурс] // Студопедия:[сайт]. URL: https://studopedia.ru/3_33047_ob-ektno-orientirovanniy-analiz-i-proektirovanie.html (дата обращения: 29.09.2019).
- 4 Методология объектно-ориентированного анализа и проектирования [Электронный ресурс] // ВикиЧтение: [сайт]. URL: <https://it.wikireading.ru/5076> (дата обращения: 29.09.2019)
- 5 Классификация АЗС [Электронный ресурс] // Нефтепродуктообеспечение: [сайт]. URL: <http://proofoil.ru/Petrochemical/Petrochemical1.html> (дата обращения: 29.09.2019)
- 6 Классификация и общая характеристика АЗС [Электронный ресурс] // Персональная АЗС: [сайт]. URL: https://www.personalazs.ru/documentation/safety/characteristic_azs/ (дата обращения: 30.09.2019)
- 7 Топливораздаточные колонки (TPK) [Электронный ресурс] // Роспайл: [сайт]. URL: <http://ros-pipe.ru/clauses/toplivorazdatochnye-kolonki-trk/> (дата обращения: 30.09.2019)
- 8 Резервуар для АЗС. Емкости для АЗС [Электронный ресурс] // Дмитриевский завод металлоконструкций: [сайт]. URL: <http://www.dzm-k.ru/index.php/component/content/article/2-uncategorised/39-rezervuary-dlay-azs> (дата обращения: 30.09.2019)

- 9 Резервуары стальные горизонтальный для нефтепродуктов. Технические условия [Электронный ресурс] // Электронный фонд: [сайт]. URL: <http://docs.cntd.ru/document/1200084951> (дата обращения: 30.09.2019)
- 10 Бензин [Электронный ресурс] // Википедия:[сайт]. URL: https://ru.wikipedia.org/wiki/Бензин#Разновидности_бензина (дата обращения: 30.09.2019).
- 11 Технические характеристики автомобиля [Электронный ресурс] // Avto-Russia:[сайт]. URL: https://avto-russia.ru/autos/lamborghini/lamborghini_diablo_6-0_mt.html (дата обращения: 30.09.2019).
- 12 Транспортный поток [Электронный ресурс] // Студопедия:[сайт]. URL: https://studopedia.ru/7_31440_transportniy-potok.html (дата обращения: 30.09.2019).
- 13 Поток событий [Электронный ресурс] // Научная библиотека:[сайт]. URL: http://scask.ru/a_book_tp.php?id=114 (дата обращения: 07.10.2019).
- 14 Моделирование случайных величин [Электронный ресурс] // Студопедия:[сайт]. URL: <http://studepedia.org/index.php?vol=1&post=75284> (дата обращения: 07.10.2019).
- 15 Моделирование случайных величин [Электронный ресурс] // Студенческие материалы:[сайт]. URL: https://studref.com/362644/ekonomika/modelirovaniye_sluchaynyh_velichin (дата обращения: 07.10.2019).
- 16 Равномерный и нормальный законы распределения непрерывных случайных величин [Электронный ресурс] // Математика и информатика:[сайт]. URL: http://edu.tltsu.ru/er/book_view.php?book_id=1cee&page_id=19506 (дата обращения: 07.10.2019).
- 17 Плотность вероятности [Электронный ресурс] // Википедия:[сайт]. URL: https://ru.wikipedia.org/wiki/Плотность_вероятности (дата обращения: 07.10.2019).

- 18 Функция распределения [Электронный ресурс] // Википедия:[сайт]. URL: https://ru.wikipedia.org/wiki/Функция_распределения (дата обращения: 07.10.2019).
- 19 Моделирование нормально распределенных случайных величин [Электронный ресурс] // Учебник «Моделирование системы»:[сайт]. URL: <http://stratum.ac.ru/education/textbooks/modelir/lection25.html> (дата обращения: 07.10.2019).
- 20 AnyLogic [Электронный ресурс] // Википедия:[сайт]. URL: <https://ru.wikipedia.org/wiki/AnyLogic> (дата обращения: 07.10.2019).
- 21 VisSim [Электронный ресурс] // Википедия:[сайт]. URL: <https://ru.wikipedia.org/wiki/VisSim> (дата обращения: 07.10.2019).
- 22 Дьяконов В.П. VisSim+Mathcad+MATLAB. Визуальное математическое моделирование. Изд. Солон-пресс, 2009. 384 с.
- 23 Анализ предметной области. Выявление функциональных требований к приложению [Электронный ресурс] // Интуит: [сайт]. URL: <http://www.intuit.ru/studies/courses/574/430/lecture/9749> (дата обращения: 07.10.2019).
- 24 Декомпозиция [Электронный ресурс] // Бизнес прост: [сайт] URL: <https://biznes-prost.ru/dekompoziciya.html> (дата обращения: 07.10.2019).
- 25 Виды архитектуры информационной системы [Электронный ресурс] // Архитектура информационной системы: [сайт]. URL: https://spravochnick.ru/bazy_dannyh/bazy_dannyh_vvedenie/arhitektura_informacionnoy_sistemy/ (дата обращения: 07.10.2019).
- 26 Архитектура клиент-сервер [Электронный ресурс] // Архитектура информационной системы: [сайт]. URL: https://life-prog.ru/1_62377_arhitektura-klient-server.html (дата обращения: 07.10.2019).

- 27 Клиент – сервер [Электронный ресурс] // Википедия: [сайт]. URL: https://ru.wikipedia.org/wiki/Клиент_—_сервер (дата обращения: 14.10.2019).
- 28 Многоуровневая архитектура [Электронный ресурс] // Lawbooks: [сайт]. URL: <https://lawbooks.news/telekommunikatsionnyie-sistemy-i-kompyuternyie/mnogourovnevaya-arhitektura-57298.html> (дата обращения: 14.10.2019).
- 29 Система [Электронный ресурс] // Википедия: [сайт] URL: <https://ru.wikipedia.org/wiki/Система> (дата обращения: 14.10.2019).
- 30 Проектирование системы [Электронный ресурс] // Студенческая библиотека онлайн: [сайт]. URL: https://mobile.studbooks.net/2034986/informatika/proektirovaniye_sistemy (дата обращения: 14.10.2019).
- 31 Зеленко, Л.С. Программная инженерия. Курс лекций [Текст]: учебное пособие /Л.С. Зеленко. – Самара: изд-во СГАУ, 2012 – 148 с.
- 32 Области знаний программной инженерии и стандарты ЖЦ программного обеспечения [Электронный ресурс] // Интуит [сайт]. URL: https://www.intuit.ru/studies/professional_retraining/945/courses/237/lecture/6118 (дата обращения: 14.10.2019).
- 33 Требования к программному обеспечению [Электронный ресурс] // Интуит [сайт]. URL: https://life-prog.ru/1_16776_trebovaniya-k-programmnому-obespecheniyu.html (дата обращения: 21.10.2019).
- 34 Подходы к управлению требованиями в SWEBOK [Электронный ресурс] // Бизнес-Анализ в России [сайт]. URL: <https://analytics.infozone.pro/requirements-in-swebok/> (дата обращения: 21.10.2019).
- 35 Свойства пользовательского интерфейса [Электронный ресурс] // Научный словарь-справочник [сайт]. URL: https://spravochnick.ru/informatika/arhitektura_personalnogo_kompyutera/polzovatelskiy_interfeys/ (дата обращения: 21.10.2019).

- 36 Виды интерфейсов. Технологии реализации интерфейсов [Электронный ресурс] Студопедия [сайт]. URL: <https://studopedia.org/5-44000.html> (дата обращения: 04.11.2019).
- 37 Унифицированный язык моделирования UML [Электронный ресурс] Язык UML [сайт]. URL: <http://samara.mgpu.ru/~dzhadzha/dis/15/200.html> (дата обращения: 04.11.2019).
- 38 Элементы графической нотации диаграммы вариантов использования [Электронный ресурс] Интуит [сайт]. URL: <https://www.intuit.ru/studies/courses/32/32/lecture/1004> (дата обращения: 04.11.2019).
- 39 Спецификация требований и рекомендации по написанию эффективных вариантов использования [Электронный ресурс] Интуит [сайт]. URL: <https://www.intuit.ru/studies/courses/32/32/lecture/1006?page=1> (дата обращения: 04.11.2019).
- 40 Диаграмма классов [Электронный ресурс] Студопедия [сайт]. URL: <https://studopedia.info/10-59449.html> (дата обращения: 04.11.2019).
- 41 Диаграмма состояний [Электронный ресурс] Самоучитель UML [сайт]. URL: <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl6/gl6.html> (дата обращения: 04.11.2019).
- 42 Диаграмма деятельности [Электронный ресурс] Самоучитель UML [сайт]. URL: <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html> (дата обращения: 04.11.2019).
- 43 Диаграмма последовательностей [Электронный ресурс] Мега Предмет [сайт]. URL: <https://megapredmet.ru/1-78846.html> (дата обращения: 04.11.2019).
- 44 Язык C# [Электронный ресурс] Microsoft [сайт]. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/> (дата обращения: 16.11.2019).

