

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

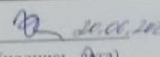
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

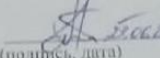
на тему

**ПРОГРАММА РАСЧЁТА ХАРАКТЕРИСТИК
СТОХАТИЧЕСКИХ СЕТЕЙ МАССОВОГО
ОБСЛУЖИВАНИЯ**

Студент

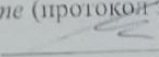
 20.06.2025 Репин И.В.
(подпись, дата)

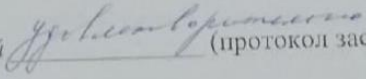
Руководитель

 Бикташев Р.А.
(подпись, дата)

Нормоконтролёр

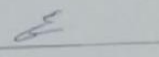
 23.06.2025 Бычков А.С.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025г. № 16)
Заведующий кафедрой  М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от

24 июня 2025 № 18)

Секретарь ГЭК

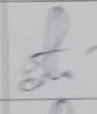
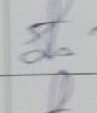
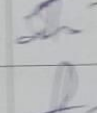
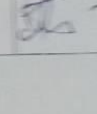
 Е.И. Гурий
(подпись) (фамилия, инициалы)

Пенза 2025

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить аналоги существующих программ расчёта стохастических систем, изучить актуальные способы сохранения данных о проекте и построения диаграмм, изучить способы проектирования и разработки кроссплатформенного приложения. Также необходимо изучить функционал библиотек GTK+, Cairo.

6. Календарный график выполнения работы

№п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	06.05.2025 08.05.2025	
2.	Анализ требований	10	09.05.2025 10.05.2025	
3.	Выбор технологий, применяемых при проектировании и разработке	50	11.05.2025 13.05.2025	
4.	Описание алгоритмов	40	13.05.2025 15.05.2025	
5.	Разработка приложения «Программа расчёта характеристик стохастических сетей массового обслуживания»	150	16.05.2025 27.05.2025	
6.	Оформление пояснительной записки	24	27.05.2025 30.05.2025	

Дата выдачи задания "06" 05 2025г.

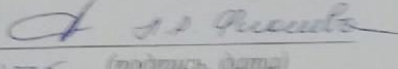
Руководитель бакалаврской работы

 06.05.25
(подпись, дата)

Задание к исполнению принял студент

 06.05.25
(подпись, дата)

Работу к защите допустить
Декан факультета

 23.06.2025 (подпись, дата)

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 02 » 05 20 25 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Ренин Иван Вадимович, 21ВВВ1 факультета вычислительной техники

2. Тема работы: «Программа расчёта характеристик стохастических сетей массового обслуживания»

Тема утверждена приказом ПГУ № 601/о от "6" мая 2025г.

3. Руководитель бакалаврской работы к.т.н., доцент Бикташев Р.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Программа расчёта характеристик стохастических сетей массового обслуживания», выполняющее расчёт стохастических сетей массового обслуживания и вывод характеристик в виде отчёта, диаграмм и графиков. Приложение должно быть кроссплатформенным, поддерживать как OS Windows, так и Linux. Программа должна обладать графическим интерфейсом, позволяющим редактировать систему как с помощью графических элементов, так и с помощью матрицы вероятностей передач. Система должна включать в себя функционал сохранения и импорта проекта, расчёта основных характеристик сети, а также вывода результата в файл и в виде диаграмм и графов.

Реферат

Бакалаврская работа содержит 150 листов, 38 рисунков, 7 таблиц, 7 источников и 1 приложение.

**СТОХАСТИЧЕСКИЕ СЕТИ, МАССОВОЕ ОБСЛУЖИВАНИЕ, ТЕОРИЯ
ОЧЕРЕДЕЙ, РАСЧЁТ, ВРЕМЯ ОЖИДАНИЯ, ПРОПУСКНАЯ
СПОСОБНОСТЬ, СИМУЛЯЦИЯ**

Цель работы – разработать программу для расчёта основных характеристик стохастических сетей массового обслуживания, позволяющее моделировать процессы обслуживания и анализировать показатели эффективности системы, такие как число заявок в очередях, число заявок в сети, время ожидания в очередях, время пребывания заявки в сети.

Объектом разработки является пользовательское приложение для расчёта характеристик стохастических сетей массового обслуживания, включая моделирование процессов и вычисление ключевых показателей.

В результате проделанной работы разработано приложение, полностью соответствующее поставленной задаче, а именно: реализован графический интерфейс для создания и редактирования стохастических сетей массового обслуживания, обеспечена возможность задания параметров узлов и переходов между ними, реализованы алгоритмы расчёта ключевых характеристик сетей с учётом вариаций параметров, а также предусмотрена функция сохранения и загрузки проектов в формате JSON. Приложение позволяет пользователю интерактивно моделировать сети, выполнять вычисления и визуализировать результаты в виде отчётов и диаграмм.

					ПГУ 1.090301.18.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Репин И.В.		23.06.25	Программа расчёта характеристик стохастических сетей массового обслуживания Пояснительная записка		
Провер.		Бикташев Р.А.		23.06.25			
Н. Контр		Бычков А.С.		23.06.25			
Утверд.		Митрохин М.А.		23.06.25			
					Лит.	Лист	Листов
						2	
					ФВТ, каф. ВТ		
					гр. 21ВВВ1		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Программа расчёта характеристик стохастических сетей массового обслуживания	7
1.1 Анализ предметной области.....	7
1.1.1 Обзор аналогов.....	8
1.1.2 Анализ требований к разработке.....	9
1.1.3 Выбор архитектуры приложения	10
1.1.4 Выбор языка программирования.....	12
1.1.5 Выбор библиотеки для разработки кроссплатформенного приложения	13
1.1.6 Хранение данных.....	15
1.1.7 Выбор библиотеки для работы с JSON.....	16
1.1.8 Выбор библиотеки для реализации графического модуля.....	17
1.1.9 Выбор среды разработки.....	19
1.2 Проектирование и разработка	21
1.2.1 Реализация сериализации данных.....	21
1.2.2 Реализация графической части.....	22
1.2.3 Разработка структур.....	24
1.2.4 Разработка расчётной части	25
1.2.5 Разработка построения диаграмм	28
1.3 Анализ результатов	31
1.3.1 Проверка результатов расчётов	32
1.3.2 Клиентская программа.....	35
2 Охрана труда	48
2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ	48

2.2 Классификация и категорирование помещения, в котором производится разработка	48
2.2.1 Классификация по взрывопожарной и пожарной опасности	48
2.2.2 Классификация по опасности поражения электрическим током	49
2.3 Санитарно-гигиеническая характеристика производства	50
2.4 Расчет контура заземления производственного помещения, где установлены ПЭВМ.....	51
Заключение.....	56
Список использованных источников.....	57
Приложение А – UML-диаграммы	58
Приложение Б – Листинг программы.....	63
Приложение В – Презентация	136

Перечень сокращений и обозначений

ООП – объектно-ориентированное программирование

ПЭВМ – персональная электронно-вычислительная машина

СМО – сеть массового обслуживания

ЭВМ – электронно-вычислительная машина

GUI (Graphical User Interface) – графический пользовательский интерфейс

OS (Operating System) – операционная система

UML (Unified Modeling Language) – унифицирован язык моделирования

					ПГУ 1.090301.18.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Современные информационные системы и сети коммуникации стремительно развиваются, что приводит к росту объёмов обрабатываемых заявок и усложнению структуры систем массового обслуживания (СМО).

Сеть массового обслуживания представляет собой совокупность конечного числа N обслуживающих узлов [1]. Эффективное моделирование и анализ таких систем является важной задачей для оптимизации работы серверов, сетевых устройств, служб поддержки и других областей, где требуется управление потоками заявок и очередями.

Актуальность темы обусловлена необходимостью создания удобных и наглядных инструментов для построения, моделирования и расчёта параметров стохастических сетей, которые позволят инженерам и программистами быстро оценивать характеристики систем и принимать решения по их оптимизации.

Цель работы – разработать программу для расчёта основных характеристик стохастических сетей массового обслуживания, позволяющее моделировать процессы обслуживания и анализировать показатели эффективности системы, такие как число заявок в очередях, число заявок в сети, время ожидания в очередях, время пребывания заявки в сети.

В рамках работы необходимо реализовать программу построения и редактирования сетей СМО, алгоритмы расчёта ключевых характеристик систем, а также обеспечить возможность редактирования и сохранение данных в формате JSON.

На сегодняшний день существует множество разработок и программ для анализа СМО, однако многие из них либо ориентированы на узконаправленные задачи, либо имеют сложный интерфейс, либо являются коммерческими продуктами без поддержки кроссплатформенности.

С целью закрытия этой позиции на нише программного обеспечения было принято решение разработать собственное приложение.

1 Программа расчёта характеристик стохастических сетей массового обслуживания

1.1 Анализ предметной области

Системы массового обслуживания широко применяются в различных областях — от компьютерных сетей до логистики и сервисных центров. Они позволяют моделировать процессы поступления заявок, их обработки и ожидания в очередях, что можно использовать для оценки производительности и качества обслуживания. В современном мире эффективное управление потоками заявок и оптимизация ресурсов становится критически важным для обеспечения стабильной работы сервисов и улучшения пользовательского опыта.

Стохастические СМО представляют собой обобщённую модель, включающую несколько связанных между собой СМО, что позволяет анализировать сложные системы с различными вариациями параметров. Такие модели помогают прогнозировать поведение системы при изменении нагрузки, конфигурации или параметров обслуживания, что важно для принятия управленческих решений.

На сегодняшний день существует множество разработок и программ для анализа СМО, однако многие из них либо ориентированы на узконаправленные задачи, либо имеют сложный интерфейс, затрудняющий визуализацию и настройку моделей, либо являются коммерческими продуктами без поддержки кроссплатформенности.

С целью закрытия этой позиции на нише программного обеспечения было принято решение разработать собственное кроссплатформенное приложение с графическим редактором и интуитивно понятным интерфейсом. Такое приложение, будучи бесплатным к использованию, будет выгодно выигрывать у конкурентов.

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

1.1.1 Обзор аналогов

Вместе с развитием информационных технологий, развивались и математические методы расчёта стохастических цепей. На данный момент существует огромное количество программ, позволяющих пользователю рассчитывать самые сложные СМО. В данном разделе мы рассмотрим несколько из этих программ и детально проанализируем их функционал.

Начнем с наиболее популярных приложений: Arena и AnyLogic. Arena предоставляет различные средства для моделирования сетей, такие как поддержка сетей массового обслуживания, удобным графический редактор, интеграция с таблицами Excel, имитационное моделирование, анализ производительности, визуализация расчётов. Однако, из минусов стоит отметить её коммерческую лицензию. AnyLogic отличается гибкостью и возможностью моделировать гибридные системы, сочетая агентное и событийное моделирование, что полезно для систем с множественными потоками заявок.

Среди открытых решений выделяется SimPy — библиотека на Python для моделирования дискретных событий, включая СМО. SimPy ориентирован на программирование моделей и подходит для пользователей, комфортно работающих с кодом, однако не имеет визуального редактора и требует самостоятельной реализации интерфейса.

Другим примером является OMNeT++ — платформа с открытым исходным кодом для моделирования сетевых и распределённых систем, включающая модули для массового обслуживания. OMNeT++ предлагает мощные средства визуализации и расширяемость, но требует значительного времени на освоение.

Существуют и специализированные инструменты, например, QtsPlus, ориентированные на анализ конкретных классов сетей массового обслуживания с удобным графическим

интерфейсом и возможностью экспорта данных в стандартных форматах.

Однако большинство существующих решений либо сложны в освоении, либо являются коммерческими продуктами.

Таким образом, разработка клиентского приложения с понятным графическим интерфейсом и поддержкой визуального построения моделей позволит заполнить существующую нишу.

Сравнение возможностей решений разных компаний представлена в таблице 1.1.

Таблица 1.1 – Сравнение различных решений разных компаний

Название	Тип лицензии	Интерфейс	Визуальное моделирование	Платформа
Arena	Коммерческая	Графический	Да	Windows
AnyLogic	Коммерческая	Графический	Да	Windows/ Linux
SimPy	Открытая	Код	Нет	Windows/ Linux
OMNeT++	Открытая	Смешанный	Частично	Windows/ Linux
QtsPlus	Коммерческая	Графический	Да	Windows

1.1.2 Анализ требований к разработке

Создание Open Source решения направлено на предоставление широкому кругу пользователей возможности эффективно создавать, визуализировать и рассчитывать характеристики сетей с различными параметрами.

Открытый исходный код обеспечит прозрачность алгоритмов, позволит пользователям адаптировать и расширять функциональность под свои особые задачи, а разработчикам — совместно улучшать производительность и точность расчетов.

Программа должна предоставлять графический интерфейс, позволяющий создавать и редактировать структуру сети (узлы и связи с вероятностями переходов), задавать параметры СМО (время обслуживания, число каналов, размер очереди и приоритеты), выполнять расчёт характеристик сети с учётом различных вариантов параметров, визуализировать результаты в виде графиков и таблиц, сохранять и загружать проекты для дальнейшей работы.

В качестве основы для реализации выбран язык программирования C++ с использованием библиотеки GTK для создания кроссплатформенного графического интерфейса и сохранения данных.

Для успешной реализации поставлены следующие задачи:

- изучить основные модели и алгоритмы расчёта характеристик СМО, включая модели M/M/m, M/M/1, M/M/m/k [2];
- разработать удобный GUI для создания и редактирования сети и параметров СМО;
- реализовать вычислительные модули для расчёта параметров сети с возможностью варьирования входных данных по циклам;
- обеспечить сохранение и загрузку проектов в удобном формате;
- реализовать визуализацию результатов расчётов;
- провести тестирование программы.

1.1.3 Выбор архитектуры приложения

Разработка приложения для расчета характеристик СМО требует соблюдения следующих требований: производительность, интуитивно-понятный интерфейс, кроссплатформенность для работы на OS windows и Linux.

Таким образом можно прийти к выводу, что оптимальный вариант - модульная архитектура [3].

Принципиально важно, чтобы набор модулей включал в себя следующие:

- модельный модуль;
- модуль приемы и проверки параметров;
- вычислительный модуль;
- модуль вывода диаграмм;
- пользовательский интерфейс.

Таким образом, проект будет представлять собой несколько отдельно сохраненных файлов проекта, соединяющихся в единый проект. Тем самым кратно обеспечивается понятность и чистота кода, облегчается его редактирование и изменение, а также упростится редактирование и добавление функционала.

На рисунке 1 представлена архитектура приложения.

					ПГУ 1.090301.18.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

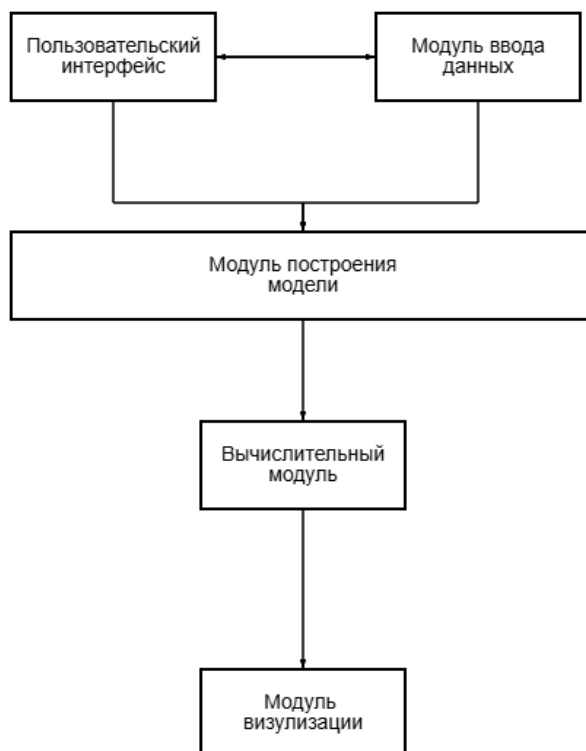


Рисунок 1 – Архитектура приложения

1.1.4 Выбор языка программирования

В современном мире существует множество языков программирования. При выборе языка для разработки программы расчета характеристик СМО следует уделить внимание таким параметрам как быстродействие, доступность и разнообразие библиотек.

Python — интерпретируемый язык программирования с простым синтаксисом. Python имеет множество библиотек, которые упрощают разработку сложных вычислительных систем с большим количеством данных. Поскольку он является интерпретируемым языком, его скорость ожидаемо уступает компилируемым, что значит, что он не является лучшим выбором для поставленной задачи.

C++ — компилируемый язык. Достаточно быстрый язык программирования с поддержкой кроссплатформенности. Однако он довольно сложен в усвоении и разработка на нем требует дополнительного обучения и навыков программирования [4].

C# — компилируемый язык, предлагающий разнообразные

инструменты для разработки программ. Хотя он уступает C++ в скорости, он гораздо удобнее. Однако для данной задачи C# не так хорош, поскольку у него нет большого количества библиотек, которые могли бы помочь для реализации поставленной задачи, а также у него весьма ограниченная кроссплатформенность.

Java представляет собой одновременно компилируемый и интерпретируемый язык программирования. Поддерживает кроссплатформенность и имеет большое количество различных библиотек. Но является далеко не самым быстрым языком, а потому не подходит для решения поставленной задачи.

Go (Golang) является компилируемым языком, разработанный Google. Довольно молодой язык, а потому не имеет большого количества библиотек и имеет проблемы с кроссплатформенностью.

Таким образом, C++ идеально подходит для поставленной задачи из-за высокой производительности и поддержки кроссплатформенности.

1.1.5 Выбор библиотеки для разработки кроссплатформенного приложения

Кроссплатформенность всегда была преимуществом приложения, по этой причине существует множество библиотек, позволяющих реализовать кроссплатформенность, самые популярные из них: Qt, wxWidgets и GTK+. Рассмотрим особенность каждой из них и определим наиболее подходящую.

Qt является широко используемой библиотекой, поддерживающей язык программирования C++. Эта библиотека предоставляет пользователю возможность работать с 2D и 3D графикой, удобно редактировать интерфейс приложения. Из минусов можно отметить наличие платной подписки, что делает эту библиотеку не подходящей для использования в данном проекте.

wxWidgets — это еще одна кроссплатформенная библиотека, которая позволяет создавать приложения для OS Windows, macOS и Linux. За счёт того, что библиотека использует элементы управления операционной системы, программы, написанные на ней, ведут себя стандартно независимо от платформы. Однако данная библиотека может быть сложна в освоении, а также она не самая популярная, что значит, при возникновении проблем при написании программы, помощь будет найти сложнее.

GTK+ — это библиотека с открытым исходным кодом, поддерживает кроссплатформенность, а именно работу с OS Windows и Linux. Достаточно популярна, что значит, что помощь при возникновении проблем будет найти проще. Имеет высокую интеграцию с другими библиотеками GNOME, имеет адаптивный дизайн и высокую производительность. Так же стоит отметить, что библиотека бесплатная, что хорошо подходит для поставленной задачи.

В таблице 1.2 представлено сравнение библиотек.

Учитывая все преимущества и недостатки представленных библиотек — была выбрана библиотека GTK+.

Таблица 1.2 – Сравнение библиотек

Характеристика	Qt	wxWidgets	GTK3
Кроссплатформенность	Да	Да	Да
Лицензия	Коммерческая	Открытая	Открытая
Сложность	Средняя	Высокая	Высокая

1.1.6 Хранение данных

Для реализации возможности сохранения проекта и сериализации данных, необходимо определиться с форматах хранения данных. Требования к формату хранения следующие: читаемость, оптимизированность, гибкость, поддержка C++, желательно интеграция с GTK+. Для данного проекта был выбран формат JSON, имеющий ряд преимуществ перед другими форматами. Рассмотрим каждое из них.

Преимущества хранения данных в формате JSON:

- данные JSON имеют удобный и читаемый формат, что значительно упрощает их отладку [5];
- данный формат не зависит от языка, что значит, что данные JSON можно использовать и в других приложениях;
- легкая сериализация и десериализация позволяет задавать и редактировать структуру данных;
- файлы JSON обладают низким весом, что облегчает их загрузку и делает программу более производительной;
- формат имеет поддержку работы с массивами данных, что позволяет сохранять списки, динамические массивы в этом формате;
- формат JSON не требует строгой схемы хранения данных, это позволяет удобно задавать собственную структуру файла проекта;
- поддержка работы с вложенными структурами, это особенно важно для данного проекта;
- формат довольно популярен и имеет активное комьюнити;
- язык C++ имеет множество библиотек для работы с этим форматом, что позволяет выбрать наиболее подходящую для данной задачи;
- формат поддерживает кроссплатформенность, а в частности, работу на Linux, что критически важно в конкретной задаче.

1.1.7 Выбор библиотеки для работы с JSON

Выбор библиотеки – важный шаг в реализации данного проекта. Каждая из существующих сегодня библиотек предлагает множество возможностей, плюсов и минусов. При выборе важно опираться на такие факты, как производительность, совместимость с языком C++, простота использования, поддержка структур, массивов и различных типов данных.

Рассмотрим самые популярные библиотеки для C++ и выделим их плюсы и минусы.

json-c представляет из себя лёгкую и компактную библиотеку, имеющую простое API, поддерживает основные типы данных, имеет хорошую производительность и активное сообщество, из минусов – сложен для новичков и имеет ограниченную документацию.

cJSON имеет хорошую документацию и примеры, а так же поддерживает кроссплатформенность, но не имеет поддержки UTF-8, что критично в данном проекте, потому что имена СМО могут требовать такой поддержки.

У библиотеки plohmann/json стоит отдельно отметить поддержку работы с вложенными структурами и хорошую совместимость, но одной из главных проблем библиотеки является её небезопасность, и в целом данная библиотека будет избыточной для данной задачи.

json-glib имеет интеграцию работы с библиотеками GTK+ и Glib, поддерживает кроссплатформенность и UTF-8, из минусов – не самая высокая производительность, что не критично для конкретной задачи[6].

Из всех рассмотренных библиотек, json-glib выглядит наиболее подходящей библиотекой, в частности из-за интеграции с GTK+ и Glib, а также из-за поддержки формата UTF-8. Еще из преимуществ этой библиотеки стоит отметить поддержку схемы JSON Schema, хорошую обработку ошибок и кроссплатформенность. Однако библиотека зависима от Glib и не так популярна.

Проанализировав достоинства и недостатки библиотек, было принято решение использовать json-glib. В таблице 1.3 приведены плюсы и минусы каждой библиотеки.

Таблица 1.3 – Сравнение различных библиотек.

Название	Производительность	Поддержка GObject
json-c	Средняя	Нет
cJSON	Высокая	Нет
nlohmann/json	Средняя	Нет
json-glib	Высокая	Да

1.1.8 Выбор библиотеки для реализации графического модуля

Что бы реализовать возможность редактирования и представления СМО в виде графа, необходимо использовать графическую библиотеку, наиболее подходящие библиотеки – Cairo и Skia. Рассмотрим их плюсы и минусы

Cairo представляет из себя библиотеку для работы с 2D графикой. Она поддерживает вывод изображений такие форматы, как JPEG, PNG, SVG, а так же имеет интеграцию с библиотекой GTK+, что является важным фактором для данной программы, поскольку она будет разработана с использованием GTK+. Cairo является крайне гибкой библиотекой, поддерживающей множество элементов рисования.

Из минусов стоит отметить, что Cairo не имеет инструментов для разработки пользовательского интерфейса, что незначительно для данной задачи, поскольку интерфейс будет реализован с помощью GTK+. Так же Cairo может требовать некоторых навыков и имеет не самую обширную документацию.

На рисунке 2 – представлено типовое приложение на Cairo.

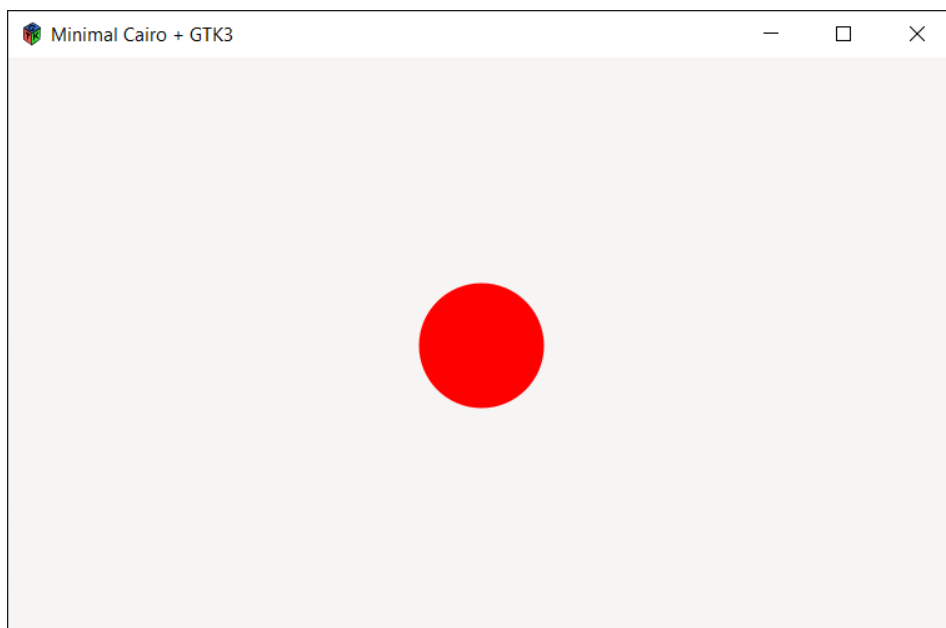


Рисунок 2 – Типовое приложение на Cairo

Еще одна библиотека — Skia — является библиотекой, разработанной Google. Библиотека способная проводить рендеринг 2D графики и выводить результат в браузеры, мобильные приложения и приложения на компьютере. Skia имеет поддержку кроссплатформенности, позволяет рисовать различные векторные изображения, имеет поддержку работы с растровыми изображениями и текстом. Однако она не имеет интеграции с GTK+, что является важным недочетом и в целом для данного проекта она будет избыточна. На рисунке 3 и 4 показаны примеры того, на что способна Skia

Проанализировав достоинства и недостатки обоих фреймворков, было решено использовать Cairo.



Рисунок 3 – Пример Skia

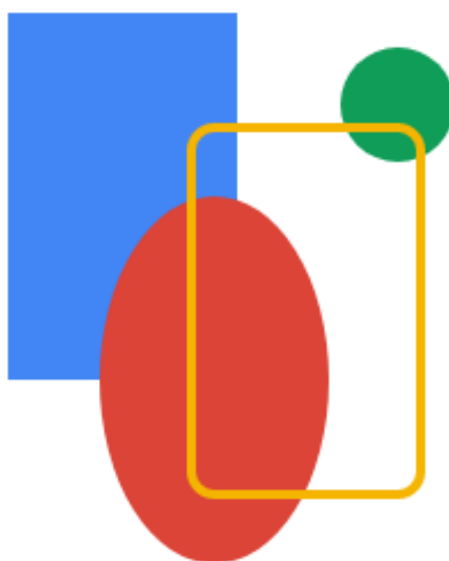


Рисунок 4 – Пример Skia

1.1.9 Выбор среды разработки

Для разработки данного проекта выбор среды разработки не критичен, поскольку компиляция происходит в командной строке MSYS2, тем не менее, для удобства следует уделить внимание этому пункту и выбрать подходящую среду разработки. В данном пункте будут рассмотрены различные IDE,

позволяющие которые предлагают расширенные возможности, такие как отладка кода, управление проектами и интеграция с системами контроля версий. Наиболее популярные среды разработки для C++ являются: Visual Studio, Code::Blocks, CLion и Qt Creator.

Visual Studio является IDE, разработанной компанией Microsoft специально для работы с языками C, C++, C#, что уже является огромным плюсом, Visual Studio не поддерживает кроссплатформенность, но это не большая проблема, так как разработку можно вести на Windows[7].

Code::Blocks – IDE, поддерживающая C++, но по сравнению с Visual Studio не так развита.

CLion от компании JetBrains – это IDE, так же поддерживающая C++ и являющаяся кроссплатформенной, однако она доступна по платной подписке, в то время как Visual Studio бесплатная, что делает её не самым лучшим выбором.

Qt Creator – специальная IDE, созданная для разработки Qt-приложений. Несмотря на то, что она довольно удобна для создания интерфейсов, она не подходит для данного проекта, так как тут не используется Qt. А еще фреймворк Qt платный.

Наиболее подходящим IDE является Visual Studio, поскольку оно бесплатное и поддерживает разработку на C++.

1.2 Проектирование и разработка

Специализированная программа предназначена для моделирования систем и сетей массового обслуживания. С её помощью можно исследовать различные ВС, которые можно представить в виде систем массового обслуживания типа $M/M/1$, $M/M/m/k$, $M/M/m$ (в соответствии с классификацией Кендалла), в том числе и систем с ограничением очереди.

Разрабатываемая программа для расчета вероятностно-временных характеристик, представляемых СМО, предназначена для анализа характеристик функционирования многопроцессорных систем, представляемой в виде аналитических моделей устройств, каждому из которых ставится в соответствие одноканальная или многоканальная СМО.

1.2.1 Реализация сериализации данных

Для сериализации данных о каждой СМО в сети с целью сохранения проекта и параметров каждой СМО Была выбрана библиотека `json-glib`. Она предоставляет богатый функционал и поддерживает почти все необходимые функции, а также библиотека хорошо оптимизирована. Реализация сериализации включает следующие этапы:

- преобразование внутренних структур данных в формат, пригодный для записи;
- запись сериализованных данных в файл;
- при необходимости — десериализация, то есть восстановление структур данных из сохранённого представления.

Основной структурой, подлежащей сериализации, является структура `Circles`, содержащая:

- координаты узлов сети (x, y);
- массив связей (`lines`), описывающих переходы между узлами с весами;
- параметры каждого узла (`circle_params`), включающие имя, время обслуживания, число каналов, параметры приоритетов и другие настройки.

Для сохранения состояния вызывается функция `serialize_circles_to_json`, которая с помощью объекта `JsonBuilder` формирует JSON-объект, включающий:

- общее количество узлов и связей;
- массив узлов с их координатами;
- массив связей с индексами узлов и весами;
- массив параметров узлов, включая вложенные массивы приоритетов для соответствующих узлов.

Далее сформированный JSON-объект записывается в файл функцией `save_circles_to_file` через `JsonGenerator`. UML-диаграмма классов сериализации представлена на рисунке 5.

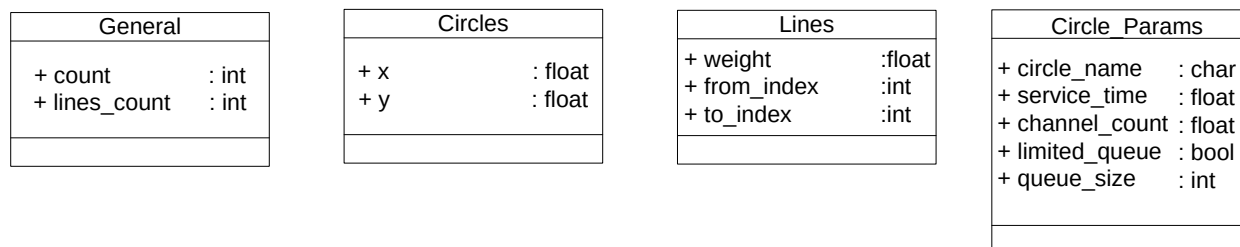


Рисунок 5 – UML-диаграмма классов сериализации

1.2.2 Реализация графической части

Графическая часть программы реализована с использованием библиотеки GTK, которая обеспечивает создание кроссплатформенного GUI на языке C++. Основные элементы интерфейса включают окно приложения, панель инструментов с кнопками, меню и область для визуализации сети массового обслуживания.

Основные пункты интерфейса:

- главное окно;
- панель кнопок;
- меню.

Для отображения узлов и связей используется GtkDrawingArea. На нём происходит отрисовка графа сети в функции on_draw_event, которая вызывается при необходимости перерисовки окна.

Узлы сети отображаются в виде окружностей с радиусом 20 пикселей и подписаны как «S0», «S1» и т.д. Связи между узлами рисуются линиями с учётом веса (интенсивности перехода), который влияет на оттенок цвета линии (чем больше вес, тем темнее линия). Для петель (связей узла с самим собой) рисуется окружность-дуга рядом с узлом. На концах связей рисуются красные кружочки, обозначающие направление.

Область рисования поддерживает обработку событий мыши:

- добавление узлов;
- удаление узлов;
- перемещение узлов;
- добавление и удаление переходов;
- редактирование веса связи.

На рисунке 6 представлено графическое представление сети.

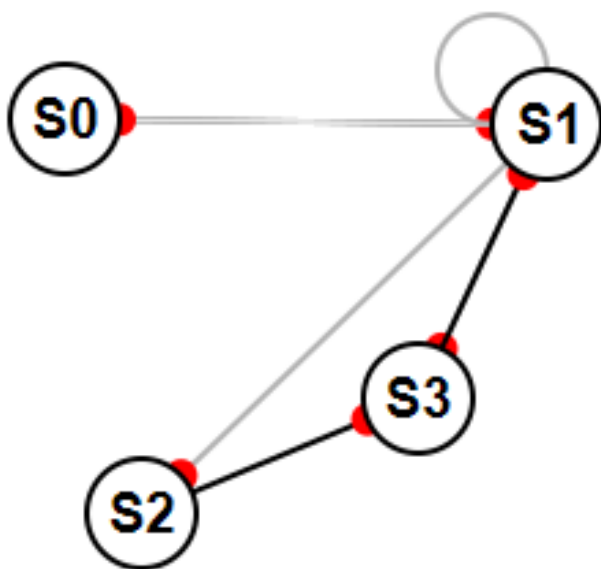


Рисунок 6 – Графическое представление сети

Программа также поддерживает построение графиков и диаграмм с библиотеки Cairo: отображение диаграмм характеристик сети и отдельных СМО, различные столбчатые диаграммы.

