TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования

Школа: Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
ООП: Разработка программно-информационных систем
Отделение школы (НОЦ): Отделение информационных технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема работы
Программный комплекс для обработки данных на буровой установке

УДК 004.65:622.242.2

Обучающийся

Группа	ФИО	Подпись	Дата
8К91	Туев Дмитрий Васильевич		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Кузнецов Дмитрий Юрьевич	канд. техн. наук		

Консультант от инженерно-производственного центра «Геофит»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Инженер-программист	Дранов И.Г.			

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание		Подпись	
Профессор ОСГН	Гасанов Магеррам Али оглы	док. эконом. наук			

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ст. преподаватель	Мезенцева Ирина Леонидовна			

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП, должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Чердынцев Евгений Сергеевич	канд. техн. наук		

Томск – 2023 г.

Планируемые результаты освоения ООП

Код компетенции	Наименование компетенции
УК(У)-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач
УК(У)-2	Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений
УК(У)-3	Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде
УК(У)-4	Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(-ых) языке(-ах)
УК(У)-5	Способен воспринимать межкультурное разнообразие общества в социально-историческом, этическом и философском контекстах
УК(У)-6	Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни
УК(У)-7	Способен поддерживать должный уровень физической подготовленности для обеспечения полноценной социальной и профессиональной деятельности
УК(У)-8	Способен создавать и поддерживать безопасные условия жизнедеятельности, в том числе при возникновении чрезвычайных ситуаций
ОПК(У)-1	Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности
ОПК(У)-2	Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности
ОПК(У)-3	Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности
ОПК(У)-4	Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью
ОПК(У)-5	Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем
ОПК(У)-6	Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов
ОПК(У)-7	Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой
ОПК(У)-8	Способен осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в

Код компетенции	Наименование компетенции
	требуемом формате с использованием информационных, компьютерных и сетевых технологий.
ПК(У)-1	Владение навыками разработки требований и проектирования программного обеспечения
ПК(У)-2	Владение навыками разработки документов и стратегии тестирования программного обеспечения
ПК(У)-3	Владение навыками моделирования, анализа и использования формальных методов конструирования программного обеспечения
ПК(У)-4	Владение навыками использования операционных систем, сетевых технологий, средств разработки программного интерфейса, применения языков и методов формальных спецификаций, систем управления базами данных
ПК(У)-5	Владение концепциями и атрибутами качества программного обеспечения (надежности, безопасности, удобства использования), в том числе роли людей, процессов, методов, инструментов и технологий обеспечения качества

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования

Школа: Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
ООП: Разработка программно-информационных систем
Отделение школы (НОЦ): Отделение информационных технологий

УТВЕРЖДАЮ
Руководитель ООП/ОПОП
_____ ___.____ Чердынцев Е.С.
(Подпись) (Дата) (ФИО)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Обучающийся

Группа	ФИО
8К91	Туев Дмитрий Васильевич

Тема работы

Программный комплекс для обработки данных на буровой установке	
Утверждена приказом директора	№102-28/с от 12.04.2023

Срок сдачи обучающимся выполненной работы:	17.06.2023

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Работа направлена на разработку программного комплекса для обработки данных на буровой установке.
Перечень разделов пояснительной записки, подлежащих исследованию, проектированию и разработке	1. Исследование предметной области 2. Проектирование программной системы 3. Реализация программной системы 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение 5. Социальная ответственность
Перечень графического материала	Презентация в формате *.pptx
Консультанты по разделам выпускной квалификационной работы	
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Гасанов Магеррам Али оглы
Социальная ответственность	Мезенцева Ирина Леонидовна

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	02.03.2023

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Кузнецов Дмитрий Юрьевич	канд. техн. наук		02.03.2023

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
8К91	Туев Дмитрий Васильевич		02.03.2023

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования

Школа: Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
ООП: Разработка программно-информационных систем
Отделение школы (НОЦ): Отделение информационных технологий
Период выполнения: весенний семестр 2022/2023 учебного года

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Обучающийся

Группа	ФИО
8К91	Туев Дмитрий Васильевич

Тема работы

Программный комплекс для обработки данных на буровой установке

Срок сдачи обучающимся выполненной работы:	17.06.2023

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
17.03.2023	Исследование предметной области	25
14.04.2023	Проектирование программной системы	25
20.05.2023	Реализация программной системы	20
24.05.2023	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	15
24.05.2023	Социальная ответственность	15

СОСТАВИЛ
Руководитель ВКР:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Кузнецов Дмитрий Юрьевич	канд. техн. наук		02.03.2023

Консультанты:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ОСГН	Гасанов Магеррам Али оглы	док. эконом. наук		02.03.2023
Ст. преподаватель	Мезенцева Ирина Леонидовна			02.03.2023

СОГЛАСОВАНО
Руководитель ООП/ОПОП

Руководитель ООП, должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Чердынцев Евгений Сергеевич	канд. техн. наук		03.03.2023

Обучающийся

Группа	ФИО	Подпись	Дата
8К91	Туев Дмитрий Васильевич		02.03.2023

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа		ФИО	
8К91		Туев Дмитрий Васильевич	
Школа	ИШИТР	**Отделение (НОЦ)**	ОИТ
Уровень образования	Бакалавриат	**Направление/ специальность**	09.03.04 Программная инженерия