- 45 Windows [Электронный ресурс] Wikipedia [сайт]. URL: <https://ru.wikipedia.org/wiki/Windows> (дата обращения: 16.11.2019).
- 46 Windows 10 [Электронный ресурс] 3dnews [сайт]. URL: <https://3dnews.ru/993393> (дата обращения: 16.11.2019).
- 47 Visual Studio [Электронный ресурс] Microsoft [сайт]. URL: <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019> (дата обращения: 16.11.2019).
- 48 MS SQL Server [Электронный ресурс] Metanit [сайт]. URL: <https://metanit.com/sql/sqlserver/1.1.php> (дата обращения: 16.11.2019).
- 49 Диаграммы реализации [Электронный ресурс] Языки программирования [сайт]. URL: <http://www.maksakov-sa.ru/ModelUML/DiagrReal/index.html> (дата обращения: 16.11.2019).
- 50 Элементы графической нотации диаграммы компонентов [Электронный ресурс] Интуит [сайт]. URL: <https://www.intuit.ru/studies/courses/32/32/lecture/1022> (дата обращения: 16.11.2019).
- 51 Диаграмма развёртывания [Электронный ресурс] Wikipedia [сайт]. URL: https://ru.wikipedia.org/wiki/Диаграмма_развёртывания (дата обращения: 16.11.2019).
- 52 Физическая модель базы данных [Электронный ресурс] Helpiks [сайт]. URL: <https://helpiks.org/2-92209.html> (дата обращения: 16.11.2019).

ПРИЛОЖЕНИЕ А

Руководство пользователя

A.1 Назначение системы

Данная программа позволяет моделировать работу автозаправочной станции (АЗС). С помощью данной программы пользователь может заниматься конструированием топологии АЗС, размещать на ней шаблонные элементы (ШЭ), настраивать их, а также проводить процессы моделирования.

A.2 Условия работы системы

Для корректной работы системы необходимо наличие соответствующих программных и аппаратных средств.

1) Требования к техническому обеспечению:

- ЭВМ типа IBM PC;
- объем ОЗУ – не менее 512 Мб;
- объем свободного дискового пространства – не менее 10 Гб;
- клавиатура или иное устройство ввода;
- мышь или иное манипулирующее устройство;
- процессор – Intel или AMD с частотой не менее 1,5 ГГц;
- монитор с разрешающей способностью не ниже 800 x 600;
- процессор типа x86 или x64 тактовой частоты 1400 МГц и выше.

2) Требования к программному обеспечению:

- операционная система Windows 10;
- установленная платформа .Net версии 4.0 и выше;
- установленная СУБД Microsoft SQL Server.

A.3 Установка системы

Система поставляется в виде rar-архива. Данный файл необходимо распаковать в любую директорию на жестком диске. Запускаемым файлом системы является файл GasStationMs.App.exe.

A.4 Работа с системой

A.4.1 Начало работы

При запуске системы отобразит окно, в котором пользователю будет предложено создать новую топологию или загрузить существующую топологию. На рисунке А.1 представлена начальная экранная форма системы.

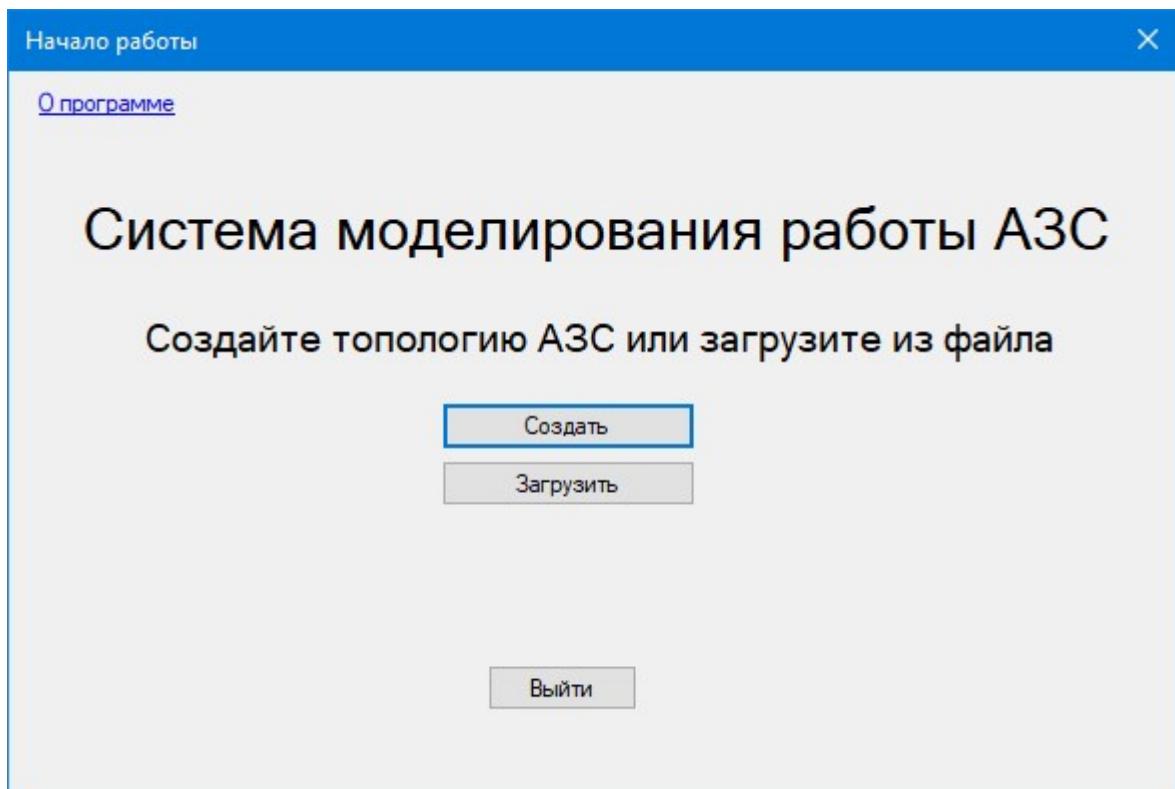


Рисунок А.1 – Начальная экранная форма системы

При нажатии на гиперссылку «О программе» система откроет браузер по умолчанию, в котором отобразит информацию о возможностях программы. Информация о программе представлена в виде HTML-страницы.

Если файл с информацией «О программе» будет поврежден или отсутствовать, то система выдаст ошибку об этом. На рисунке А.2 приведено окно с сообщением об ошибке.

При нажатии на кнопку «Загрузить» откроется диалоговое окно, в котором пользователь должен выбрать файл с существующей топологией.

Если выбранный файл будет поврежден, система сообщит об ошибке, выбранный файл не будет загружен.

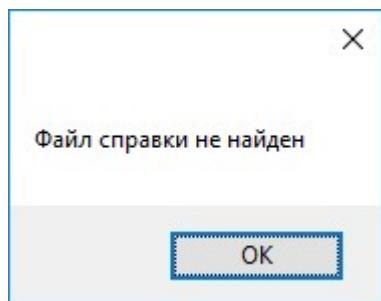


Рисунок А.2 – Сообщение об ошибке «Файл справки не найден»

При нажатии на кнопку «Создать» откроется экранная форма «Создание топологии», изображенная на рисунке А.3, в которой пользователю будет предоставлена возможность создать топологию. Пользователь должен указать расположение файла топологии, нажав на кнопку «Обзор».