1.2.3 Разработка структур

Для реализации модели сети массового обслуживания в программе были разработаны несколько ключевых структур данных на языке Си. Эти структуры обеспечивают хранение информации об узлах СМО, связях между ними, а также параметрах моделирования и визуализации.

Основной элемент модели — узел, представляющий отдельную систему массового обслуживания. Для описания узла используется структура, включающая следующие поля:

- координаты;
- идентификатор;
- параметры обслуживания;
- массив весов связей;
- структура для хранения характеристик узла СМО.

Структура для хранения характеристик узла СМО представляет из себя динамический массив узлов и матрица весов переходов. Основные элемент структуры:

- название СМО;
- время обслуживания;
- число каналов;
- есть ли ограничение очереди;
- длина очереди.

Для взаимодействия с пользователя и отображения модели используются структуры графического интерфейса:

- структуры описании кнопок меню;
- структура для хранения состояние интерфейса;
- структура для хранения состояние интерфейса;

- структура для диалоговых окон.

Для реализации перетаскивание узлов по графической части и выбора связей используются структуры, в которых хранятся состояния текущей операции:

- флаги нажатия и отпускания кнопки мыши;
- индекс выбранного узла;
- координаты начала и конца перетаскивания.

1.2.4 Разработка алгоритмов решения задачи

Алгоритм реализован в виде модульного приложения с графическим интерфейсом, предназначенного для моделирования и анализа СМО типа М/М/т, М/М/1, М/М/т/к. Основная задача — расчёт характеристик сети при заданных параметрах узлов и связей между ними, а также визуализация результатов. Алгоритм состоит из следующих этапов:

- ввод структуры;
- задание параметров каждого узла;
- расчёт интенсивностей заявок в каждом узле;
- вычисление характеристик отдельных узлов и всей сети;
- сохранения отчёта;
- построение диаграмм и графиков.

Сеть массового обслуживания представляется в виде графа, где вершины – узлы СМО, рёбра – переходы между СМО с вероятностями. Каждый узел характеризуется временем обслуживания, числом каналов СМО, ограниченностью очереди. Входящий поток заявок подаётся на начальный узел сети с заданной интенсивностью. Заявки передаются по сети с определённой вероятностью.

Входные данные: начальная интенсивность, матрица весов переходов, число узлов.

Для расчёта интенсивности в каждом узле используется итерационный метод. Он заключается в суммировании произведений интенсивности всех переходов поступающий в узел умноженный на вероятность переходов до тех пор, пока изменения интенсивностей не станут незначительными.

После, рассчитываются основные показатели для каждого узла: устойчивость системы, вероятность простоя, среднее число заявок в очереди, среднее время ожидания, эффективная интенсивность, среднее число заявок в системе, среднее время пребывания заявки в системе.

Так же в программе присутствует варьирование параметров. Варьируется входная интенсивность, число каналов для каждой СМО, время обслуживания для каждой СМО, размер очереди. Для каждого параметра задаются начальное и конечное значение, размер шага рассчитывается в программе. Так же в случае с варьированием параметров задаётся число циклов варьирования. Блок схема алгоритма решения задачи приведена в рисунке 7.

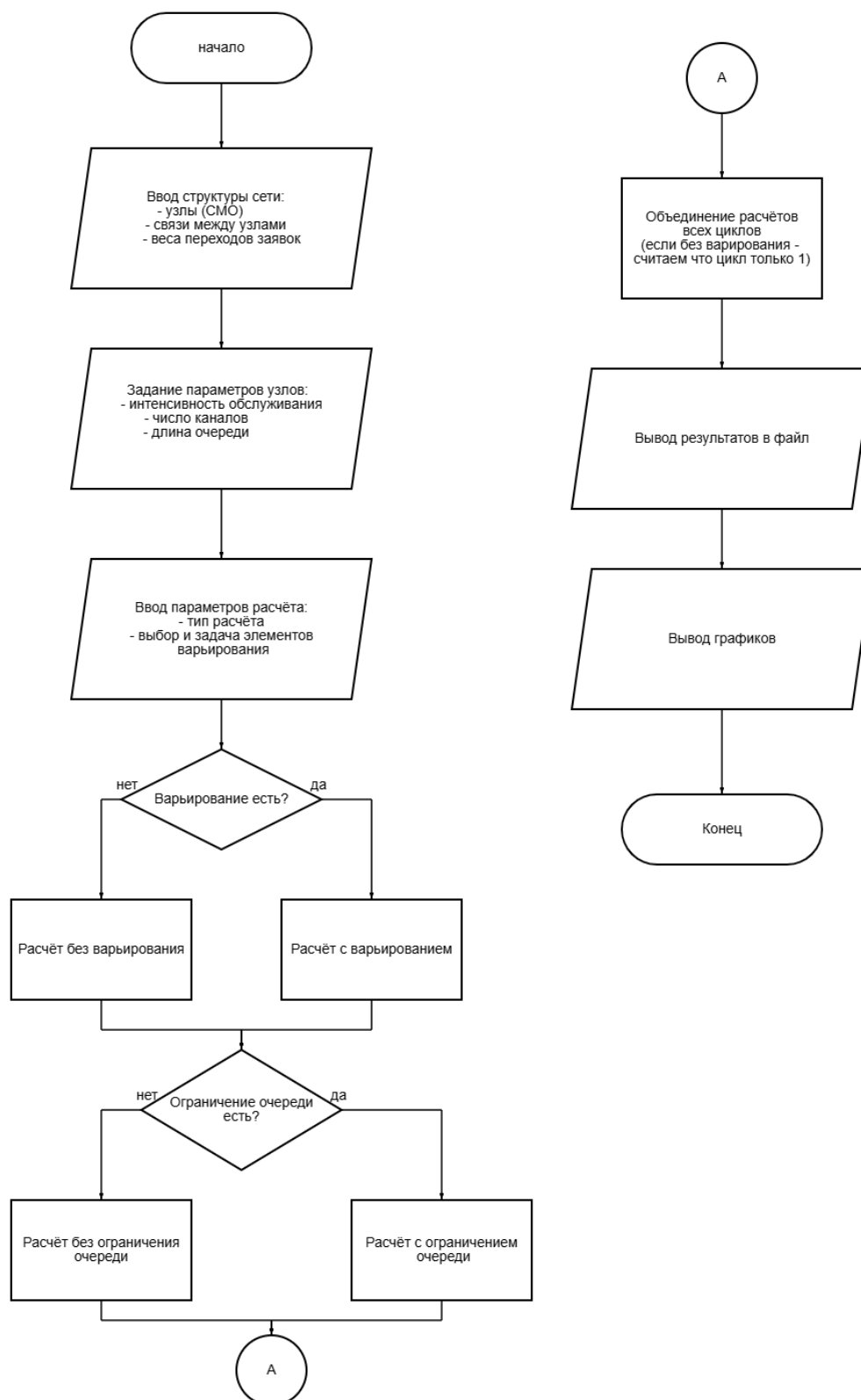


Рисунок 7 – Алгоритм решения задачи

После расчета создается файл отчета со структурой показанной на рисунке 8.

Описание варьируемых параметров:

Общее число циклов:

Расчёт:

Цикл номер:

Исходные данные:

| Номер СМО | Наименование СМО | Время обслуживания | Число каналов | Число мест |

Характеристики отдельных СМО:

| Номер СМО | Коэфф. передачи | Интенсивность входного потока | Коэфф. загрузки | Ср. число занятых каналов | Ср. число заявок в СМО

| Ср. длина очереди | Ср. время ожидания в очереди | Ср. время пребывания в СМО |

Характеристики сети в целом:

| Интенсивность входящего потока:

| Среднее число заявок в очередях:

| Среднее число заявок в сети:

| Среднее время ожидания в очередях:

| Среднее время пребывания заявки в сети:

Рисунок 8 – Структура отчёта

1.2.5 Разработка построения диаграмм.

В данной программе есть два типа диаграмм: Диаграмма сети и диаграмма характеристик СМО. Диаграмма сети отображает усредненные характеристики по всем циклам варьирования параметров. Диаграмма характеристик СМО позволяет отобразить изменение характеристик конкретной СМО по ходу варьирования параметров системы.

Логика отрисовки диаграммы:

- определение размеров области рисования;
- вычисление максимального значения для нормализации высоты столбцов/линий;
- построение осей координат;
- отрисовка шкалы и подписей;
- построение диаграммы.

На рисунке 9 показана диаграмма сети, на рисунке 10 показана диаграмма характеристик СМО. На рисунке 11 показана блок схема алгоритма построения диаграммы.

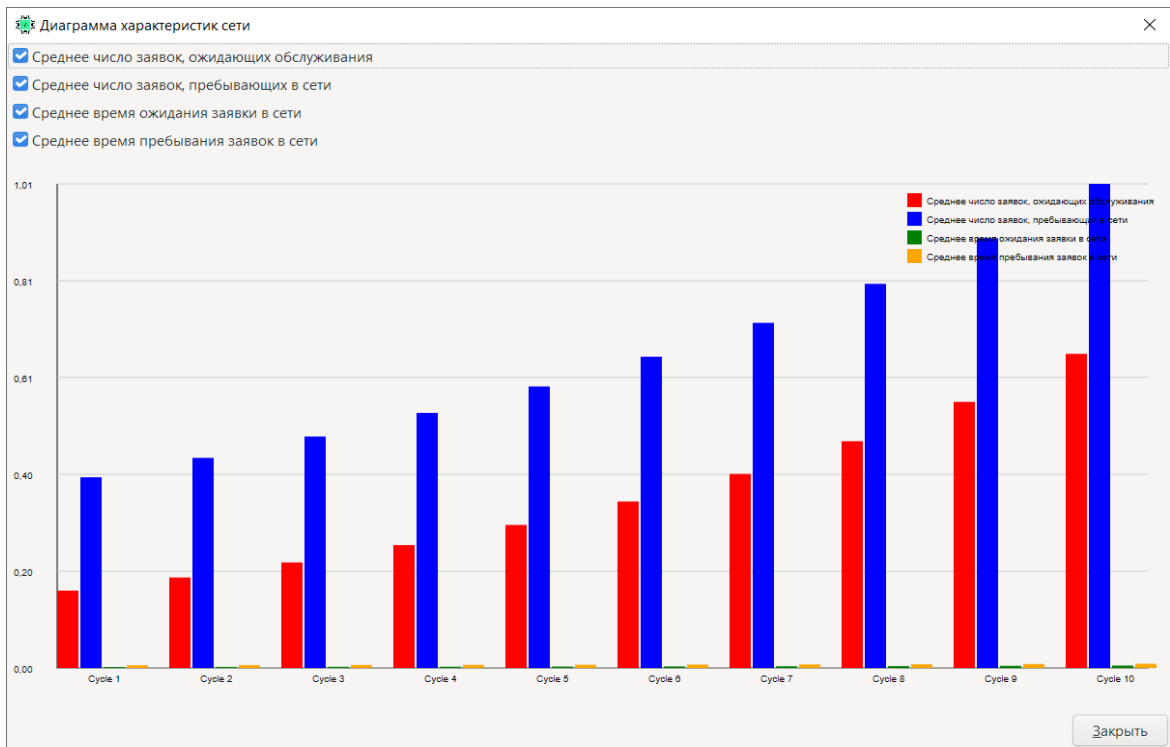


Рисунок 9 – Диаграмма сети

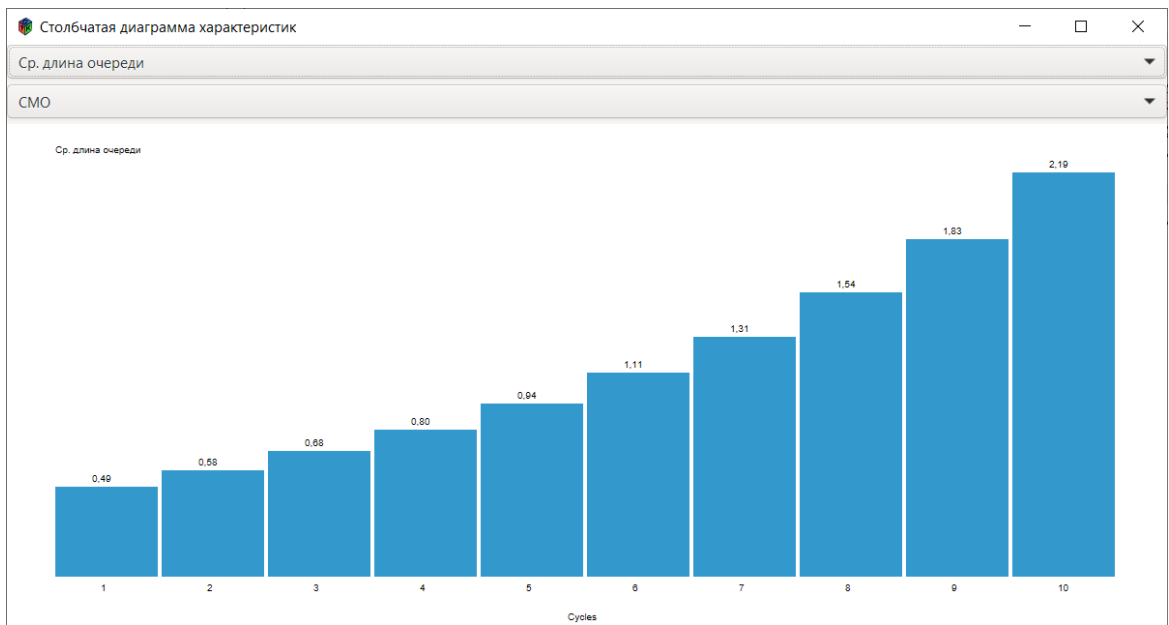


Рисунок 10 – Диаграмма характеристик СМО.



Рисунок 11 – Блок схема алгоритма построения диаграммы.

1.3 Анализ результатов

Разработанная программа реализует моделирование СМО с возможностью варьирования параметров и визуализации результатов. Рассчитываются основные характеристики, такие как среднее число заявок в очереди, среднее время ожидания, коэффициент загрузки и среднее время пребывания в системе. Программа поддерживает работу с такими типами сетей, как M/M/m, M/M/1, M/M/m/k.

Пользователю доступно ограниченное количество действий:

- добавление узла СМО;
- удаление узла СМО;
- перемещение узла СМО;
- создание перехода;
- удаление перехода;
- редактирование веса перехода;
- расчёт характеристик сети;
- построение диаграммы сети;
- построение диаграммы характеристик СМО;
- отображение файла последнего отчёта;
- сохранение проекта;
- начало нового проекта;
- загрузка сохраненного проекта;
- получение справки о программе;
- получение информации об авторе.

На рисунке 12 представлена UML-диаграмма вариантов использования.

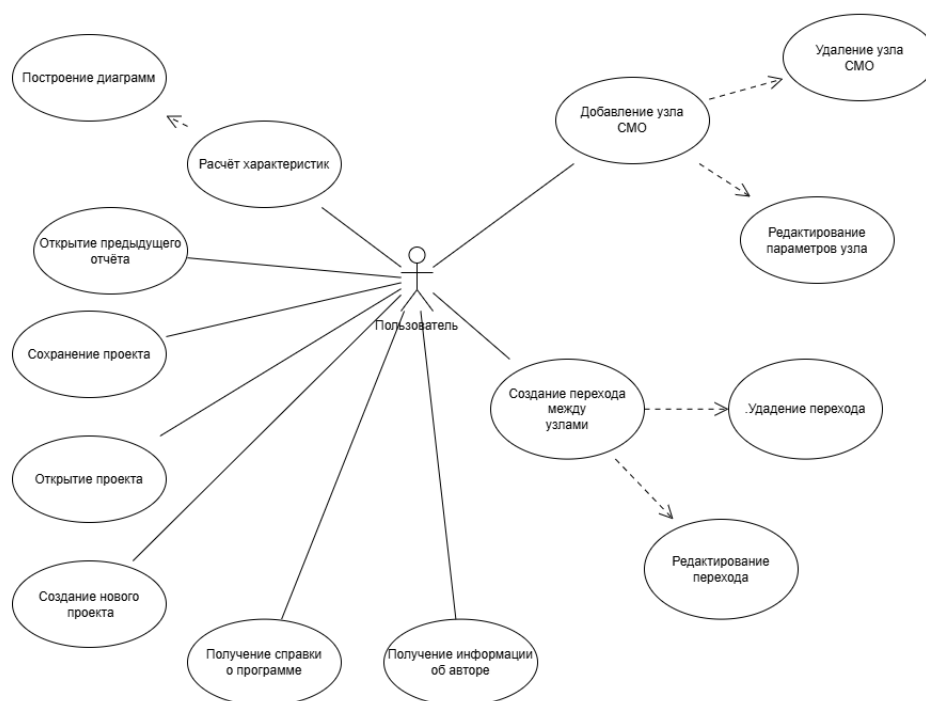


Рисунок 12 – UML-диаграмма вариантов использования

1.3.1 Проверка результатов расчётов

Для тестирования результатов расчётов была использована программа STSET в качестве проверочной программы, которая обладает схожим функционалом, но не имеет кроссплатформенности. В качестве тестовых данных мы будем использовать матрицу переходов, представленную в таблице 1.4.

Таблица 1.4 – Матрица переходов.

	S0	S1	S2	S3
S0	0	1	0	0
S1	0.1	0.3	0.3	0.3
S2	0	0	0	1
S3	0	1	0	0

Варьируемые параметры каждой СМО представлены в таблицу 1.5.

Таблица 1.5 – Варьируемые параметры СМО

	S1	S2	S3
Число каналов	1 - 2	1 - 2	1-2
Время обслуживания	0,0005-0,001	0,001-0,002	0,002-0,003

На рисунке 13 представлен результат расчётов программы. На рисунке 14 представлен результат расчётов проверочной программы.

Описание варьируемых параметров:

Параметр - интенсивность входящего потока - начальное значение - 10 - Конечное значение - 15 - шаг - 2,5
Параметр - время обслуживания СМО S1 - начальное значение - 0,0005 - Конечное значение - 0,001 - шаг - 0,00025
Параметр - время обслуживания СМО S2 - начальное значение - 0,001 - Конечное значение - 0,002 - шаг - 0,0005
Параметр - время обслуживания СМО S3 - начальное значение - 0,002 - Конечное значение - 0,003 - шаг - 0,0005
Параметр - число каналов СМО S1 - начальное значение - 1 - Конечное значение - 2 - шаг - 0
Параметр - число каналов СМО S2 - начальное значение - 1 - Конечное значение - 2 - шаг - 0
Параметр - число каналов СМО S3 - начальное значение - 1 - Конечное значение - 2 - шаг - 0
Общее число циклов: -3-

Расчёт:

Цикл номер: -1-

Исходные данные:

Номер СМО	Наименование СМО	Время обслуживания	Число каналов	Число мест
0	СМО	0,0001	1	0
1	СМО	0,0005	1	1
2	СМО	0,001	1	1
3	СМО	0,002	1	1

Характеристики отдельных СМО:

Номер СМО	Коэфф. передачи	Интенсивность входного потока	Коэфф. загрузки	Ср. число занятых каналов	Ср. число заявок в СМО	Ср. длина очереди	Ср. время ожидания в очереди	Ср. время пребывания в СМО
0	0	0	0	0	0	0	0	0
1	9,94791	99,4791	0,0497396	0,0497396	0,0513431	0,00268352	2,61715e-05	0,000526172
2	2,98354	29,8354	0,0298354	0,0298354	0,0307529	0,000917524	3,07529e-05	0,00103075
3	5,96619	59,6619	0,119324	0,119324	0,135491	0,0161673	0,002270982	0,00227098

Характеристики сети в целом:

Интенсивность входящего потока: 10
Среднее число заявок в очередях: 0,0196884
Среднее число заявок в сети: 0,218587
Среднее время ожидания в очередях: 0,00196884
Среднее время пребывания заявки в сети: 0,0218587

Цикл номер: -2-

Исходные данные:

Номер СМО	Наименование СМО	Время обслуживания	Число каналов	Число мест
0	СМО	0,0001	1	0
1	СМО	0,00075	2	1
2	СМО	0,0015	2	1
3	СМО	0,0025	2	1

Характеристики отдельных СМО:

Номер СМО	Коэфф. передачи	Интенсивность входного потока	Коэфф. загрузки	Ср. число занятых каналов	Ср. число заявок в СМО	Ср. длина очереди	Ср. время ожидания в очереди	Ср. время пребывания в СМО
0	0	0	0	0	0	0	0	0
1	9,94791	124,349	0,0466309	0,0932617	0,0934649	0,000203234	1,63438e-06	0,000751634
2	2,98354	37,2942	0,0279707	0,0559413	0,0559851	4,38004e-05	1,17446e-06	0,00150117
3	5,96619	74,5774	0,0932217	0,186443	0,188078	0,00163445	2,19162e-05	0,00252192

Характеристики сети в целом:

Интенсивность входящего потока: 12,5
Среднее число заявок в очередях: 0,00180149
Среднее число заявок в сети: 0,337528
Среднее время ожидания в очередях: 0,000150519
Среднее время пребывания заявки в сети: 0,0270022

Цикл номер: -3-

Исходные данные:

Номер СМО	Наименование СМО	Время обслуживания	Число каналов	Число мест
0	СМО	0,0001	1	0
1	СМО	0,001	2	1
2	СМО	0,002	2	1
3	СМО	0,003	2	1

Характеристики отдельных СМО:

Номер СМО	Коэфф. передачи	Интенсивность входного потока	Коэфф. загрузки	Ср. число занятых каналов	Ср. число заявок в СМО	Ср. длина очереди	Ср. время ожидания в очереди	Ср. время пребывания в СМО
0	0	0	0	0	0	0	0	0
1	9,94791	149,219	0,0746094	0,149219	0,150054	0,00035204	5,59772e-06	0,0010056
2	2,98354	44,753	0,0447531	0,0895061	0,0896857	0,000179626	4,01371e-06	0,00200401
3	5,96619	89,4929	0,134239	0,268479	0,273405	0,00492682	5,50526e-05	0,00305505

Характеристики сети в целом:

Интенсивность входящего потока: 15
Среднее число заявок в очередях: 0,00594173
Среднее число заявок в сети: 0,513145
Среднее время ожидания в очередях: 0,000396115
Среднее время пребывания заявки в сети: 0,0342097

Рисунок 13 – Результат расчётов программы

Характеристики сети в целом:

Интенсивность входящего потока:	10.000000000
Среднее число заявок в очередях:	0.0199230504
Среднее число заявок в сети:	0.2199230504
Среднее время ожидания в очередях:	0.0019923050
Среднее время пребывания заявки в сети:	0.0219923050

Характеристики сети в целом:

Интенсивность входящего потока:	12.500000000
Среднее число заявок в очередях:	0.0019135387
Среднее число заявок в сети:	0.3394135387
Среднее время ожидания в очередях:	0.0001530831
Среднее время пребывания заявки в сети:	0.0271530831

Интенсивность входящего потока:	15.000000000
Среднее число заявок в очередях:	0.0060432382
Среднее число заявок в сети:	0.5160432382
Среднее время ожидания в очередях:	0.0004028825
Среднее время пребывания заявки в сети:	0.0344028825

Рисунок 14 – Результат расчётов проверочной программы

На рисунке 15 показана диаграмма сети, построенная из рассчитанных параметров.

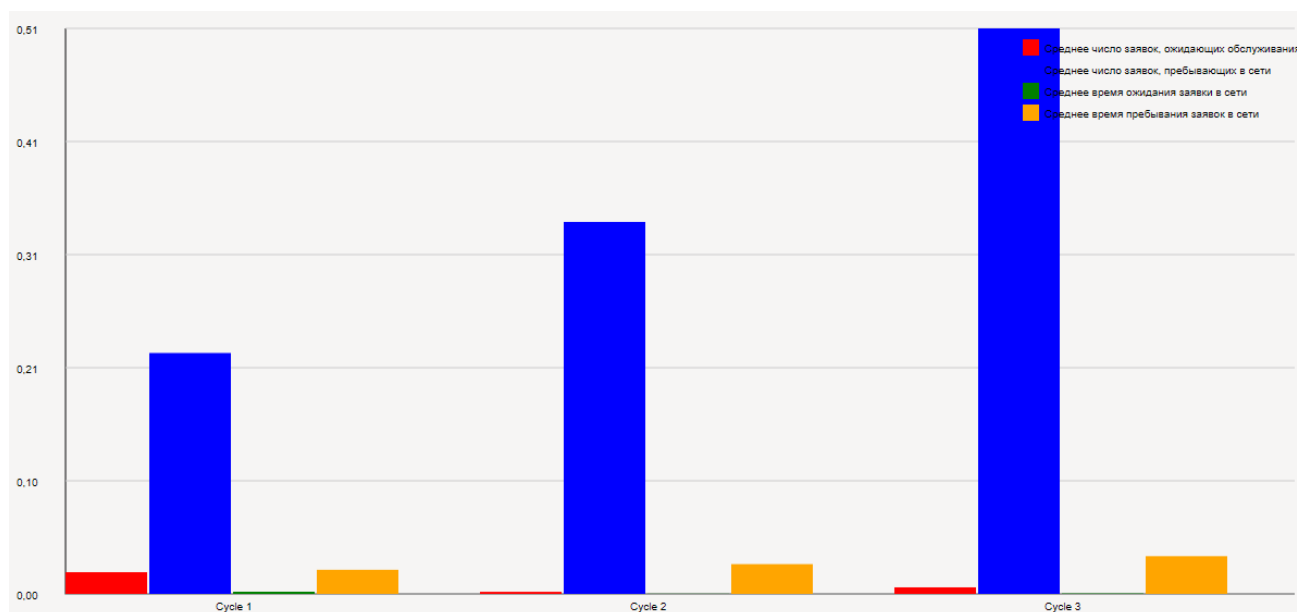


Рисунок 15 – Диаграмма сети

На рисунке 16 показана диаграмма сети проверочной программы.

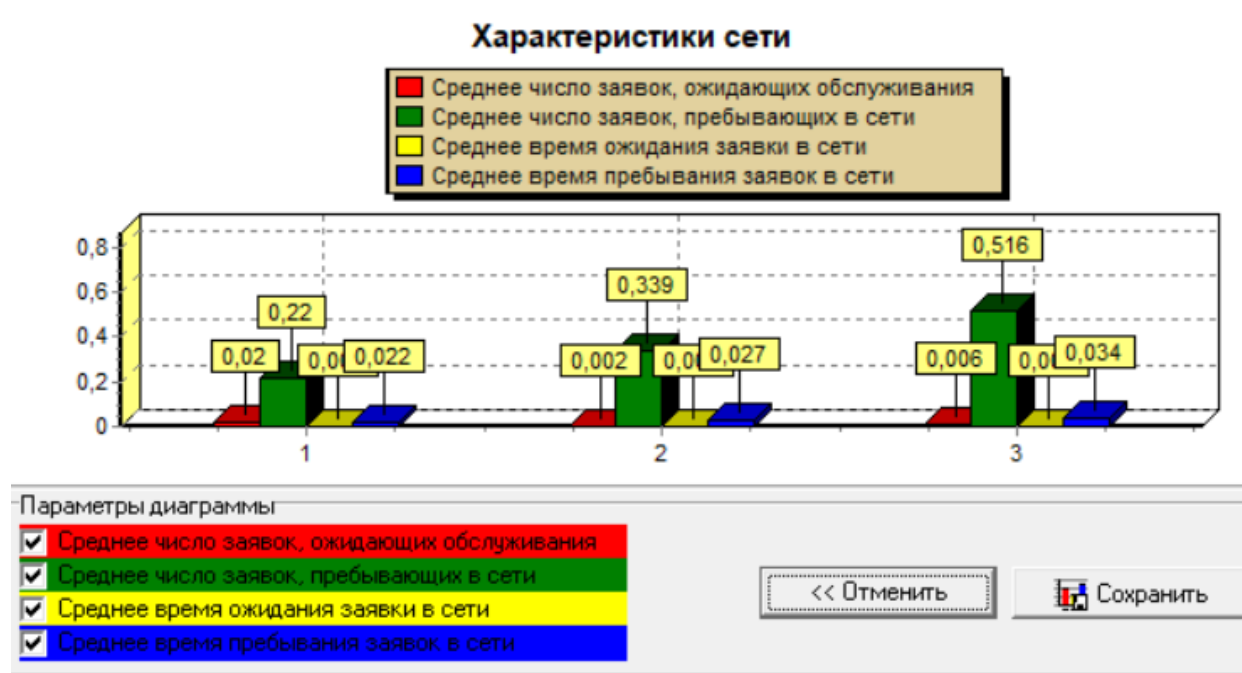


Рисунок 16 – Диаграмма сети проверочной программы

Результаты тестов расчётной части программы показывают, что программа рассчитывает значения верно, в пределах допустимой погрешности.

1.3.2 Клиентская программа

При запуске клиентской программы пользователь встречает начальное окно с пустым холстом для построения графа сети и панелью инструментов. Часть функционала программы (расчёт характеристик, отображение диаграмм) заблокирована до создания сети или загрузки проекта. На рисунке 17 показано стартовое окно приложения.

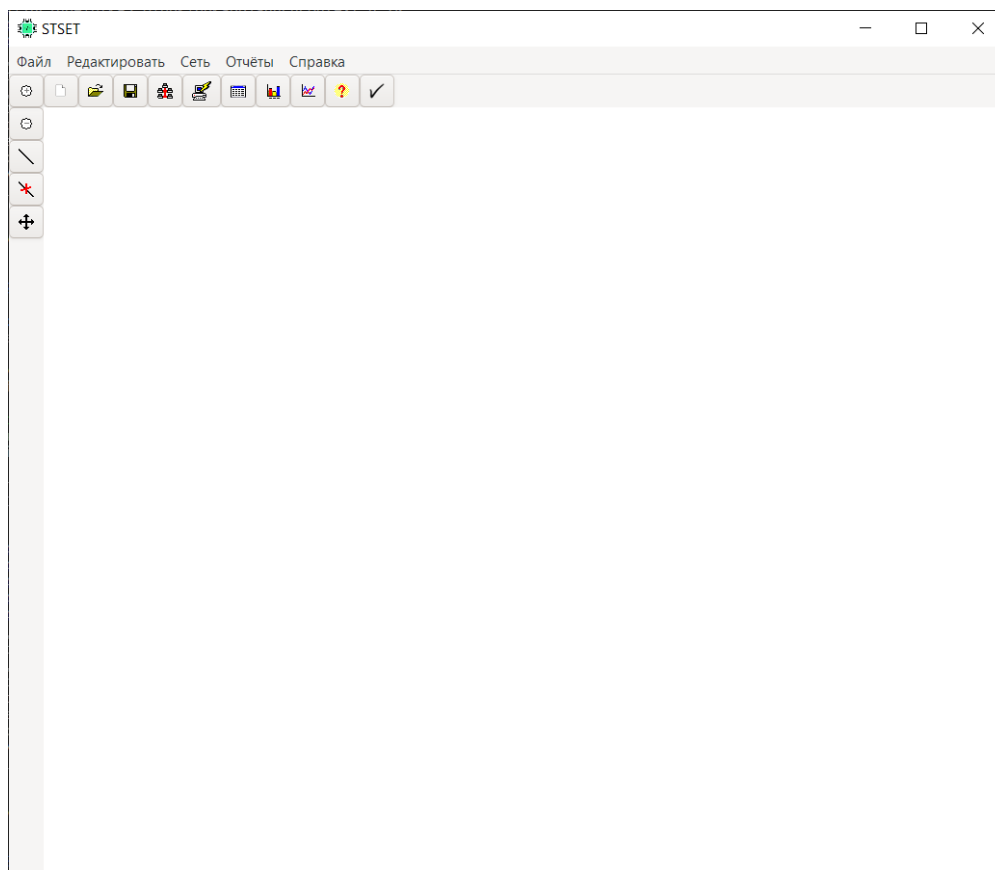


Рисунок 17 – Стартовое окно приложения

Чтобы разблокировать функционал программы, пользователь переходит к созданию графа сети. На рисунке 18 представлено панель управления элементами:

- добавить СМО;
- удалить СМО;
- провести связь;
- удалить связь;
- переместить СМО.



Рисунок 18 – Панель управления элементами

Что бы создать узел СМО, необходимо нажать на кнопку «Добавить СМО», после нажать на любую точку рабочей области. На рисунке 19 представлено внешний вид узла СМО.



Рисунок 19 – Внешний вид узла СМО

Что бы удалить узел СМО, необходимо нажать на кнопку «Удалить СМО».

Что бы переместить узел СМО, необходимо нажать на кнопку «Переместить СМО».

Что бы создать переход между узлами необходимо нажать на кнопку «Провести связь», затем нажать на одно СМО, затем нажать на второе СМО (или на то же самое), после назначить вероятность перехода в соответствующем окне. На рисунке 20 представлено окно ввода вероятности перехода СМО и сам внешний вид перехода.