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Оклад научного руководителя – 30 000 руб*. Оклад научного консультанта от организации – 50 000 руб*. Оклад студента – 30 000 руб*. * Значения выбраны для демонстрации методики расчета и не являются настоящими (см. раздел 4.3.4.3)
2. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	Коэффициент отчислений во внебюджетные фонды – 30%
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого и инновационного потенциала НТИ	1. Описание потребителей разрабатываемого продукта 2. QuaD-анализ 3. SWOT-анализ
2. Разработка устава научно-технического проекта	1. Структура работ в рамках ВКР
3. Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок	1. Определение трудоемкости работ. 2. Разработка графика проведения научного исследования.
4. Определение ресурсной, финансовой, экономической эффективности	1. Расчет показателей финансовой эффективности, ресурсоэффективности и эффективности исполнения
Перечень графического материала:	
1. Оценка конкурентоспособности технических решений *2. Матрица SWOT* *3. График проведения и бюджет НТИ* *4. Оценка ресурсной, финансовой и экономической эффективности НТИ*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ОСГН	Гасанов Магеррам Али оглы	Доктор экономических наук		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К91	Туев Дмитрий Васильевич		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа		ФИО	
8К91		Туев Дмитрий Васильевич	
Школа	ИШИТР	**Отделение (НОЦ)**	ОИТ
Уровень образования	Бакалавриат	**Направление/ специальность**	09.03.04 Программная инженерия

Тема ВКР:

Программный комплекс для обработки данных на буровой установке	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	*Объект исследования:* программный комплекс для обработки данных, получаемых с системы телеметрического сопровождения процесса бурения *Область применения:* бурение нефтяных скважин *Рабочая зона:* офис на территории инженерно-производственного центра «Геофит» *Размеры помещения:* 5х5 метра Количество и наименование оборудования рабочей зоны: компьютер Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне: проектирование и разработка программы
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности при разработке проектного решения: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	– Трудовой кодекс РФ; – ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования» – ГОСТ 21889-76 «Система человек-машина. Кресло человека-оператора. Общие эргономические требования» – СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» – СП 2.2.3670-20 «Санитарно-эпидемиологические требования к условиям труда»
2. Производственная безопасность при разработке проектного решения: – Анализ выявленных вредных и опасных производственных факторов	**Вредные:** – повышенный уровень шума; – наличие электромагнитных полей промышленных и радиочастот; – отсутствие или недостаток необходимого искусственного освещения; – повышенная пульсация светового потока; – нервно-психические перегрузки, связанные с напряженностью трудового процесса; **Опасные:**

	– факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов. **Требуемые средства коллективной защиты от выявленных факторов:** – системы естественного освещения; – приборы искусственного освещения; – предохранительные устройства.
3. Экологическая безопасность при разработке проектного решения	Воздействие на селитебную зону не выявлено. Воздействие на литосферу из-за неверного способа утилизации рабочей техники. Воздействие на гидросферу из-за неверного способа утилизации рабочей техники. Воздействие на атмосферу из-за неверного способа утилизации рабочей техники.
4. Безопасность в чрезвычайных ситуациях при разработке проектного решения	**Возможные ЧС:** – природные (землетрясения, ураганы и т.д.); – техногенные (транспортные аварии, пожары и т.д.); – биолого-социальные (эпидемии). **Наиболее типичная ЧС:** пожар
Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель	Мезенцева Ирина Леонидовна			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К91	Туев Дмитрий Васильевич		

Реферат

Выпускная квалификационная работа содержит: 129 страниц, 32 рисунка, 18 таблиц, 5 листингов, 34 источника, 2 приложения.

Ключевые слова: разработка, проектирование, программный комплекс, телеметрическая система, измерение во время бурения, каротаж во время бурения.

Объектом исследования является один из модулей комплекса наземной аппаратуры и ПО системы телеметрического сопровождения процесса бурения.

Цель работы: разработка программного комплекса для приема, обработки и отправки данных с акцентом на расширяемость системы и упрощение процесса внедрения новой функциональности, регулярно возникающей в связи с бизнес-потребностями.

В процессе исследования был произведен анализ существующей версии программы. Были выявлены факторы, затрудняющие расширение существующей версии программы. Было принято решение разработать новую версию программы с новой архитектурой.

В результате разработки был спроектирован программный комплекс. Были реализованы ядро системы, часть модулей системы, значительная часть пользовательского интерфейса.

Степень внедрения: система находится в процессе доработки сотрудниками отдела инженерных разработок.

Область применения: система заменит существующую версию программы для обработки данных на буровой установки.

Оглавление

Список терминов и сокращений

WITS (Wellsite Information Transfer Specification) – спецификация для формата обмена информации о процессе бурения [27].

WITSML (WITS XML) – спецификация для формата обмена информации о процессе бурения на основе формата XML [33].

WAKE – основанный на SLIP двоичный протокол передачи данных, предназначенный для обмена данными между компьютером и микроконтроллерными устройствами посредством последовательного порта или аналогичного ему канала связи (например, TCP/IP) [5].

SLIP (Serial Link Interface Protocol) – описанный в RFC 1055 протокол для передачи IP пакетов посредством последовательного порта [24].

Блок СПД (сбора и подготовки данных) – программно-аппаратный комплекс для считывания показаний с различных датчиков, производимый в «Геофит».

КСПД (контроллер блока сбора и подготовки данных) – модуль блока СПД, отвечающий за управление блоком СПД и за передачу данных на внешние устройства.

Модуль приема наземных данных (ПНД) – модуль СПД, отвечающий за считывание показаний с датчиков, их оцифровку, фильтрацию и передачу в КСПД.