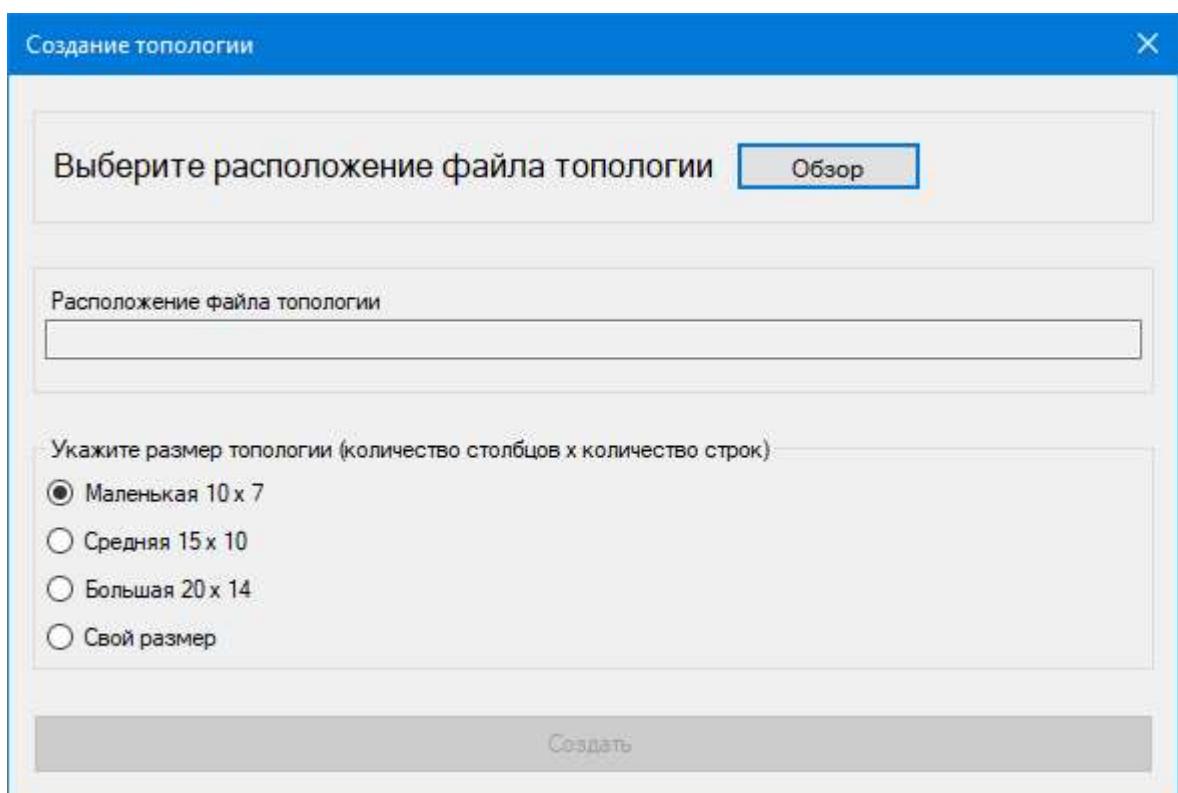


Рисунок А.3 – Экранная форма «Создание топологии»

После этого, пользователь может выбрать стандартные размеры топологии (маленькая, средняя, большая) или же задать свои размеры.

Если пользователь решил задать свои размеры для топологии, то ему станут видимы и доступны поля для задания размеров по горизонтали и вертикали. На рисунке А.4 изображена настройка своих размеров топологии

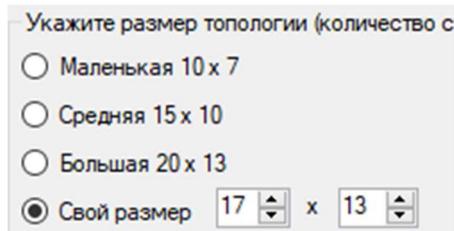


Рисунок А.4 – Настройка своих размеров топологии

Кнопка «Создать» станет доступной после того, как пользователь задал все параметры.

A.4.2 Работа с конструктором

После того как была создана и настроена или загружена из файла топология, система отобразит экранную форму «Конструктор топологии», изображенную на рисунке А.5, в котором пользователь может выполнять различные действия: размещать ШЭ, удалять ШЭ, настраивать ШЭ.

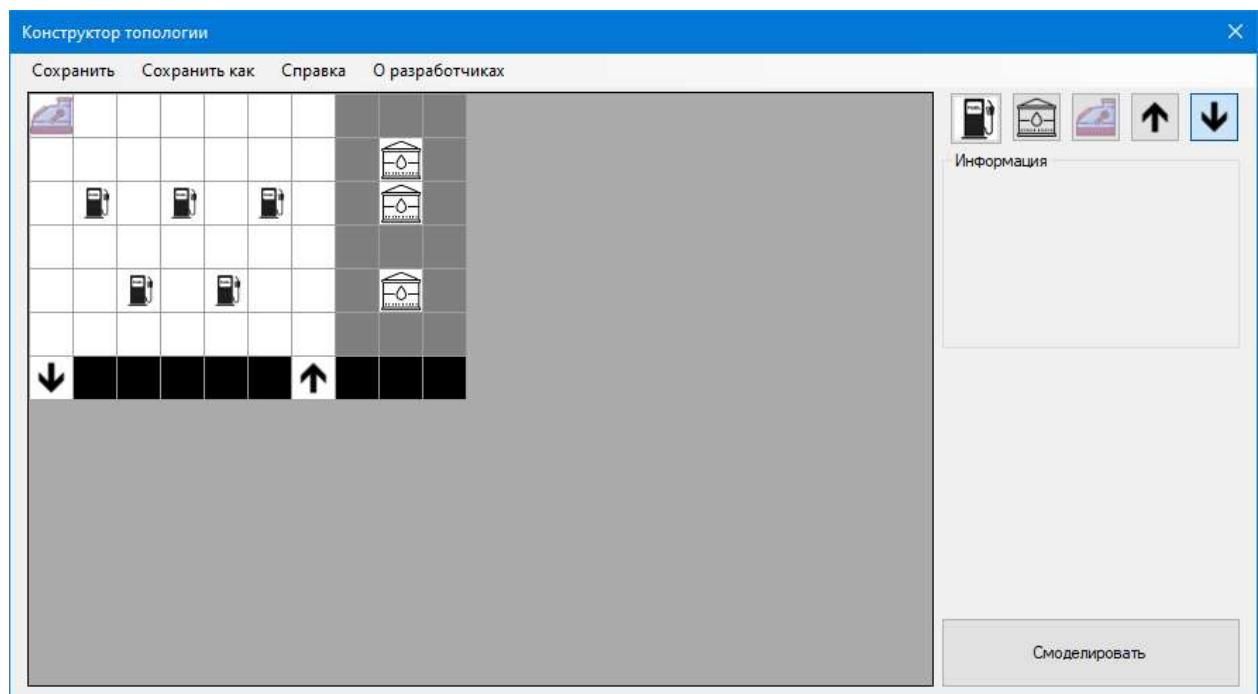


Рисунок А.5 – Экранная форма «Конструктор топологии»

В левом верхнем углу располагаются кнопки «Сохранить», «Сохранить как», «Справка» и «О разработчиках». С помощью кнопок «Сохранить» и «Сохранить как» пользователь может сохранить топологию в файл, в котором осуществляется работа или же сохранить топологию в новый файл. При успешном сохранении, система уведомляет пользователя о том, что топология успешно сохранена в файл. На рисунке А.6 система уведомляет пользователя с помощью модального окна.

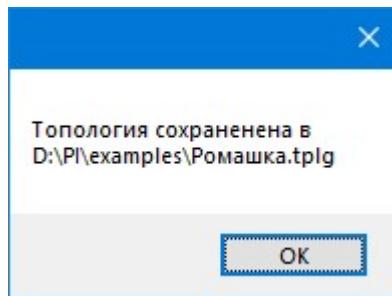


Рисунок А.6 – Сохранение прошло успешно

Нажав на кнопки «Справка» и «О разработчика», пользователь может посмотреть информацию о конструкторе и разработчиках соответственно.

Если файлы с информацией будут повреждены или отсутствовать, то система выдаст ошибку об этом. На рисунке А.7 приведено окно с сообщением об ошибке.

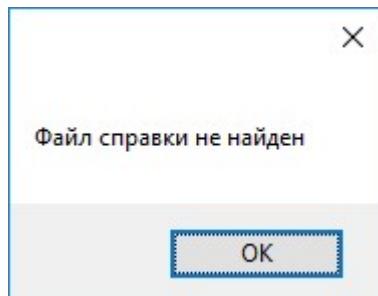


Рисунок А.7 – Сообщение об ошибке «Файл справки не найден»

В правом верхнем углу расположена панель ШЭ (ТРК, ТБ, касса, въезд, выезд). Под этой панелью располагается окно, в котором отображается информация о выбранном ШЭ, где пользователь может настроить параметры ШЭ.

Большая часть экрана отображает пространство, в котором расположено сама топология и расположенные на ней ШЭ. Чтобы добавить ШЭ на топологию, пользователь должен выбрать ШЭ и, кликая ЛКМ, размещать ШЭ. Пользователь может размещать ШЭ с помощью функции «Drag and drop». Чтобы удалить ШЭ с топологии, пользователь должен кликнуть правой кнопкой мыши (ПКМ) по ШЭ, который он хочет удалить.

Если пользователь размещает большее количество ШЭ, чем предусмотрено системой, то в результате действия пользователя система вывод сообщение об ошибке соответственно. На рисунках А.8 и А.9 представлены различные варианты ошибок.