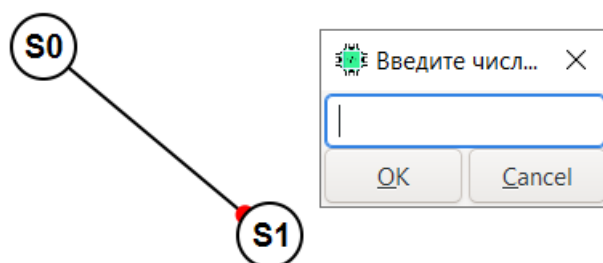


Рисунок 20 – Окно ввода вероятности перехода СМО и его внешний вид

Для редактирования параметров СМО, необходимо нажать на него. На рисунке 21 показано окно параметров СМО.

Параметры СМО

Наименование: СМО

Время обслуживания: 0

Число каналов: 1

☐ Ограниченная длина очереди

Число мест: 1

OK Отмена

Рисунок 21 – Окно параметров СМО

Для задачи интенсивности входного потока нужно нажать на нулевой узел. На рисунке 22 представлено окно задачи интенсивности входного потока.

Введите интенсивность входяще...

Интенсивность (число с плавающей точкой):

1,000000

OK Отмена

Рисунок 22 – Окно ввода интенсивности входного потока

На рисунке 23 показана верхняя панель управления, содержащая элементы:

- новый проект;
- открыть проект;
- сохранить проект;
- редактор матрицы вероятностей передач;
- рассчитать;
- показать отчёт;
- показать диаграмму сети;
- показать диаграмму характеристик СМО;
- справка;
- о программе.



Рисунок 23 – Верхняя панель управления

Что бы редактировать вероятность перехода, необходимо нажать на сам переход или на кнопку «Редактор матрицы вероятности передач». На рисунке 24 показано окно редактирования вероятности перехода и таблица передач.

Окно «Редактирование матрицы весов» содержит таблицу с 5 строками (S0-S4) и 5 столбцами (S0-S4). В ячейке S0-S0 введено значение 0,00. В каждой ячейке есть кнопки «-» и «+». В центре окна открыто диалоговое окно «Введите числ...» с полем ввода и кнопками «OK» и «Cancel». Внизу окна находится кнопка «Применить».

	S0	S1	S2	S3	S4
S0	0,00 - +	1,00 - +	0,00 - +	0,00 - +	0,00 - +
S1	0,00 - +	0,00 - +	0,00 - +	0,00 - +	0,00 - +
S2	0,00 - +	0,00 - +	0,00 - +	0,00 - +	0,00 - +
S3	0,00 - +	0,00 - +	0,00 - +	0,00 - +	0,00 - +
S4	0,00 - +	0,00 - +	0,00 - +	0,00 - +	0,00 - +

Рисунок 24 – Окно редактирования вероятности перехода и таблица передач

После задачи параметров можно провести расчёт. Можно выбрать вариант расчёта: с фиксированными или варьируемыми параметрами. При выборе фиксированных параметров будет произведен расчёт.

Что бы показать отчёт необходимо начать кнопку «Показать отчёт». На рисунке 25 показано окно настройки параметров расчёта, на рисунке 26 показан пример отчёта

Рисунок 25 – Окно настройки параметров расчёта

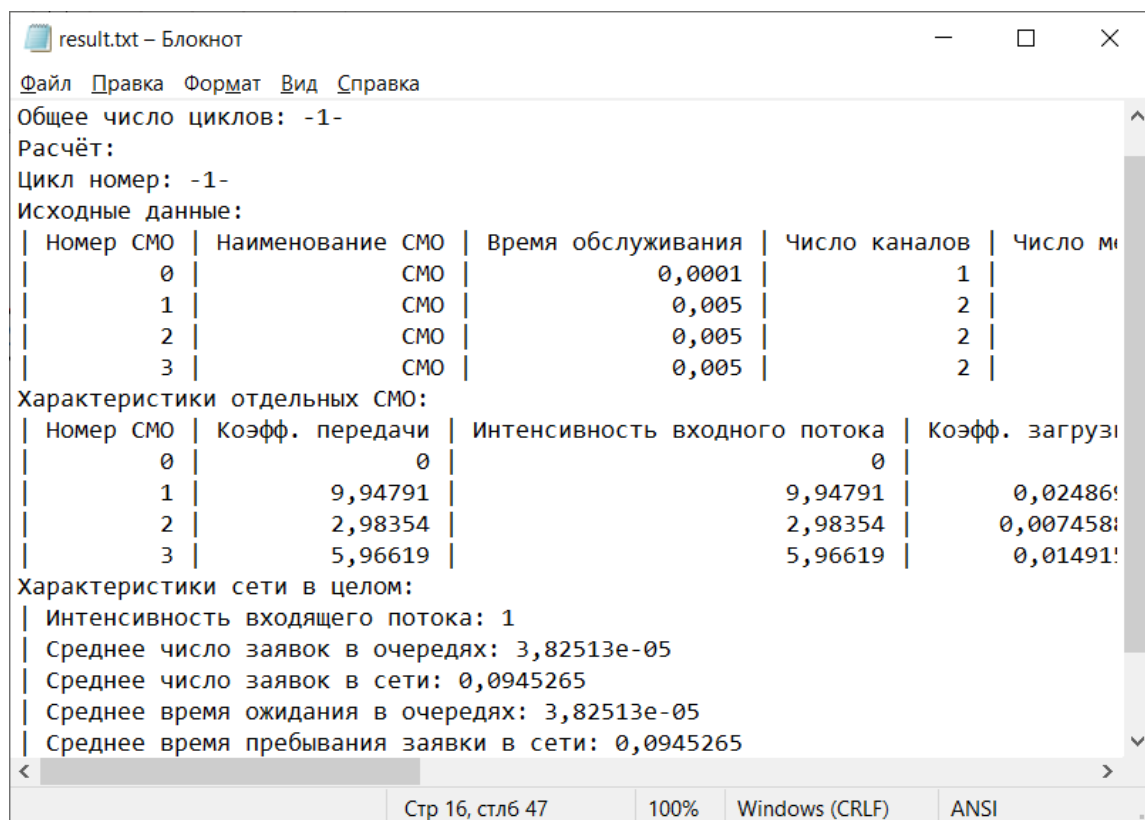


Рисунок 26 – Пример отчёта

Изначально пользователь может задать любое значение вероятности перехода, но для того, чтобы провести расчёт необходимо чтобы вероятности исходящих связей из СМО были равно 1, иначе программа выдаст ошибку. На рисунке 27 показано окно ошибки.

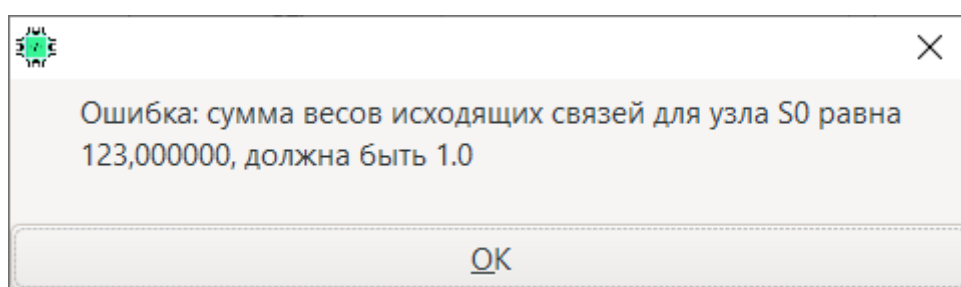


Рисунок 27 – Окно ошибки

Если система нестабильна, программа так же выдаст ошибку. На рисунке 28 показано окно ошибки.

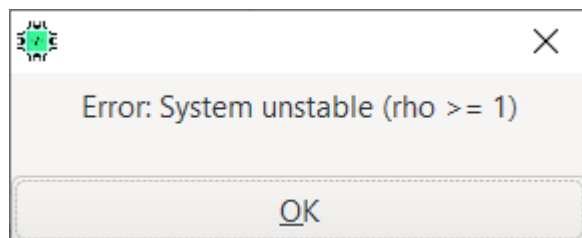


Рисунок 28 – Окно ошибки

Если пользователь хочет провести расчёт с варьированием ему необходимо будет ввести число циклов варьирования (минимум 2), выбрать какие именно параметры варьировать, и задать варьирование для каждой СМО по отдельности при необходимости. Пример настройки варьирования изображен на рисунке 29.

 A screenshot of a software dialog box titled "Расчет сети" (Network Calculation). It contains several configuration options:

- Two radio buttons: "Расчет сети с фиксированными параметрами" (unselected) and "Расчет сети с варьируемыми параметрами" (selected).
- A checked checkbox "Изменять интенсивность" (Change intensity) with input fields for "от:" (10) and "до:" (15).
- A checked checkbox "Изменять время обслуживания в СМО" (Change service time in queue) with a dropdown menu set to "S3" and input fields for "от:" (0,0005) and "до:" (0,001).
- An unchecked checkbox "Изменять число каналов в СМО" (Change number of channels in queue) with a dropdown menu set to "Выберите круг" (Select circle) and empty input fields for "от:" and "до:".
- An unchecked checkbox "Изменять число мест в СМО" (Change number of places in queue) with a dropdown menu set to "Выберите круг" (Select circle) and empty input fields for "от:" and "до:".
- A label "Число циклов:" (Number of cycles) with an input field containing the value "100".
- At the bottom, there are two buttons: "Рассчитать" (Calculate) and "Заккрыть" (Close).

Рисунок 29 – Пример настройки варьирования

Построение диаграмм возможно только при расчётах с варьированием, так как диаграммы строятся по циклам. При попытке построить диаграмму при расчёте в фиксированными параметрами, диаграмма не отобразится. Чтобы построить диаграмму необходимо нажать кнопку «Показать диаграмму характеристик сети» или «Показать диаграмму характеристик СМО». Окно диаграммы характеристик состоит из самой диаграммы и 4 флагов отображения характеристик, которые можно отображать или скрывать по желанию. Окно диаграммы характеристик СМО состоит из диаграммы, и 2 выпадающих списков, первый из них – выбор характеристики для отображения, второй – выбор СМО. На рисунке 30 изображена диаграмма характеристик сети, на рисунке 31 изображена диаграмма характеристик СМО.

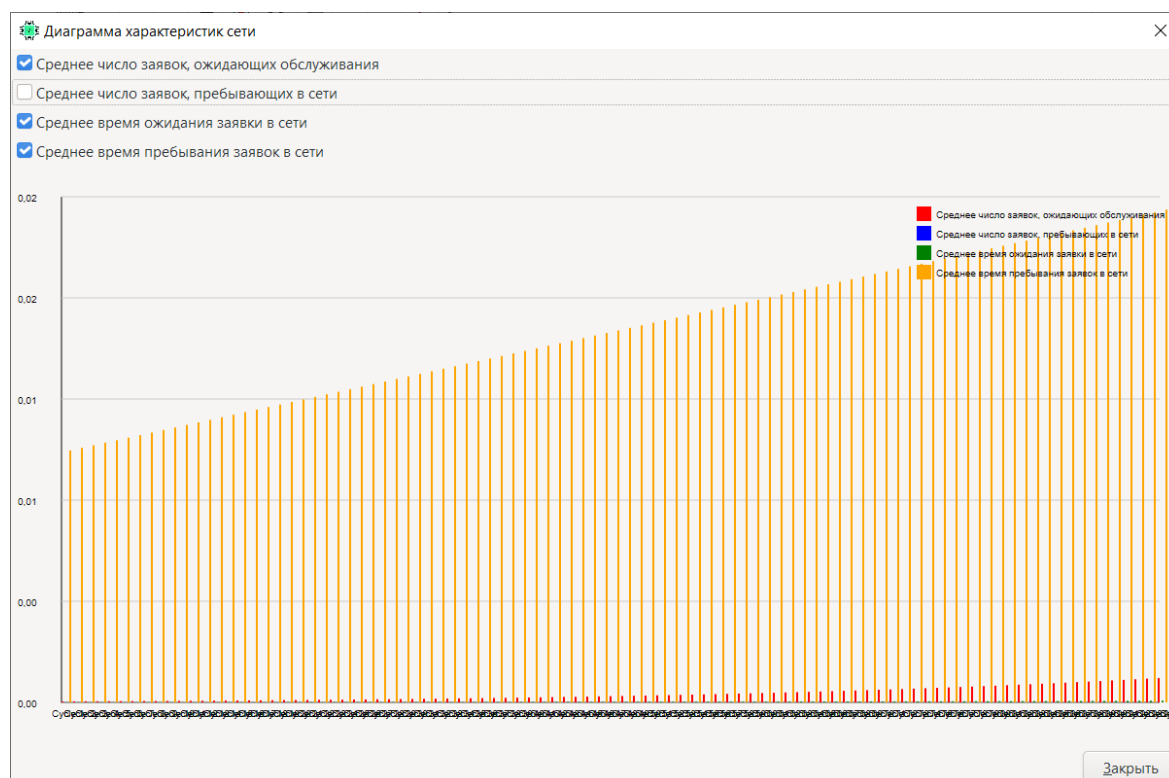


Рисунок 30 – Диаграмма характеристик сети

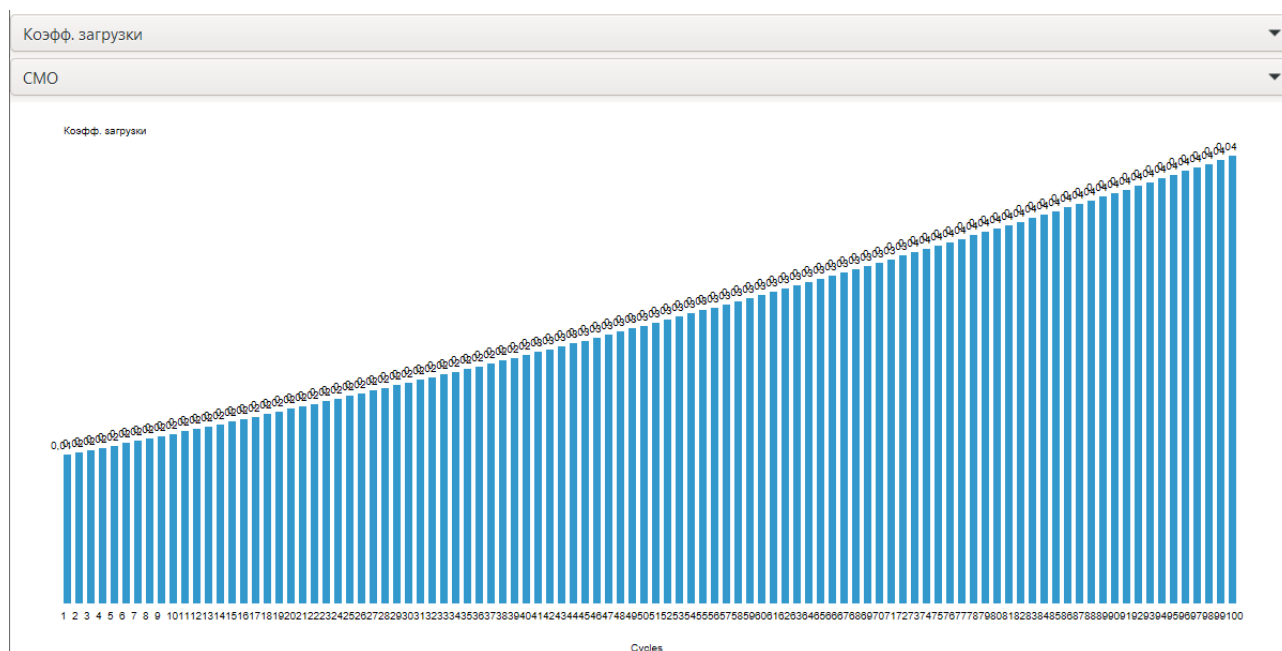


Рисунок 31 – Диаграмма характеристик СМО

Кнопка «Справка» открывает HTML-документ в браузере, описывающий функционал программы. На рисунке 32 показан этот документ.

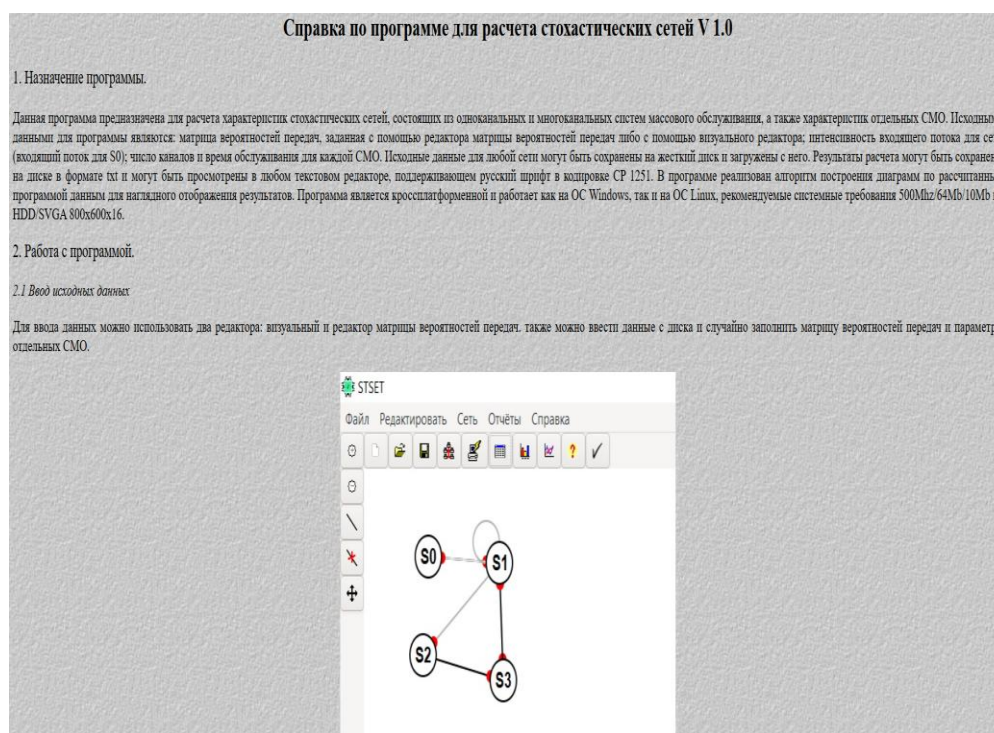


Рисунок 32 – Справочный HTML-документ

Кнопка «О программе» позволяет получить информацию о разработчике. На рисунке 33 показано окно «О программе».

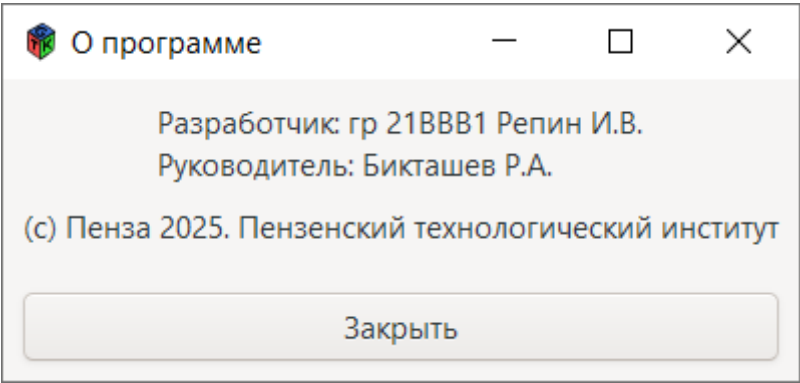


Рисунок 33 – Окно «О программе»

Что бы сохранить проект, необходимо нажать кнопку «Сохранить», затем выбрать путь сохранения и имя файла, а затем нажать сохранить. На рисунке 34 показано окно сохранения проекта.

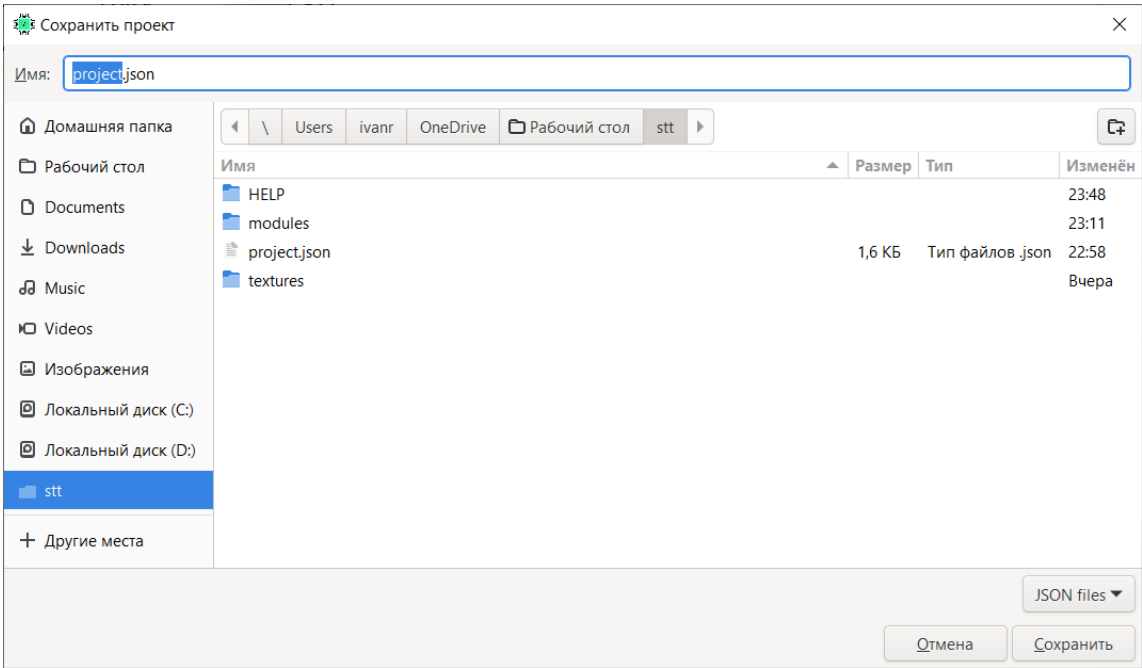


Рисунок 34 – Окно сохранения проекта

Что бы открыть готовый проект нужно нажать кнопку «Открыть» и выбрать файл проекта. На рисунке 35 показано окно выбора файла проекта.

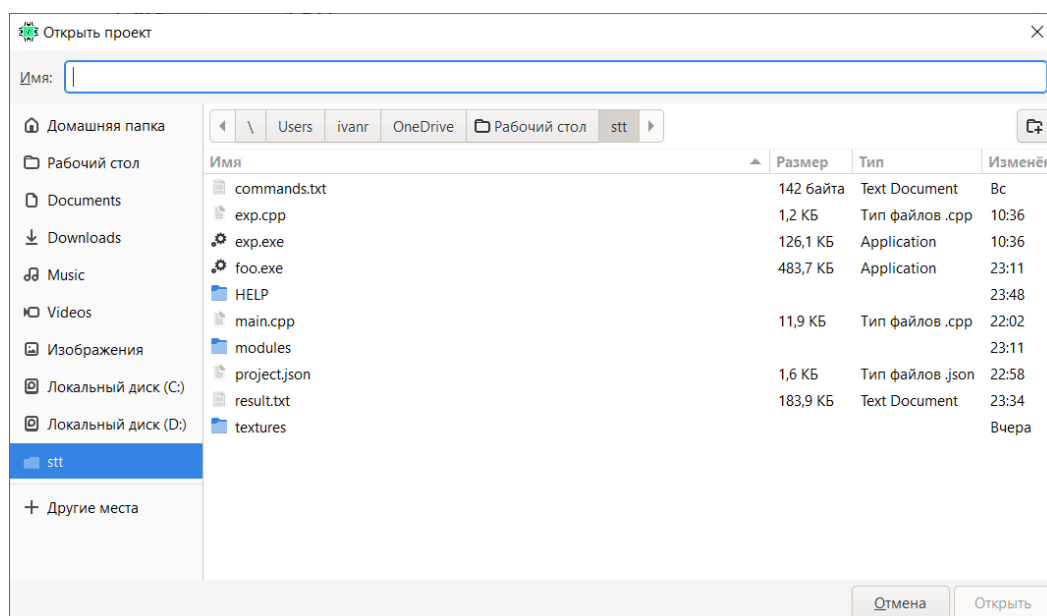


Рисунок 35 – Окно выбора файла проекта

Можно так же очистить проект. Для этого нужно нажать кнопку «Новый проект». После этого проект будет очищен. Если пользователь не сохранил текущий проект перед выходом из программы или перед началом нового проекта, пользователь будет предупрежден и ему будет предложено сохранить проект. На рисунке 36 показано окно предупреждения при создании нового проекта, на рисунке 37 показано окно предупреждения при закрытии приложения.

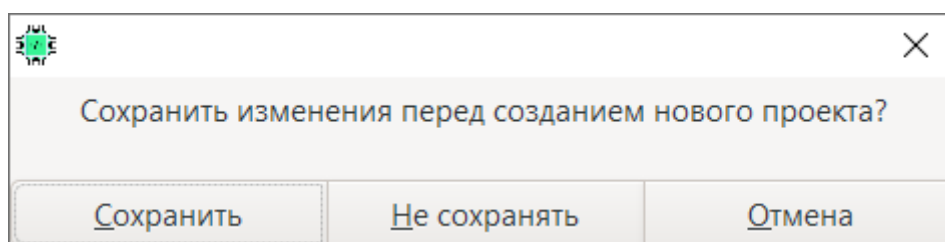


Рисунок 36 – Окно предупреждения

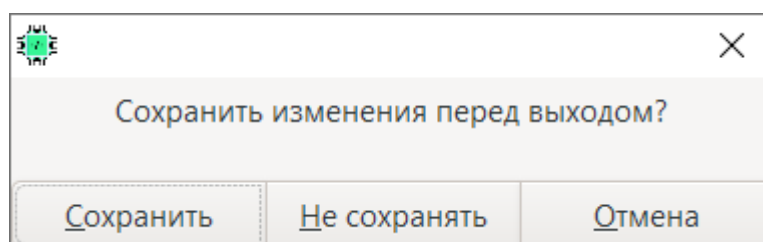


Рисунок 37 – Окно предупреждения

Так же в приложении реализовано меню. Но оно не привносит новых функций, все функции меню так же присутствуют в боковой и верхней панелях. На рисунке 38 представлено верхнее меню.

Файл Редактировать Сеть Отчёты Справка

Рисунок 38 – Верхнее меню

2 Охрана труда

В этой части будут рассмотрены необходимые требования по обеспечению безопасности жизнедеятельности и охраны труда операторов ПЭВМ, принимающих участие в разработке и эксплуатации программного обеспечения, а также способы минимизации вероятности получения травмы работником.

2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ

Несмотря на то, что ПЭВМ позволяют нам решать самые разные задачи существует так же вероятность получения травмы пользователем или всех находящимся людям поблизости. С целью предотвращения таких сценариев, работникам следует придерживаться разработанных правил безопасности на рабочем месте и правил эксплуатации ПЭВМ, таких как нормирование времени работы, правильное расположение рабочего места, соблюдение нормы уровня освещенности и шума.

2.2 Классификация и категорирование помещения, в котором производится разработка

2.2.1 Классификация по взрывопожарной и пожарной опасности

По взрывопожарной и пожарной опасности помещения подразделяются на категории А, Б, В1-В4, Г и Д, а здания - на категории А, Б, В, Г и Д.

Горючие вещества и материалы, их количество и пожароопасность влияют на классификацию помещений и зданий по пожарной опасности. Эта классификация также учитывает особенности планировки помещений и характер технологических процессов, которые в них осуществляются. Ниже представлены категории помещений с описанием свойств находящихся в них веществ и материалов:

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

- категория «А» — повышенная взрывопожарная опасность.

К этой группе относятся помещения, где содержатся горючие газы или легковоспламеняющиеся жидкости с температурой вспышки не выше 28 °С в объёме, достаточном для образования взрывоопасных парогазовых смесей. При воспламенении таких смесей избыточное давление взрыва в помещении может превысить 5 кПа;

- категория «Б» — взрывопожарная опасность.

В таких помещениях могут находиться горючие пыли и волокна, а также легковоспламеняющиеся жидкости с температурой вспышки выше 28 °С и другие горючие жидкости, способные образовывать взрывоопасные пылевоздушные или паровоздушные смеси. При воспламенении этих смесей расчётное избыточное давление взрыва также превышает 5 кПа;

- категории «В1–В4» — пожароопасность.

Сюда относятся вещества и материалы, которые могут гореть при определённых условиях, включая горючие и трудногорючие жидкости, а также вещества, склонные к самовозгоранию при контакте с кислородом или влагой, но не образующие взрывоопасных смесей;

- категория «Г» — умеренная пожарная опасность.

Это помещения, в которых обрабатываются негорючие вещества в нагретом или раскалённом состоянии;

- категория «Д» — пониженная пожарная опасность.

Относится к помещениям, где находятся негорючие вещества и материалы в холодном состоянии, не представляющие значительной опасности.

2.2.2 Классификация по опасности поражения электрическим током

Условия, при которых возрастает риск поражения человека электрическим током:

- повышенная влажность воздуха — в помещении образуется конденсат в виде мелких капель, а относительная влажность превышает 75%;

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		49

- наличие проводящей пыли — технологическая или иная пыль оседает на проводах и попадает внутрь оборудования, ухудшая изоляцию и теплоотвод, но не создающая риска возгорания или взрыва;

- присутствие токопроводящих полов и поверхностей;

- повышенная температура окружающей среды.

Условия особой опасности поражения людей электрическим током:

- сырость помещения;

- химически активная среда;

- наличие одновременно двух или более факторов повышенной опасности.

Таким образом можно выделить три категории опасности помещений: безопасные, с повышенной опасностью и особо опасные.

Для снижения вероятности поражения электрическим током рекомендуется организовывать рабочие места в помещениях, соответствующих следующим нормам:

- температура воздуха должна быть в пределах 18–20 °С, но не превышать 18-20°С;

- полы необходимо выполнять из диэлектрических материалов, исключающих возможность прохождения тока;

- обязательно наличие системы вентиляции;

- уровень влажности не должен превышать 60%.

2.3 Санитарно-гигиеническая характеристика производства

Обязательными санитарно-гигиеническими нормами являются: сухость помещения, комфортная температура, достаточная освещённость. Кроме того, установлены следующие базовые нормы для организации безопасных условий труда:

					ПГУ 1.090301.18.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

- объём каждого рабочего места должен составлять не менее 15 м³;
- высота потолков — не ниже 3,2м;
- ширина дверного проёма — минимум 0,8 м;
- минимальная ширина коридоров — 1,4м.

Естественное освещение должно быть организовано так, чтобы каждый рабочий участок получал достаточное количество света. Все окна необходимо очищать от загрязнений и обеспечивать возможность проветривания. В случае недостатка света в помещении должны быть установлены искусственные источники освещения. Световые фонари должны быть выполнены из армированного стекла и защищены металлическими решётками или сетками.

Нормальной считается температура 18-21°C, при влажности 40-60%.

Работодатели обязаны обеспечить доступ к чистой питьевой воде. Рекомендуются обновлять запас воды не реже одного раза в 24 часа.

Помещения, где уровень шума превышает допустимые значения, подлежат обязательной маркировке предупредительными знаками. Каждый сотрудник должен быть ознакомлен с мерами предосторожности при повышенном уровне шума.

2.4 Расчет контура заземления производственного помещения, где установлены ПЭВМ

При использовании любого электрического устройства, подключённого к сети, существует риск поражения электрическим током. Во избежание этого в помещениях, где эксплуатируются персональные электронно-вычислительные машины (ПЭВМ), устанавливают системы защитного заземления. Для изготовления заземляющих магистралей применяется стальная полоса или круглая сталь сечением не менее 48 мм².

В таблице 2.1 приведены минимальные размеры стальных заземлителей и заземляющих проводников, мм.

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		51

Таблица 2.1 - Минимальные размеры стальных заземлителей и заземляющих проводников, мм

Заземлитель	Место расположения		
	В зданиях	В наружных установках	В земле
Круглые, диаметром, мм	5	6	10
Прямоугольные, сечением, мм ²	24	48	48
Прямоугольные, толщиной, мм ²	3	4	4
Угловая сталь с толщиной полос, мм	2	2,5	4
Стальные водопроводные (некондиционные) трубы с толщиной стенок, мм	2,5	2,5	3,5

Заземляющий проводник соединяется с контуром заземления посредством сварки в двух и более точках. Длина нахлесточного соединения должна составлять не менее двух ширин используемого проводника. Выбор диаметра болтового соединения зависит от номинального тока потребителя: при токе потребителя до 16 А, диаметр болта 4 мм потребителя 250-30 А диаметр болта 10 мм.

В таблице 2.2 представлены допустимые значения сопротивлений заземляющих устройств электроустановок.

Таблица 2.2 - Наибольшие допустимые значения сопротивлений заземляющих устройств электроустановок

Характеристика объекта	Сопротивление, Ом
Электроустановки напряжением 110кВ и выше сетей с эффективным заземлением нейтрали, выполненные по нормам на сопротивление	0,5 $0,002 \cdot 0,5\rho$
Электроустановки 3-35 кВ сетей с изолированной нейтралью	
Электроустановки сетей напряжением до 1000 В с глухозаземленной нейтралью напряжением:	
660/380 В	$15^{**} (15 \cdot 0,01\rho)$
380/220 В	$30^{**} (30 \cdot 0,01\rho)$
220/127 В	$60^{**} (60 \cdot 0,01\rho)$
Электроустановки сетей напряжением до 1000 В с изолированной нейтралью при мощности источника питания:	
более 100 кВА	$50/I_p^*$, но не более 4 Ом
До 100 кВА	$50/I_p^*$, но не более 10 Ом

П р и м е ч а н и е

1 * - I_p – расчетный ток замыкания на землю

2 ** - сопротивление заземляющего устройства с учетом повторных заземлений нулевого провода должно быть не более 2, 4 и 8 Ом при линейных напряжениях соответственно 660, 380 и 220В источника трехфазного тока и напряжениях 380, 220 и 127 В источника однофазного тока.