SibReceiver – программа для декодирования телеметрии, разработанная в «Геофит».

GS Terminal (GeoScan Terminal) – исследуемая программа для обработки и обмена данными, разработанная в «Геофит».

Инклинометрия – методика определения пространственного положения ствола буровой скважины путем непрерывного замера угла наклона, азимута и глубины сенсоров по стволу.

Вертикальная глубина – расстояние от скважины на поверхности до точки в скважине по вертикали.

Глубина по стволу – расстояние от скважины на поверхности до точки в скважине, замеренное по стволу скважине.

Measurement While Drilling (MWD) – телеметрическое сопровождение процесса бурения, позволяющее определять пространственное положение ствола скважины и управлять направлением бурения.

Logging While Drilling (LWD) – проведение геофизических исследований скважины в процессе бурения.

Unified Modelling Language (UML) – язык графического анализа и проектирования программных систем, а также моделирования бизнес-процессов.

Фреймворк – набор библиотек и инструментов для разработки программного обеспечения.

Hyper-Text Markup Language (HTML) – язык разметки, применяемый для написания веб-страниц [15].

Cascading Style Sheets (CSS) – язык описания стиля веб-страницы [6].

JavaScript – скриптовый язык программирования, применяемый в веб-страницах.

Hyper-Text Transfer Protocol (HTTP) – протокол передачи данных, используемый в сети Интернет [25].

JavaScript Object Notation (JSON) – формат обмена данными, основанный на синтаксисе языка JavaScript [9].

Введение

Бурение нефтяных и газовых скважин представляет собой сложный технологический процесс, контроль за состоянием которого требует сбора и обработки большого набора информации, поступающей как от оборудования, расположенного непосредственно на буровой установке, так и от забойного оборудования, расположенного в скважине. Для передачи информации от забоя ствола скважины на поверхность без прерывания процесса бурения применяются системы, называемые в иностранной литературе Measurement While Drilling (MWD) и Logging While Drilling (LWD) – измерение во время бурения и каротаж во время бурения, соответственно. Собираемая информация в дальнейшем передается различным экспертам для анализа ситуации и принятия оперативных решений во время процесса бурения.

Одним из разработчиков и производителей систем MWD-LWD, а также ПО для информационного обеспечения процесса бурения в России, является инженерно-производственный центр «Геофит» (филиал ООО «Технологическая Компания Шлюмберже», г. Томск, Россия). В частности, одним из продуктов «Геофит» является «блок СПД» – блок сбора и подготовки данных, получаемых от забойного оборудования, а также от оборудования, расположенного непосредственно на буровой установке. Блок СПД передает данные на персональный компьютер (ПК) посредством Ethernet, используя несколько протоколов передачи данных (WITS, WAKE, HTTP).

Для конфигурации наземного оборудования; сбора данных, принимаемых из разных источников; их визуального отображения и анализа, а также их дальнейшей передачи экспертам, вовлеченным в процесс бурения, «Геофит» предлагает пакет специализированного ПО – «SibReceiver», «GS Terminal», и др. Задачу по приему пакетов данных из различных источников (блок СПД, «SibReceiver», сторонние подразделения) с последующим формированием пакетов данных для пересылки в другие модули системы выполняет программа «GS Terminal». Также эта программа выполняет

функцию коррекции принимаемых данных и расчета на их основе ряда дополнительных параметров. Основным пользователем «GS Terminal» является инженер по телеметрическому сопровождению процесса бурения, однако в программе реализована возможность отправки данных по протоколу WITS сторонним подразделениям (например, экспертам по геолого-технологическим исследованиям и т.д.).

Первая версия программы «GS Terminal» была создана около восьми лет назад. На тот момент архитектура программы проектировалась для работы с существовавшими на тот момент комплексом наземной аппаратуры и сопутствующим ПО. К настоящему моменту элементы комплекса наземной аппаратуры и сопутствующее ПО претерпели существенные изменения с целью увеличения количества функциональных возможностей и улучшения качества собираемых данных. Оригинальная архитектура программы позволяла проводить такие изменения ранее, однако сейчас для упрощения внедрения ряда запланированных модификаций комплекса наземной аппаратуры необходимо внести изменения в архитектуру самой программы «GS Terminal».

Объектом исследования данной работы является программа «GS Terminal» как один из модулей комплекса наземной аппаратуры и ПО системы телеметрического сопровождения процесса бурения.

Целью данной работы является проектирование и реализация программного комплекса для приема данных с различных источников (блок СПД, специализированное ПО и т.д.), их обработки и последующей отправки в другие модули системы. Разрабатываемая программа должна предусматривать возможность быстрого внедрения регулярно возникающих потребностей пользователей и клиентов в изменении функциональности по сбору, обработке и визуальному представлению данных.

Для достижения поставленной цели сформулированы следующие задачи.

1. Определить всю существующую функциональность программы «GS Terminal».
2. Выполнить анализ исходного кода и архитектуры программы «GS Terminal» для выявления факторов, усложняющих внедрение запланированных изменений.
3. Составить и согласовать с руководством производственного центра «Геофит» техническое задание (ТЗ) по разработке новой версии программы «GS Terminal». ТЗ должно включать в себя функциональность, определенную в пункте 1, а также новые потребности заказчика.
4. Выполнить проектирование системы с учётом возможности запланированного расширения функциональности программы, используя диаграммы UML.
5. Спроектировать пользовательский интерфейс в эскизном виде, основываясь на обратной связи, полученной от пользователей.
6. Провести обзор доступных технических средств для разработки и выбрать наиболее подходящее.
7. Реализовать спроектированные программные модули.