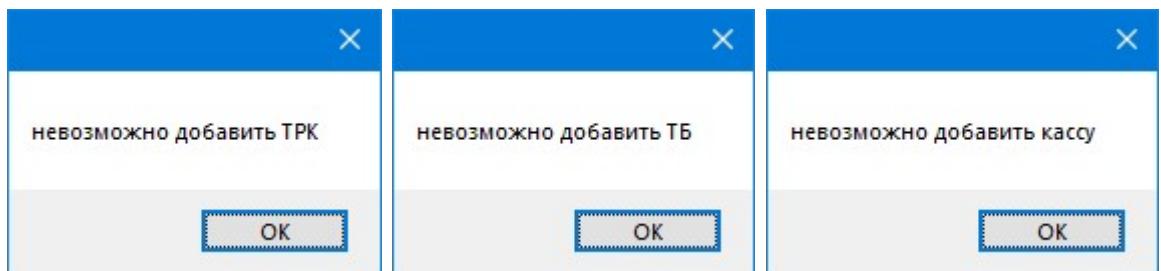


Рисунок А.8 – Сообщение о невозможности размещения ШЭ

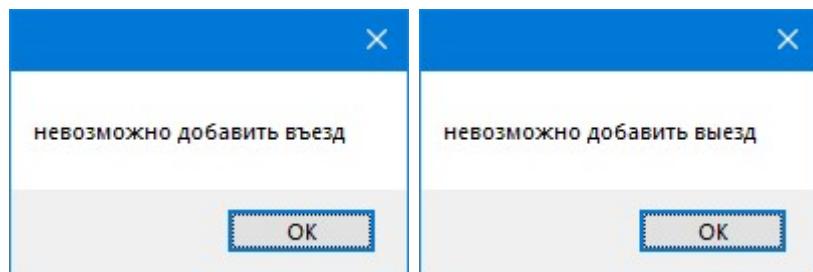


Рисунок А.9 – Сообщение о невозможности размещения ШЭ

Чтобы настроить ШЭ «ТРК», пользователь должен кликнуть ЛКМ по ТРК на топологии, после чего станет доступным параметр выбранной ТРК, а именно параметр «Скорость подачи». В данном поле пользователь может указать любое число. Если ведённое значение выходит за пределы допустимых значений, то присвоено будет минимальное или максимальное значение соответственно. На рисунке А.10 изображена область, в которой происходит настройка ТРК.

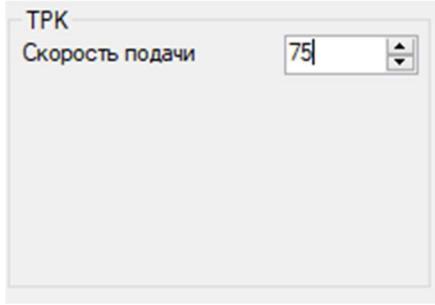


Рисунок А.10 – Настройка ТРК

Чтобы настроить ШЭ «ТБ», пользователь должен кликнуть ЛКМ по ТБ на топологии, после чего станут доступны такие параметры как «Объем» и «Объем топлива» и «Топливо». На рисунке А.11 изображена настройка параметров ТБ.

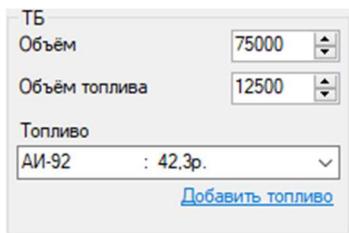


Рисунок А.11 – Настройка ТБ

В поле «Объем» пользователь указывает размер ТБ. В поле «Объем топлива» пользователь указывает объем выбранного топлива в ТБ. В поле «Топливо» пользователь указывает какой вид топлива будет находиться в выбранном ТБ. Для того чтобы выбрать топливо, пользователю необходимо открыть список видов топлива, кликнув ЛКМ по стрелке в конце поля. После этого откроется список доступных видов топлива, где он может выбрать нужный ему вид топлива.

Если пользователь захочет создать новый вид топлива, то он кликает на гиперссылку «Добавить топливо». После нажатия на эту кнопку, откроется модальное окно «Добавление топлива». В данном поле пользователю необходимо заполнить поля «Название» и «Цена за литр, руб». После успешного заполнения соответствующих полей, создается новый вид топлива и выбранный ТБ выбирает новое топливо в качестве параметров. На рисунке А.12 изображена экранная форма «Добавление топлива».

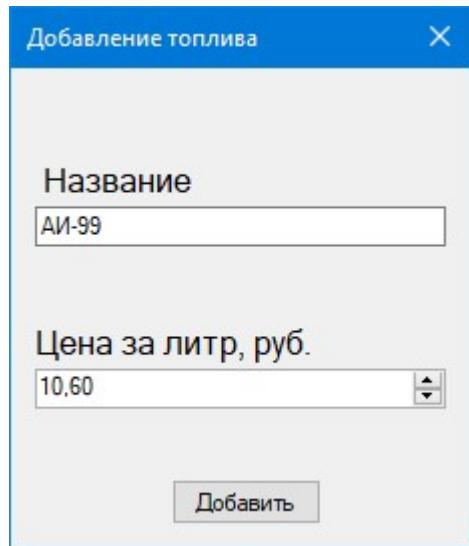


Рисунок А.12 – Экранная форма «Добавление топлива»

Если пользователь неправильно указывает параметры при добавлении топлива, то система выдает ошибку, с указанием соответствующего текста. На рисунке А.13 система ошибку пользователя при добавлении топлива.

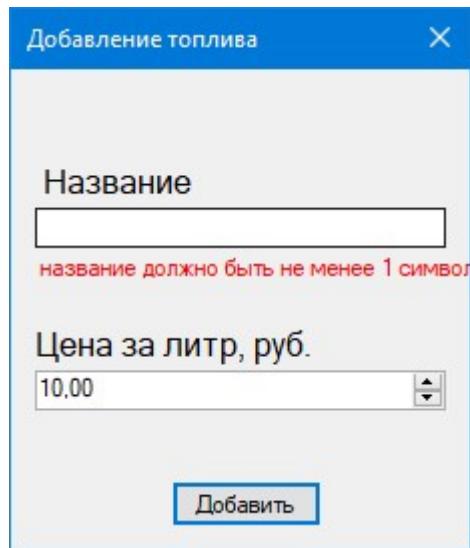


Рисунок А.13 – Ошибка добавления топлива

ШЭ «касса», «въезд» и «выезд» не имеют настраиваемых параметров.

При нажатии на кнопку «Смоделировать» система выполняет валидацию. Если валидация пройдет успешно, то осуществится переход к экранной форме «Настройка транспортного потока», иначе система выдаст сообщение об ошибке с указанием ее причины. На рисунках А.14–А.19 указаны различные ошибки валидации конструктора.

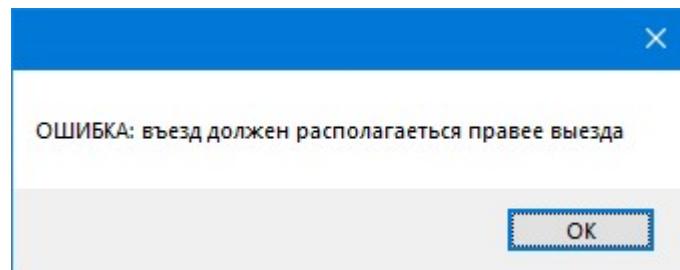


Рисунок А.14 – Ошибка валидации конструктора

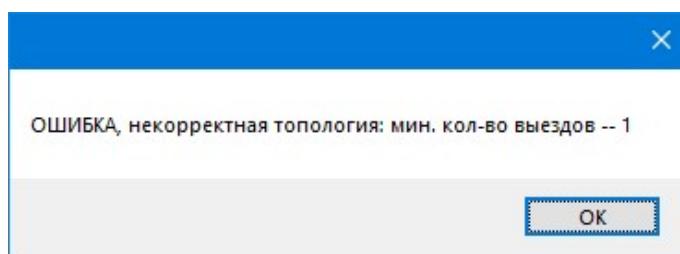


Рисунок А.15 – Ошибка валидации конструктора

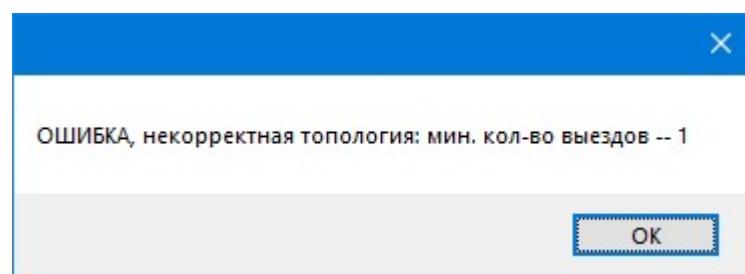


Рисунок А.16 – Ошибка валидации конструктора

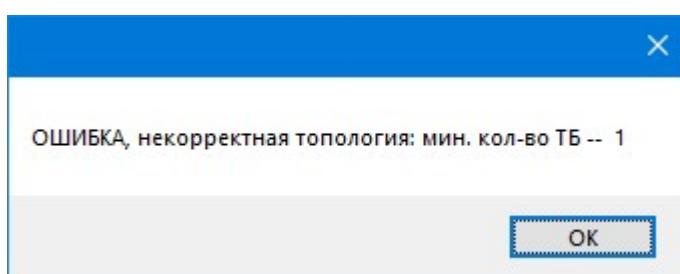


Рисунок А.17 – Ошибка валидации конструктора

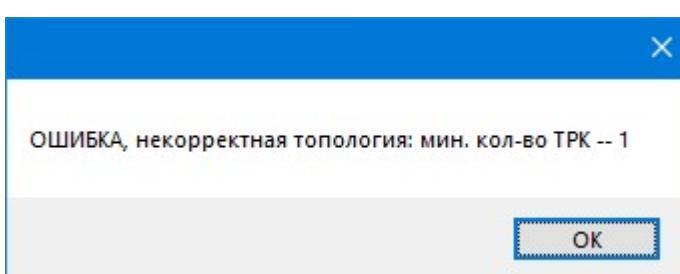


Рисунок А.18 – Ошибка валидации конструктора

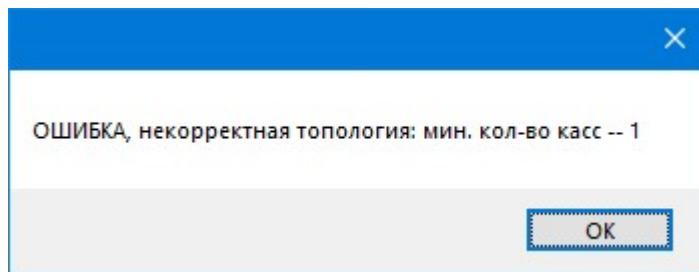


Рисунок А.19 – Ошибка валидации конструктора

A.4.3 Настройка транспортного потока

В экранной форме «Настройка транспортного потока» пользователь может настроить параметры транспортного потока или перейти к процессу моделирования с помощью нажатия кнопки «Смоделировать».