Имеется заземлитель диаметром 25...30 мм. Рассчитаем сопротивление растеканию тока по формуле, представленной ниже:

$$R = 0,9 \cdot (p / l), \quad (1)$$

где p – сопротивление грунта,

l – длина трубы.

В нашем случае:

$$R = 0,9 \cdot (420 / 4) = 94,5 \text{ (Ом)}.$$

Определим примерное число заземлителей без учёта коэффициента экранирования при допустимом сопротивлении заземляющего устройства 2 Ом:

$$n = 94,5 / 2 = 47,25 \text{ (шт)}$$

Определим число вертикальных заземлителей с учетом коэффициента экранирования:

$$n_1 = 47,25 / 0,36 = 131,25 \text{ (шт)}$$

Рассчитаем длину соединительной полосы, при расстоянии между трубами 2,75 м:

$$l_n = n_1 \cdot a = 131,25 \cdot 2,75 = 360,94, \quad (2)$$

где a - расстояние между вертикальными заземлителями.

Периметр помещения с ПЭВМ 200 (м). Расчетная длина соединительной полосы не менее периметра помещения. Рассчитаем сопротивление растеканию электрического тока через соединительную полосу:

$$R_n = 2,1 \cdot (420 / 360,94) = 2,44$$

Определим результирующее сопротивление растеканию тока всего устройства, Ом.

$$R_3 = (94,5 \cdot 2,44) / (0,21 \cdot 94,5 + 0,36 \cdot 2,44 \cdot 131,25) = 230,58 / (19,85 + 115,29) = 230,58 / 135,14 = 1,71$$

Допустимое сопротивление заземляющего устройства на электрических установках напряжением до 1000 В равно 1,71 Ом. Это

значение не превышает 4 Ом, следовательно, результирующее сопротивление растеканию тока установлено верно и соответствует норме.

					ПГУ 1.090301.18.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе реализации данной работы были изучены методы моделирования СМО, включая расчет характеристик отдельных узлов и всей сети в целом. Были освоены алгоритмы итерационного определения интенсивностей заявок, проверки устойчивости системы ($\rho_{ho} < 1$), а также методы построения диаграмм и графиков для отображения результатов. Особое внимание уделено применению библиотек GTK+ и Cairo для разработки графического интерфейса и отрисовки диаграмм, а также использованию структур данных для хранения параметров узлов и переходов между ними.

В процессе выполнения работы были получены навыки разработки программного обеспечения с использованием GTK+ для создания кроссплатформенной программы, интерфейса, реализации алгоритмов моделирования СМО, работы с графиками, отрисовкой и диаграммами через библиотеку Cairo, проектирования пользовательских интерфейсов и управления состоянием программы. Также были углублены знания в области объектно-ориентированного программирования для структурирования кода и упрощения взаимодействия между модулями.

Разработанная программа может быть применена в разработке сетей телекоммуникации, логистике, управлении ресурсами, в качестве учебного пособия и других областях, где требуется анализ и оптимизация СМО. Ее развитие открывает возможности для создания профессионального инструмента для инженеров и исследователей.

Список использованных источников

1. Моделирование биологических процессов и систем. — URL: <https://studfile.net/preview/7423983/> (дата обращения 10.04.2025).
2. Классификация систем массового обслуживания. — URL: https://koi.tspu.ru/koi_books/gorchakov/klassif2.htm (дата обращения 10.04.2025).
3. Модульная архитектура: что, как и почему. — URL: <https://proglib.io/p/modulnaya-arhitektura-cto-kak-i-pochemu-2023-04-04> (дата обращения 10.04.2025).
4. Введение в C++. — URL: <https://metanit.com/cpp/tutorial/1.1.php> (дата обращения 10.04.2025).
5. Документация JSON. — URL: <https://www.json.org/json-ru.html> (дата обращения 10.04.2025).
6. Документация json-glib. — URL: <https://wiki.gnome.org/Projects/JsonGlib> (дата обращения 10.04.2025).
7. Документация Visual Studio. — URL: <https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022> (дата обращения 10.04.2025).

Приложение А

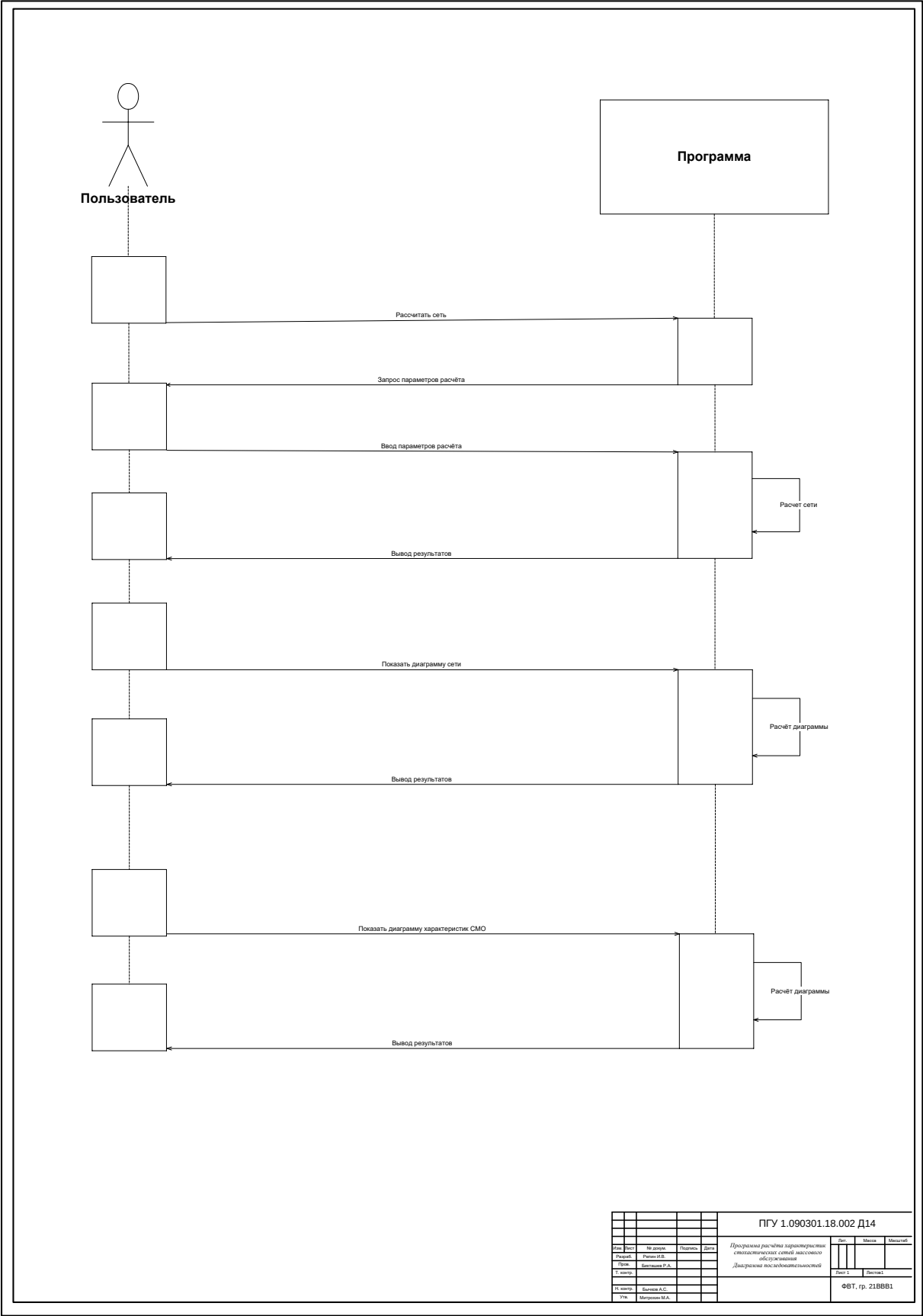
(обязательное)

Программа расчёта характеристик стохастических сетей массового
обслуживания

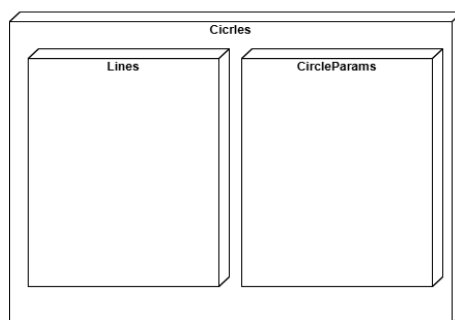
UML-диаграммы

					ПГУ 1.090301.18.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 - Диаграмма последовательностей

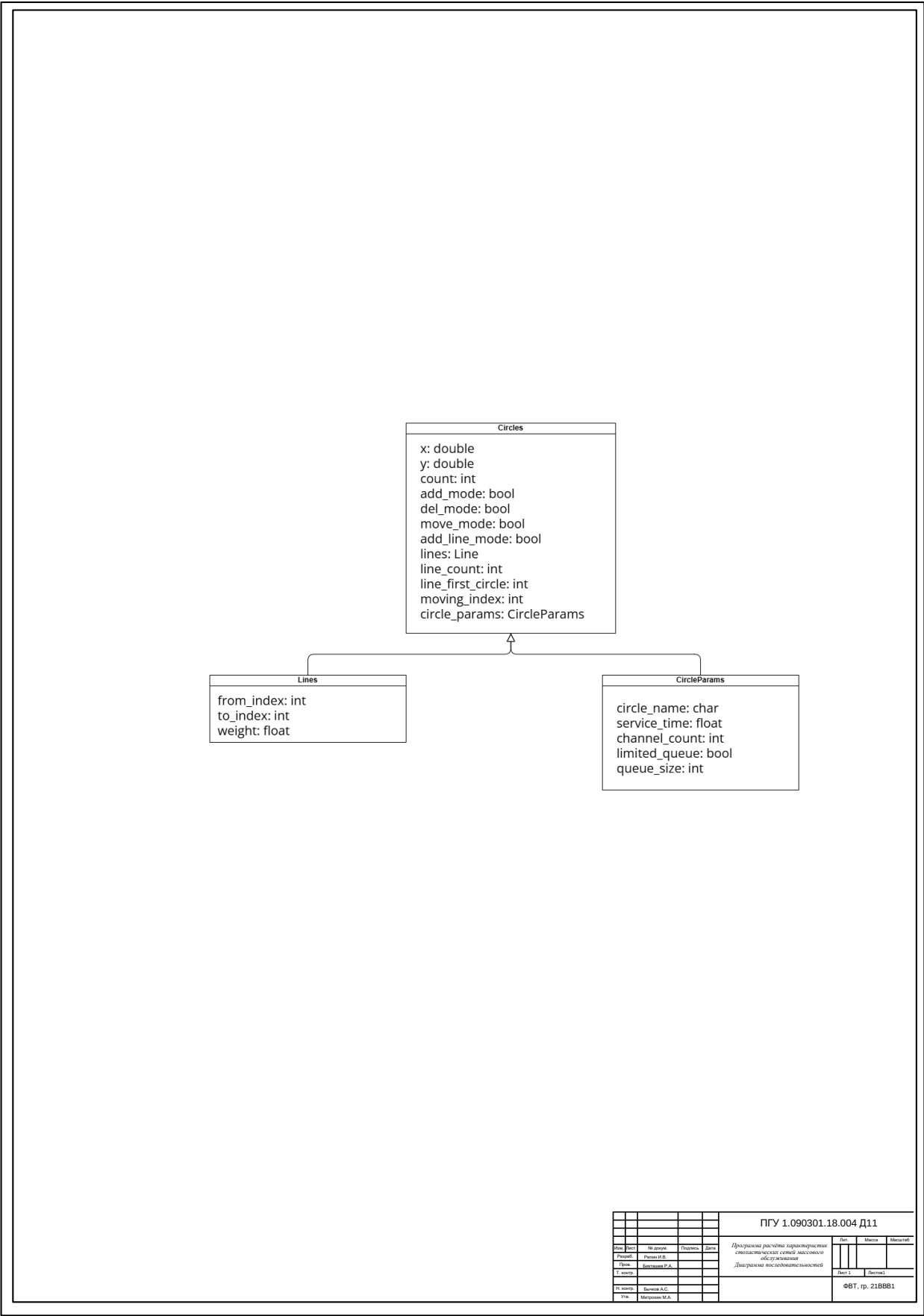


Приложение А.2 - Диаграмма развертывания



						ПГУ 1.090301.18.003 Д19			
						Программа расчёта характеристик статистических сетей массового обслуживания Диаграмма востребованностей			
Имя	Фамилия	№ докум.	Подпись	Дата					
Рисовал	Решал								
Прок.	Выполнил								
Т. номер									
И. номер	Выполнил								
						Лист 1 Листов 1			
						ФБТ, пр. 218ВВ1			

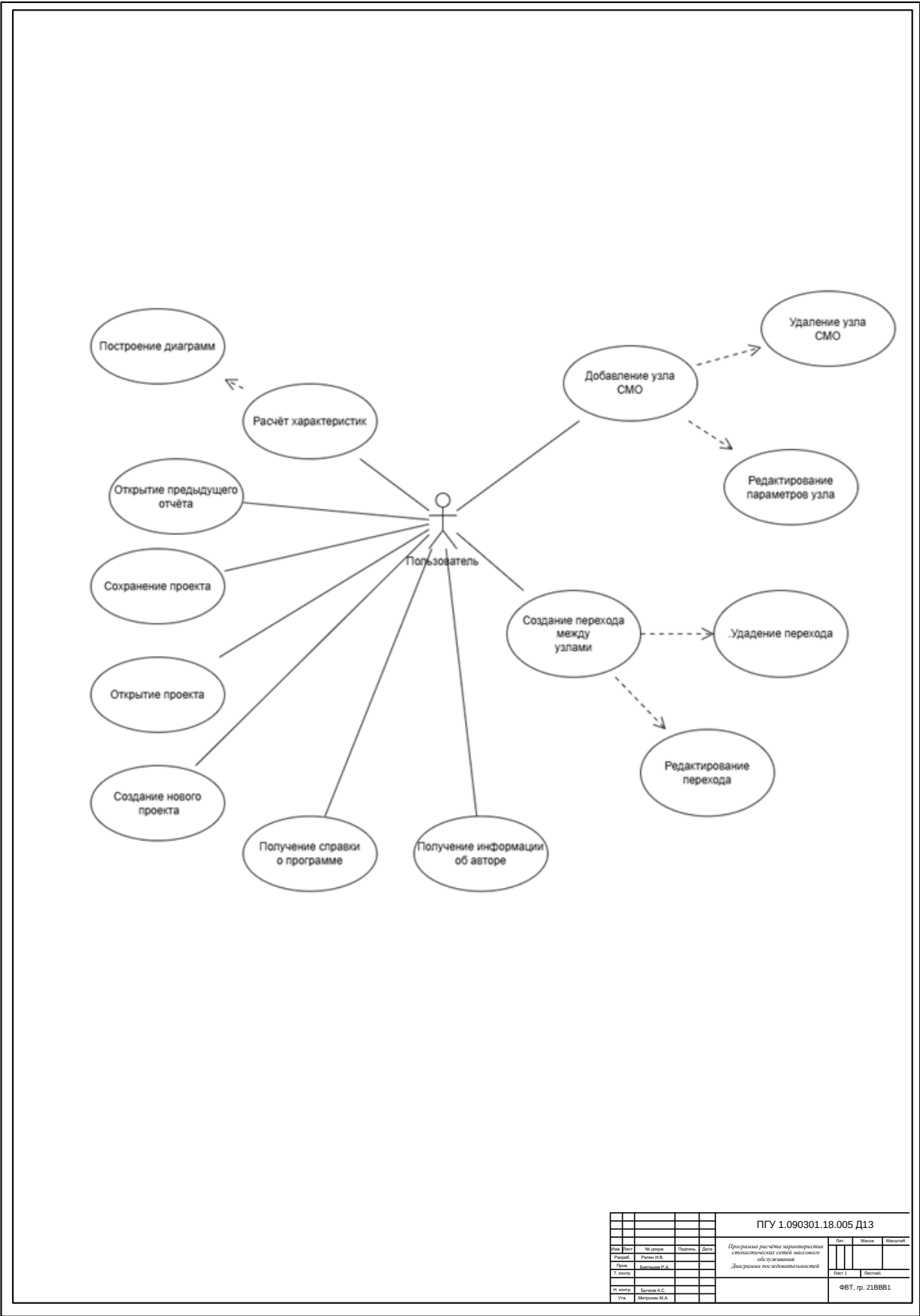
Приложение А.3 - Диаграмма классов



ПГУ 1.090301.18.004 Д11									
Имя	Фамилия	№ докум.	Подпись	Дата	Программа расчета характеристик статистических сетей массового обслуживания				
Рисовал	Рисовал И.Б.	Деталь	Деталь И.А.		Диаграмма последовательностей				
1. лист					Лист 1				
25.01.2018	Воснован И.С.				ФБТ, р. 218881				
2018	Михайлов И.А.								

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4 - Диаграмма прецедентов



ПГУ 1.090301.18.005 Д13									
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг
Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг	Имя	Долг

Изм.	Лист	№ докум.	Подпись	Дата

Приложение Б

(обязательное)

Программа расчёта характеристик стохастических сетей массового
обслуживания

Листинг программы

					ПГУ 1.090301.18.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

Файл main.h

```
#ifndef MAIN_H
#define MAIN_H

#include <stdio.h>
#include <gtk/gtk.h>
#include <glib.h>
#include <math.h>
#include <json-glib/json-glib.h>

#endif
```

Файл main.cpp

```
#include "modules/ui.cpp"
#include "modules/model.h"

static void activate(GtkApplication* app, gpointer user_data)
{
    GtkWidget* window = gtk_application_window_new(app);
    gtk_window_set_title(GTK_WINDOW(window), "STSET");
    gtk_window_set_default_size(GTK_WINDOW(window), 1000, 800);
    gtk_window_set_icon_from_file(GTK_WINDOW(window),
    "textures/icon_programm.png", NULL);
    GtkWidget* grid = gtk_grid_new();
    GtkWidget* vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 0);
    gtk_container_add(GTK_CONTAINER(window), vbox);
    GtkWidget* menubar = gtk_menu_bar_new();
    gtk_box_pack_start(GTK_BOX(vbox), menubar, FALSE, FALSE, 0);

    GtkWidget* addCMO = my_button_new("Добавить СМО",
    "textures/ADD.png");
    GtkWidget* delCMO = my_button_new("Удалить СМО",
    "textures/DEL.png");
    GtkWidget* addLine = my_button_new("Провести связь",
    "textures/LINK.png");
    GtkWidget* delLine = my_button_new("Удалить связь",
    "textures/DLINK.png");
    GtkWidget* moveCMO = my_button_new("Переместить СМО",
    "textures/MOV.png");
    GtkWidget* reNew = my_button_new("Удалить холст",
    "textures/REF.png");

    GtkWidget* newProject = my_button_new("Новый",
    "textures/NEW.png");
    GtkWidget* openProject = my_button_new("Открыть",
    "textures/OPEN.png");
    GtkWidget* saveProject = my_button_new("Сохранить",
    "textures/SAVE.png");
    GtkWidget* editMatrix = my_button_new("Редактор матрицы
    вероятностей передач", "textures/NETSERV.png");
```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		64

```

        GtkWidget* calcNet = my_button_new("Рассчитать сеть",
"textures/Connect.png");
        GtkWidget* showReport = my_button_new("Показать отчет",
"textures/TABLE.png");
        GtkWidget* showDiagNet = my_button_new("Показать диаграмму
характеристик сети", "textures/GRPHBAR.png");
        GtkWidget* showDiagChar = my_button_new("Показать диаграмму
характеристик СМО", "textures/GRF.png");
        GtkWidget* reference = my_button_new("Справка",
"textures/HELP.png");
        GtkWidget* aboutProgramm = my_button_new("О программе",
"textures/TICK.png");

        gtk_grid_attach(GTK_GRID(grid), addCMO, 0, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), delCMO, 0, 2, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), addLine, 0, 3, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), delLine, 0, 4, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), moveCMO, 0, 5, 1, 1);

        gtk_grid_attach(GTK_GRID(grid), newProject, 1, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), openProject, 2, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), saveProject, 3, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), editMatrix, 4, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), calcNet, 5, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), showReport, 6, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), showDiagNet, 7, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), showDiagChar, 8, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), reference, 9, 1, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), aboutProgramm, 10, 1, 1, 1);

        GtkWidget* drawing_area = gtk_drawing_area_new();
        gtk_widget_set_size_request(drawing_area, 1000, 800);
        gtk_grid_attach(GTK_GRID(grid), drawing_area, 1, 2, 1000, 1000);

        Circles* circles = g_new0(Circles, 1);
        circles->add_mode = FALSE;
        circles->del_mode = FALSE;

        g_signal_connect(G_OBJECT(drawing_area), "draw",
G_CALLBACK(on_draw_event), circles);
        g_signal_connect(drawing_area, "button-press-event",
G_CALLBACK(on_button_press), circles);
        g_signal_connect(drawing_area, "motion-notify-event",
G_CALLBACK(on_motion_notify), circles);
        g_signal_connect(drawing_area, "button-release-event",
G_CALLBACK(on_button_release), circles);
        gtk_widget_add_events(drawing_area, GDK_BUTTON_PRESS_MASK |
GDK_BUTTON_RELEASE_MASK | GDK_POINTER_MOTION_MASK);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		65

```

        g_signal_connect(G_OBJECT(addCMO), "clicked",
G_CALLBACK(on_add_cmo_clicked), circles);
        g_signal_connect(G_OBJECT(delCMO), "clicked",
G_CALLBACK(on_del_cmo_clicked), circles);
        g_signal_connect(G_OBJECT(moveCMO), "clicked",
G_CALLBACK(on_move_cmo_clicked), circles);
        g_signal_connect(G_OBJECT(addLine), "clicked",
G_CALLBACK(on_add_line_clicked), circles);
        g_signal_connect(G_OBJECT(delLine), "clicked",
G_CALLBACK(on_delLine_button_clicked), circles);
        g_signal_connect(G_OBJECT(editMatrix), "clicked",
G_CALLBACK(open_edit_weights_window), circles);
        g_signal_connect(G_OBJECT(calcNet), "clicked",
G_CALLBACK(on_calcNet_clicked), circles);
        g_signal_connect(G_OBJECT(saveProject), "clicked",
G_CALLBACK(on_save_project_clicked), circles);
        g_signal_connect(G_OBJECT(openProject), "clicked",
G_CALLBACK(on_open_project_clicked), circles);
        g_signal_connect(window, "delete-event",
G_CALLBACK(on_main_window_delete_event), circles);
        g_signal_connect(G_OBJECT(newProject), "clicked",
G_CALLBACK(on_new_project_button_clicked), circles);
        g_signal_connect(G_OBJECT(showReport), "clicked",
G_CALLBACK(on_show_report_button_clicked), circles);
        g_signal_connect(G_OBJECT(showDiagNet), "clicked",
G_CALLBACK(on_showDiagNet_clicked), window);
        g_signal_connect(G_OBJECT(showDiagChar), "clicked",
G_CALLBACK(show_diag_char), circles);
        g_signal_connect(G_OBJECT(reference), "clicked",
G_CALLBACK(on_reference_button_clicked), window);
        g_signal_connect(G_OBJECT(aboutProgramm), "clicked",
G_CALLBACK(on_about_programm_clicked), NULL);

GtkWidget* file_menu = gtk_menu_new();
GtkWidget* file_item = gtk_menu_item_new_with_label(
        g_locale_to_utf8("Файл", -1, NULL, NULL, NULL));
gtk_menu_item_set_submenu(GTK_MENU_ITEM(file_item), file_menu);

GtkWidget* new_item = gtk_menu_item_new_with_label(
        g_locale_to_utf8("Новый", -1, NULL, NULL, NULL));
GtkWidget* open_item = gtk_menu_item_new_with_label(
        g_locale_to_utf8("Открыть", -1, NULL, NULL, NULL));
GtkWidget* save_item = gtk_menu_item_new_with_label(
        g_locale_to_utf8("Сохранить", -1, NULL, NULL, NULL));
gtk_menu_shell_append(GTK_MENU_SHELL(file_menu), new_item);
gtk_menu_shell_append(GTK_MENU_SHELL(file_menu), open_item);
gtk_menu_shell_append(GTK_MENU_SHELL(file_menu), save_item);

GtkWidget* edit_menu = gtk_menu_new();
GtkWidget* edit_item = gtk_menu_item_new_with_label(
        g_locale_to_utf8("Редактировать", -1, NULL, NULL, NULL));

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		66

```

gtk_menu_item_set_submenu(GTK_MENU_ITEM(edit_item), edit_menu);

GtkWidget* edit_matrix = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Редактор матрицы вероятностей передач", -
1, NULL, NULL, NULL));
GtkWidget* add_smo = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Добавить СМО", -1, NULL, NULL, NULL));
GtkWidget* delete_smo = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Удалить СМО", -1, NULL, NULL, NULL));
GtkWidget* move_smo = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Переместить СМО", -1, NULL, NULL, NULL));
GtkWidget* make_link = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Провести связь", -1, NULL, NULL, NULL));
GtkWidget* delete_link = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Удалить связь", -1, NULL, NULL, NULL));

gtk_menu_shell_append(GTK_MENU_SHELL(edit_menu), edit_matrix);
gtk_menu_shell_append(GTK_MENU_SHELL(edit_menu), add_smo);
gtk_menu_shell_append(GTK_MENU_SHELL(edit_menu), delete_smo);
gtk_menu_shell_append(GTK_MENU_SHELL(edit_menu), move_smo);
gtk_menu_shell_append(GTK_MENU_SHELL(edit_menu), make_link);
gtk_menu_shell_append(GTK_MENU_SHELL(edit_menu), delete_link);

GtkWidget* network_menu = gtk_menu_new();
GtkWidget* network_item = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Сеть", -1, NULL, NULL, NULL));
gtk_menu_item_set_submenu(GTK_MENU_ITEM(network_item),
network_menu);

GtkWidget* calculate_item = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Рассчитать", -1, NULL, NULL, NULL));
gtk_menu_shell_append(GTK_MENU_SHELL(network_menu),
calculate_item);

GtkWidget* reports_menu = gtk_menu_new();
GtkWidget* reports_item = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Отчёты", -1, NULL, NULL, NULL));
gtk_menu_item_set_submenu(GTK_MENU_ITEM(reports_item),
reports_menu);

GtkWidget* view_last_report = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Посмотреть последний отчёт", -1, NULL,
NULL, NULL));
GtkWidget* view_network_chart = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Посмотреть диаграмму характеристик сети",
-1, NULL, NULL, NULL));
GtkWidget* view_smo_chart = gtk_menu_item_new_with_label(
    g_locale_to_utf8("Посмотреть диаграмму характеристик СМО", -
1, NULL, NULL, NULL));

```

```

        gtk_menu_shell_append(GTK_MENU_SHELL(reports_menu),
view_last_report);
        gtk_menu_shell_append(GTK_MENU_SHELL(reports_menu),
view_network_chart);
        gtk_menu_shell_append(GTK_MENU_SHELL(reports_menu),
view_smo_chart);

        GtkWidget* help_menu = gtk_menu_new();
        GtkWidget* help_item = gtk_menu_item_new_with_label(
                g_locale_to_utf8("Справка", -1, NULL, NULL, NULL));
        gtk_menu_item_set_submenu(GTK_MENU_ITEM(help_item), help_menu);

        GtkWidget* help_about = gtk_menu_item_new_with_label(
                g_locale_to_utf8("Справка", -1, NULL, NULL, NULL));
        GtkWidget* help_about_programm = gtk_menu_item_new_with_label(
                g_locale_to_utf8("О программе", -1, NULL, NULL, NULL));

        gtk_menu_shell_append(GTK_MENU_SHELL(help_menu), help_about);
        gtk_menu_shell_append(GTK_MENU_SHELL(help_menu),
help_about_programm);

        gtk_menu_shell_append(GTK_MENU_SHELL(menuubar), file_item);
        gtk_menu_shell_append(GTK_MENU_SHELL(menuubar), edit_item);
        gtk_menu_shell_append(GTK_MENU_SHELL(menuubar), network_item);
        gtk_menu_shell_append(GTK_MENU_SHELL(menuubar), reports_item);
        gtk_menu_shell_append(GTK_MENU_SHELL(menuubar), help_item);

        gtk_box_pack_start(GTK_BOX(vbox), menuubar, FALSE, FALSE, 0);

        g_signal_connect(add_smo, "activate",
G_CALLBACK(on_add_cmo_clicked), circles);
        g_signal_connect(delete_smo, "activate",
G_CALLBACK(on_del_cmo_clicked), circles);
        g_signal_connect(move_smo, "activate",
G_CALLBACK(on_move_cmo_clicked), circles);
        g_signal_connect(make_link, "activate",
G_CALLBACK(on_add_line_clicked), circles);
        g_signal_connect(delete_link, "activate",
G_CALLBACK(on_delLine_button_clicked), circles);
        g_signal_connect(edit_matrix, "activate",
G_CALLBACK(open_edit_weights_window), circles);
        g_signal_connect(calculate_item, "activate",
G_CALLBACK(on_calcNet_clicked), circles);
        g_signal_connect(save_item, "activate",
G_CALLBACK(on_save_project_clicked), circles);
        g_signal_connect(open_item, "activate",
G_CALLBACK(on_open_project_clicked), circles);
        g_signal_connect(new_item, "activate",
G_CALLBACK(on_new_project_button_clicked), circles);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		68

```

        g_signal_connect(view_last_report, "activate",
G_CALLBACK(on_show_report_button_clicked), circles);
        g_signal_connect(view_network_chart, "activate",
G_CALLBACK(on_showDiagNet_clicked), window);
        g_signal_connect(view_smo_chart, "activate",
G_CALLBACK(show_diag_char), circles);
        g_signal_connect(help_about, "activate",
G_CALLBACK(on_reference_button_clicked), window);
        g_signal_connect(help_about_programm, "activate",
G_CALLBACK(on_about_programm_clicked), NULL);

        gtk_box_pack_start(GTK_BOX(vbox), grid, TRUE, TRUE, 0);
        gtk_container_add(GTK_CONTAINER(window), grid);
        gtk_widget_show_all(window);
    }

    int main(int argc, char** argv)
    {
        GtkApplication* app;
        int status;
        app = gtk_application_new("org.gtk.examples",
G_APPLICATION_FLAGS_NONE);

        g_signal_connect(app, "activate", G_CALLBACK(activate), NULL);
        status = g_application_run(G_APPLICATION(app), argc, argv);
        g_object_unref(app);
        return status;
    }

```

Файл ui.h

```

#ifndef UI_H
#define UI_H

void on_add_cmo_clicked(GtkButton* button, gpointer user_data);
void on_del_cmo_clicked(GtkButton* button, gpointer user_data);
void on_move_cmo_clicked(GtkButton* button, gpointer user_data);
void on_add_line_clicked(GtkButton* button, gpointer user_data);
void on_calcNet_clicked(GtkButton* button, gpointer user_data);
static void on_save_project_clicked(GtkButton* button, gpointer
user_data);
void on_new_project_button_clicked(GtkButton* button, gpointer
user_data);
void on_show_report_button_clicked(GtkButton* button, gpointer
user_data);
void on_open_project_clicked(GtkButton* button, gpointer user_data);
void on_showDiagNet_clicked(GtkButton* button, gpointer user_data);
void on_reference_button_clicked(GtkButton* button, gpointer
user_data);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		69

```

        void on_about_programm_clicked(GtkButton* button, gpointer
user_data);
        gboolean on_main_window_delete_event(GtkWidget* widget, GdkEvent*
event, gpointer user_data);
        GtkWidget* my_button_new(const char* ttips, const gchar* icon_name);
        void show_diag_char(GtkWidget* button, gpointer user_data);
        void open_edit_weights_window(GtkWidget* button, gpointer
user_data);
        void apply_matrix_weights_editing(GtkWidget* button, gpointer
user_data);
        static gboolean on_button_press(GtkWidget* widget, GdkEventButton*
event, gpointer user_data);
        static gboolean on_motion_notify(GtkWidget* widget, GdkEventMotion*
event, gpointer user_data);
        static gboolean on_button_release(GtkWidget* widget, GdkEventButton*
event, gpointer user_data);

#endif

```

Файл ui.cpp

```

#include "visualize.cpp"
#include "model.h"
#include "ui.h"

void on_add_cmo_clicked(GtkButton* button, gpointer user_data)
{
    Circles* circles = (Circles*)user_data;
    circles->add_mode = TRUE;
}

void on_del_cmo_clicked(GtkButton* button, gpointer user_data)
{
    Circles* circles = (Circles*)user_data;
    circles->del_mode = TRUE;
}

void on_move_cmo_clicked(GtkButton* button, gpointer user_data)
{
    Circles* circles = (Circles*)user_data;
    circles->move_mode = TRUE;
    circles->moving_index = -1;
}

void on_add_line_clicked(GtkButton* button, gpointer user_data)
{
    Circles* circles = (Circles*)user_data;
    circles->add_line_mode = TRUE;
    circles->line_first_circle = -1;
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		70