Глава 1 Исследование предметной области

1.1 Описание предметной области

Буровая установка – это комплекс буровых машин, механизмов и оборудования, смонтированный на точке бурения и обеспечивающий с помощью бурового инструмента выполнение технологических операций по строительству скважин [2]. В процессе бурения скважины участвует как наземное оборудование, расположенное на буровой установке, так и забойное оборудование, расположенное внутри скважины.

Разрушение породы при вращательном бурении осуществляет долото, расположенное на забое ствола скважины и соединенное с наземным оборудованием с помощью бурильной колонны [2]. Бурильная колонна включает в себя бурильные трубы и компоновку низа бурильной колонны (КНБК). КНБК может иметь в своем составе приборы телеметрии и каротажа, забойный двигатель и т.д. Буровая установка, помимо прочего, содержит буровые насосы (для циркуляции бурового раствора через бурильную колонну), спускоподъемный комплекс (для контроля веса и положения КНБК в скважине), ротор или силовой привод (для вращения бурильной колонны).

Информационное обеспечение процесса бурения включает в себя сбор, обработку, передачу и анализ набора данных о состоянии оборудования и процесса бурения в целом [3]. В частности, для этого на наземное оборудование устанавливаются различные датчики (например, датчики положения талевого блока, давления бурового раствора, скорости вращения буровой колонны, и т.п.). Помимо этого, для сбора данных изнутри скважины без прерывания процесса бурения используются телеметрические системы, передающие информацию от датчиков в КНБК на поверхность. Телеметрическая система состоит из прибора телеметрии, установленного в КНБК, телеметрического канала передачи данных и наземного оборудования, декодирующего телеметрические данные. Одним из типов телеметрического канала является гидравлический канал: оборудование КНБК имеет

возможность вызывать изменение давления бурового раствора, которое фиксируется датчиком на поверхности.

В системе, разработанной в инженерно-производственном центре «Геофит», для сбора информации с наземных датчиков используется «блок СПД» – блок сбора и подготовки данных. Он состоит из модуля контроллера СПД (КСПД) и модулей приема наземных данных (ПНД). Модули ПНД собирают, оцифровывают и передают показания с датчиков наземного оборудования, а также с датчиков телеметрии в КСПД. КСПД передает данные на персональный компьютер (ПК) посредством Ethernet, на котором данные обрабатываются специализированным ПО. Задачу по декодированию телеметрических данных выполняет программа «SibReceiver», которая принимает поток данных по протоколам WITS [27] и WAKE [5] и передает декодированные данные по протоколу WITS. Задачу по приему пакетов данных из различных источников (блок СПД, «SibReceiver», сторонние подразделения) с последующим формированием пакетов данных для пересылки в другие модули системы выполняет программа «GS Terminal». Основным пользователем «GS Terminal» является инженер по телеметрическому сопровождению процесса бурения, однако в программе реализована возможность отправки данных по протоколу WITS/WITSML сторонним подразделениям (например, экспертам по геолого-технологическим исследованиям и т.д.).

1.2 Описание протоколов WITS и WITSML

Одним из протоколов для обмена данными, часто используемых на буровой установке, является протокол WITS [27]. Это многоуровневый протокол. На уровне 0 обмен данными происходит при помощи текстовых сообщений, каждое из которых включает маркер начала, маркер конца и набор элементов. Каждый элемент, в свою очередь, состоит из номера записи (две десятичные цифры), номера элемента записи (две десятичные цифры) и текстового или числового значения. Пара номера записи и номера элемента

записи в дальнейшем будут именоваться «адресом» или «каналом» WITS. Пример сообщения WITS представлен в листинге 1. Набор всех возможных адресов описан в спецификации WITS [27], но встречаются отклонения от нее, поэтому важно иметь возможность настраивать адрес для каждого получаемого или передаваемого вида информации.

Листинг 1 – пример сообщения WITS в текстовом формате. Значения каждого элемента выделены цветами.

```
&&
0101Identifier
01020
0103123
01041
0105230430
0106133712
01070
01081118.12
01101234.56
!!
```

Сообщения WITS могут генерироваться периодически (например, раз в 1 секунду) или по некому событию (например, при возникновении новых данных на датчике). Примером периодически отправляемых данных являются непрерывно измеряемые величины: среднее давление бурового раствора, положение талевого блока и т.п. Примером данных, отправляемых по событию, являются данные КНБК, поступающие по каналу телеметрии.

Для замены достаточно старого протокола WITS был разработан протокол WITSML [33]. Сообщения по этому протоколу закодированы в формате XML. По сравнению с протоколом WITS, номера записей и номера элементов в записи были заменены текстовыми обозначениями, а также была добавлена передача единиц измерения.