Пользователь может выбрать детерминированный или случайный поток с помощью выбора соответствующих радиокнопок.

Если выбран «Детерминированный поток», то система отображает поле «Время между появлением автомобилей, с». В данном поле пользователь может указать интервал времени, по истечению которого будут появляться новые машины в транспортном потоке. На рисунке А.20 изображена экранная форма настройки детерминированного потока.

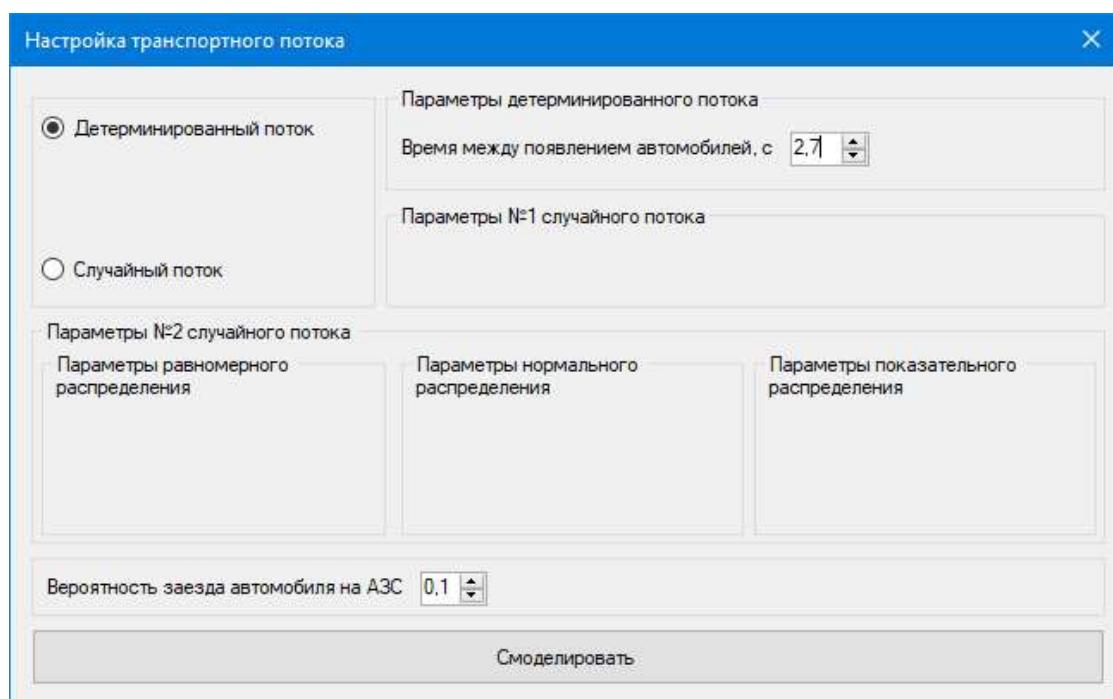


Рисунок А.20 – Экранная форма настройки детерминированного потока

Если выбран «Случайный поток», то система отобразит поле, где пользователю необходимо выбрать ЗР (равномерный, нормальный и показательный) и задать его параметры. На рисунках А.21–А.23 изображена настройка параметров ЗР.