```

void on_delLine_button_clicked(GtkButton* button, gpointer user_data) {
    Circles* circles = (Circles*)user_data;
    circles->delLineMode = TRUE;
}

void on_calcNet_clicked(GtkButton* button, gpointer user_data) {
    Circles* circles = (Circles*)user_data;

    GtkWidget* dialog = gtk_dialog_new_with_buttons(
        g_locale_to_utf8("Расчет сети", -1, NULL, NULL, NULL),
        GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
        GTK_DIALOG_MODAL,
        g_locale_to_utf8("_Рассчитать", -1, NULL, NULL, NULL),
        GTK_RESPONSE_OK,
        g_locale_to_utf8("_Закрыть", -1, NULL, NULL, NULL),
        GTK_RESPONSE_CLOSE,
        NULL);

    GtkWidget* content_area =
gtk_dialog_get_content_area(GTK_DIALOG(dialog));

    GtkWidget* main_grid = gtk_grid_new();
    gtk_grid_set_row_spacing(GTK_GRID(main_grid), 8);
    gtk_grid_set_column_spacing(GTK_GRID(main_grid), 10);
    gtk_container_set_border_width(GTK_CONTAINER(main_grid), 10);
    gtk_container_add(GTK_CONTAINER(content_area), main_grid);

    CalcNetWidgets* w = g_new0(CalcNetWidgets, 1);

    w->radio_fixed = gtk_radio_button_new_with_label(NULL,
g_locale_to_utf8("Расчет сети с фиксированными параметрами", -1, NULL, NULL,
NULL));
    w->radio_vary =
gtk_radio_button_new_with_label_from_widget(GTK_RADIO_BUTTON(w->radio_fixed),
g_locale_to_utf8("Расчет сети с варьируемыми параметрами", -1, NULL, NULL,
NULL));

    gtk_grid_attach(GTK_GRID(main_grid), w->radio_fixed, 0, 0, 2, 1);
    gtk_grid_attach(GTK_GRID(main_grid), w->radio_vary, 0, 1, 2, 1);

    gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w->radio_fixed),
!calc_vary_mode);
    gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w->radio_vary),
calc_vary_mode);

    g_signal_connect(w->radio_fixed, "toggled",
G_CALLBACK(on_radio_mode_toggled), w);
    g_signal_connect(w->radio_vary, "toggled",
G_CALLBACK(on_radio_mode_toggled), w);

    w->check_vary_intensity =
gtk_check_button_new_with_label(g_locale_to_utf8("Изменять интенсивность", -1,
NULL, NULL, NULL));
    gtk_grid_attach(GTK_GRID(main_grid), w->check_vary_intensity, 0, 2, 2,
1);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		71

```

        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w->check_vary_intensity),
vary_intensity);
        g_signal_connect(w->check_vary_intensity, "toggled",
G_CALLBACK(on_check_vary_intensity_toggled), NULL);

        GtkWidget* label_intensity_from = gtk_label_new(g_locale_to_utf8("от:", -
1, NULL, NULL, NULL));
        w->entry_intensity_from = gtk_entry_new();
        GtkWidget* label_intensity_to = gtk_label_new(g_locale_to_utf8("до:",
-1, NULL, NULL, NULL));
        w->entry_intensity_to = gtk_entry_new();

        char buf[64];
        snprintf(buf, sizeof(buf), "%.3f", intensity_from);
        gtk_entry_set_text(GTK_ENTRY(w->entry_intensity_from), buf);
        snprintf(buf, sizeof(buf), "%.3f", intensity_to);
        gtk_entry_set_text(GTK_ENTRY(w->entry_intensity_to), buf);

        gtk_grid_attach(GTK_GRID(main_grid), label_intensity_from, 0, 3, 1,
1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_intensity_from, 1, 3, 1,
1);
        gtk_grid_attach(GTK_GRID(main_grid), label_intensity_to, 0, 4, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_intensity_to, 1, 4, 1,
1);

        g_signal_connect(w->entry_intensity_from, "changed",
G_CALLBACK(on_entry_intensity_from_changed), NULL);
        g_signal_connect(w->entry_intensity_to, "changed",
G_CALLBACK(on_entry_intensity_to_changed), NULL);

        w->check_vary_service =
gtk_check_button_new_with_label(g_locale_to_utf8("Изменять время обслуживания в
СМО", -1, NULL, NULL, NULL));
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w->check_vary_service),
vary_service_time);
        gtk_grid_attach(GTK_GRID(main_grid), w->check_vary_service, 0, 5, 2,
1);

        w->combo_service_circle = gtk_combo_box_text_new();
        gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(w->combo_service_circle),
NULL, g_locale_to_utf8("Выберите крур", -1, NULL, NULL, NULL));
        for (int i = 1; i < circles->count; i++) {
            char buf2[32];
            snprintf(buf2, sizeof(buf2), "S%d", i);
            gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(w-
>combo_service_circle), buf2, buf2);
        }
        gtk_combo_box_set_active(GTK_COMBO_BOX(w->combo_service_circle), 0);

        gtk_grid_attach(GTK_GRID(main_grid), w->combo_service_circle, 0, 6, 2,
1);

        GtkWidget* label_service_from = gtk_label_new(g_locale_to_utf8("от:",
-1, NULL, NULL, NULL));
        w->entry_service_from = gtk_entry_new();

```

```

        GtkWidget* label_service_to = gtk_label_new(g_locale_to_utf8("до:", -1,
NULL, NULL, NULL));
        w->entry_service_to = gtk_entry_new();

        gtk_grid_attach(GTK_GRID(main_grid), label_service_from, 0, 7, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_service_from, 1, 7, 1,
1);

        gtk_grid_attach(GTK_GRID(main_grid), label_service_to, 0, 8, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_service_to, 1, 8, 1, 1);

        g_signal_connect(w->check_vary_service, "toggled",
G_CALLBACK(on_check_vary_service_toggled), w);


        g_signal_connect(w->combo_service_circle, "changed",
G_CALLBACK(on_combo_service_changed), w);
        g_signal_connect(w->entry_service_from, "changed",
G_CALLBACK(on_entry_service_from_changed), w);
        g_signal_connect(w->entry_service_to, "changed",
G_CALLBACK(on_entry_service_to_changed), w);

        w->check_vary_channels =
gtk_check_button_new_with_label(g_locale_to_utf8("Изменять число каналов в СМО",
-1, NULL, NULL, NULL));
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_channels), vary_channel_count);
        gtk_grid_attach(GTK_GRID(main_grid), w->check_vary_channels, 0, 9, 2,
1);

        w->combo_channel_circle = gtk_combo_box_text_new();
        gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(w->combo_channel_circle),
NULL, g_locale_to_utf8("Выберите круг", -1, NULL, NULL, NULL));
        for (int i = 1; i < circles->count; i++) {
            char buf2[32];
            snprintf(buf2, sizeof(buf2), "S%d", i);
            gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(w-
>combo_channel_circle), buf2, buf2);
        }
        gtk_combo_box_set_active(GTK_COMBO_BOX(w->combo_channel_circle), 0);

        gtk_grid_attach(GTK_GRID(main_grid), w->combo_channel_circle, 0, 10,
2, 1);

        GtkWidget* label_channel_from = gtk_label_new(g_locale_to_utf8("от:",
-1, NULL, NULL, NULL));
        w->entry_channel_from = gtk_entry_new();
        GtkWidget* label_channel_to = gtk_label_new(g_locale_to_utf8("до:", -
1, NULL, NULL, NULL));
        w->entry_channel_to = gtk_entry_new();

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		73

```

        gtk_grid_attach(GTK_GRID(main_grid), label_channel_from, 0, 11, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_channel_from, 1, 11, 1,
1);

        gtk_grid_attach(GTK_GRID(main_grid), label_channel_to, 0, 12, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_channel_to, 1, 12, 1,
1);

        g_signal_connect(w->check_vary_channels, "toggled",
G_CALLBACK(on_check_vary_channels_toggled), w);
        g_signal_connect(w->combo_channel_circle, "changed",
G_CALLBACK(on_combo_channel_changed), w);
        g_signal_connect(w->entry_channel_from, "changed",
G_CALLBACK(on_entry_channel_from_changed), w);
        g_signal_connect(w->entry_channel_to, "changed",
G_CALLBACK(on_entry_channel_to_changed), w);

        w->check_vary_queue =
gtk_check_button_new_with_label(g_locale_to_utf8("Изменять число мест в СМО", -
1, NULL, NULL, NULL));
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w->check_vary_queue),
vary_queue_size);
        gtk_grid_attach(GTK_GRID(main_grid), w->check_vary_queue, 0, 13, 2,
1);


        w->combo_queue_circle = gtk_combo_box_text_new();
        gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(w->combo_queue_circle),
NULL, g_locale_to_utf8("Выберите кпур", -1, NULL, NULL, NULL));
        for (int i = 1; i < circles->count; i++) {
            char buf2[32];
            snprintf(buf2, sizeof(buf2), "S%d", i);
            gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(w-
>combo_queue_circle), buf2, buf2);
        }
        gtk_combo_box_set_active(GTK_COMBO_BOX(w->combo_queue_circle), 0);

        gtk_grid_attach(GTK_GRID(main_grid), w->combo_queue_circle, 0, 14, 2,
1);

        GtkWidget* label_queue_from = gtk_label_new(g_locale_to_utf8("от:", -
1, NULL, NULL, NULL));
        w->entry_queue_from = gtk_entry_new();
        GtkWidget* label_queue_to = gtk_label_new(g_locale_to_utf8("до:", -1,
NULL, NULL, NULL));
        w->entry_queue_to = gtk_entry_new();

        gtk_grid_attach(GTK_GRID(main_grid), label_queue_from, 0, 15, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_queue_from, 1, 15, 1,
1);

        gtk_grid_attach(GTK_GRID(main_grid), label_queue_to, 0, 16, 1, 1);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		74

```

        gtk_grid_attach(GTK_GRID(main_grid), w->entry_queue_to, 1, 16, 1, 1);

        g_signal_connect(w->check_vary_queue, "toggled",
G_CALLBACK(on_check_vary_queue_toggled), w);
        g_signal_connect(w->combo_queue_circle, "changed",
G_CALLBACK(on_combo_queue_changed), w);
        g_signal_connect(w->entry_queue_from, "changed",
G_CALLBACK(on_entry_queue_from_changed), w);
        g_signal_connect(w->entry_queue_to, "changed",
G_CALLBACK(on_entry_queue_to_changed), w);

        GtkWidget* label_cycles = gtk_label_new(g_locale_to_utf8("Число
циклов:", -1, NULL, NULL, NULL));
        w->entry_cycles_count = gtk_entry_new();
        snprintf(buf, sizeof(buf), "%d", cycles_count);
        gtk_entry_set_text(GTK_ENTRY(w->entry_cycles_count), buf);

        gtk_grid_attach(GTK_GRID(main_grid), label_cycles, 0, 17, 1, 1);
        gtk_grid_attach(GTK_GRID(main_grid), w->entry_cycles_count, 1, 17, 1,
1);

        g_signal_connect(w->entry_cycles_count, "changed",
G_CALLBACK(on_entry_cycles_count_changed), NULL);

        gtk_widget_show_all(dialog);

        gint response = gtk_dialog_run(GTK_DIALOG(dialog));

        if (response == GTK_RESPONSE_OK) {
            if (calc_vary_mode) {
                vary_and_calculate(circles, button);
            }

else {
                cycles_count = 1;
                calculate_network(circles, input_intensity, 0, button);
            }

            gtk_widget_destroy(dialog);

            g_free(w);
        }

        static void on_save_project_clicked(GtkButton* button, gpointer user_data)
{
            Circles* circles = (Circles*)user_data;

```

```

        GtkWidget* dialog = gtk_file_chooser_dialog_new(
            g_locale_to_utf8("Сохранить проект", -1, NULL, NULL, NULL),
            GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
            GTK_FILE_CHOOSER_ACTION_SAVE,
            g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
            GTK_RESPONSE_CANCEL,
            g_locale_to_utf8("_Сохранить", -1, NULL, NULL, NULL),
            GTK_RESPONSE_ACCEPT,
            NULL);
        GtkFileFilter* filter = gtk_file_filter_new();
        gtk_file_filter_set_name(filter, "JSON files");
        gtk_file_filter_add_pattern(filter, "*.json");
        gtk_file_chooser_add_filter(GTK_FILE_CHOOSER(dialog), filter);

        gtk_file_chooser_set_current_name(GTK_FILE_CHOOSER(dialog),
            "project.json");

        if (gtk_dialog_run(GTK_DIALOG(dialog)) == GTK_RESPONSE_ACCEPT) {
            char* filename =
            gtk_file_chooser_get_filename(GTK_FILE_CHOOSER(dialog));
            if (filename) {
                if (!save_circles_to_file(circles, filename)) {
                    GtkWidget* error_dialog =
            gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
                GTK_DIALOG_MODAL,
                GTK_MESSAGE_ERROR,
                GTK_BUTTONS_OK,
                g_locale_to_utf8("Не удалось сохранить файл: %s", -1,
            NULL, NULL, NULL), filename);
                    gtk_dialog_run(GTK_DIALOG(error_dialog));
                    gtk_widget_destroy(error_dialog);
                }
                g_free(filename);
            }
        }
        project_modified = false;
        gtk_widget_destroy(dialog);
    }

    void on_new_project_button_clicked(GtkButton* button, gpointer user_data)
    {
        Circles* circles = (Circles*)user_data;
        if (project_modified) {
            GtkWidget* dialog =
            gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),

            GTK_DIALOG_MODAL,
            GTK_MESSAGE_QUESTION,
            GTK_BUTTONS_NONE,
            g_locale_to_utf8("Сохранить изменения перед созданием нового
            проекта?", -1, NULL, NULL, NULL));

```

```

        gtk_dialog_add_buttons(GTK_DIALOG(dialog),
                               g_locale_to_utf8("_Сохранить", -1, NULL, NULL, NULL),
GTK_RESPONSE_YES,
                               g_locale_to_utf8("_Не сохранять", -1, NULL, NULL, NULL),
GTK_RESPONSE_NO,
                               g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
GTK_RESPONSE_CANCEL,
                               NULL);

        gint response = gtk_dialog_run(GTK_DIALOG(dialog));
        gtk_widget_destroy(dialog);

        switch (response) {
        case GTK_RESPONSE_YES: {
            GtkWidget* file_dialog =
gtk_file_chooser_dialog_new("Сохранить проект",
                             GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
                             GTK_FILE_CHOOSER_ACTION_SAVE,
                             g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
GTK_RESPONSE_CANCEL,
                             g_locale_to_utf8("_Сохранить", -1, NULL, NULL, NULL),
GTK_RESPONSE_ACCEPT,
                             NULL);

            gtk_file_chooser_set_do_overwrite_confirmation(GTK_FILE_CHOOSER(file_dialog),
TRUE);

            gint save_response = gtk_dialog_run(GTK_DIALOG(file_dialog));
            if (save_response == GTK_RESPONSE_ACCEPT) {
                char* filename =
gtk_file_chooser_get_filename(GTK_FILE_CHOOSER(file_dialog));
                bool save_ok = save_circles_to_file(circles, filename);
                g_free(filename);
                gtk_widget_destroy(file_dialog);

                if (!save_ok) {
                    GtkWidget* error_dialog =
gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
                        GTK_DIALOG_MODAL,
                        GTK_MESSAGE_ERROR,
                        GTK_BUTTONS_OK,
                        g_locale_to_utf8("_Отмена", -1, NULL, NULL,
NULL));

                    gtk_dialog_run(GTK_DIALOG(error_dialog));
                    gtk_widget_destroy(error_dialog);
                    return;
                }
            }
            else {
                gtk_widget_destroy(file_dialog);
                return;
            }
        }
    }
}

```

```

        break;
    }
    case GTK_RESPONSE_NO:

        break;
    case GTK_RESPONSE_CANCEL:
    default:

        return;
    }
}

clear_circles(circles);

project_modified = false;
}

void on_show_report_button_clicked(GtkButton* button, gpointer user_data)
{
    const char* filename = "result.txt";

    char* current_dir = g_get_current_dir();
    if (!current_dir) {
        GtkWidget* error_dialog =
gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
                        GTK_DIALOG_MODAL,
                        GTK_MESSAGE_ERROR,
                        GTK_BUTTONS_OK,
                        g_locale_to_utf8("Не удалось получить текущую директорию.", -
1, NULL, NULL, NULL));
        gtk_dialog_run(GTK_DIALOG(error_dialog));
        gtk_widget_destroy(error_dialog);
        return;
    }

    char* filepath = g_build_filename(current_dir, "result.txt", NULL);
    g_free(current_dir);

    if (!g_file_test(filepath, G_FILE_TEST_EXISTS)) {
        GtkWidget* error_dialog =
gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
                        GTK_DIALOG_MODAL,
                        GTK_MESSAGE_ERROR,
                        GTK_BUTTONS_OK,
                        g_locale_to_utf8("Файл отчета не найден.", -1, NULL, NULL,
NULL));
        gtk_dialog_run(GTK_DIALOG(error_dialog));
        gtk_widget_destroy(error_dialog);
        g_free(filepath);
        return;
    }

    char* file_uri = g_filename_to_uri(filepath, NULL, NULL);
    if (!file_uri) {
        GtkWidget* error_dialog =
gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
                        GTK_DIALOG_MODAL,
                        GTK_MESSAGE_ERROR,

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		78

```

        GTK_BUTTONS_OK,
        g_locale_to_utf8("Не удалось сформировать URI для файла.", -1,
        NULL, NULL, NULL));
        gtk_dialog_run(GTK_DIALOG(error_dialog));
        gtk_widget_destroy(error_dialog);
        return;
    }

    GError* error = NULL;
    gboolean success =
gtk_show_uri_on_window(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
file_uri, GDK_CURRENT_TIME, &error);

    g_free(file_uri);

    if (!success) {
        GtkWidget* error_dialog =
gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
        GTK_DIALOG_MODAL,
        GTK_MESSAGE_ERROR,
        GTK_BUTTONS_OK,
        g_locale_to_utf8("Не удалось открыть файл", -1, NULL, NULL,
        NULL));

        gtk_dialog_run(GTK_DIALOG(error_dialog));
        gtk_widget_destroy(error_dialog);
        g_error_free(error);
    }
}

void on_open_project_clicked(GtkButton* button, gpointer user_data) {
    GtkWidget* dialog;
    Circles* circles = (Circles*)user_data;

    dialog = gtk_file_chooser_dialog_new(
        g_locale_to_utf8("Открыть проект", -1, NULL, NULL, NULL),
        GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(button))),
        GTK_FILE_CHOOSER_ACTION_SAVE,
        g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
        GTK_RESPONSE_CANCEL,
        g_locale_to_utf8("_Открыть", -1, NULL, NULL, NULL),
        GTK_RESPONSE_ACCEPT,
        NULL);

    if (gtk_dialog_run(GTK_DIALOG(dialog)) == GTK_RESPONSE_ACCEPT) {
        char* filename;
        filename =
gtk_file_chooser_get_filename(GTK_FILE_CHOOSER(dialog));
        if (load_circles_from_file(filename, circles)) {
            g_print("Project loaded from %s\n", filename);

            gtk_widget_queue_draw(GTK_WIDGET(gtk_widget_get_toplevel(GTK_WIDGET(button))));
        }
        else {
            g_printerr("Error loading project %s\n", filename);
        }
        g_free(filename);
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		79

```

gtk_widget_destroy(dialog);
}

void on_showDiagNet_clicked(GtkButton* button, gpointer user_data) {
    GtkWidget* parent = GTK_WIDGET(user_data);
    show_diag_net_dialog(parent);
}

void on_reference_button_clicked(GtkButton* button, gpointer user_data) {
    GtkWidget* parent_window = GTK_WINDOW(user_data);
    char* path = g_strdup("HELP/HELP.HTM");
    char* abs_path = g_canonicalize_filename(path, NULL);
    g_free(path);

    char* uri = g_filename_to_uri(abs_path, NULL, NULL);
    g_free(abs_path);

    GError* error = NULL;
    gboolean res = gtk_show_uri_on_window(parent_window, uri,
GDK_CURRENT_TIME, &error);
    g_free(uri);

    if (!res) {
        g_printerr("Error: %s\n", error->message);
        g_clear_error(&error);
    }
}

void on_about_programm_clicked(GtkButton* button, gpointer user_data) {
    GtkWidget* dialog = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(dialog), g_locale_to_utf8("O
программе", -1, NULL, NULL, NULL));
    gtk_window_set_default_size(GTK_WINDOW(dialog), 300, 150);
    gtk_window_set_position(GTK_WINDOW(dialog), GTK_WIN_POS_CENTER);

    GtkWidget* vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 10);
    gtk_container_set_border_width(GTK_CONTAINER(vbox), 10);
    gtk_container_add(GTK_CONTAINER(dialog), vbox);

    GtkWidget* label_desc = gtk_label_new(
        g_locale_to_utf8("Разработчик: гр 21BBB1 Репин И.В.\nРуководитель:
Бикташев Р.А.", -1, NULL, NULL, NULL));
    gtk_label_set_justify(GTK_LABEL(label_desc), GTK_JUSTIFY_LEFT);
    gtk_box_pack_start(GTK_BOX(vbox), label_desc, FALSE, FALSE, 0);

    GtkWidget* label_copy = gtk_label_new(g_locale_to_utf8("(с) Пенза
2025. Пензенский технологический институт", -1, NULL, NULL, NULL));
    gtk_label_set_justify(GTK_LABEL(label_copy), GTK_JUSTIFY_CENTER);
    gtk_box_pack_start(GTK_BOX(vbox), label_copy, FALSE, FALSE, 0);

    GtkWidget* close_button =
gtk_button_new_with_label(g_locale_to_utf8("Закрыть", -1, NULL, NULL, NULL));
    gtk_box_pack_end(GTK_BOX(vbox), close_button, FALSE, FALSE, 0);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		80

```

        g_signal_connect_swapped(close_button, "clicked",
G_CALLBACK(gtk_widget_destroy), dialog);

        gtk_widget_show_all(dialog);
    }

    gboolean on_main_window_delete_event(GtkWidget* widget, GdkEvent* event,
gpointer user_data) {
        if (!project_modified) {
            return FALSE;
        }

        GtkWidget* dialog = gtk_message_dialog_new(GTK_WINDOW(widget),
            GTK_DIALOG_MODAL,
            GTK_MESSAGE_QUESTION,
            GTK_BUTTONS_NONE,
            g_locale_to_utf8("Сохранить изменения перед выходом?", -1, NULL,
NULL, NULL));
        gtk_dialog_add_buttons(GTK_DIALOG(dialog),
            g_locale_to_utf8("_Сохранить", -1, NULL, NULL, NULL),
GTK_RESPONSE_YES,
            g_locale_to_utf8("_Не сохранять", -1, NULL, NULL, NULL),
GTK_RESPONSE_NO,
            g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
GTK_RESPONSE_CANCEL,
            NULL);

        gint response = gtk_dialog_run(GTK_DIALOG(dialog));
        gtk_widget_destroy(dialog);

        switch (response) {
        case GTK_RESPONSE_YES: {
            GtkWidget* file_dialog = gtk_file_chooser_dialog_new(
                g_locale_to_utf8("Сохранить проект", -1, NULL, NULL, NULL),
                GTK_WINDOW(widget),
                GTK_FILE_CHOOSER_ACTION_SAVE,
                g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
GTK_RESPONSE_CANCEL,
                g_locale_to_utf8("_Сохранить", -1, NULL, NULL, NULL),
GTK_RESPONSE_ACCEPT,
                NULL);

            gtk_file_chooser_set_do_overwrite_confirmation(GTK_FILE_CHOOSER(file_dialog),
TRUE);

            gint save_response = gtk_dialog_run(GTK_DIALOG(file_dialog));
            if (save_response == GTK_RESPONSE_ACCEPT) {
                char* filename =
gtk_file_chooser_get_filename(GTK_FILE_CHOOSER(file_dialog));
                Circles* circles = (Circles*)user_data;
                bool save_ok = save_circles_to_file(circles, filename);
                g_free(filename);
                gtk_widget_destroy(file_dialog);

                if (save_ok) {
                    project_modified = false;
                    return FALSE;
                }
            }
            else {

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		81

```

GtkWidget* error_dialog = gtk_message_dialog_new(GTK_WINDOW(widget),
                                                GTK_DIALOG_MODAL,
                                                GTK_MESSAGE_ERROR,
                                                GTK_BUTTONS_OK,
                                                g_locale_to_utf8("Ошибка сохранения проекта!", -1,
NULL, NULL, NULL));

        gtk_dialog_run(GTK_DIALOG(error_dialog));
        gtk_widget_destroy(error_dialog);
        return TRUE;
    }
}
else {
    gtk_widget_destroy(file_dialog);
    return TRUE;
}
}
case GTK_RESPONSE_NO: {
    return FALSE;
}
case GTK_RESPONSE_CANCEL:
default: {
    return TRUE;
}
}
}

GtkWidget* my_button_new(const char* ttips, const gchar* icon_name)
{
    gchar* msg;
    msg = g_locale_to_utf8(ttips, -1, NULL, NULL, NULL);
    GtkWidget* button = gtk_button_new();
    GtkWidget* child = gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 5);
    GtkWidget* image = gtk_image_new_from_file(icon_name);
    gtk_box_pack_start(GTK_BOX(child), image, FALSE, FALSE, 0);
    gtk_widget_set_tooltip_text(button, msg);
    gtk_container_add(GTK_CONTAINER(button), child);
    g_free(msg);
    return button;
}

void show_diag_char(GtkWidget* button, gpointer user_data) {
    Circles* circles = (Circles*)user_data;
    DiagWindowData* data = g_new0(DiagWindowData, 1);
    data->selected_char = 0;
    data->selected_smo = 1;

    data->window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(data->window),
g_locale_to_utf8("Столбчатая диаграмма характеристик", -1, NULL, NULL, NULL));
    gtk_window_set_default_size(GTK_WINDOW(data->window), 1200, 600);

    GtkWidget* vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 6);
    gtk_container_add(GTK_CONTAINER(data->window), vbox);

    data->char_combo = gtk_combo_box_text_new();
    for (int i = 0; i < CHAR_COUNT; i++) {

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		82

```

        gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(data->char_combo),
characteristic_names[i]);
    }
    gtk_combo_box_set_active(GTK_COMBO_BOX(data->char_combo), 0);
    gtk_box_pack_start(GTK_BOX(vbox), data->char_combo, FALSE, FALSE, 0);

    data->smo_combo = gtk_combo_box_text_new();
    for (int i = 0; i < results_acc.circle_count; i++) {
        gtk_combo_box_text_append_text(GTK_COMBO_BOX_TEXT(data-
>smo_combo), circles->circle_params[i].circle_name);
    }
    gtk_combo_box_set_active(GTK_COMBO_BOX(data->smo_combo), 0);
    gtk_box_pack_start(GTK_BOX(vbox), data->smo_combo, FALSE, FALSE, 0);

    data->drawing_area = gtk_drawing_area_new();
    gtk_widget_set_vexpand(data->drawing_area, TRUE);
    gtk_box_pack_start(GTK_BOX(vbox), data->drawing_area, TRUE, TRUE, 0);

    g_signal_connect(data->char_combo, "changed",
G_CALLBACK(on_combo_changed), data);
    g_signal_connect(data->smo_combo, "changed",
G_CALLBACK(on_combo_changed), data);
    g_signal_connect(data->drawing_area, "draw",
G_CALLBACK(draw_bar_chart), data);

    gtk_widget_show_all(data->window);
}

void open_edit_weights_window(GtkWidget* button, gpointer user_data) {
    Circles* circles = (Circles*)user_data;

    if (matrix_window != NULL) {
        gtk_window_present(GTK_WINDOW(matrix_window));
        return;
    }

    matrix_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(matrix_window),
g_locale_to_utf8("Редактирование матрицы весов", -1, NULL, NULL));
    gtk_window_set_default_size(GTK_WINDOW(matrix_window), 500, 500);
    gtk_container_set_border_width(GTK_CONTAINER(matrix_window), 10);

    GtkWidget* vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 6);
    gtk_container_add(GTK_CONTAINER(matrix_window), vbox);

    GtkWidget* grid = gtk_grid_new();
    gtk_grid_set_row_spacing(GTK_GRID(grid), 2);
    gtk_grid_set_column_spacing(GTK_GRID(grid), 2);
    gtk_box_pack_start(GTK_BOX(vbox), grid, TRUE, TRUE, 0);

    for (int i = 0; i < circles->count; i++) {
        char label_text[16];
        snprintf(label_text, sizeof(label_text), "S%d", i);
        GtkWidget* label_col = gtk_label_new(label_text);
        gtk_grid_attach(GTK_GRID(grid), label_col, i + 1, 0, 1, 1);
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		83

```

    GtkWidget* label_row = gtk_label_new(label_text);
    gtk_grid_attach(GTK_GRID(grid), label_row, 0, i + 1, 1, 1);
}

for (int i = 0; i < circles->count; i++) {
    for (int j = 0; j < circles->count; j++) {
        GtkAdjustment* adjustment = gtk_adjustment_new(0.0, 0.0,
1000.0, 0.1, 1.0, 0.0);
        GtkWidget* spin = gtk_spin_button_new(adjustment, 0.1, 2);

        double weight = 0.0;
        for (int k = 0; k < circles->line_count; k++) {
            if (circles->lines[k].from_index == i && circles-
>lines[k].to_index == j) {
                weight = circles->lines[k].weight;
                break;
            }
        }
        gtk_spin_button_set_value(GTK_SPIN_BUTTON(spin), weight);

        gtk_grid_attach(GTK_GRID(grid), spin, j + 1, i + 1, 1, 1);
    }
}

GtkWidget* apply_btn =
gtk_button_new_with_label(g_locale_to_utf8("Применить", -1, NULL, NULL, NULL));
gtk_box_pack_start(GTK_BOX(vbox), apply_btn, FALSE, FALSE, 0);
g_signal_connect(apply_btn, "clicked",
G_CALLBACK(apply_matrix_weights_editing), circles);

g_signal_connect(matrix_window, "destroy",
G_CALLBACK(gtk_widget_destroyed), &matrix_window);

gtk_widget_show_all(matrix_window);
}

void apply_matrix_weights_editing(GtkWidget* button, gpointer user_data) {
    Circles* circles = (Circles*)user_data;

    GtkWidget* vbox = gtk_bin_get_child(GTK_BIN(matrix_window));
    if (!GTK_IS_BOX(vbox)) {
        g_warning("Expected vbox container");
        return;
    }

    GtkWidget* grid = NULL;
    GList* children = gtk_container_get_children(GTK_CONTAINER(vbox));
    for (GList* iter = children; iter != NULL; iter = iter->next) {
        if (GTK_IS_GRID(iter->data)) {
            grid = GTK_WIDGET(iter->data);
            break;
        }
    }
}

```

```

g_list_free(children);

if (!grid) {
    g_warning("Grid not found in vbox");
    return;
}

circles->line_count = 0;

for (int i = 0; i < circles->count; i++) {
    for (int j = 0; j < circles->count; j++) {

        GtkWidget* child = gtk_grid_get_child_at(GTK_GRID(grid), j +
1, i + 1);

        if (!child) continue;

        double weight =
gtk_spin_button_get_value(GTK_SPIN_BUTTON(child));

        if (weight > 0.0) {
            if (circles->line_count < MAX_LINES) {
                circles->lines[circles->line_count].from_index = i;
                circles->lines[circles->line_count].to_index = j;
                circles->lines[circles->line_count].weight = weight;
                circles->line_count++;
            }
        }
    }
}

gtk_widget_queue_draw(GTK_WIDGET(gtk_widget_get_toplevel(GTK_WIDGET(button))));

gtk_widget_destroy(matrix_window);
matrix_window = NULL;

}

static gboolean on_button_press(GtkWidget* widget, GdkEventButton* event,
gpointer user_data)
{
    Circles* circles = (Circles*)user_data;
    const double radius = 30.0;

    if (event->button == GDK_BUTTON_PRIMARY) {
        if (circles->add_line_mode) {
            int clicked_circle = -1;
            for (int i = 0; i < circles->count; i++) {
                double dx = event->x - circles->x[i];
                double dy = event->y - circles->y[i];
                if (sqrt(dx * dx + dy * dy) <= radius) {
                    clicked_circle = i;
                    break;
                }
            }
            if (clicked_circle == -1)
                return FALSE;
        }
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		85