1.3 Программа «GS Terminal»

1.3.1 Описание программы «GS Terminal»

Программа «GS Terminal» в ее текущем виде выполняет функции приема и обработки данных от разных источников (наиболее часто используемая конфигурация – блок СПД и программа «SibReceiver»), а также

для передачи данных сторонним подразделениям (например, экспертам по геолого-технологическим исследованиям и т.д.) и другому ПО «Геофит». Диаграмма обмена данными изображена на рисунке 1.В задачи программы входят:

1. прием данных, передаваемых по TCP, от КСПД по двум протоколам:
    a. WITS – текстовый протокол с частотой сообщений порядка 1 Гц и объемом данных около 1 КБ/с;
    b. WAKE – двоичный протокол с частотой сообщений порядка 500 Гц и объемом данных около 50 КБ/с;
2. перенаправление полученных данных в «SibReceiver» по протоколам WITS и WAKE, передаваемых по TCP;
3. прием данных от источника WITS (чаще всего из «SibReceiver») по протоколу WITS;
4. отправка данных по протоколам WITS или WITSML сторонним подразделениям и в иное ПО «Геофит» (чаще всего в «SibReceiver»);
5. автоматическое определение режимов процесса бурения («бурение», «над забоем», «клинья»);
6. расчет значений ряда величин (глубина забоя, глубина долота, скорость проходки и др.) с возможностью их ручной корректировки пользователем либо на основании автоматического определения режимов бурения;
7. отправка рассчитанных значений (упомянутых выше):
    a. по WAKE обратно в КСПД;
    b. по WITS/WITSML в «SibReceiver» и сторонним подразделениям;
8. графическое отображение информации, полученной по протоколу WITS, на веб-странице «Монитор бурильщика».

«Монитор бурильщика» – это компонент программы «GS Terminal», выводящий заданный набор величин пользователю в числовом и графическом виде. Он представляет собой веб-страницу, разработанную с помощью HTML,

CSS и JavaScript, которая принимает данные из «GS Terminal» в текстовом формате WITS по протоколу HTTP. Снимок экрана «монитора бурильщика» представлен на рисунке А.6 в приложении А.

Программа «GS Terminal» посредством непрерывного суммирования значений датчика положения талевого блока и с учетом режимов процесса бурения производит расчет глубины забоя скважины, а также глубины долота (по стволу).

Общая концепция алгоритма автоматического определения режимов бурения основывается на анализе данных, полученных от датчиков веса, давления бурового раствора и талевого блока. В случае наличия давления бурового раствора, веса и опускания талевого блока текущий режим определяется как «бурение», а также происходит изменение глубины долота. В случае если глубина долота достигает значения глубины забоя, значение глубины забоя также начинает увеличиваться. В случае подъема талевого блока и при наличии веса происходит уменьшение глубины долота и режим определяется как «над забоем». В случае отсутствия веса режим определяется как «клинья», и изменение положения талевого блока не влияет на значения глубины долота и забоя. Это может происходить в момент наращивания бурильной колонны, когда она фиксируется в специальном устройстве, называемом «клинья», и отсоединяется от талевого блока. В случае отсутствия данных с датчиков веса либо давления, установка режима «клинья» производится пользователем вручную.

За формирование пакетов на основе данных, полученных из различных источников, и предназначенных для отправки внешним клиентам по протоколу WITS, отвечает компонент программы, далее называемый в этой работе «WITS хаб».