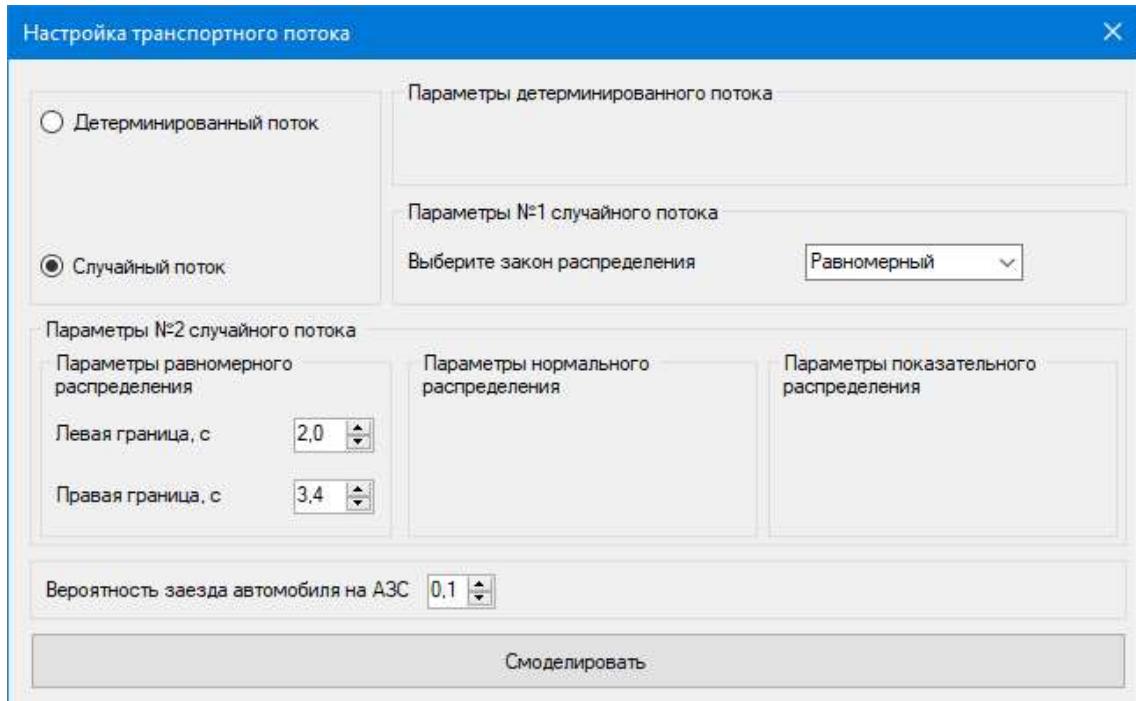


Рисунок А.21 – Экранная форма настройки равномерного ЗР

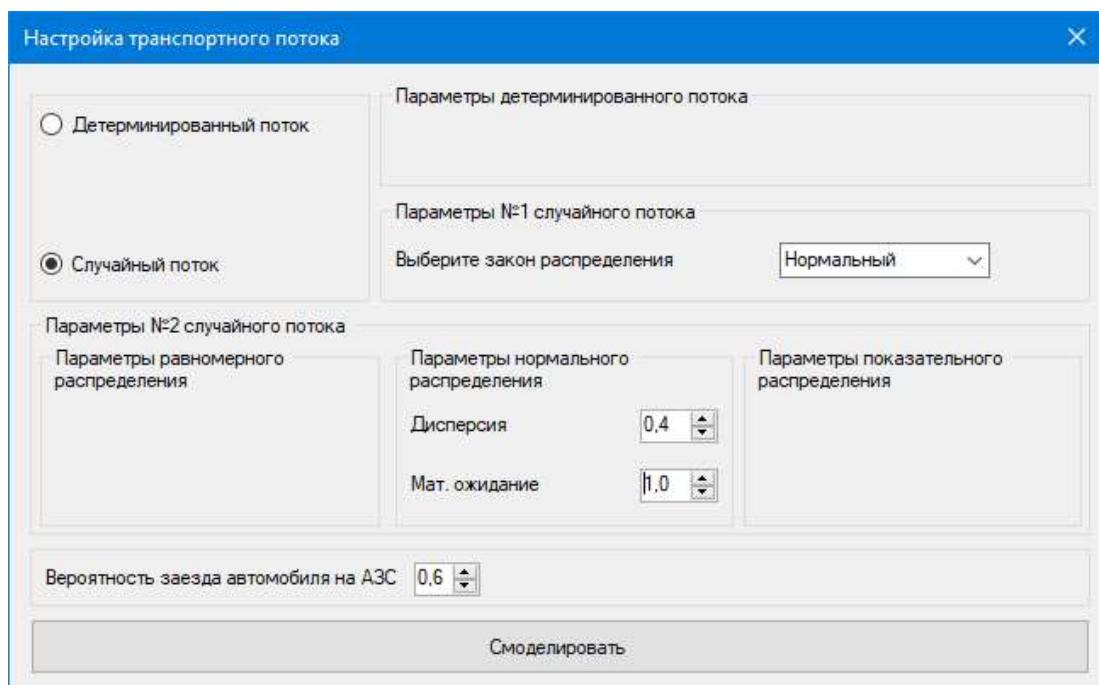


Рисунок А.22 – Экранная форма настройки нормального ЗР

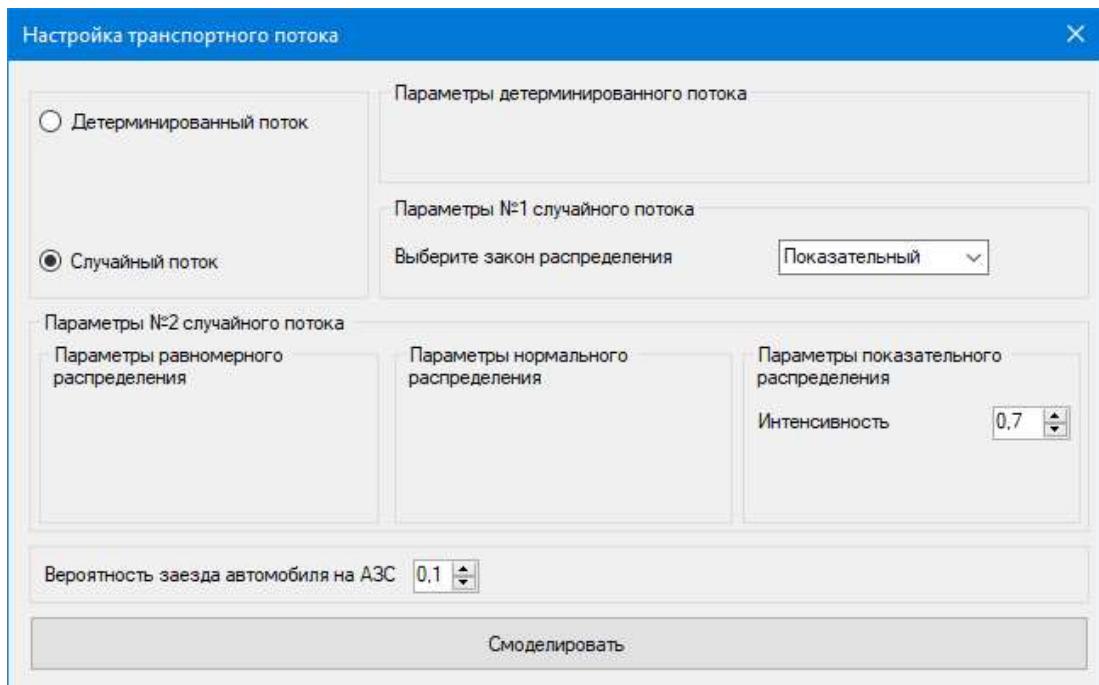


Рисунок А.23 – Экранная форма настройки показательного ЗР

Пользователь может менять параметр поля, который отвечает за вероятность заезда автомобиля на АЗС. На рисунке А.24 изображена экранная форма настройки данного параметра.

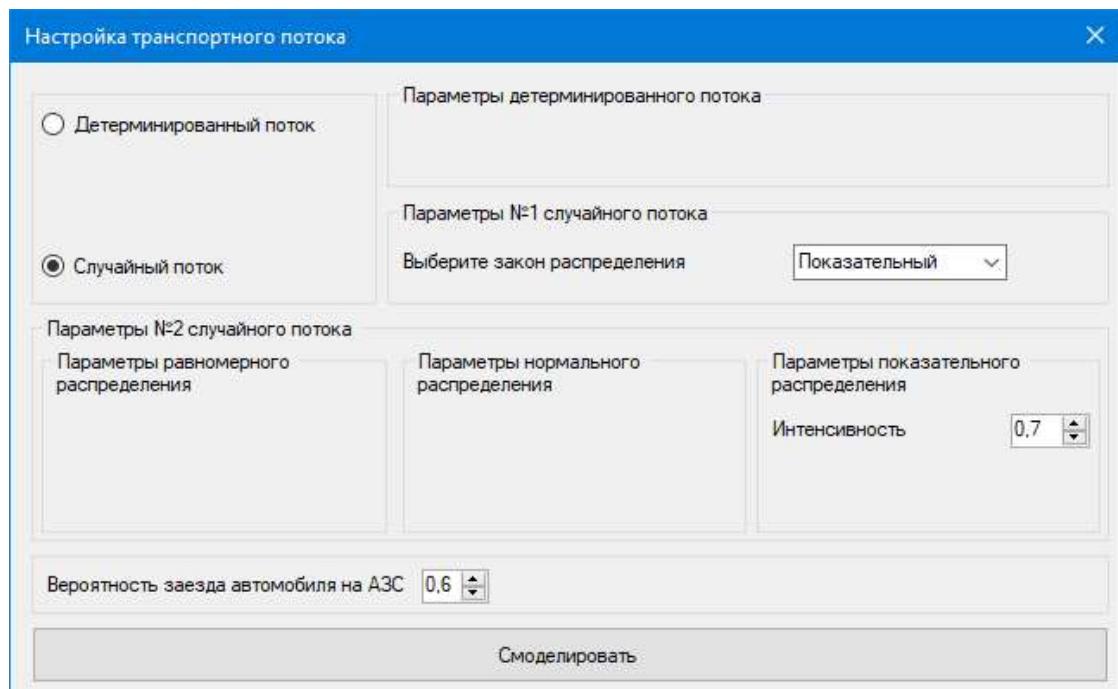


Рисунок А.24 – Экранная форма настройки вероятности заезда автомобиля на АЗС

Если указанные параметры введены верно, то при нажатии на кнопку «Смоделировать» система отобразит экранную форму «Моделирование», иначе система выдаст ошибку, в которой будет указано, какие параметры были указаны неверно.

A.4.4 Моделирование системы АЗС

Экранная форма «Моделирование» представляет моделирование работы АЗС с учетом указанных ранее настроек транспортного потока и выбранной топологии. В правой части экрана отображается состояние системы: «Активна» или «Пауза». Можно менять состояние процесса, с помощью нажатия на кнопки «►» или «||», которые расположены в нижней части окна. Данные кнопки меняют друг друга, в зависимости от того, в каком состоянии находится система в данный момент времени.

Под строкой состояния отображается информация о кассе, последнем выбранном ТРК и ТБ. Ниже отображается информация о выбранном пользователем элементе. В качестве элемента можно будет выбрать кассу, ТБ, ТРК, въезд, выезд, автомобиль, автомобиль инкасации, дозаправщика и сервисную зону. На рисунке А.25 изображено моделирование работы АЗС.