```

        if (circles->line_first_circle == -1) {
            circles->line_first_circle = clicked_circle;
        }
        else {
            float weight =
prompt_user_for_weight(GTK_WINDOW(gtk_widget_get_toplevel(widget)));
            if (circles->line_count < MAX_LINES) {
                circles->lines[circles->line_count].from_index =
circles->line_first_circle;
                circles->lines[circles->line_count].to_index =
clicked_circle;
                circles->lines[circles->line_count].weight = weight;
                circles->line_count++;
            }
            circles->line_first_circle = -1;
            circles->add_line_mode = FALSE;
        }
        gtk_widget_queue_draw(widget);
        project_modified = true;
        return TRUE;
    }
    else if (circles->add_mode) {
        if (circles->count < MAX_CIRCLES) {
            circles->x[circles->count] = event->x;
            circles->y[circles->count] = event->y;
            if (circles->count > 0) {
                circles->circle_params[circles->count].circle_name =
"CMO";
                circles->circle_params[circles->count].service_time =
0.0;
                circles->circle_params[circles->count].channel_count =
1;
                circles->circle_params[circles->count].request_type =
0;
                circles->circle_params[circles->count].variation_coeff
= 0.0;
                circles->circle_params[circles-
>count].priority_class_count = 1;
                circles->circle_params[circles->count].priority_table
= g_new0(PriorityRow, 1);
                circles->circle_params[circles->count].priority_table-
>priority_class = 1;
                circles->circle_params[circles->count].priority_table-
>input_share = 1;
                circles->circle_params[circles->count].priority_table-
>service_time = 0.0;
                circles->circle_params[circles->count].threshold_ap =
0;
                circles->circle_params[circles->count].threshold_op =
0;
                circles->circle_params[circles->count].limited_queue =
FALSE;
                circles->circle_params[circles->count].queue_size = 1;
            }

            circles->count++;
            circles->add_mode = FALSE;
            gtk_widget_queue_draw(widget);
        }
    }

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		86

```

project_modified = true;
    return TRUE;
}
else if (circles->del_mode) {
    for (int i = 0; i < circles->count; i++) {

double dx = event->x - circles->x[i];
        double dy = event->y - circles->y[i];
        if (sqrt(dx * dx + dy * dy) <= radius) {
            for (int j = i; j < circles->count - 1; j++) {
                circles->x[j] = circles->x[j + 1];
                circles->y[j] = circles->y[j + 1];
            }
            circles->count--;
            circles->del_mode = FALSE;
            gtk_widget_queue_draw(widget);
            break;
        }
    }
    project_modified = true;
    return TRUE;
}
else if (circles->move_mode) {
    for (int i = 0; i < circles->count; i++) {
        double dx = event->x - circles->x[i];
        double dy = event->y - circles->y[i];
        if (sqrt(dx * dx + dy * dy) <= radius) {
            circles->moving_index = i;
            project_modified = true;
            return TRUE;
        }
    }
}
else if (circles->delLineMode && event->button == 1) {
    double click_x = event->x;
    double click_y = event->y;

    for (int i = 0; i < circles->line_count; i++) {
        int from = circles->lines[i].from_index;
        int to = circles->lines[i].to_index;
        double x1 = circles->x[from];
        double y1 = circles->y[from];
        double x2 = circles->x[to];
        double y2 = circles->y[to];

        double dist = point_to_segment_distance(click_x, click_y,
x1, y1, x2, y2);

        if (dist < 5.0) {
            for (int j = i; j < circles->line_count - 1; j++) {
                circles->lines[j] = circles->lines[j + 1];
            }
            circles->line_count--;

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		87

```

        circles->delLineMode = false;
        gtk_widget_queue_draw(widget);
        project_modified = true;
        return TRUE;
    }
}
else {
    const double click_tolerance = 5.0;

    for (int i = 0; i < circles->count; i++) {
        double cx = circles->x[i];
        double cy = circles->y[i];
        double dx = event->x - cx;
        double dy = event->y - cy;
        if (dx * dx + dy * dy <= 400) {
            if (i == 0) {
                float val =
prompt_user_for_intensity(GTK_WINDOW(gtk_widget_get_toplevel(widget)));
                if (val >= 0.0f) {
                    input_intensity = val;
                    g_print("Интенсивность входящего потока
установлена: %f\n", input_intensity);
                }
                project_modified = true;
                return TRUE;
            }
            else {
                GtkWindow* parent_window =
GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(widget)));
                show_circle_params_dialog(parent_window, circles,
i);

                return TRUE;
            }
        }
    }
    for (int i = 0; i < circles->line_count; i++) {
        int from = circles->lines[i].from_index;
        int to = circles->lines[i].to_index;
        double x1 = circles->x[from];
        double y1 = circles->y[from];
        double x2 = circles->x[to];
        double y2 = circles->y[to];

        double dist = point_to_segment_distance(event->x, event-
>y, x1, y1, x2, y2);
        if (dist <= click_tolerance) {

            float new_weight =
prompt_user_for_weight(GTK_WINDOW(gtk_widget_get_toplevel(widget)));
            if (new_weight != -1) {
                circles->lines[i].weight = new_weight;
                gtk_widget_queue_draw(widget);
            }
        }
    }
}

```

```

        project_modified = true;
        return TRUE;
    }
}

return FALSE;
}

static gboolean on_motion_notify(GtkWidget* widget, GdkEventMotion* event,
gpointer user_data)
{
    Circles* circles = (Circles*)user_data;

    if (circles->move_mode && circles->moving_index != -1) {
        circles->x[circles->moving_index] = event->x;
        circles->y[circles->moving_index] = event->y;
        gtk_widget_queue_draw(widget);
        return TRUE;
    }
    return FALSE;
}

static gboolean on_button_release(GtkWidget* widget, GdkEventButton*
event, gpointer user_data)
{
    Circles* circles = (Circles*)user_data;

    if (event->button == GDK_BUTTON_PRIMARY && circles->move_mode &&
circles->moving_index != -1) {
        circles->moving_index = -1;
        circles->move_mode = FALSE;
        return TRUE;
    }
    return FALSE;
}

```

Файл vizualise.h

```

#ifndef VISUALIZE_H
#define VISUALIZE_H

static gboolean on_draw_event(GtkWidget* widget, cairo_t* cr, gpointer
user_data);
static gboolean on_draw_chart(GtkWidget* widget, cairo_t* cr, gpointer
user_data);
static void draw_bar_chart(GtkWidget* widget, cairo_t* cr, gpointer
user_data);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		89

```
void show_diag_net_dialog(GtkWidget* parent);
```

```
#endif
```

Файл vizualise.cpp

```
#include "compute.cpp"
```

```
#include "model.h"
```

```
#include "visualize.h"
```

```
static gboolean on_draw_event(GtkWidget* widget, cairo_t* cr,
gpointer user_data)
{
    Circles* circles = (Circles*)user_data;

    GtkAllocation allocation;
    gtk_widget_get_allocation(widget, &allocation);
    cairo_set_source_rgb(cr, 1, 1, 1);
    cairo_paint(cr);

    const double radius = 20.0;

    cairo_select_font_face(cr, "Sans", CAIRO_FONT_SLANT_NORMAL,
CAIRO_FONT_WEIGHT_BOLD);
    cairo_set_font_size(cr, 20);

    for (int i = 0; i < circles->line_count; i++) {
        if (circles->lines[i].weight == 0.0f) {
            continue;
        }

        int from = circles->lines[i].from_index;
        int to = circles->lines[i].to_index;
        float weight = circles->lines[i].weight;

        double x1 = circles->x[from];
        double y1 = circles->y[from];
        double x2 = circles->x[to];
        double y2 = circles->y[to];

        double dx = x2 - x1;
        double dy = y2 - y1;
        double dist = sqrt(dx * dx + dy * dy);
        if (dist == 0) dist = 1;

        double nx = dx / dist;
        double ny = dy / dist;
```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		90

```

double start_x = x1 + nx * radius;
double start_y = y1 + ny * radius;
double end_x = x2 - nx * radius;
double end_y = y2 - ny * radius;

double gray = 1.0 - weight;
cairo_set_source_rgb(cr, gray, gray, gray);
cairo_set_line_width(cr, 2.0);

if (from == to) {
    double loop_radius = 20;

    cairo_save(cr);
    cairo_set_line_width(cr, 2);
    cairo_arc(cr, x1 - radius, y1 - radius, loop_radius, 0,
2 * M_PI);

    cairo_stroke(cr);
    cairo_restore(cr);
}
else {
    cairo_move_to(cr, start_x, start_y);
    cairo_line_to(cr, end_x, end_y);
    cairo_stroke(cr);
}

cairo_set_source_rgb(cr, 1, 0, 0);
cairo_arc(cr, end_x, end_y, 6, 0, 2 * M_PI);
cairo_fill(cr);
}

for (int i = 0; i < circles->count; i++) {
    double cx = circles->x[i];
    double cy = circles->y[i];

    cairo_new_path(cr);

    cairo_set_source_rgb(cr, 1, 1, 1);
    cairo_arc(cr, circles->x[i], circles->y[i], radius, 0, 2 *
M_PI);

    cairo_fill(cr);

    cairo_set_source_rgb(cr, 0, 0, 0);
    cairo_set_line_width(cr, 2.0);
    cairo_arc(cr, cx, cy, radius, 0, 2 * G_PI);
    cairo_stroke(cr);

    char text[16];
    snprintf(text, sizeof(text), "S%d", i);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		91

```

    cairo_text_extents_t extents;
    cairo_text_extents(cr, text, &extents);

    double tx = cx - extents.width / 2 - extents.x_bearing;
    double ty = cy - extents.height / 2 - extents.y_bearing;

    cairo_move_to(cr, tx, ty);
    cairo_show_text(cr, text);
}

return FALSE;
}

static gboolean on_draw_chart(GtkWidget* widget, cairo_t* cr,
gpointer user_data) {
    if (results_acc.cycle_count == 0) {
        cairo_move_to(cr, 10, 20);
        cairo_show_text(cr, g_locale_to_utf8("Нет данных для
отображения", -1, NULL, NULL, NULL));
        return FALSE;
    }

    int width = gtk_widget_get_allocated_width(widget);
    int height = gtk_widget_get_allocated_height(widget);

    int margin_left = 50;
    int margin_bottom = 50;
    int margin_top = 30;
    int margin_right = 20;

    int chart_width = width - margin_left - margin_right;
    int chart_height = height - margin_top - margin_bottom;

    int cycles = results_acc.cycle_count;
    int bar_width = chart_width / (cycles * 5);

    double max_val = 0.0;

    double avg_queue_length[cycles] = { 0.0 };
    double avg_network_length[cycles] = { 0.0 };
    double avg_queue_wait[cycles] = { 0.0 };
    double avg_network_wait[cycles] = { 0.0 };
    for (int c = 0; c < cycles; c++) {
        double total_avg_queue_length = 0.0;
        double total_avg_network_length = 0.0;
        double total_avg_queue_wait = 0.0;
        double total_avg_network_wait = 0.0;

```

```

        for (int i = 0; i < results_acc.circle_count; i++) {
            CircleCalcResult* res = &results_acc.results[i][c];
            total_avg_queue_length += res->avg_in_queue;
            total_avg_network_length += res->avg_in_system;
            total_avg_queue_wait += res->avg_wait_queue * res-
>throughput;
            total_avg_network_wait += res->avg_time_system * res-
>throughput;
        }
        avg_queue_length[c] = total_avg_queue_length;
        avg_network_length[c] = total_avg_network_length;
        avg_queue_wait[c] = total_avg_queue_wait;
        avg_network_wait[c] = total_avg_network_wait;
    }

    for (int c = 0; c < cycles; c++) {
        if (diag_settings.show_avg_waiting && avg_queue_length[c] >
max_val)
            max_val = avg_queue_length[c];
        if (diag_settings.show_avg_in_system &&
avg_network_length[c] > max_val)
            max_val = avg_network_length[c];
        if (diag_settings.show_avg_wait_time && avg_queue_wait[c] >
max_val)
            max_val = avg_queue_wait[c];
        if (diag_settings.show_avg_soj_time && avg_network_wait[c] >
max_val)
            max_val = avg_network_wait[c];
    }
    if (max_val == 0) max_val = 1;

    GdkRGBA colors[4];
    gdk_rgba_parse(&colors[0], "red");
    gdk_rgba_parse(&colors[1], "blue");
    gdk_rgba_parse(&colors[2], "green");
    gdk_rgba_parse(&colors[3], "orange");

    const char* labels[4] = {
        g_locale_to_utf8("Среднее число заявок, ожидающих
обслуживания", -1, NULL, NULL, NULL),
        g_locale_to_utf8("Среднее число заявок, пребывающих в сети",
-1, NULL, NULL, NULL),
        g_locale_to_utf8("Среднее время ожидания заявки в сети", -1,
NULL, NULL, NULL),
        g_locale_to_utf8("Среднее время пребывания заявок в сети", -
1, NULL, NULL, NULL)
    };

    double* data[4] = {
        avg_queue_length,
        avg_network_length,
        avg_queue_wait,
        avg_network_wait
    };

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		93

```

gboolean show_flags[4] = {
    diag_settings.show_avg_waiting,
    diag_settings.show_avg_in_system,
    diag_settings.show_avg_wait_time,
    diag_settings.show_avg_soj_time
};

cairo_set_source_rgb(cr, 0, 0, 0);
cairo_set_line_width(cr, 1);
cairo_move_to(cr, margin_left, margin_top);
cairo_line_to(cr, margin_left, height - margin_bottom);
cairo_line_to(cr, width - margin_right, height - margin_bottom);
cairo_stroke(cr);

for (int i = 0; i <= 5; i++) {
    double y = margin_top + i * (chart_height / 5.0);
    double val = max_val * (1 - i / 5.0);
    char buf[32];
    snprintf(buf, sizeof(buf), "%.2f", val);
    cairo_move_to(cr, 5, y + 5);
    cairo_show_text(cr, buf);

    cairo_set_source_rgb(cr, 0.8, 0.8, 0.8);
    cairo_move_to(cr, margin_left, y);
    cairo_line_to(cr, width - margin_right, y);
    cairo_stroke(cr);

    cairo_set_source_rgb(cr, 0, 0, 0);
}

int group_width = 0;
for (int i = 0; i < 4; i++)
    if (show_flags[i]) group_width++;

if (group_width == 0) return FALSE;

int bar_spacing = 2;
int total_bar_width = bar_width * group_width + bar_spacing *
(group_width - 1);

for (int c = 0; c < cycles; c++) {
    int x0 = margin_left + c * (total_bar_width + bar_width);

    int bar_index = 0;
    for (int i = 0; i < 4; i++) {
        if (!show_flags[i]) continue;
        double val = data[i][c];
        double bar_height = (val / max_val) * chart_height;
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		94

```

gdk_cairo_set_source_rgba(cr, &colors[i]);
    cairo_rectangle(cr,
        x0 + bar_index * (bar_width + bar_spacing),
        height - margin_bottom - bar_height,
        bar_width,
        bar_height);
    cairo_fill(cr);

    bar_index++;
}

}

cairo_set_source_rgb(cr, 0, 0, 0);
for (int c = 0; c < cycles; c++) {
    int x0 = margin_left + c * (total_bar_width + bar_width) +
total_bar_width / 2;
    char buf[16];
    snprintf(buf, sizeof(buf), "Cycle %d", c + 1);
    cairo_move_to(cr, x0 - 15, height - margin_bottom + 15);
    cairo_show_text(cr, buf);
}

int legend_x = width - margin_right - 250;
int legend_y = margin_top + 10;
for (int i = 0; i < 4; i++) {
    gdk_cairo_set_source_rgba(cr, &colors[i]);
    cairo_rectangle(cr, legend_x, legend_y + i * 20, 15, 15);
    cairo_fill(cr);

    cairo_set_source_rgb(cr, 0, 0, 0);
    cairo_move_to(cr, legend_x + 20, legend_y + i * 20 + 12);
    cairo_show_text(cr, labels[i]);
}

return FALSE;
}

static void draw_bar_chart(GtkWidget* widget, cairo_t* cr, gpointer
user_data) {
    DiagWindowData* data = (DiagWindowData*)user_data;

    int width = gtk_widget_get_allocated_width(widget);
    int height = gtk_widget_get_allocated_height(widget);

    cairo_set_source_rgb(cr, 1, 1, 1);
    cairo_paint(cr);

    if (data->selected_smo < 0 || data->selected_smo >=
results_acc.circle_count) return;
    if (cycles_count == 0) return;

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		95

```

double values[cycles_count];
for (int i = 0; i < cycles_count; i++) {
    switch (data->selected_char) {
        case CHAR_TRANS_PROB:
            values[i] = results_acc.results[data-
>selected_smo][i].throughput;
            break;
        case CHAR_INTENSITY:
            values[i] = results_acc.results[data-
>selected_smo][i].input_intensity;
            break;
        case CHAR_LOAD_COEFF:
            values[i] = results_acc.results[data-
>selected_smo][i].load_factor;
            break;
        case CHAR_BUSY_CHANNELS:
            values[i] = results_acc.results[data-
>selected_smo][i].busy_channels;
            break;
        case CHAR_AVG_REQUESTS:
            values[i] = results_acc.results[data-
>selected_smo][i].avg_in_system;
            break;
        case CHAR_AVG_QUEUE_LEN:
            values[i] = results_acc.results[data-
>selected_smo][i].avg_in_queue;
            break;
        case CHAR_AVG_WAIT_TIME:
            values[i] = results_acc.results[data-
>selected_smo][i].avg_wait_queue;
            break;
        case CHAR_AVG_STAY_TIME:
            values[i] = results_acc.results[data-
>selected_smo][i].avg_time_system;
            break;
        default:
            values[i] = 0;
    }
}

double max_val = 0;
for (int i = 0; i < cycles_count; i++) {
    if (values[i] > max_val) max_val = values[i];
}
if (max_val == 0) max_val = 1;

int margin = 50;
int chart_width = width - 2 * margin;
int chart_height = height - 2 * margin;
int bar_width = chart_width / cycles_count;

cairo_set_source_rgb(cr, 0.2, 0.6, 0.8);

```

```

    for (int i = 0; i < cycles_count; i++) {
        double val = values[i];
        int bar_height = (int)(val / max_val * chart_height);
        int x = margin + i * bar_width;
        int y = height - margin - bar_height;

        cairo_rectangle(cr, x, y, bar_width - 4, bar_height);
        cairo_fill(cr);

        cairo_set_source_rgb(cr, 0, 0, 0);
        cairo_select_font_face(cr, "Sans", CAIRO_FONT_SLANT_NORMAL,
CAIRO_FONT_WEIGHT_NORMAL);
        cairo_set_font_size(cr, 10);
        char label[16];
        snprintf(label, sizeof(label), "%d", i + 1);
        cairo_move_to(cr, x + (bar_width - 4) / 2 - 6, height -
margin + 15);
        cairo_show_text(cr, label);

        snprintf(label, sizeof(label), "%.2f", val);
        cairo_move_to(cr, x + (bar_width - 4) / 2 - 15, y - 5);
        cairo_show_text(cr, label);

        cairo_set_source_rgb(cr, 0.2, 0.6, 0.8);
    }

    cairo_set_source_rgb(cr, 0, 0, 0);
    cairo_move_to(cr, margin, margin - 20);
    cairo_show_text(cr, characteristic_names[data->selected_char]);
    cairo_move_to(cr, width / 2 - 20, height - 5);
    cairo_show_text(cr, "Cycles");
}

void show_diag_net_dialog(GtkWidget* parent) {
    GtkWidget* dialog =
gtk_dialog_new_with_buttons(g_locale_to_utf8("Диаграмма характеристик
сети", -1, NULL, NULL, NULL),
        GTK_WINDOW(parent),
        GTK_DIALOG_MODAL,
        g_locale_to_utf8("_Заккрыть", -1, NULL, NULL, NULL),
GTK_RESPONSE_CLOSE,
        NULL);

    GtkWidget* content_area =
gtk_dialog_get_content_area(GTK_DIALOG(dialog));

    GtkWidget* vbox = gtk_box_new(GTK_ORIENTATION_VERTICAL, 5);
    gtk_container_add(GTK_CONTAINER(content_area), vbox);

    const char* labels[4] = {
        g_locale_to_utf8("Среднее число заявок, ожидающих
обслуживания", -1, NULL, NULL, NULL),

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		97

```

        g_locale_to_utf8("Среднее число заявок, пребывающих в сети", -1,
NULL, NULL, NULL),
        g_locale_to_utf8("Среднее время ожидания заявки в сети", -1,
NULL, NULL, NULL),
        g_locale_to_utf8("Среднее время пребывания заявок в сети", -
1, NULL, NULL, NULL)
    };

    for (int i = 0; i < 4; i++) {
        GtkWidget* chk = gtk_check_button_new_with_label(labels[i]);
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(chk), TRUE);
        g_object_set_data(G_OBJECT(chk), "id", GINT_TO_POINTER(i));
        g_object_set_data(G_OBJECT(chk), "drawing_area", NULL);

        g_signal_connect(chk, "toggled",
G_CALLBACK(on_checkbox_toggled), &diag_settings);
        gtk_box_pack_start(GTK_BOX(vbox), chk, FALSE, FALSE, 0);
    }

    GtkWidget* drawing_area = gtk_drawing_area_new();
    gtk_widget_set_size_request(drawing_area, 1200, 600);
    gtk_box_pack_start(GTK_BOX(vbox), drawing_area, TRUE, TRUE, 0);

    GList* children =
gtk_container_get_children(GTK_CONTAINER(vbox));
    for (GList* l = children; l != NULL; l = l->next) {
        GtkWidget* w = GTK_WIDGET(l->data);
        if (GTK_IS_TOGGLE_BUTTON(w)) {
            g_object_set_data(G_OBJECT(w), "drawing_area",
drawing_area);
        }
    }
    g_list_free(children);

    g_signal_connect(drawing_area, "draw",
G_CALLBACK(on_draw_chart), NULL);

    gtk_widget_show_all(dialog);
    gtk_widget_queue_draw(drawing_area);

    g_signal_connect_swapped(dialog, "response",
G_CALLBACK(gtk_widget_destroy), dialog);
}

```

Файл compute.h

```

#ifndef COMPUTE_H
#define COMPUTE_H

CircleCalcResult calculate_mm_m(double lambda, double mu, int m,
double lambda0, GtkWidget* button, CircleParams* params);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		98

```

        void calculate_network(Circles* circles, double current_intensity,
int cycle_index, GtkWidget* button);
        void vary_and_calculate(Circles* circles, GtkWidget* button);
        static double factorial(int n);
        static double point_to_segment_distance(double px, double py, double
x1, double y1, double x2, double y2);

#endif

```

Файл compute.cpp

```

#include "input.cpp"
#include "model.h"
#include "compute.h"

CircleCalcResult calculate_mm_m(double lambda, double mu, int m,
double lambda0, GtkWidget* button, CircleParams* params) {
    CircleCalcResult res = { 0 };

    if (m <= 0 || mu <= 0 || lambda <= 0) {
        return res;
    }

    double rho = lambda / (m * mu);
    if (rho >= 1.0) {
        char message[256];
        snprintf(message, sizeof(message), "Error: System unstable
(rho >= 1)");
        show_warning_dialog(button, message);
        return res;
    }
    if (params->limited_queue) {
        int K = m + params->queue_size;
        if (K < m) K = m;

        double sum1 = 0.0;
        for (int n = 0; n < m; n++) {
            sum1 += pow(m * rho, n) / factorial(n);
        }

        double sum2 = 0.0;
        if (rho != 1.0) {
            sum2 = (pow(m * rho, m) / factorial(m)) * (1 - pow(rho,
K - m + 1)) / (1 - rho);
        }
        else {
            sum2 = (pow(m * rho, m) / factorial(m)) * (K - m + 1);
        }
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		99

```

double P0 = 1.0 / (sum1 + sum2);

double PK = (pow(m * rho, K) / (factorial(m) * pow(m, K -
m))) * P0;

double Ls = 0.0;
for (int n = 0; n <= K; n++) {
    double Pn = 0.0;
    if (n < m) {
        Pn = (pow(m * rho, n) / factorial(n)) * P0;
    }
    else {
        Pn = (pow(m, m) * pow(rho, n) / factorial(m)) * P0;
    }
    Ls += n * Pn;
}

double sumP0 = 0.0;
for (int n = 0; n < m; n++) {
    sumP0 += pow(m * rho, n) / factorial(n);
}
double busy_channels = m * (1 - P0 * sumP0);

double Lq = Ls - busy_channels;

double effective_lambda = lambda * (1 - PK);
double W = Ls / effective_lambda;
double Wq = W - 1.0 / mu;

res.throughput = lambda / lambda0;
res.input_intensity = lambda;
res.load_factor = rho;
res.busy_channels = busy_channels;
res.avg_in_system = Ls;
res.avg_in_queue = Lq;
res.avg_wait_queue = Wq;
res.avg_time_system = W;
}
else {
    double sum = 0.0;
    for (int k = 0; k < m; k++) {
        sum += pow(m * rho, k) / factorial(k);
    }
    double pm = pow(m * rho, m) / (factorial(m) * (1 - rho));
    double P0 = 1.0 / (sum + pm);

    double Lq = P0 * pow(m * rho, m) * rho / (factorial(m) *
pow(1 - rho, 2));

    double Wq = Lq / lambda;
    double W = Wq + 1.0 / mu;

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		100

```

double busy_channels = m * rho;

double L = busy_channels + Lq;

res.throughput = lambda / lambda0;
res.input_intensity = lambda;
res.load_factor = rho;
res.busy_channels = busy_channels;
res.avg_in_system = L;
res.avg_in_queue = Lq;
res.avg_wait_queue = Wq;
res.avg_time_system = W;
}

return res;
}

void calculate_network(Circles* circles, double current_intensity,
int cycle_index, GtkWidget* button) {
    if (!circles) return;
    if (circles->count < 1) return;

    double lambda[MAX_CIRCLES] = { 0 };
    const int max_iter = 100;
    const double epsilon = 1e-6;

    for (int i = 0; i < circles->count; i++) {
        double sum_weights = 0.0;
        for (int j = 0; j < circles->line_count; j++) {
            if (circles->lines[j].from_index == i) {
                sum_weights += circles->lines[j].weight;
            }
        }
        if (fabs(sum_weights - 1.0) > 1e-6) {
            char message[256];
            snprintf(message, sizeof(message),
                g_locale_to_utf8("Ошибка: сумма весов исходящих
связей для узла %d равна %.6f, должна быть 1.0", -1, NULL, NULL, NULL),
                i, sum_weights);
            show_warning_dialog(button, message);
            return;
        }
    }

    lambda[0] = current_intensity;

    for (int iter = 0; iter < max_iter; iter++) {
        double new_lambda[MAX_CIRCLES] = { 0 };
        new_lambda[0] = current_intensity;

        for (int i = 0; i < circles->line_count; i++) {
            int from = circles->lines[i].from_index;

```

```

        int to = circles->lines[i].to_index;
        double weight = circles->lines[i].weight;
        new_lambda[to] += lambda[from] * weight;
    }

    double max_diff = 0.0;
    for (int i = 1; i < circles->count; i++) {
        double diff = fabs(new_lambda[i] - lambda[i]);
        if (diff > max_diff) max_diff = diff;
        lambda[i] = new_lambda[i];
    }

    if (max_diff < epsilon) {
        printf("Converged after %d iterations\n", iter + 1);
        break;
    }
}

for (int i = 1; i < circles->count; i++) {
    CircleParams* p = &circles->circle_params[i];
    double mu = (p->service_time > 0) ? 1.0 / p->service_time :
1.0;

    int m = p->channel_count > 0 ? p->channel_count : 1;

    CircleCalcResult res = calculate_mm_m(lambda[i], mu, m,
lambda[0], button, p);

    results_acc.results[i][cycle_index] = res;

    g_print("Queue S%d:\n", i);
    g_print(" Throughput coefficient: %f\n", res.throughput);
    g_print(" Input intensity lambda: %f\n",
res.input_intensity);
    g_print(" Load factor p: %f\n", res.load_factor);
    g_print(" Average busy channels: %f\n", res.busy_channels);
    g_print(" Average number in system (L): %f\n",
res.avg_in_system);
    g_print(" Average queue length (Lq): %f\n",
res.avg_in_queue);
    g_print(" Average queue waiting time (Wq): %f\n",
res.avg_wait_queue);
    g_print(" Average time in system (W): %f\n\n",
res.avg_time_system);
}
if (cycles_count == 1) {
    save_results_to_file("result.txt", circles);
}
}

```

```

void vary_and_calculate(Circles* circles, GtkWidget* button) {
    if (!circles) return;
    if (cycles_count < 2) cycles_count = 2;

    results_acc.cycle_count = cycles_count;
    results_acc.circle_count = circles->count;

    for (int cycle = 0; cycle < cycles_count; cycle++) {
        double t = (cycles_count == 1) ? 0.0 : (double)cycle /
(cycles_count - 1);

        double current_intensity = input_intensity;
        if (vary_intensity) {
            current_intensity = intensity_from + t * (intensity_to -
intensity_from);
        }

        for (int i = 0; i < circles->count; i++) {
            CircleParams* p = &circles->circle_params[i];

            if (vary_service_time) {
                p->service_time = service_time_from[i] + t *
(service_time_to[i] - service_time_from[i]);
                if (p->service_time < 0.0001) p->service_time =
0.0001;
            }

            if (vary_channel_count) {
                float val = channel_count_from[i] + t *
(channel_count_to[i] - channel_count_from[i]);
                int m = (int)(val + 0.5f);
                if (m < 1) m = 1;
                p->channel_count = m;
            }

            if (vary_queue_size) {
                float val = queue_size_from[i] + t *
(queue_size_to[i] - queue_size_from[i]);
                int m = (int)(val + 0.5f);
                if (m < 1) m = 1;
                p->queue_size = m;
            }
        }

        g_print("=== Cycle %d / %d ===\n", cycle + 1, cycles_count);
        g_print("Cycle parameters:\n");
        g_print("  Input intensity: %f\n", current_intensity);
        for (int i = 0; i < circles->count; i++) {
            CircleParams* p = &circles->circle_params[i];

```

```

        g_print(" Queue S%d: service_time=%f, channel_count=%d,
queue_size=%d\n",
                i, p->service_time, p->channel_count, p-
>queue_size);
    }
    calculate_network(circles, current_intensity, cycle,
button);

    g_print("\n");
}

g_print("=== Average values over %d cycles ===\n",
cycles_count);
for (int i = 1; i < circles->count; i++) {
    CircleCalcResult avg = { 0 };
    for (int cycle = 0; cycle < cycles_count; cycle++) {
        avg.throughput +=
results_acc.results[i][cycle].throughput;
        avg.input_intensity +=
results_acc.results[i][cycle].input_intensity;
        avg.load_factor +=
results_acc.results[i][cycle].load_factor;
        avg.busy_channels +=
results_acc.results[i][cycle].busy_channels;
        avg.avg_in_system +=
results_acc.results[i][cycle].avg_in_system;
        avg.avg_in_queue +=
results_acc.results[i][cycle].avg_in_queue;
        avg.avg_wait_queue +=
results_acc.results[i][cycle].avg_wait_queue;
        avg.avg_time_system +=
results_acc.results[i][cycle].avg_time_system;
    }
    double c = (double)cycles_count;
    g_print("Queue S%d (average):\n", i);
    g_print(" Throughput coefficient: %f\n", avg.throughput /
c);
    g_print(" Input intensity lambda: %f\n",
avg.input_intensity / c);
    g_print(" Load factor p: %f\n", avg.load_factor / c);
    g_print(" Average busy channels: %f\n", avg.busy_channels /
c);
    g_print(" Average number in system (L): %f\n",
avg.avg_in_system / c);
    g_print(" Average queue length (Lq): %f\n",
avg.avg_in_queue / c);
    g_print(" Average queue waiting time (Wq): %f\n",
avg.avg_wait_queue / c);
    g_print(" Average time in system (W): %f\n\n",
avg.avg_time_system / c);
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		104

```

    save_results_to_file("result.txt", circles);
}

static double factorial(int n) {
    static double fact_cache[101] = { 1 };
    if (n < 0) return 1;
    if (fact_cache[n] != 0) return fact_cache[n];
    fact_cache[n] = n * factorial(n - 1);
    return fact_cache[n];
}

static double point_to_segment_distance(double px, double py, double
x1, double y1, double x2, double y2)
{
    double dx = x2 - x1;
    double dy = y2 - y1;
    if (dx == 0 && dy == 0) {
        dx = px - x1;
        dy = py - y1;
        return sqrt(dx * dx + dy * dy);
    }

    double t = ((px - x1) * dx + (py - y1) * dy) / (dx * dx + dy *
dy);
    if (t < 0) t = 0;
    else if (t > 1) t = 1;

    double proj_x = x1 + t * dx;
    double proj_y = y1 + t * dy;

    dx = px - proj_x;
    dy = py - proj_y;
    return sqrt(dx * dx + dy * dy);
}

```

Файл input.h

```

#ifndef INPUT_H
#define INPUT_H

float prompt_user_for_weight(GtkWindow* parent);
float prompt_user_for_intensity(GtkWindow* parent);
void show_circle_params_dialog(GtkWindow* parent, Circles* circles,
int circle_index);
void on_priority_class_count_changed(GtkEditable* editable, gpointer
user_data);
static void on_entry_intensity_from_changed(GtkEditable* editable,
gpointer user_data);
static void on_entry_intensity_to_changed(GtkEditable* editable,
gpointer user_data);
static void on_check_vary_service_toggled(GtkToggleButton* toggle,
gpointer user_data);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		105

```

static void on_combo_service_changed(GtkComboBox* combo, gpointer
user_data);
static void on_entry_service_from_changed(GtkEditable* editable,
gpointer user_data);
static void on_entry_service_to_changed(GtkEditable* editable,
gpointer user_data);
static void on_check_vary_channels_toggled(GtkToggleButton* toggle,
gpointer user_data);
static void on_combo_channel_changed(GtkComboBox* combo, gpointer
user_data);
static void on_entry_channel_from_changed(GtkEditable* editable,
gpointer user_data);
static void on_entry_channel_to_changed(GtkEditable* editable,
gpointer user_data);
static void on_check_vary_queue_toggled(GtkToggleButton* toggle,
gpointer user_data);
static void on_combo_queue_changed(GtkComboBox* combo, gpointer
user_data);
static void on_entry_queue_from_changed(GtkEditable* editable,
gpointer user_data);
static void on_entry_queue_to_changed(GtkEditable* editable,
gpointer user_data);
static void on_entry_cycles_count_changed(GtkEditable* editable,
gpointer user_data);
static void on_combo_changed(GtkComboBox* combo, gpointer
user_data);
static void on_radio_mode_toggled(GtkToggleButton* toggle, gpointer
user_data);
static void on_check_vary_intensity_toggled(GtkToggleButton* toggle,
gpointer user_data);
static void on_checkbox_toggled(GtkToggleButton* toggle, gpointer
user_data);
static void show_warning_dialog(GtkButton* button, const char*
message);

#endif