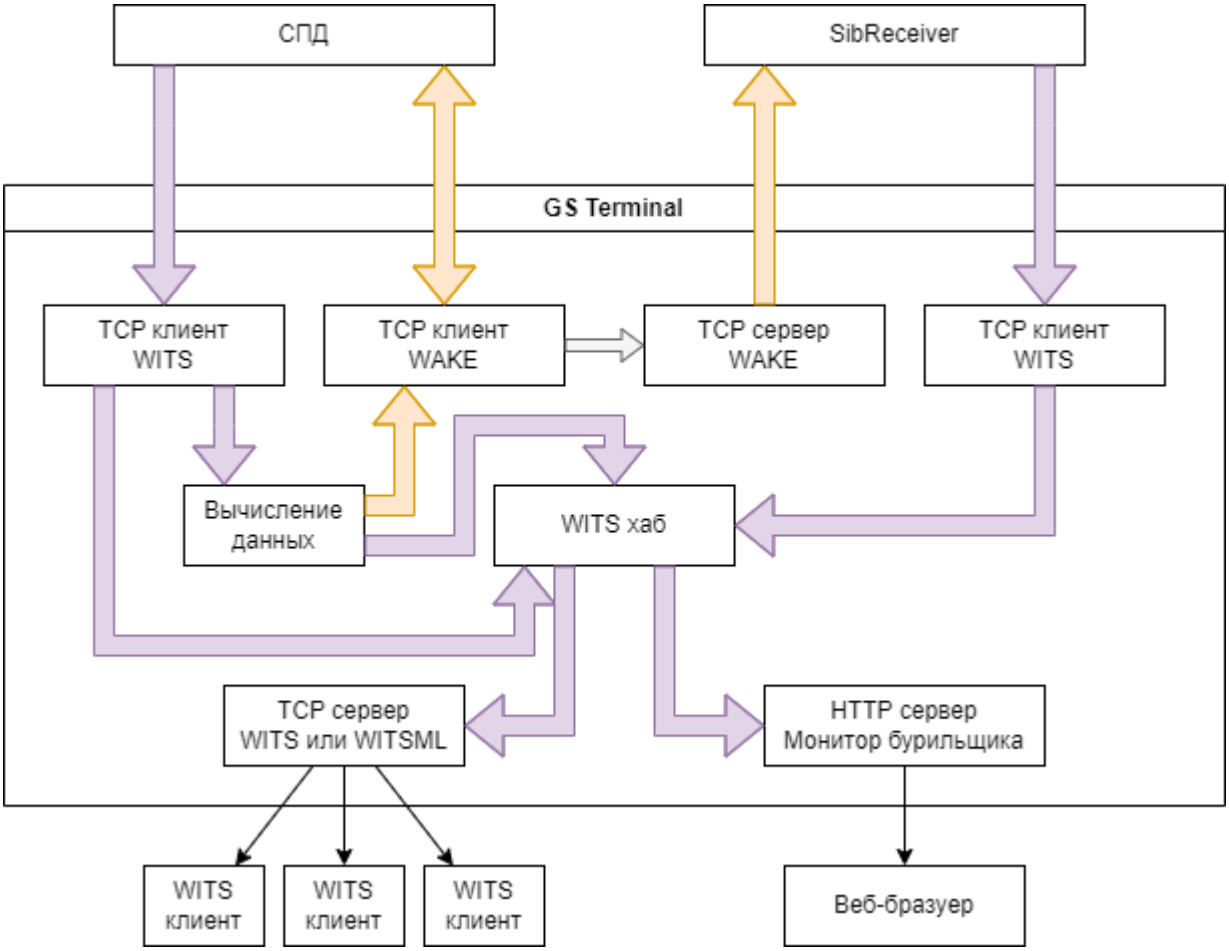


Рисунок 1 – диаграмма обмена данными. Фиолетовым цветом изображены данные по протоколу WITS, оранжевым – по протоколу WAKE.

1.3.1 Элементы пользовательского интерфейса «GS Terminal»

Пользовательский интерфейс программы «GS Terminal» состоит из основного окна с панелью вкладок, отдельного окна конфигурации каналов WITS и веб-страницы «Монитор бурильщика», подробно описанной выше.

Основное окно «GS Terminal» состоит из следующего набора вкладок, отвечающих за различные модули программы.

1. «СПД». Настройки соединения с блоком СПД по протоколу WAKE, а также настройки одного из подключений WITS для приема данных, которое чаще всего используется для подключения к блоку СПД.
2. «SibReceiver». Настройки сервера WAKE для перенаправления данных в программу «SibReceiver», а также настройки второго подключения по

WITS для приема данных, которое чаще всего используется именно для подключения к «SibReceiver».

3. «GeoScan». Данная вкладка отвечает за настройки подключения «GS Terminal» к оборудованию, которое к настоящему моменту является устаревшим и выведено из эксплуатации.
4. «WITS Tx». Данная вкладка предназначена для текстового отображения принимаемых и передаваемых данных по протоколам WITS и WITSML, а также для вызова окна конфигурации WITS.
5. «Вычисления». Данная вкладка предназначена для настройки параметров алгоритма расчета глубины забоя, глубины долота и скорости проходки. Помимо этого, имеется возможность ручной корректировки рассчитанных значений пользователем в случае отсутствия/сбоев датчиков, необходимых для автоматического расчёта.

Снимки экрана графического интерфейса текущей версии доступны в приложении А.

1.3.2 Обратная связь от пользователей

С момента выпуска первой версии программы регулярно собирались и анализировались отзывы пользователей. На их основе был составлен список особенностей текущей версии программы, которые вызывают ряд неудобств со стороны пользователей.

1. Упростить интерфейс программы, удалив вкладки с настройками оборудования, которое выведено из эксплуатации.
2. В ряде случаев пользователям необходимо получать данные с множества источников – блок СПД, «SibReceiver», а также данные, предоставляемыми по протоколу WITS другими подразделениями и организациями, работающими на буровой. Однако в данной реализации допустимо не более двух подключений по WITS для приема данных.
3. В ряде случаев, используя протоколы WITS/WITSML, необходимо передавать различные по структуре пакеты данных разным адресатам.

Однако в данной реализации доступен только один сервер с протоколом или WITS, или WITSML, использующий одну конфигурацию пакетов данных.

4. В программе не предусмотрен выбор источника данных по талевому блоку, давлению и весу, используемых для расчёта величин.
5. Отсутствуют проверки конфигурации WITS/WITSML на ввод пользователем ошибочных значений.
6. Нет возможности разделить данные с одинаковыми адресами, полученные по протоколу WITS от разных источников.
7. «Монитор бурильщика» использует те же пакеты данных, что отправляются по протоколу WITS сервером в «GS Terminal». Таким образом, все данные, выводимые в «мониторе бурильщика», необходимо включать в список данных, отправляемых сервером WITS.
8. «Монитор бурильщика» не поддерживает протокол WITSML. В случае использования данного протокола на сервере, его работа невозможна.

На основе пожеланий пользователей был составлен следующий список пожеланий по расширению функциональных возможностей программы.

1. Поддержка ряда проприетарных протоколов приема/передачи данных.
2. Логирование выбранных данных в исходном виде (как они были получены по TCP/IP) для диагностики и отладки.
3. Возможность вывода получаемых данных в виде временных диаграмм в реальном времени.
4. Перенос интерфейса настройки и калибровки блока СПД из веб-интерфейса КСПД в «GS Terminal» для минимизации действий пользователя.

1.3.3 Анализ исходного кода существующей версии программы

Программа была разработана на языке C# для версий .NET Framework 4.5-4.8. Анализ исходного кода был проведен для каждого класса, представляющего основные элементы пользовательского интерфейса, а также

всех классов, касающихся оборудования и протоколов, все еще используемых на практике. Были составлены диаграмма классов в формате UML, диаграмма вызовов событий (C# events), а также были добавлены новые комментарии к методам, свойствам и полям. Программа не использует сторонние библиотеки и состоит из одного модуля, который компилируется в .exe файл. Для графического интерфейса программы используется в основном фреймворк Windows Forms с элементами фреймворка Windows Presentation Foundation. «Монитор бурильщика» был разработан с помощью HTML, CSS и JavaScript.