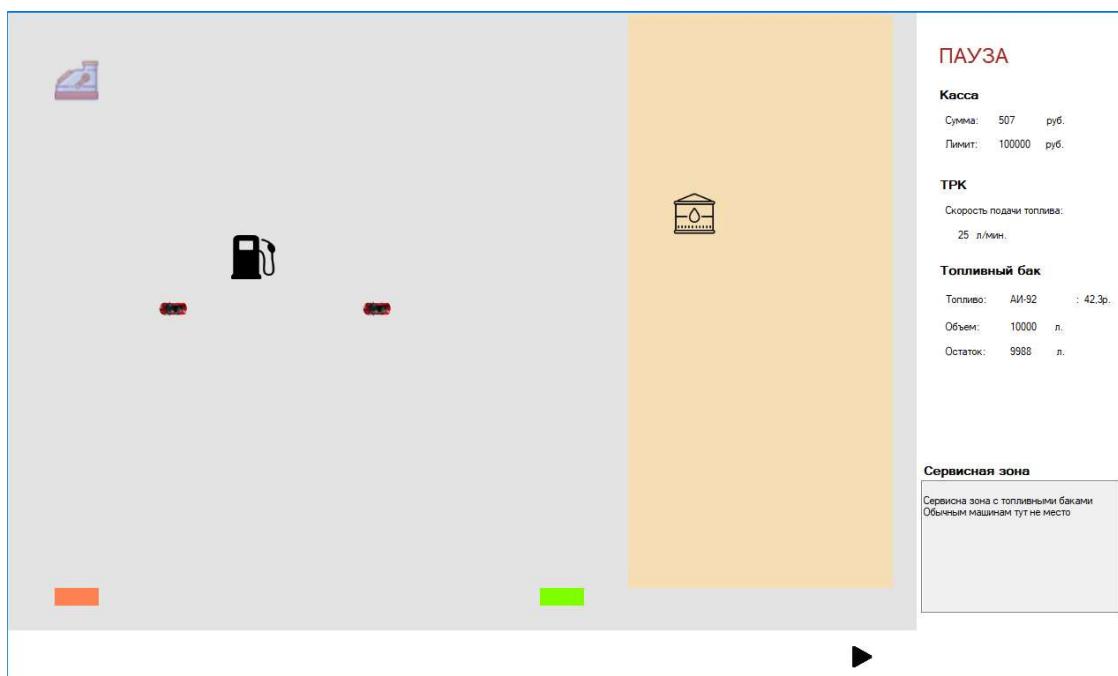


Рисунок А.25 – Моделирование работы АЗС

ПРИЛОЖЕНИЕ Б

Листинг модулей программы

```
using GasStationMs.App.TemplateElements;
using GasStationMs.App.Topology.TopologyBuilderHelpers;
using System;
using System.Windows.Forms;

namespace GasStationMs.App.Topology
{
    public partial class TopologyBuilder
    {
        private DataGridView field;
        private int serviceAreaInCells;
        private int serviceAreaBorderColIndex;

        public TopologyBuilder(DataGridView dgv)
        {
            field = dgv ?? throw new NullReferenceException();

            AddDgvCols(Topology.MinColsCount);
            field.RowCount = Topology.MinRowsCount;

            SetupDgv();
            serviceAreaInCells = RecalculateServiceArea();
            SetupServiceArea();
            SetupRoad();
        }

        public TopologyBuilder(DataGridView field, int cols, int rows)
        {
            this.field = field ?? throw new NullReferenceException();

            if (cols < Topology.MinColsCount ||
                cols > Topology.MaxColsCount)
            {
                throw new ArgumentOutOfRangeException();
            }

            if (rows < Topology.MinRowsCount ||
                rows > Topology.MaxRowsCount)
            {
                throw new ArgumentOutOfRangeException();
            }

            AddDgvCols(cols);
            this.field.RowCount = rows;

            SetupDgv();
            serviceAreaInCells = RecalculateServiceArea();
            SetupServiceArea();
            SetupRoad();
        }

        public TopologyBuilder(DataGridView field, Topology topology)
        {
            this.field = field ?? throw new NullReferenceException();

            if (topology == null)
                throw new NullReferenceException();
        }
    }
}
```

```

AddDgvCols(topology.ColsCount);
this.field.RowCount = topology.RowsCount;

SetupDgv();
serviceAreaInCells = RecalculateServiceArea();
SetupServiceArea();
SetupRoad();

SetField(topology);
}

private void SetupDgv()
{
    field.RowHeadersVisible = false;
    field.ColumnHeadersVisible = false;

    field.AllowUserToResizeColumns = false;
    field.AllowUserToResizeRows = false;

    field.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;
    field.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
}

private void AddDgvCols(int colsCount)
{
    for (int i = 0; i < colsCount; i++)
        field.Columns.Add(new BlankTopologyBuilderCol());
}

private void AddDgvRows(int rowsCount)
{
    for (int i = 0; i < rowsCount; i++)
        field.Rows.Add();
}

private void SetupServiceArea()
{
    int lastRowIndex = field.RowCount - 1;

    int cellsLeftToAdd = serviceAreaInCells;
    int cellsAdded = 0;

    DataGridViewImageCell cell;

    for (int currCol = field.ColumnCount - 1; currCol >= 0; currCol--)
    {
        for (int currRow = 0; currRow < lastRowIndex; currRow++)
        {
            cell = (DataGridViewImageCell)field.Rows[currRow].Cells[currCol];
            cell.Tag = new ServiceArea();
            cell.Value = ServiceArea.Image;

            cellsAdded++;
            cellsLeftToAdd--;
        }

        if (cellsLeftToAdd <= 0)
        {
            serviceAreaInCells = cellsAdded;
            serviceAreaBorderColIndex = currCol;
            break;
        }
    }
}

```

```

}

private void SetupRoad()
{
    DataGridViewImageCell cell;
    for (int currCol = 0, lastRow = RoadRowIndex; currCol < field.ColumnCount; currCol++)
    {
        cell = (DataGridViewImageCell)field.Rows[lastRow].Cells[currCol];
        cell.Tag = new Road();
        cell.Value = Road.Image;
    }
}

private void SetField(Topology topology)
{
    field.ColumnCount = topology.ColsCount;
    field.RowCount = topology.RowsCount;

    SetupServiceArea();
    SetupRoad();

    IGasStationElement gse;
    for (int y = 0; y <= topology.LastY; y++)
        for (int x = 0; x <= topology.LastX; x++)
        {
            gse = topology[x, y];

            if (gse == null)
                AddBlank(x, y);
            else if (gse is CashCounter cashCounter)
                AddCashCounter(x, y, cashCounter);
            else if (gse is Entry entry)
                AddEntry(x, y, entry);
            else if (gse is Exit exit)
                AddExit(x, y, exit);
            else if (gse is FuelDispenser fuelDispenser)
                AddFuelDispenser(x, y, fuelDispenser);
            else if (gse is FuelTank fuelTank)
                AddFuelTank(x, y, fuelTank);
            else if (gse is Road road)
                AddRoad(x, y, road);
            else if (gse is ServiceArea serviceArea)
                AddServiceArea(x, y, serviceArea);
            else
                throw new InvalidCastException();
        }
}

private void AddBlank(int x, int y)
{
    DataGridViewImageCell cell = (DataGridViewImageCell)field.Rows[y].Cells[x];

    cell.Value = null;
    cell.Tag = null;
}

private void AddRoad(int x, int y)
{
    DataGridViewImageCell cell = (DataGridViewImageCell)field.Rows[y].Cells[x];

    cell.Value = Road.Image;
    cell.Tag = new Road();
}

```

```

private void AddRoad(int x, int y, Road road)
{
    DataGridViewImageCell cell = (DataGridViewImageCell)field.Rows[y].Cells[x];
    cell.Value = Road.Image;
    cell.Tag = road;
}

private void AddServiceArea(int x, int y)
{
    DataGridViewImageCell cell = (DataGridViewImageCell)field.Rows[y].Cells[x];
    cell.Value = ServiceArea.Image;
    cell.Tag = new ServiceArea();
}

private void AddServiceArea(int x, int y, ServiceArea serviceArea)
{
    DataGridViewImageCell cell = (DataGridViewImageCell)field.Rows[y].Cells[x];
    cell.Value = ServiceArea.Image;
    cell.Tag = serviceArea;
}

public int ColsCount
{
    get
    {
        return field.ColumnCount;
    }
}

public int RowsCount
{
    get
    {
        return field.RowCount;
    }
}

public int RoadRowIndex
{
    get
    {
        return field.Rows.GetLastRow(DataGridViewElementStates.Visible);
    }
}

private int RecalculateServiceArea()
{
    return (int)(RowsCount * ColsCount * Topology.ServiceAreaInShares);
}

public Topology ToTopology()
{
    CheckTopologyCorrectness();

    IGasStationElement[,] gseArr = new IGasStationElement[RowsCount, ColsCount];

    DataGridViewImageCell cell;
    for (int currRow = 0; currRow < gseArr.GetLength(0); currRow++)
    {

```

```

        for (int currCol = 0; currCol < gseArr.GetLength(1); currCol++)
    {
        cell = (DataGridViewImageCell)field.Rows[currRow].Cells[currCol];
        gseArr[currRow, currCol] = (IGasStationElement)cell.Tag;
    }
}

return new Topology(gseArr, serviceAreaBorderColIndex);
}

private void CheckTopologyCorrectness()
{
    CheckMinNumOfTemplatesElements();
    CheckMaxNumOfTemplatesElements();
    CheckEntryAndExitOrder();
}

private void CheckMinNumOfTemplatesElements()
{
    if (cashCountersCount < Topology.MinCashCountersCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: мин. кол-во касс -- \" + Topology.MinCashCountersCount);

    if (EntriesCount < Topology.MinEntriesCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: мин. кол-во въездов -- \" + Topology.MinEntriesCount);

    if (ExitsCount < Topology.MinExitsCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: мин. кол-во выездов -- \" + Topology.MinExitsCount);

    if (FuelDispensersCount < Topology.MinFuelDispensersCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: мин. кол-во ТРК -- \" + Topology.MinFuelDispensersCount);

    if (FuelTanksCount < Topology.MinFuelTanksCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: мин. кол-во ТБ -- \" + Topology.MinFuelTanksCount);
}

private void CheckMaxNumOfTemplatesElements()
{
    if (cashCountersCount > Topology.MaxCashCountersCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: макс. кол-во касс -- \" + Topology.MaxCashCountersCount);

    if (EntriesCount > Topology.MaxEntriesCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: макс. кол-во въездов -- \" + Topology.MaxEntriesCount);

    if (ExitsCount > Topology.MaxExitsCount)
        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: макс. кол-во выездов -- \" + Topology.MaxExitsCount);

    if (FuelDispensersCount > Topology.MaxFuelDispensersCount)