```

Файл input.cpp

```

#include "io.cpp"
#include "model.h"
#include "input.h"

float prompt_user_for_weight(GtkWindow* parent)
{
    gchar* msg;
    msg = g_locale_to_utf8("Введите вероятность перехода", -1, NULL,
NULL, NULL);
    GtkWidget* dialog = gtk_dialog_new_with_buttons(msg,
parent,
GTK_DIALOG_MODAL,
"_OK",

```

```

        GTK_RESPONSE_OK,
        "_Cancel",
        GTK_RESPONSE_CANCEL,
        NULL);
    GtkWidget* content_area =
gtk_dialog_get_content_area(GTK_DIALOG(dialog));
    GtkWidget* entry = gtk_entry_new();
    gtk_entry_set_placeholder_text(GTK_ENTRY(entry), "0.0 - 1.0");
    gtk_container_add(GTK_CONTAINER(content_area), entry);
    gtk_widget_show_all(dialog);

    float value = -1.0f;
    if (gtk_dialog_run(GTK_DIALOG(dialog)) == GTK_RESPONSE_OK) {
        const char* text = gtk_entry_get_text(GTK_ENTRY(entry));
        char buf[256];
        strncpy(buf, text, sizeof(buf));
        buf[sizeof(buf) - 1] = 0;

        for (char* p = buf; *p; p++) {
            if (*p == '.') *p = ',';
        }

        value = strtod(buf, NULL);
    }
    gtk_widget_destroy(dialog);
    return value;
}

float prompt_user_for_intensity(GtkWindow* parent) {
    GtkWidget* dialog = gtk_dialog_new_with_buttons(
        g_locale_to_utf8("Введите интенсивность входящего потока", -
1, NULL, NULL, NULL),
        parent,
        GTK_DIALOG_MODAL,
        "_OK",
        GTK_RESPONSE_OK,
        g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
        GTK_RESPONSE_CANCEL,
        NULL);
    GtkWidget* content_area =
gtk_dialog_get_content_area(GTK_DIALOG(dialog));
    GtkWidget* label = gtk_label_new(g_locale_to_utf8("Интенсивность
(число с плавающей точкой):", -1, NULL, NULL, NULL));
    gtk_container_add(GTK_CONTAINER(content_area), label);
    char buf[64];
    GtkWidget* entry = gtk_entry_new();
    snprintf(buf, sizeof(buf), "%f", input_intensity);
    gtk_entry_set_text(GTK_ENTRY(entry), buf);
    gtk_entry_set_placeholder_text(GTK_ENTRY(entry),
g_locale_to_utf8("Например: 3.14", -1, NULL, NULL, NULL));
    gtk_container_add(GTK_CONTAINER(content_area), entry);

```

```

gtk_widget_show_all(dialog);

float intensity = -1.0f;
if (gtk_dialog_run(GTK_DIALOG(dialog)) == GTK_RESPONSE_OK) {
    const char* text = gtk_entry_get_text(GTK_ENTRY(entry));
    if (text && strlen(text) > 0) {
        char* endptr = NULL;
        strncpy(buf, text, sizeof(buf));
        buf[sizeof(buf) - 1] = 0;

        for (char* p = buf; *p; p++) {
            if (*p == '.') *p = ',';
        }
        float val = strtod(buf, NULL);
        if (endptr != text) {
            intensity = val;
        }
    }
}

gtk_widget_destroy(dialog);
return intensity;
}

void show_circle_params_dialog(GtkWindow* parent, Circles* circles,
int circle_index) {
    CircleParams* params = &circles->circle_params[circle_index];

    GtkWidget* dialog = gtk_dialog_new_with_buttons(
        g_locale_to_utf8("Параметры СМО", -1, NULL, NULL, NULL),
        parent,
        GTK_DIALOG_MODAL,
        g_locale_to_utf8("_OK", -1, NULL, NULL, NULL),
GTK_RESPONSE_OK,
        g_locale_to_utf8("_Отмена", -1, NULL, NULL, NULL),
GTK_RESPONSE_CANCEL,
        NULL);

    GtkWidget* content_area =
gtk_dialog_get_content_area(GTK_DIALOG(dialog));

    GtkWidget* grid = gtk_grid_new();
    gtk_grid_set_row_spacing(GTK_GRID(grid), 6);
    gtk_grid_set_column_spacing(GTK_GRID(grid), 10);
    gtk_container_set_border_width(GTK_CONTAINER(grid), 10);
    gtk_container_add(GTK_CONTAINER(content_area), grid);

    GtkWidget* label_name =
gtk_label_new(g_locale_to_utf8("Наименование:", -1, NULL, NULL, NULL));
    GtkWidget* entry_name = gtk_entry_new();
    if (params->circle_name)
        gtk_entry_set_text(GTK_ENTRY(entry_name), params-
>circle_name);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		108

```

gtk_grid_attach(GTK_GRID(grid), label_name, 0, 0, 1, 1);
gtk_grid_attach(GTK_GRID(grid), entry_name, 1, 0, 1, 1);

GtkWidget* label_service_time =
gtk_label_new(g_locale_to_utf8("Время обслуживания:", -1, NULL, NULL,
NULL));

GtkWidget* entry_service_time = gtk_entry_new();
char buf[256];
snprintf(buf, sizeof(buf), "%.6g", params->service_time);
gtk_entry_set_text(GTK_ENTRY(entry_service_time), buf);
gtk_grid_attach(GTK_GRID(grid), label_service_time, 0, 1, 1, 1);
gtk_grid_attach(GTK_GRID(grid), entry_service_time, 1, 1, 1, 1);

GtkWidget* label_channel_count =
gtk_label_new(g_locale_to_utf8("Число каналов:", -1, NULL, NULL, NULL));
GtkWidget* entry_channel_count = gtk_entry_new();
snprintf(buf, sizeof(buf), "%d", params->channel_count);
gtk_entry_set_text(GTK_ENTRY(entry_channel_count), buf);
gtk_grid_attach(GTK_GRID(grid), label_channel_count, 0, 2, 1,
1);

gtk_grid_attach(GTK_GRID(grid), entry_channel_count, 1, 2, 1,
1);

/*// Заявки (тип int с выбором)
GtkWidget* label_request_type =
gtk_label_new(g_locale_to_utf8("Заявки:", -1, NULL, NULL, NULL));
GtkWidget* combo_request_type = gtk_combo_box_text_new();

gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(combo_request_type), "0",
g_locale_to_utf8("Без приоритетов", -1, NULL, NULL, NULL));

gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(combo_request_type), "1",
g_locale_to_utf8("Относительные приоритеты", -1, NULL, NULL, NULL));

gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(combo_request_type), "2",
g_locale_to_utf8("Абсолютные приоритеты", -1, NULL, NULL, NULL));

gtk_combo_box_text_append(GTK_COMBO_BOX_TEXT(combo_request_type), "3",
g_locale_to_utf8("Смешанные приоритеты", -1, NULL, NULL, NULL));
gtk_combo_box_set_active(GTK_COMBO_BOX(combo_request_type),
params->request_type);
gtk_grid_attach(GTK_GRID(grid), label_request_type, 0, 3, 1, 1);
gtk_grid_attach(GTK_GRID(grid), combo_request_type, 1, 3, 1, 1);

// Коэффициент вариации (double)
GtkWidget* label_variation_coeff =
gtk_label_new(g_locale_to_utf8("Коэффициент вариации:", -1, NULL, NULL,
NULL));

GtkWidget* entry_variation_coeff = gtk_entry_new();

```

```

    snprintf(buf, sizeof(buf), "%.6g", params->variation_coeff);
    gtk_entry_set_text(GTK_ENTRY(entry_variation_coeff), buf);
    gtk_grid_attach(GTK_GRID(grid), label_variation_coeff, 0, 4, 1,
1);
    gtk_grid_attach(GTK_GRID(grid), entry_variation_coeff, 1, 4, 1,
1);

    // Попор АП (int)
    GtkWidget* label_threshold_ap =
gtk_label_new(g_locale_to_utf8("Попор АП:", -1, NULL, NULL, NULL));
    GtkWidget* entry_threshold_ap = gtk_entry_new();
    snprintf(buf, sizeof(buf), "%d", params->threshold_ap);
    gtk_entry_set_text(GTK_ENTRY(entry_threshold_ap), buf);
    gtk_grid_attach(GTK_GRID(grid), label_threshold_ap, 0, 6, 1, 1);
    gtk_grid_attach(GTK_GRID(grid), entry_threshold_ap, 1, 6, 1, 1);

    // Попор ОП (int)
    GtkWidget* label_threshold_op =
gtk_label_new(g_locale_to_utf8("Попор ОП:", -1, NULL, NULL, NULL));
    GtkWidget* entry_threshold_op = gtk_entry_new();
    snprintf(buf, sizeof(buf), "%d", params->threshold_op);
    gtk_entry_set_text(GTK_ENTRY(entry_threshold_op), buf);
    gtk_grid_attach(GTK_GRID(grid), label_threshold_op, 0, 7, 1, 1);
    gtk_grid_attach(GTK_GRID(grid), entry_threshold_op, 1, 7, 1,
1);*/

    GtkWidget* check_limited_queue =
gtk_check_button_new_with_label(g_locale_to_utf8("Ограниченная длина
очереди", -1, NULL, NULL, NULL));

    gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(check_limited_queue),
params->limited_queue);
    gtk_grid_attach(GTK_GRID(grid), check_limited_queue, 0, 8, 2,
1);

    GtkWidget* label_queue_size =
gtk_label_new(g_locale_to_utf8("Число мест:", -1, NULL, NULL, NULL));
    GtkWidget* entry_queue_size = gtk_entry_new();
    snprintf(buf, sizeof(buf), "%d", params->queue_size);
    gtk_entry_set_text(GTK_ENTRY(entry_queue_size), buf);
    gtk_grid_attach(GTK_GRID(grid), label_queue_size, 0, 9, 1, 1);
    gtk_grid_attach(GTK_GRID(grid), entry_queue_size, 1, 9, 1, 1);

    /*// Метка и entry для числа классов приоритета
    GtkWidget* label_priority_class_count =
gtk_label_new(g_locale_to_utf8("Число классов приоритета:", -1, NULL,
NULL, NULL));
    GtkWidget* entry_priority_class_count = gtk_entry_new();
    snprintf(buf, sizeof(buf), "%d", params->priority_class_count);

```

```

        gtk_entry_set_text(GTK_ENTRY(entry_priority_class_count), buf);
        gtk_grid_attach(GTK_GRID(grid), label_priority_class_count, 0,
5, 1, 1);
        gtk_grid_attach(GTK_GRID(grid), entry_priority_class_count, 1,
5, 1, 1);

        // Создаем GtkScrolledWindow для таблицы
        GtkWidget* scrolled_window = gtk_scrolled_window_new(NULL,
NULL);
        gtk_widget_set_size_request(scrolled_window, 400, 200); //
задаём размер области прокрутки

        gtk_scrolled_window_set_policy(GTK_SCROLLED_WINDOW(scrolled_window),
GTK_POLICY_NEVER, GTK_POLICY_AUTOMATIC);

        // Создаем grid для таблицы
        GtkWidget* priority_grid = gtk_grid_new();
        gtk_grid_set_row_spacing(GTK_GRID(priority_grid), 4);
        gtk_grid_set_column_spacing(GTK_GRID(priority_grid), 6);

        // Добавляем grid в scrolled_window
        gtk_container_add(GTK_CONTAINER(scrolled_window),
priority_grid);

        // Добавляем scrolled_window в основной grid диалога
        gtk_grid_attach(GTK_GRID(grid), scrolled_window, 1, 10, 10, 10);

        gtk_grid_attach(GTK_GRID(priority_grid),
gtk_label_new(g_locale_to_utf8("Класс приоритета", -1, NULL, NULL,
NULL)), 0, 0, 1, 1);
        gtk_grid_attach(GTK_GRID(priority_grid),
gtk_label_new(g_locale_to_utf8("Доля входного потока", -1, NULL, NULL,
NULL)), 1, 0, 1, 1);
        gtk_grid_attach(GTK_GRID(priority_grid),
gtk_label_new(g_locale_to_utf8("Время обслуживания", -1, NULL, NULL,
NULL)), 2, 0, 1, 1);

        // Создаем структуру для хранения виджетов
        PriorityTableWidgets ptw = { 0 };
        ptw.priority_grid = priority_grid;

        // Изначально создаем 1 строку

        load_priority_table_rows(&ptw, params);

        // Подключаем сигнал изменения текста entry
        g_signal_connect(entry_priority_class_count, "changed",
G_CALLBACK(on_priority_class_count_changed), &ptw);*/
        gtk_widget_show_all(dialog);

        int response = gtk_dialog_run(GTK_DIALOG(dialog));

```

```

        if (response == GTK_RESPONSE_OK) {
            project_modified = true;
            const char* name_text =
gtk_entry_get_text(GTK_ENTRY(entry_name));
            /*if (params->circle_name) {
                g_free(params->circle_name);
            }*/
            params->circle_name = g_strdup(name_text);

            const char* service_time_text =
gtk_entry_get_text(GTK_ENTRY(entry_service_time));
            strncpy(buf, service_time_text, sizeof(buf));
            buf[sizeof(buf) - 1] = 0;

            for (char* p = buf; *p; p++) {
                if (*p == '.') *p = ',';
            }

            params->service_time = strtod(buf, NULL);

            const char* channel_count_text =
gtk_entry_get_text(GTK_ENTRY(entry_channel_count));
            params->channel_count = atoi(channel_count_text);

            // Тип заявок
            /*params->request_type =
gtk_combo_box_get_active(GTK_COMBO_BOX(combo_request_type));

            // Коэффициент вариации
            const char* variation_text =
gtk_entry_get_text(GTK_ENTRY(entry_variation_coeff));
            params->variation_coeff = atof(variation_text);

            // Порог АП
            const char* threshold_ap_text =
gtk_entry_get_text(GTK_ENTRY(entry_threshold_ap));
            params->threshold_ap = atoi(threshold_ap_text);

            // Порог ОП
            const char* threshold_op_text =
gtk_entry_get_text(GTK_ENTRY(entry_threshold_op));
            params->threshold_op = atoi(threshold_op_text);*/

            params->limited_queue =
gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(check_limited_queue));

            const char* queue_size_text =
gtk_entry_get_text(GTK_ENTRY(entry_queue_size));
            params->queue_size = atoi(queue_size_text);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		112

```

        /*// Число классов приоритета
            const char* priority_class_count_text =
gtk_entry_get_text(GTK_ENTRY(entry_priority_class_count));
            int new_priority_class_count =
atoi(priority_class_count_text);
            if (new_priority_class_count < 1) new_priority_class_count =
1;
            if (new_priority_class_count > MAX_PRIORITY_CLASSES)
new_priority_class_count = MAX_PRIORITY_CLASSES;
            params->priority_class_count = new_priority_class_count;

            // Сохраняем данные из таблицы приоритетов
            save_priority_table_from_widgets(params, &ptw);*/
    }

    gtk_widget_destroy(dialog);

}

static void on_radio_mode_toggled(GtkToggleButton* toggle, gpointer
user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    if (gtk_toggle_button_get_active(toggle)) {
        calc_vary_mode = (toggle == GTK_TOGGLE_BUTTON(w-
>radio_vary));
        g_print("calc_vary_mode = %d\n", calc_vary_mode);
    }
}

static void on_check_vary_intensity_toggled(GtkToggleButton* toggle,
gpointer user_data) {
    vary_intensity = gtk_toggle_button_get_active(toggle);
    g_print("vary_intensity = %d\n", vary_intensity);
}

static void on_entry_intensity_from_changed(GtkEditable* editable,
gpointer user_data) {
    const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
    intensity_from = strtod(text, NULL);
    g_print("intensity_from = %f\n", intensity_from);
}

static void on_entry_intensity_to_changed(GtkEditable* editable,
gpointer user_data) {
    const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
    intensity_to = strtod(text, NULL);
    g_print("intensity_to = %f\n", intensity_to);
}

```

```

static void on_check_vary_service_toggled(GtkToggleButton* toggle,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    vary_service_time = gtk_toggle_button_get_active(toggle);
    g_print("vary_service_time = %d\n", vary_service_time);
}

static void on_combo_service_changed(GtkComboBox* combo, gpointer
user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(combo);
    g_print("combo_service_circle changed: %d\n", idx);
    if (idx > 0) {
        char buf[32];
        snprintf(buf, sizeof(buf), "%.3f", service_time_from[idx]);
        gtk_entry_set_text(GTK_ENTRY(w->entry_service_from), buf);
        snprintf(buf, sizeof(buf), "%.3f", service_time_to[idx]);
        gtk_entry_set_text(GTK_ENTRY(w->entry_service_to), buf);
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_service), vary_service_time);
    }
    else {
        gtk_entry_set_text(GTK_ENTRY(w->entry_service_from), "0.0");
        gtk_entry_set_text(GTK_ENTRY(w->entry_service_to), "0.0");
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_service), FALSE);
    }
}

static void on_entry_service_from_changed(GtkEditable* editable,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(GTK_COMBO_BOX(w-
>combo_service_circle));
    if (idx > 0) {
        const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
        service_time_from[idx] = strtod(text, NULL);
        g_print("service_time_from[%d] = %f\n", idx,
service_time_from[idx]);
    }
}

static void on_entry_service_to_changed(GtkEditable* editable,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(GTK_COMBO_BOX(w-
>combo_service_circle));

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		114

```

        if (idx > 0) {
            const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
            service_time_to[idx] = strtod(text, NULL);
            g_print("service_time_to[%d] = %f\n", idx,
service_time_to[idx]);
        }
    }

    static void on_check_vary_channels_toggled(GtkToggleButton* toggle,
gpointer user_data) {
        CalcNetWidgets* w = (CalcNetWidgets*)user_data;
        vary_channel_count = gtk_toggle_button_get_active(toggle);
        g_print("vary_channel_count = %d\n", vary_channel_count);
    }

    static void on_combo_channel_changed(GtkComboBox* combo, gpointer
user_data) {
        CalcNetWidgets* w = (CalcNetWidgets*)user_data;
        int idx = gtk_combo_box_get_active(combo);
        g_print("vary_channel_count changed: %d\n", idx);
        if (idx > 0) {
            char buf[32];
            snprintf(buf, sizeof(buf), "%i", channel_count_from[idx]);
            gtk_entry_set_text(GTK_ENTRY(w->entry_channel_from), buf);
            snprintf(buf, sizeof(buf), "%i", channel_count_to[idx]);
            gtk_entry_set_text(GTK_ENTRY(w->entry_channel_to), buf);
            gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_channels), vary_channel_count);
        }
        else {
            gtk_entry_set_text(GTK_ENTRY(w->entry_channel_from), "0");
            gtk_entry_set_text(GTK_ENTRY(w->entry_channel_to), "0");
            gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_channels), FALSE);
        }
    }

    static void on_entry_channel_from_changed(GtkEditable* editable,
gpointer user_data) {
        CalcNetWidgets* w = (CalcNetWidgets*)user_data;
        int idx = gtk_combo_box_get_active(GTK_COMBO_BOX(w-
>combo_channel_circle));
        if (idx > 0) {
            const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
            channel_count_from[idx] = strtod(text, NULL);
            g_print("channel_count_from[%d] = %i\n", idx,
channel_count_from[idx]);
        }
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		115

```

static void on_entry_channel_to_changed(GtkEditable* editable,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(GTK_COMBO_BOX(w-
>combo_channel_circle));
    if (idx > 0) {
        const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
        channel_count_to[idx] = strtouf(text, NULL);
        g_print("channel_count_to[%d] = %i\n", idx,
channel_count_to[idx]);
    }
}

static void on_check_vary_queue_toggled(GtkToggleButton* toggle,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    vary_queue_size = gtk_toggle_button_get_active(toggle);
    g_print("vary_queue_size = %d\n", vary_queue_size);
}

static void on_combo_queue_changed(GtkComboBox* combo, gpointer
user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(combo);
    g_print("vary_queue_size changed: %d\n", idx);
    if (idx > 0) {
        char buf[32];
        snprintf(buf, sizeof(buf), "%i", queue_size_from[idx]);
        gtk_entry_set_text(GTK_ENTRY(w->entry_queue_from), buf);
        snprintf(buf, sizeof(buf), "%i", queue_size_to[idx]);
        gtk_entry_set_text(GTK_ENTRY(w->entry_queue_to), buf);
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_queue), vary_queue_size);
    }
    else {
        gtk_entry_set_text(GTK_ENTRY(w->entry_queue_from), "0");
        gtk_entry_set_text(GTK_ENTRY(w->entry_queue_to), "0");
        gtk_toggle_button_set_active(GTK_TOGGLE_BUTTON(w-
>check_vary_queue), FALSE);
    }
}

static void on_entry_queue_from_changed(GtkEditable* editable,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(GTK_COMBO_BOX(w-
>combo_queue_circle));
    if (idx > 0) {
        const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
        queue_size_from[idx] = strtouf(text, NULL);
        g_print("queue_size_from[%d] = %i\n", idx,
queue_size_from[idx]);
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		116

```

static void on_entry_queue_to_changed(GtkEditable* editable,
gpointer user_data) {
    CalcNetWidgets* w = (CalcNetWidgets*)user_data;
    int idx = gtk_combo_box_get_active(GTK_COMBO_BOX(w-
>combo_queue_circle));
    if (idx > 0) {
        const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
        queue_size_to[idx] = strtouf(text, NULL);
        g_print("queue_size_to[%d] = %i\n", idx,
queue_size_to[idx]);
    }
}

static void on_entry_cycles_count_changed(GtkEditable* editable,
gpointer user_data) {
    const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
    int val = atoi(text);
    if (val < 1) val = 1;
    cycles_count = val;
    g_print("cycles_count = %d\n", cycles_count);
}

static void on_combo_changed(GtkComboBox* combo, gpointer user_data)
{
    DiagWindowData* data = (DiagWindowData*)user_data;

    if (combo == GTK_COMBO_BOX(data->char_combo)) {
        data->selected_char = gtk_combo_box_get_active(combo);
    }
    else if (combo == GTK_COMBO_BOX(data->smo_combo)) {
        data->selected_smo = gtk_combo_box_get_active(combo) + 1;
    }

    gtk_widget_queue_draw(data->drawing_area);
}

static void on_checkbox_toggled(GtkToggleButton* toggle, gpointer
user_data) {
    ShowDiagSettings* settings = (ShowDiagSettings*)user_data;

    int id = GPOINTER_TO_INT(g_object_get_data(G_OBJECT(toggle),
"id"));
    gboolean active = gtk_toggle_button_get_active(toggle);

    switch (id) {
        case 0: settings->show_avg_waiting = active; break;
        case 1: settings->show_avg_in_system = active; break;
        case 2: settings->show_avg_wait_time = active; break;
        case 3: settings->show_avg_soj_time = active; break;
    }
}

```

```

        GtkWidget* drawing_area =
GTK_WIDGET(g_object_get_data(G_OBJECT(toggle), "drawing_area"));
        gtk_widget_queue_draw(drawing_area);
    }

    static void show_warning_dialog(GtkButton* button, const char*
message) {
        GtkWidget* dialog =
gtk_message_dialog_new(GTK_WINDOW(gtk_widget_get_toplevel(GTK_WIDGET(butt
on))),
                        GTK_DIALOG_MODAL,
                        GTK_MESSAGE_WARNING,
                        GTK_BUTTONS_OK,
                        "%s", message);
        gtk_dialog_run(GTK_DIALOG(dialog));
        gtk_widget_destroy(dialog);
    }

```

Файл io.h

```

#ifndef IO_H
#define IO_H

static JsonNode* serialize_circles_to_json(const Circles* circles);
static gboolean save_circles_to_file(const Circles* circles, const
char* filename);
bool load_circles_from_file(const char* filename, Circles* circles);
void save_results_to_file(const char* filename, Circles* circles);

#endif

```

Файл io.cpp

```

#include "model.cpp"
#include "model.h"
#include "io.h"

static JsonNode* serialize_circles_to_json(const Circles* circles) {
    JsonBuilder* builder = json_builder_new();

    json_builder_begin_object(builder);

    json_builder_set_member_name(builder, "general");
    json_builder_begin_array(builder);

    json_builder_begin_object(builder);

    json_builder_set_member_name(builder, "count");
    json_builder_add_int_value(builder, circles->count);
    json_builder_set_member_name(builder, "line_count");

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		118

```

json_builder_add_int_value(builder, circles->line_count);
json_builder_end_object(builder);

json_builder_end_array(builder);

json_builder_set_member_name(builder, "circles");
json_builder_begin_array(builder);
for (int i = 0; i < circles->count; i++) {
    json_builder_begin_object(builder);

    json_builder_set_member_name(builder, "x");
    json_builder_add_double_value(builder, circles->x[i]);

    json_builder_set_member_name(builder, "y");
    json_builder_add_double_value(builder, circles->y[i]);

    json_builder_end_object(builder);
}
json_builder_end_array(builder);

json_builder_set_member_name(builder, "lines");
json_builder_begin_array(builder);
for (int i = 0; i < circles->line_count; i++) {
    json_builder_begin_object(builder);

    json_builder_set_member_name(builder, "weight");
    json_builder_add_double_value(builder, circles-
>lines[i].weight);

    json_builder_set_member_name(builder, "from_index");
    json_builder_add_int_value(builder, circles-
>lines[i].from_index);

    json_builder_set_member_name(builder, "to_index");
    json_builder_add_int_value(builder, circles-
>lines[i].to_index);

    json_builder_end_object(builder);
}
json_builder_end_array(builder);

json_builder_set_member_name(builder, "CircleParams");
json_builder_begin_array(builder);
for (int i = 0; i < circles->count; i++) {
    json_builder_begin_object(builder);

    json_builder_set_member_name(builder, "circle_name");
    json_builder_add_string_value(builder, circles-
>circle_params[i].circle_name);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		119

```

        json_builder_set_member_name(builder, "service_time");
        json_builder_add_double_value(builder, circles-
>circle_params[i].service_time);

        json_builder_set_member_name(builder, "channel_count");
        json_builder_add_int_value(builder, circles-
>circle_params[i].channel_count);

        json_builder_set_member_name(builder, "request_type");
        json_builder_add_int_value(builder, circles-
>circle_params[i].request_type);

        json_builder_set_member_name(builder, "variation_coeff");
        json_builder_add_double_value(builder, circles-
>circle_params[i].variation_coeff);

        if (i > 0) {
            json_builder_set_member_name(builder,
"priority_class_count");
            json_builder_add_int_value(builder, circles-
>circle_params[i].priority_class_count);

            json_builder_set_member_name(builder, "priority_table");
            json_builder_begin_array(builder);
            for (int j = 0; j < circles-
>circle_params[i].priority_class_count; j++) {
                json_builder_begin_object(builder);
                json_builder_set_member_name(builder,
"priority_class");
                json_builder_add_int_value(builder, circles-
>circle_params[i].priority_table[j].priority_class);

                json_builder_set_member_name(builder,
"input_share");
                json_builder_add_double_value(builder, circles-
>circle_params[i].priority_table[j].input_share);

                json_builder_set_member_name(builder,
"service_time");
                json_builder_add_double_value(builder, circles-
>circle_params[i].priority_table[j].service_time);
                json_builder_end_object(builder);
            }
            json_builder_end_array(builder);

            json_builder_set_member_name(builder, "threshold_ap");
            json_builder_add_int_value(builder, circles-
>circle_params[i].threshold_ap);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		120

```

        json_builder_set_member_name(builder, "threshold_op");
        json_builder_add_int_value(builder, circles-
>circle_params[i].threshold_op);

        json_builder_set_member_name(builder, "limited_queue");
        json_builder_add_boolean_value(builder, circles-
>circle_params[i].limited_queue);

        json_builder_set_member_name(builder, "queue_size");
        json_builder_add_int_value(builder, circles-
>circle_params[i].queue_size);

        json_builder_end_object(builder);
    }
    json_builder_end_array(builder);

    json_builder_end_object(builder);

    JsonNode* root = json_builder_get_root(builder);
    g_object_unref(builder);
    return root;
}

static gboolean save_circles_to_file(const Circles* circles, const
char* filename) {
    JsonNode* root = serialize_circles_to_json(circles);
    if (!root) return FALSE;

    JsonGenerator* generator = json_generator_new();
    json_generator_set_root(generator, root);
    gboolean success = json_generator_to_file(generator, filename,
NULL);

    g_object_unref(generator);
    json_node_free(root);

    return success;
}

bool load_circles_from_file(const char* filename, Circles* circles)
{
    JsonParser* parser = json_parser_new();
    GError* error = NULL;

    if (!json_parser_load_from_file(parser, filename, &error)) {
        g_printerr("Error load from JSON: %s\n", error->message);
        g_error_free(error);
        g_object_unref(parser);
        return false;
    }

    JsonNode* root = json_parser_get_root(parser);
    if (!JSON_NODE HOLDS_OBJECT(root)) {

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		121

```

    g_printerr("Root JSON not object\n");
    g_object_unref(parser);
    return false;
}

JsonObject* root_obj = json_node_get_object(root);

clear_circles(circles);

JsonArray* general_array =
json_object_get_array_member(root_obj, "general");
if (!general_array) {
    g_printerr("No general on json\n");
    g_object_unref(parser);
    return false;
}
JsonObject* gen_obj =
json_array_get_object_element(general_array, 0);

circles->count = json_object_get_int_member(gen_obj, "count");
if (circles->count > MAX_CIRCLES) {
    g_printerr("Too many circles\n");
    circles->count = MAX_CIRCLES;
}
circles->line_count = json_object_get_int_member(gen_obj,
"line_count");
if (circles->line_count > MAX_LINES) {
    g_printerr("Too many lines\n");
    circles->line_count = MAX_LINES;
}

JsonArray* circles_array =
json_object_get_array_member(root_obj, "circles");
if (!circles_array) {
    g_printerr("No circles on json\n");
    g_object_unref(parser);
    return false;
}

for (int i = 0; i < circles->count; i++) {
    JsonObject* circ_obj =
json_array_get_object_element(circles_array, i);

    circles->x[i] = json_object_get_double_member(circ_obj,
"x");
    circles->y[i] = json_object_get_double_member(circ_obj,
"y");
}

JsonArray* lines_array = json_object_get_array_member(root_obj,
"lines");

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		122

```

        if (!lines_array) {
            g_printerr("no lines in JSON\n");
            g_object_unref(parser);
            return false;
        }

        for (int i = 0; i < circles->line_count; i++) {
            JsonObject* line_obj =
json_array_get_object_element(lines_array, i);
            Line* line = &circles->lines[i];

            line->from_index = json_object_get_int_member(line_obj,
"from_index");
            line->to_index = json_object_get_int_member(line_obj,
"to_index");
            line->weight = json_object_get_double_member(line_obj,
"weight");
        }

        JsonArray* params_array = json_object_get_array_member(root_obj,
"CircleParams");
        if (!params_array) {
            g_printerr("no lines in JSON\n");
            g_object_unref(parser);
            return false;
        }
        for (int i = 0; i < circles->count; i++) {
            JsonObject* params_obj =
json_array_get_object_element(params_array, i);

            circles->circle_params[i].circle_name =
g_strdup(json_object_get_string_member(params_obj, "circle_name"));
            circles->circle_params[i].service_time =
json_object_get_double_member(params_obj, "service_time");
            circles->circle_params[i].channel_count =
json_object_get_int_member(params_obj, "channel_count");
            circles->circle_params[i].request_type =
json_object_get_int_member(params_obj, "request_type");
            circles->circle_params[i].variation_coeff =
json_object_get_double_member(params_obj, "variation_coeff");
            circles->circle_params[i].threshold_ap =
json_object_get_int_member(params_obj, "threshold_ap");
            circles->circle_params[i].threshold_op =
json_object_get_int_member(params_obj, "threshold_op");
            circles->circle_params[i].limited_queue =
json_object_get_boolean_member(params_obj, "limited_queue");
            circles->circle_params[i].queue_size =
json_object_get_int_member(params_obj, "queue_size");

            if (i > 0) {
                circles->circle_params[i].priority_class_count =
json_object_get_int_member(params_obj, "priority_class_count");
            }
        }
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		123

```

        if (circles->circle_params[i].priority_table) {
            g_free(circles->circle_params[i].priority_table);
            circles->circle_params[i].priority_table = NULL;
        }

        if (circles->circle_params[i].priority_class_count > 0)
        {
            circles->circle_params[i].priority_table =
g_new0(PriorityRow, circles->circle_params[i].priority_class_count);