На основе проведенного анализа кода программы, и с учетом имеющегося набора планируемых изменений в функциональности программы, был составлен список факторов, усложняющих внедрение запланированных изменений.

1. В ряде мест имеется прямая связь графического интерфейса и бизнес-логики, что затрудняет читаемость и, как следствие, модификацию исходного кода.
2. Блоки try-catch в ряде случаев не всегда содержали в себе уведомление пользователя или иные способы логирования возникшей ошибки, что иногда мешало диагностике проблем с оборудованием.
3. Наличие значительного количества взаимосвязанных объектов, обрабатываемых несколькими потоками, что ведет к существенному увеличению трудозатрат на модификацию кода для удовлетворения пожеланий пользователей.
4. Изначально уместное использование событий (C# events) для взаимодействия между классами в изначальной архитектуре после многократно вносимых модификаций привело к их повсеместному использованию, что привело к значительному усложнению понимания процессов внутри программы.

На основание проведенного функционального анализа текущей версии программы «GS Terminal» были составлены две диаграммы в нотации IDEF0.

На первой (контекстной) диаграмме IDEF0, представленной на рисунке 2, отображена система согласно описанию программы, приведенном в разделе 1.3.1. На второй диаграмме IDEF0, представленной на рисунке 3, отображена декомпозиция контекстной диаграммы с целью показать обмен данными между различными модулями программы в ходе работы. Отметим, что связи между блоками, показанные на второй диаграмме, определены в коде программы и не поддаются изменению пользователем. Более того, обмен данными между блоками осуществляется с использованием многочисленных различных внутреннних структур данных без использования единого стандарта.

Такая ситуация возникла из-за внесения многочисленных изменений в программу после начала эксплуатации программы. Выбор изначальной архитектуры, удовлетворяющей изначальным требованиям, был сделан с целью упрощения кода программы, а дальнейшего расширения функциональности в планах не было. Например, на момент проектирования программы не требовалось принимать данные от более чем одного источника по протоколу WITS. Однако в настоящий момент эта особенность архитектуры затрудняет процесс внедрения дополнительных блоков (например, логирования или проверки данных), так как их реализация должна учитывать внутреннюю структуру всех блоков, с которыми ожидается взаимодействие.

В ходе анализа исходного кода программы вместе с командой разработчиков было принято решение спроектировать новую архитектуру, не обладающую слабыми местами первой версии.

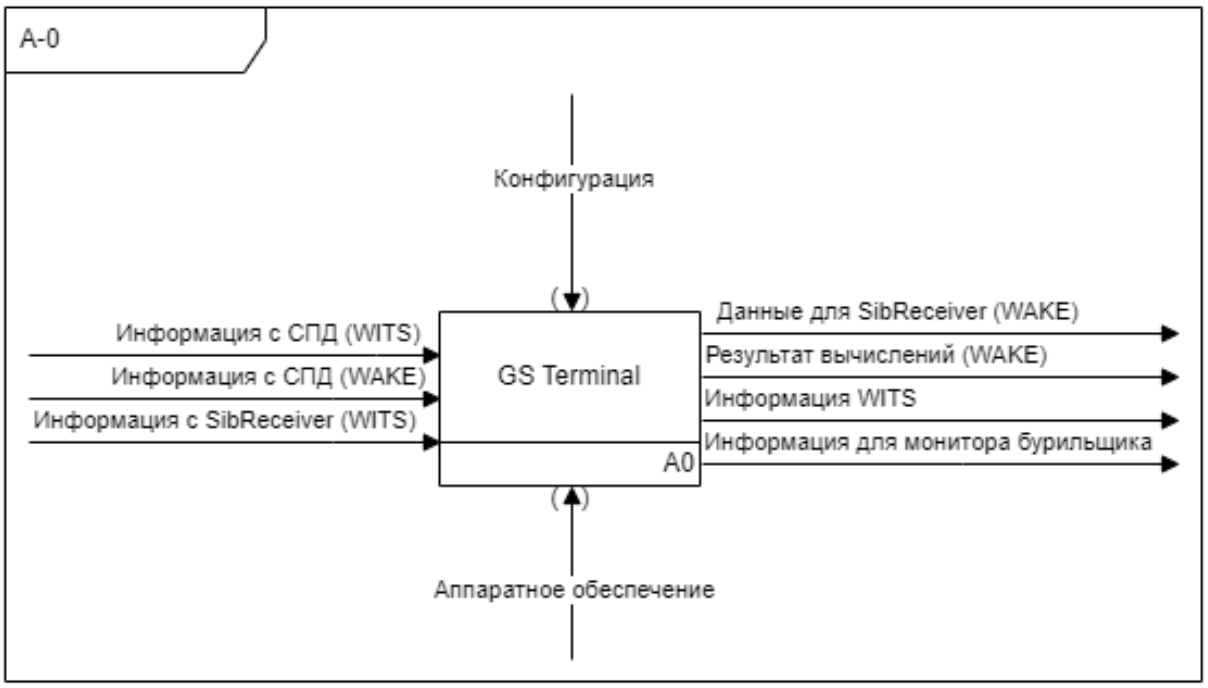


Рисунок 2 – Контекстная диаграмма IDEF0 для существующей версии «GS Terminal»