```

```

        throw new InvalidOperationException(
            "ОШИБКА, некорректная топология: макс. кол-во ТРК -- '' +"
Topology.MaxFuelDispensersCount);

        if (FuelTanksCount > Topology.MaxFuelTanksCount)
            throw new InvalidOperationException(
                "ОШИБКА, некорректная топология: макс. кол-во ТБ -- '' +"
Topology.MaxFuelTanksCount);
    }

private void CheckEntryAndExitOrder()
{
    int roadRowIndex = RoadRowIndex;
    int entryColIndex = -1;
    int exitColIndex = -1;

    DataGridViewImageCell cell;
    for (int currCol = 0; currCol < serviceAreaBorderColIndex; currCol++)
    {
        cell = (DataGridViewImageCell)field.Rows[roadRowIndex].Cells[currCol];

        if (cell.Tag is Entry)
        {
            entryColIndex = currCol;
            continue;
        }
        else if (cell.Tag is Exit)
        {
            exitColIndex = currCol;
            continue;
        }
    }

    if (entryColIndex == -1)
        throw new NullReferenceException("ОШИБКА: нет въезда");

    if (exitColIndex == -1)
        throw new NullReferenceException("ОШИБКА: нет выезда");

    if (entryColIndex < exitColIndex)
        throw new InvalidOperationException("ОШИБКА: въезд должен располагаться правее
выезда");
    }
}
}

using System;
using System.Drawing;
using GasStationMs.App.Forms;
using GasStationMs.App.Modeling.Models.PictureBoxes;
using GasStationMs.App.Modeling.Models.Views;
using System.Text;
using System.Windows.Forms;

namespace GasStationMs.App.Modeling
{
    internal static class ClickEventProvider
    {
        private static ModelingForm _modelingForm;
        private static Label _labelSelectedElement;
        private static TextBox _textBoxSelectedItemInformation;

        private static PictureBox _buttonPausePlay;
        private static Label _labelModelState;
    }
}

```

```

internal static void SetUpClickEventProvider(ModelingForm modelingForm)
{
    _modelingForm = modelingForm;
    _labelSelectedElement = modelingForm.LabelSelectedElement;
    _textBoxSelectedItemInformation = modelingForm.TextBoxSelectedItemInformation;

    _buttonPausePlay = modelingForm.ButtonPausePlay;
    _labelModelState = modelingForm.LabelModelState;
}
internal static void CarPictureBox_Click(object sender, MouseEventArgs e)
{
    var car = (CarPictureBox)sender;
    var carView = (CarView)car.Tag;
    var sumToPay = (carView.DesiredFilling - carView.FuelRemained) * carView.Fuel.Price;

    // textBoxSelectedItemInformation.Text = "";
    _labelSelectedElement.Text = "Автомобиль";

    StringBuilder carInfo = new StringBuilder();

    carInfo.Append("Топливо: " + carView.Fuel.Name);
    carInfo.Append("\r\nСтоимость: " + carView.Fuel.Price);
    carInfo.Append("\r\nОбъем бака: " + carView.TankVolume);
    carInfo.Append("\r\nТоплива в баке: "
        + (int)carView.FuelRemained);
    if (car.IsGoesFilling)
    {
        carInfo.Append("\r\nЗаплатит денег: " + sumToPay);
    }

    _textBoxSelectedItemInformation.Text = carInfo.ToString();
    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = car;
}

internal static void FuelDispenserPictureBox_Click(object sender, MouseEventArgs e)
{
    var fuelDispenser = (PictureBox)sender;
    var fuelDispenserView = (FuelDispenserView)fuelDispenser.Tag;

    _labelSelectedElement.Text = "TPK";

    StringBuilder fuelDispenserInfo = new StringBuilder();

    fuelDispenserInfo.Append("\r\nСкорость подачи топлива: " +
        fuelDispenserView.SpeedOffillingPerMinute +
        " литров/мин.");

    _textBoxSelectedItemInformation.Text = fuelDispenserInfo.ToString();

    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = fuelDispenser;
    _modelingForm.SelectedFuelDispenser = fuelDispenser;
}

internal static void FuelTankPictureBox_Click(object sender, MouseEventArgs e)
{
}

```

```

var fuelTank = (PictureBox)sender;
var fuelTankView = (FuelTankView)fuelTank.Tag;

_labelSelectedElement.Text = "Топливный бак";

StringBuilder fuelTankInfo = new StringBuilder();

fuelTankInfo.Append("\r\nОбъем: " + fuelTankView.Volume + " литров");
fuelTankInfo.Append("\r\nТопливо: " + fuelTankView.Fuel);
fuelTankInfo.Append("\r\nОстаток: " + (int)fuelTankView.CurrentFullness + " литров");

_textBoxSelectedItemInformation.Text = fuelTankInfo.ToString();

_labelSelectedElement.Visible = true;
_textBoxSelectedItemInformation.Visible = true;

_modelingForm.SelectedItem = fuelTank;
_modelingForm.SelectedFuelTank = fuelTank;
}

internal static void CashCounterPictureBox_Click(object sender, MouseEventArgs e)
{
    var cashCounter = (PictureBox)sender;
    var cashCounterView = (CashCounterView)cashCounter.Tag;

    _labelSelectedElement.Text = "Касса";

    StringBuilder cashCounterInfo = new StringBuilder();

    cashCounterInfo.Append("\r\nСумма: " + (int)cashCounterView.CurrentCashVolume + " руб.");
    cashCounterInfo.Append("\r\nЛимит кассы: " + cashCounterView.MaxCashVolume + " руб.");

    //cashCounterInfo.Append("\r\nIsFull: " + cashCounterView.IsFull);
    //cashCounterInfo.Append("\r\n-----");

    _textBoxSelectedItemInformation.Text = cashCounterInfo.ToString();

    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = cashCounter;
}

internal static void CashCollectorPictureBox_Click(object sender, MouseEventArgs e)
{
    var cashCollector = (CollectorPictureBox)sender;
    var cashCollectorView = cashCollector.Tag as CollectorView;

    _labelSelectedElement.Text = "Инкасация";

    StringBuilder cashCollectorInfo = new StringBuilder();

    cashCollectorInfo.Append("\r\nИзъято денег: " + (int)cashCollectorView.TakenCashVolume + " руб.");

    _textBoxSelectedItemInformation.Text = cashCollectorInfo.ToString();

    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = cashCollector;
}

```

```

internal static void RefuellerPictureBox_Click(object sender, MouseEventArgs e)
{
    var refueller = (RefuellerPictureBox)sender;
    var refuellerView = refueller.Tag as RefuellerView;

    _labelSelectedElement.Text = "Дозаправщик";

    StringBuilder cashCollectorInfo = new StringBuilder();

    cashCollectorInfo.Append("\r\nОсталось топлива: " + (int)refuellerView.FuelRemaining + " л.");

    _textBoxSelectedItemInformation.Text = cashCollectorInfo.ToString();

    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = refueller;
}

internal static void EnterPictureBox_Click(object sender, MouseEventArgs e)
{
    var enter = (PictureBox)sender;

    StringBuilder enterInfo = new StringBuilder();

    _labelSelectedElement.Text = "Въезд";

    enterInfo.Append("\r\nЗдесь автомобили въезжают на АЗС");

    _textBoxSelectedItemInformation.Text = enterInfo.ToString();

    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = enter;
}

internal static void ExitPictureBox_Click(object sender, MouseEventArgs e)
{
    var exit = (PictureBox)sender;

    StringBuilder exitInfo = new StringBuilder();

    _labelSelectedElement.Text = "Выезд";

    exitInfo.Append("\r\nЗдесь автомобили выезжают с АЗС");

    _textBoxSelectedItemInformation.Text = exitInfo.ToString();

    _labelSelectedElement.Visible = true;
    _textBoxSelectedItemInformation.Visible = true;

    _modelingForm.SelectedItem = exit;
}

internal static void ServiceArea_Click(object sender, MouseEventArgs e)
{
    var serviceArea = (PictureBox)sender;

    StringBuilder serviceAreaInfo = new StringBuilder();

    _labelSelectedElement.Text = "Сервисная зона";
}

```

```

serviceAreaInfo.Append("\r\nСервисна зона с топливными баками");
serviceAreaInfo.Append("\r\nОбычным машинам тут не место");

(textBoxSelectedItemInformation.Text = serviceAreaInfo.ToString());

_labelSelectedElement.Visible = true;
 textBoxSelectedItemInformation.Visible = true;

 modelingForm.SelectedItem = serviceArea;
}

internal static void PlaygrounPanel_Click(object sender, MouseEventArgs e)
{
 _labelSelectedElement.Visible = false;
 textBoxSelectedItemInformation.Visible = false;
}

internal static void PictureBoxPauseAndPlay_Click(object sender, EventArgs e)
{
 var isPaused = ModelingTimeManager.IsPaused;

 if (!isPaused)
 {
 ModelingTimeManager.IsPaused = true;
 buttonPausePlay.Image = Properties.Resources.Play;
 _labelModelState.Text = "ПАУЗА";
 _labelModelState.ForeColor = Color.Brown;
 }
 else
 {
 ModelingTimeManager.IsPaused = false;
 buttonPausePlay.Image = Properties.Resources.Pause;
 _labelModelState.Text = "АКТИВНА";
 _labelModelState.ForeColor = Color.Green;
 }

 _labelModelState.Refresh();
}
}
}

```