            JSONArray* priority_array =
json_object_get_array_member(params_obj, "priority_table");
            if (priority_array) {
                int len = json_array_get_length(priority_array);
                if (len > circles-
>circle_params[i].priority_class_count) {
                    len = circles-
>circle_params[i].priority_class_count;
                }
                for (int j = 0; j < len; j++) {
                    JsonObject* prio_obj =
json_array_get_object_element(priority_array, j);
                    circles-
>circle_params[i].priority_table[j].priority_class =
json_object_get_int_member(prio_obj, "priority_class");
                    circles-
>circle_params[i].priority_table[j].input_share =
json_object_get_double_member(prio_obj, "input_share");
                    circles-
>circle_params[i].priority_table[j].service_time =
json_object_get_double_member(prio_obj, "service_time");
                }
            }
        }
        else {
            circles->circle_params[i].priority_table = NULL;
        }
    }

    g_object_unref(parser);
    return true;
}

void save_results_to_file(const char* filename, Circles* circles) {
    FILE* f = fopen(filename, "w");
    if (!f) {
        g_printerr("Ошибка открытия файла для записи: %s\n",
filename);
        return;
    }
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		124

```

    fprintf(f, "Описание варьируемых параметров:\n");

    if (vary_intensity) {
        fprintf(f, "Параметр - Интенсивность входящего потока -
Начальное значение - %.6g - Конечное значение - %.6g - Шаг - %.6g\n",
            intensity_from, intensity_to,
            (cycles_count > 1) ? (intensity_to - intensity_from) /
(cycles_count - 1) : 0.0);
    }
    if (vary_service_time) {
        for (int i = 1; i < circles->count; i++) {
            if (service_time_to[i] != service_time_from[i]) {
                fprintf(f, "Параметр - Время обслуживания СМО S%d -
Начальное значение - %.6g - Конечное значение - %.6g - Шаг - %.6g\n",
                    i,
                    service_time_from[i], service_time_to[i],
                    (cycles_count > 1) ? (service_time_to[i] -
service_time_from[i]) / (cycles_count - 1) : 0.0);
            }
        }
    }
    if (vary_channel_count) {
        for (int i = 1; i < circles->count; i++) {
            if (channel_count_to[i] != channel_count_from[i]) {
                fprintf(f, "Параметр - Число каналов СМО S%d -
Начальное значение - %i - Конечное значение - %i - Шаг - %i\n",
                    i,
                    channel_count_from[i], channel_count_to[i],
                    (cycles_count > 1) ? (channel_count_to[i] -
channel_count_from[i]) / (cycles_count - 1) : 0.0);
            }
        }
    }
    if (vary_queue_size) {
        for (int i = 1; i < circles->count; i++) {
            if (queue_size_to[i] != queue_size_from[i]) {
                fprintf(f, "Параметр - Число мест СМО S%d -
Начальное значение - %i - Конечное значение - %i - Шаг - %i\n",
                    i,
                    queue_size_from[i], queue_size_to[i],
                    (cycles_count > 1) ? (queue_size_to[i] -
queue_size_from[i]) / (cycles_count - 1) : 0.0);
            }
        }
    }

    fprintf(f, "Общее число циклов: -%d-\n", cycles_count);
    fprintf(f, "Расчёт:\n");

    for (int cycle = 0; cycle < cycles_count; cycle++) {
        fprintf(f, "Цикл номер: -%d-\n", cycle + 1);
        fprintf(f, "Исходные данные:\n");
    }

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		125

```

        fprintf(f, "| Номер СМО | Наименование СМО | Время обслуживания |
Число каналов | Число мест |\n");
        double t = (cycles_count == 1) ? 0.0 : (double)cycle /
(cycles_count - 1);
        for (int i = 0; i < circles->count; i++) {
            CircleParams* p = &circles->circle_params[i];
            if (vary_service_time) {
                p->service_time = service_time_from[i] + t *
(service_time_to[i] - service_time_from[i]);
                if (p->service_time < 0.0001) p->service_time =
0.0001;
            }
            if (vary_channel_count) {
                float val = channel_count_from[i] + t *
(channel_count_to[i] - channel_count_from[i]);
                int m = (int)(val + 0.5f);
                if (m < 1) m = 1;
                p->channel_count = m;
            }
            if (vary_queue_size) {
                float val = queue_size_from[i] + t *
(queue_size_to[i] - queue_size_from[i]);
                int m = (int)(val + 0.5f);
                if (m < 1) m = 1;
                p->queue_size = m;
            }
            fprintf(f, "| %9d | %16s | %18.6g | %13d | %10d |\n", i,
p->circle_name ? p->circle_name : "СМО", p->service_time, p-
>channel_count, p->queue_size);
        }

        fprintf(f, "Характеристики отдельных СМО:\n");
        fprintf(f, "| Номер СМО | Коэфф. передачи | Интенсивность
входного потока | Коэфф. загрузки | Ср. число занятых каналов | Ср. число
заявок в СМО | Ср. длина очереди | Ср. время ожидания в очереди | Ср.
время пребывания в СМО |\n");

        for (int i = 0; i < circles->count; i++) {
            CircleCalcResult* res = &results_acc.results[i][cycle];
            CircleParams* p = &circles->circle_params[i];

            int priority = 0;
            double transfer_coef = 0.0;

            fprintf(f,
                "| %9d | %15.6g | %29.6g | %15.6g | %25.6g | %22.6g
| %17.6g | %28.6g | %26.6g |\n",
                i,
                res->throughput,
                res->input_intensity,

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		126

```

        res->load_factor,
                res->busy_channels,
                res->avg_in_system,
                res->avg_in_queue,
                res->avg_wait_queue,
                res->avg_time_system
        );
    }

    double total_input_intensity = 0.0;
    double total_avg_queue_length = 0.0;
    double total_avg_network_length = 0.0;
    double total_avg_queue_wait = 0.0;
    double total_avg_network_wait = 0.0;

    for (int i = 0; i < circles->count; i++) {
        CircleCalcResult* res = &results_acc.results[i][cycle];
        total_input_intensity += res->input_intensity;
        total_avg_queue_length += res->avg_in_queue;
        total_avg_network_length += res->avg_in_system;
        total_avg_queue_wait += res->avg_wait_queue * res->throughput;
        total_avg_network_wait += res->avg_time_system * res->throughput;
    }
    total_input_intensity = input_intensity;
    if (vary_intensity) {
        total_input_intensity = intensity_from + t *
(intensity_to - intensity_from);
    }
    fprintf(f, "Характеристики сети в целом:\n");
    fprintf(f, "| Интенсивность входящего потока: %.6g\n",
total_input_intensity);
    fprintf(f, "| Среднее число заявок в очередях: %.6g\n",
total_avg_queue_length);
    fprintf(f, "| Среднее число заявок в сети: %.6g\n",
total_avg_network_length);
    fprintf(f, "| Среднее время ожидания в очередях: %.6g\n",
total_avg_queue_wait);
    fprintf(f, "| Среднее время пребывания заявки в сети:
%.6g\n\n", total_avg_network_wait);
    }

    fclose(f);
    g_print("Результаты сохранены в файл %s\n", filename);
}

```

Файл model.h

```

#ifndef MODEL_H
#define MODEL_H

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		127

```

#define MAX_CIRCLES 100
#define MAX_CYCLES 10000
#define MAX_LINES 100
#define MAX_PRIORITY_CLASSES 100

typedef struct {
    int from_index;
    int to_index;
    float weight;
} Line;

typedef struct {
    int priority_class;
    double input_share;
    double service_time;
} PriorityRow;

typedef struct {
    GtkWidget* priority_grid;
    GtkWidget* priority_entries[MAX_PRIORITY_CLASSES][3];
    int current_rows;
} PriorityTableWidgets;

typedef struct {
    char*          circle_name;
    double         service_time;
    int            channel_count;
    int            request_type;
    double         variation_coeff;
    int            priority_class_count;
    PriorityRow*    priority_table;
    int            threshold_ap;
    int            threshold_op;
    gboolean       limited_queue;
    int            queue_size;
} CircleParams;

typedef struct {
    double x[MAX_CIRCLES];
    double y[MAX_CIRCLES];
    int count;

    gboolean add_mode;
    gboolean del_mode;
    gboolean move_mode;
    gboolean add_line_mode;
    gboolean delLineMode = false;

    Line lines[MAX_LINES];
    int line_count;

    int line_first_circle;
    int moving_index;

```

```

CircleParams circle_params[MAX_CIRCLES];
} Circles;

typedef struct {
    GtkWidget* radio_fixed;
    GtkWidget* radio_vary;

    GtkWidget* check_vary_intensity;
    GtkWidget* entry_intensity_from;
    GtkWidget* entry_intensity_to;

    GtkWidget* check_vary_service;
    GtkWidget* combo_service_circle;
    GtkWidget* entry_service_from;
    GtkWidget* entry_service_to;

    GtkWidget* check_vary_channels;
    GtkWidget* combo_channel_circle;
    GtkWidget* entry_channel_from;
    GtkWidget* entry_channel_to;

    GtkWidget* check_vary_queue;
    GtkWidget* combo_queue_circle;
    GtkWidget* entry_queue_from;
    GtkWidget* entry_queue_to;

    GtkWidget* entry_cycles_count;
} CalcNetWidgets;

typedef struct {
    double throughput;           // throughput coefficient K
    double input_intensity;     // lambda
    double load_factor;         // p
    double busy_channels;       // average number of busy channels
    double avg_in_system;       // L
    double avg_in_queue;        // Lq
    double avg_wait_queue;      // Wq
    double avg_time_system;     // W
} CircleCalcResult;

typedef struct {
    CircleCalcResult results[MAX_CIRCLES][MAX_CYCLES]; // results
for each M/M/m queue and cycle
    int cycle_count;
    int circle_count;
} CalcResultsAccumulator;

```

```

typedef struct {
    gboolean show_avg_waiting;    // Среднее число заявок, ожидающих
обслуживания
    gboolean show_avg_in_system; // Среднее число заявок в системе
    gboolean show_avg_wait_time; // Среднее время ожидания
    gboolean show_avg_soj_time;  // Среднее время пребывания
} ShowDiagSettings;

typedef enum {
    CHAR_TRANS_PROB,
    CHAR_INTENSITY,
    CHAR_LOAD_COEFF,
    CHAR_BUSY_CHANNELS,
    CHAR_AVG_REQUESTS,

    CHAR_AVG_QUEUE_LEN,
    CHAR_AVG_WAIT_TIME,
    CHAR_AVG_STAY_TIME,
    CHAR_COUNT
} CharacteristicType;

typedef struct {
    GtkWidget* window;
    GtkWidget* char_combo;
    GtkWidget* smo_combo;
    GtkWidget* drawing_area;
    int selected_char;
    int selected_smo;
} DiagWindowData;

void delete_circle_and_lines(Circles* circles, int circle_index);
void clear_circles(Circles* circles);
void free_priority_table(CircleParams* params);
void create_priority_table_rows(PriorityTableWidgets* ptw, int
rows);
void on_priority_class_count_changed(GtkEditable* editable, gpointer
user_data);
void update_flow_shares(PriorityTableWidgets* ptw);
void save_priority_table_from_widgets(CircleParams* params,
PriorityTableWidgets* ptw);
void load_priority_table_rows(PriorityTableWidgets* ptw,
CircleParams* params);

GtkWidget* matrix_window = NULL;
float input_intensity = 1.0f;

gboolean vary_intensity = FALSE;
float intensity_from = 0.0f;
float intensity_to = 0.0f;

```

```

gboolean vary_service_time = { FALSE };
float service_time_from[MAX_CIRCLES] = { 0.0f };
float service_time_to[MAX_CIRCLES] = { 0.0f };

gboolean vary_channel_count = { FALSE };
int channel_count_from[MAX_CIRCLES] = { 0 };
int channel_count_to[MAX_CIRCLES] = { 0 };

gboolean vary_queue_size = { FALSE };
int queue_size_from[MAX_CIRCLES] = { 0 };
int queue_size_to[MAX_CIRCLES] = { 0 };

int cycles_count = 2;

gboolean calc_vary_mode = FALSE;

CalcResultsAccumulator results_acc = { 0 };

static bool project_modified = false;


int ccount = 0;

const char* characteristic_names[CHAR_COUNT] = {
    g_locale_to_utf8("Коефф. передачи", -1, NULL, NULL, NULL),
    g_locale_to_utf8("Интенсивность входного потока", -1, NULL, NULL,
NULL),
    g_locale_to_utf8("Коефф. загрузки", -1, NULL, NULL, NULL),
    g_locale_to_utf8("Ср. число занятых каналов", -1, NULL, NULL,
NULL),
    g_locale_to_utf8("Ср. число заявок в СМО", -1, NULL, NULL, NULL),
    g_locale_to_utf8("Ср. длина очереди", -1, NULL, NULL, NULL),
    g_locale_to_utf8("Ср. время ожидания в очереди", -1, NULL, NULL,
NULL),
    g_locale_to_utf8("Ср. время пребывания в СМО", -1, NULL, NULL,
NULL)
};

static ShowDiagSettings diag_settings = {
    .show_avg_waiting = TRUE,
    .show_avg_in_system = TRUE,
    .show_avg_wait_time = TRUE,
    .show_avg_soj_time = TRUE
};

#endif

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		131

Файл model.cpp

```
#include "main.h"
#include "model.h"

void delete_circle_and_lines(Circles* circles, int circle_index) {
    for (int i = 0; i < circles->line_count; ) {
        if (circles->lines[i].from_index == circle_index || circles->lines[i].to_index == circle_index) {
            for (int j = i; j < circles->line_count - 1; j++) {
                circles->lines[j] = circles->lines[j + 1];
            }
            circles->line_count--;
        }
        else {
            i++;
        }
    }

    for (int i = circle_index; i < circles->count - 1; i++) {
        circles->x[i] = circles->x[i + 1];
        circles->y[i] = circles->y[i + 1];
    }
    circles->count--;

    for (int i = 0; i < circles->line_count; i++) {
        if (circles->lines[i].from_index > circle_index) {
            circles->lines[i].from_index--;
        }
        if (circles->lines[i].to_index > circle_index) {
            circles->lines[i].to_index--;
        }
    }
}

void clear_circles(Circles* circles) {
    for (int i = 0; i < circles->count; i++) {
        circles->x[i] = 0;
        circles->y[i] = 0;
    }
    circles->count = 0;
    circles->line_count = 0;
}

void free_priority_table(PriorityTableWidgets* ptw) {
    for (int i = 1; i <= ptw->current_rows; i++) {
        for (int j = 0; j < 3; j++) {
            GtkWidget* child = gtk_grid_get_child_at(GTK_GRID(ptw->priority_grid), j, i);
            if (child) {
                gtk_widget_destroy(child);
            }
        }
    }
}
```

```

    }
    }
    ptw->current_rows = 0;
}

void create_priority_table_rows(PriorityTableWidgets* ptw, int rows)
{
    free_priority_table(ptw);

    if (rows > MAX_PRIORITY_CLASSES) rows = MAX_PRIORITY_CLASSES;
    ptw->current_rows = rows;

    char buf[32];
    double flow_share = 1.0 / rows;
    for (int i = 0; i < rows; i++) {
        ptw->priority_entries[i][0] = gtk_entry_new();
        snprintf(buf, sizeof(buf), "%d", i + 1);
        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][0]),
buf);
        gtk_grid_attach(GTK_GRID(ptw->priority_grid), ptw-
>priority_entries[i][0], 0, i + 1, 1, 1);

        ptw->priority_entries[i][1] = gtk_entry_new();
        snprintf(buf, sizeof(buf), "%.4f", flow_share);
        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][1]),
buf);
        gtk_grid_attach(GTK_GRID(ptw->priority_grid), ptw-
>priority_entries[i][1], 1, i + 1, 1, 1);

        ptw->priority_entries[i][2] = gtk_entry_new();

        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][2]),
"0.0");
        gtk_grid_attach(GTK_GRID(ptw->priority_grid), ptw-
>priority_entries[i][2], 2, i + 1, 1, 1);
    }

    gtk_widget_show_all(ptw->priority_grid);
}

void on_priority_class_count_changed(GtkEditable* editable, gpointer
user_data) {
    PriorityTableWidgets* ptw = (PriorityTableWidgets*)user_data;
    const char* text = gtk_entry_get_text(GTK_ENTRY(editable));
    int new_count = atoi(text);
    if (new_count < 0) new_count = 0;
    if (new_count > MAX_PRIORITY_CLASSES) new_count =
MAX_PRIORITY_CLASSES;

    update_flow_shares(ptw);
    create_priority_table_rows(ptw, new_count);
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		133

```

void update_flow_shares(PriorityTableWidgets* ptw) {
    if (ptw->current_rows <= 0) return;

    double flow_share = 1.0 / ptw->current_rows;
    char buf[32];

    for (int i = 0; i < ptw->current_rows; i++) {
        snprintf(buf, sizeof(buf), "%.4f", flow_share);
        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][1]),
buf);
    }
}

void save_priority_table_from_widgets(CircleParams* params,
PriorityTableWidgets* ptw) {
    if (!params || !ptw) return;

    int count = params->priority_class_count;
    if (count < 1) count = 1;
    if (count > MAX_PRIORITY_CLASSES) count = MAX_PRIORITY_CLASSES;
    if (params->priority_table) {
        g_free(params->priority_table);
    }
    params->priority_table = g_new0(PriorityRow, count);
    for (int i = 0; i < count; i++) {
        const char* text = gtk_entry_get_text(GTK_ENTRY(ptw-
>priority_entries[i][0]));
        params->priority_table[i].priority_class = atoi(text);

        text = gtk_entry_get_text(GTK_ENTRY(ptw-
>priority_entries[i][1]));
        params->priority_table[i].input_share = atof(text);

        text = gtk_entry_get_text(GTK_ENTRY(ptw-
>priority_entries[i][2]));
        params->priority_table[i].service_time = atof(text);
    }
}

void load_priority_table_rows(PriorityTableWidgets* ptw,
CircleParams* params) {
    int rows = params->priority_class_count;
    if (rows > MAX_PRIORITY_CLASSES) rows = MAX_PRIORITY_CLASSES;
    ptw->current_rows = rows;

    char buf[32];
    double flow_share = 1.0 / rows;
    for (int i = 0; i < rows; i++) {
        ptw->priority_entries[i][0] = gtk_entry_new();
        snprintf(buf, sizeof(buf), "%i", params-
>priority_table[i].priority_class);

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		134

```

        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][0]), buf);
        gtk_grid_attach(GTK_GRID(ptw->priority_grid), ptw-
>priority_entries[i][0], 0, i + 1, 1, 1);

        ptw->priority_entries[i][1] = gtk_entry_new();
        snprintf(buf, sizeof(buf), "%.4f", params-
>priority_table[i].input_share);
        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][1]),
buf);
        gtk_grid_attach(GTK_GRID(ptw->priority_grid), ptw-
>priority_entries[i][1], 1, i + 1, 1, 1);

        ptw->priority_entries[i][2] = gtk_entry_new();
        snprintf(buf, sizeof(buf), "%.4f", params-
>priority_table[i].service_time);
        gtk_entry_set_text(GTK_ENTRY(ptw->priority_entries[i][2]),
buf);
        gtk_grid_attach(GTK_GRID(ptw->priority_grid), ptw-
>priority_entries[i][2], 2, i + 1, 1, 1);
    }

    gtk_widget_show_all(ptw->priority_grid);
}

```

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		135

Приложение В

(обязательное)

Программа расчёта характеристик стохастических сетей массового

обслуживания

Презентация.

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		136

Программа расчёта характеристик стохастичес кой массовой обслужива

боты:

студент гр. 21ВВВ1 Репин Иван Вадимович
руководитель:

					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		137

Что такое СМО?

Цель работы и объекты

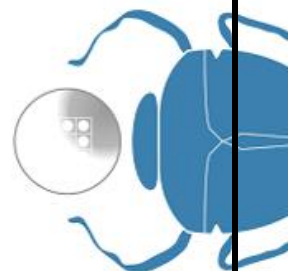
сети массового обслуживания предельно проста – разработать программу для расчёта предельно, включающую несколько связей между стохастическими сетями массового обслуживания. Цель работы – проанализировать сложность моделирования процессов обслуживания в сетях массового обслуживания. Такие модели позволяют анализировать эффективность работы системы, поведение системы при изменении параметров, число заявок в сети, время обслуживания заявок, что является важным фактором при принятии решений.

Цель работы – пользователи приложения для анализа стохастических сетей массового обслуживания.

Сравнение аналогов

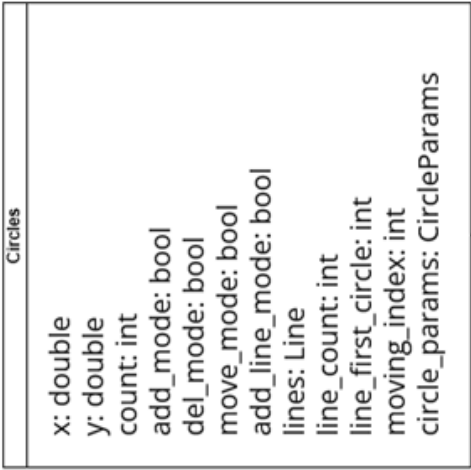
Тип лицензии	Интерфейс	Визуальное моделирование
Коммерческая	Графический	Да
Коммерческая	Графический	Да
Открытая	Код	Нет
Открытая	Смешанный	Частично

Используемые технологии



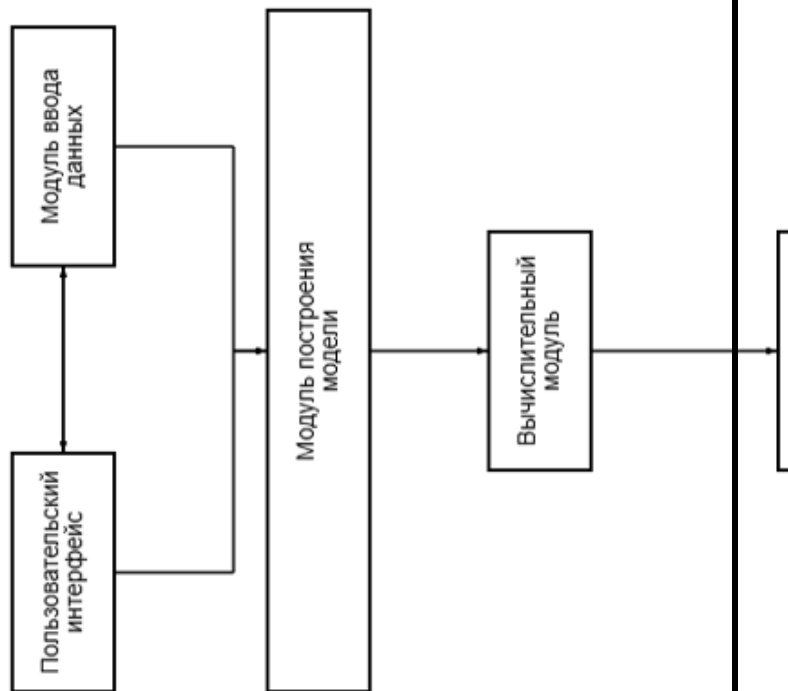
					ПГУ 1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		140

Структура Circles



Изм.	Лист	№ докум	Подпись	Дата

Архитектура приложения



Расчёт сети

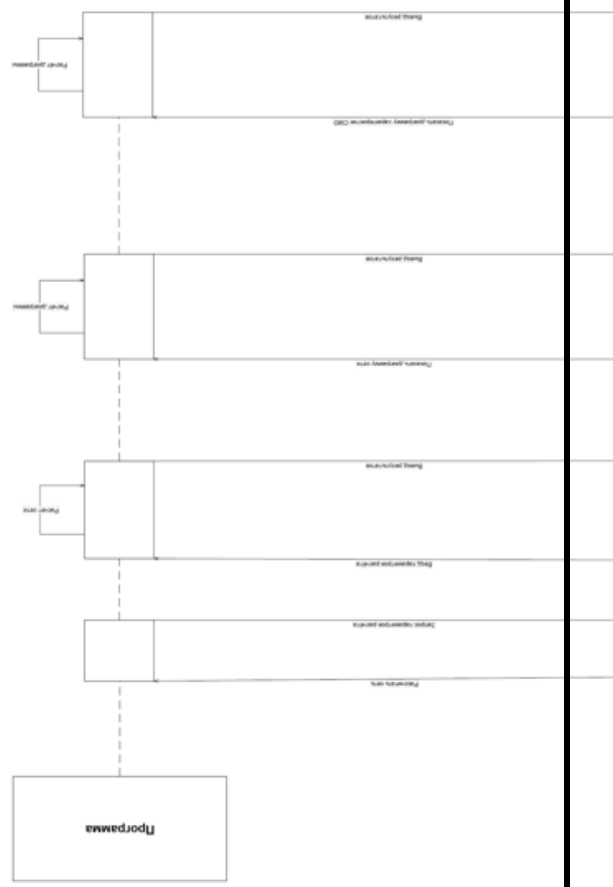
переходов:

0	1	0	0
0.1	0.3	0.3	0.3
0	0	0	1
0	1	0	0

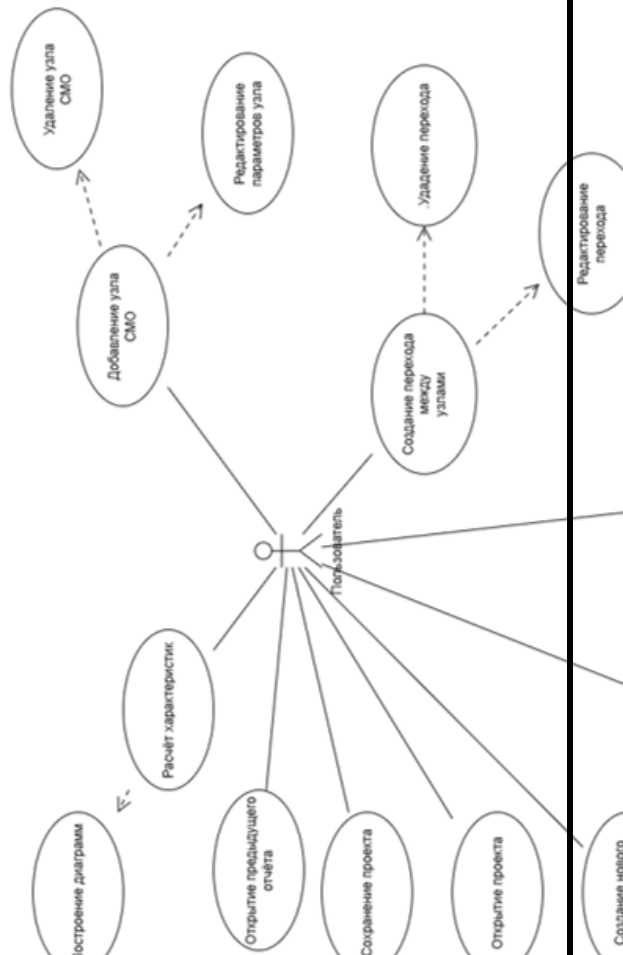
ие параметры:

	S1	S2
Число каналов	1 - 2	1 - 2
Время	0,0005-0,001	0,001-0,002

Диаграмма последовательности

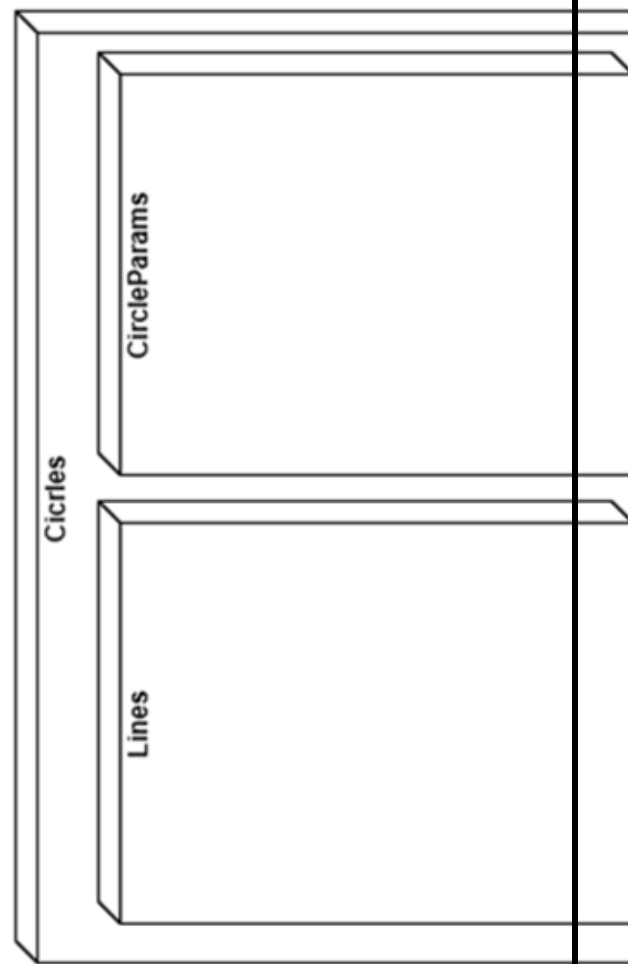


Грамматика вариантов исполнения



Изм.	Лист	№ докум	Подпись	Дата

L-диаграмма развёртывания

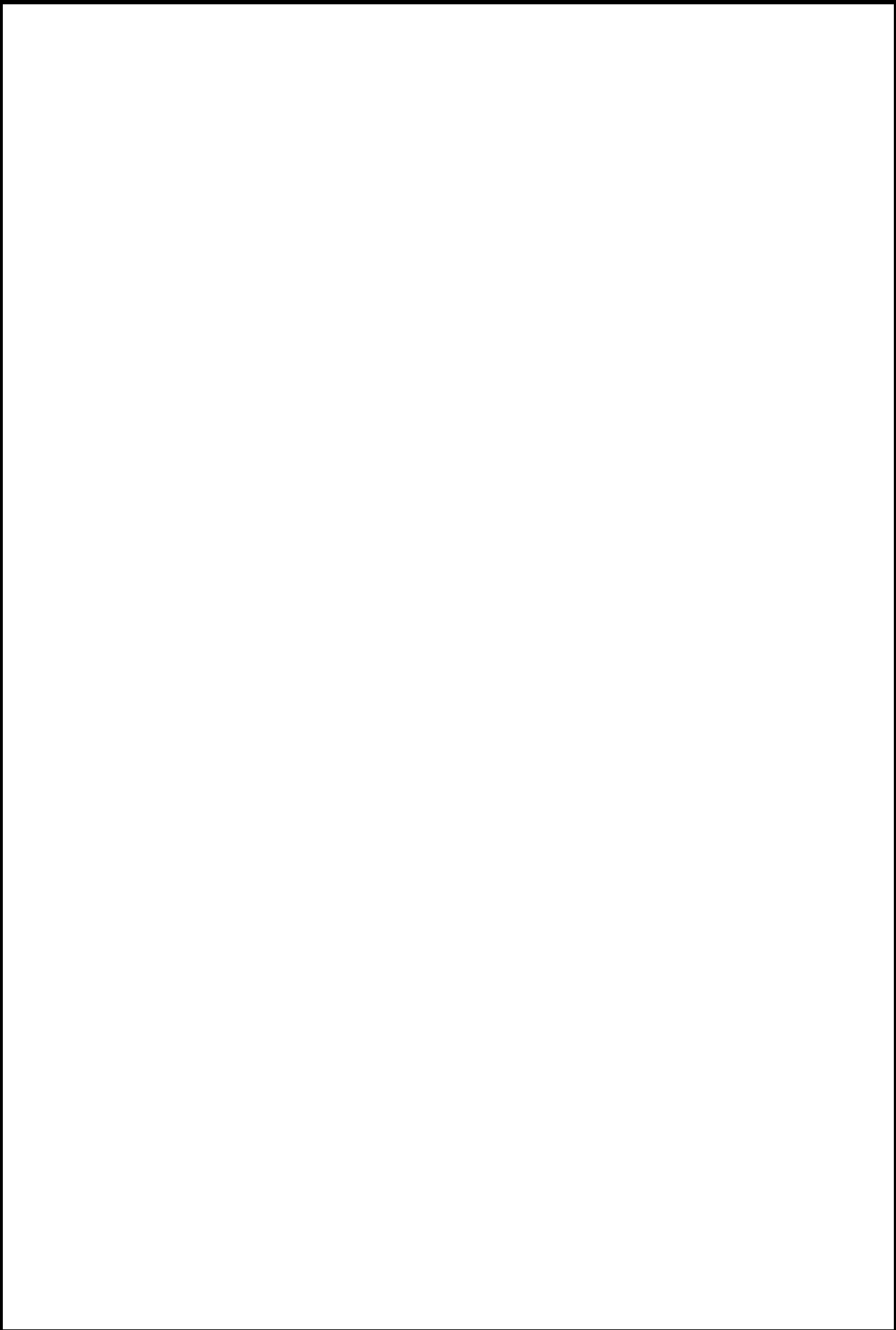


Изм.	Лист	№ докум	Подпись	Дата

ПАСИБО ЗА ВНИМАНИЕ

Результаты работы

Алгоритм расчёта СМО типа М
хранение проекта в JSON формате
Автоматизм отрисовки диаграмм и графов
Одосплатформенное приложение на



Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.18.001 ПЗ

Лис
149