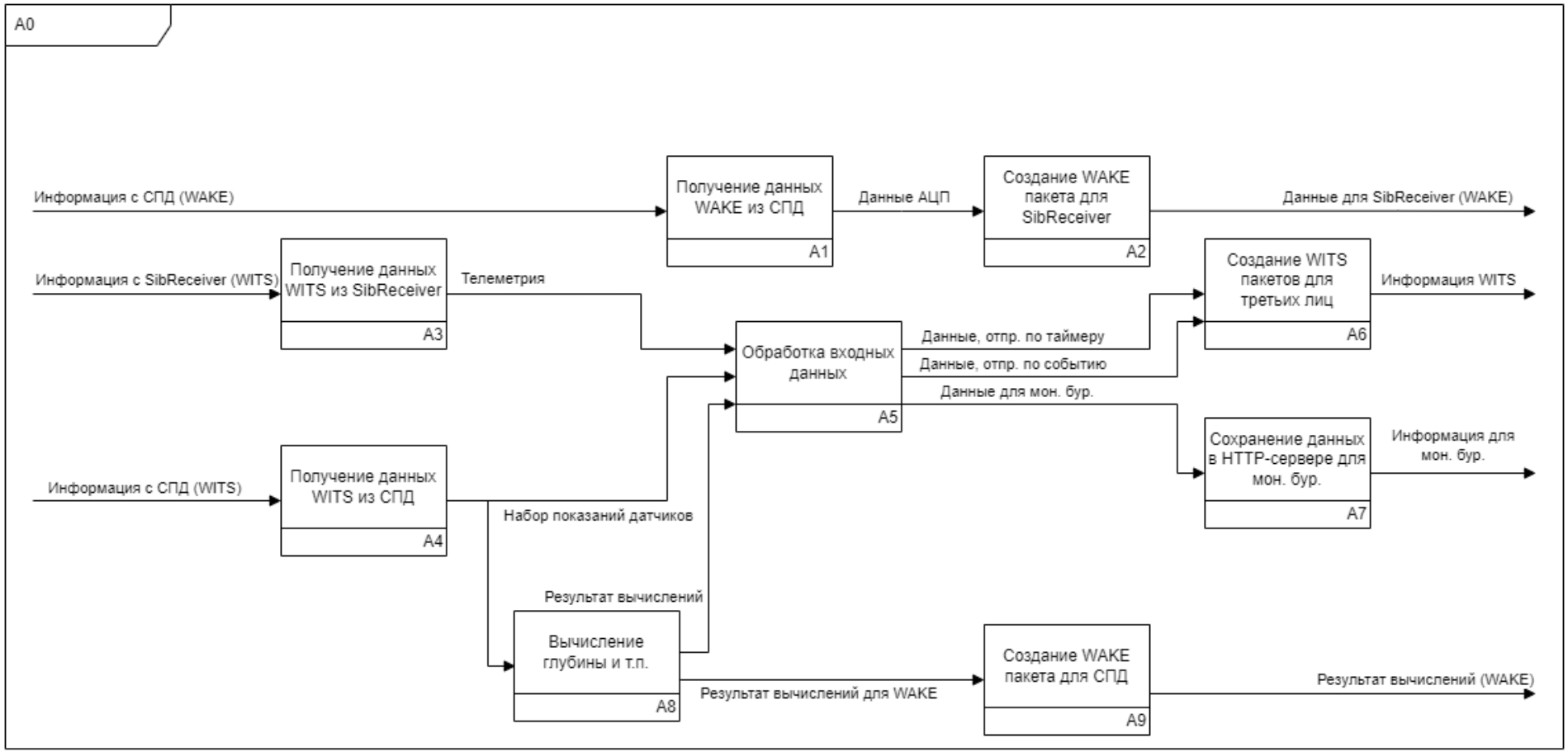


Рисунок 3 – Декомпозиция контекстной диаграммы IDEF0 для существующей версии «GS Terminal»

1.3.4 Внесение изменений в существующую версию программы

В силу регулярно возникающих бизнес-потребностей, помимо работы над проектированием новой архитектуры программы, автором был внесен ряд изменений в существующую версию программы. Изменения включают в себя следующие пункты.

1. Переработан и расширен пользовательский интерфейс конфигурации WITS/WITSML.
    - Добавлен выбор, из какого источника WITS требуется передавать данные по протоколу WITS/WITSML. Это позволяет разделять данные с одинаковыми адресами, получаемым по протоколу WITS от разных источников.
    - Добавлена проверка конфигурации WITS/WITSML на ввод пользователем ошибочных значений перед применением конфигурации.
    - Добавлена настройка адресов WITS, по которым передаются рассчитанные значения глубины и т.д.
    - Добавлена возможность указывать источник данных для «монитора бурильщика».
2. Изменен способ взаимодействия «монитора бурильщика» с «GS Terminal». Ранее «монитор бурильщика» принимал те же данные и в том же формате, что используется для отправки сервером WITS/WITSML. В измененной версии «монитор бурильщика» принимает данные в формате JSON с использованием в качестве ключей имен каналов. Таким образом, была устранена зависимость настройки «монитора бурильщика» от настройки сервера WITS.
3. Добавлено опциональное логирование всех событий изменения конфигурации пользователем и логирование всех передаваемых данных

в целях расширения методов диагностики и отладки комплекса наземной аппаратуры и сопутствующего ПО.

4. Внесены изменения в модуль расчёта значений глубины забоя, глубины долота и скорости проходки.
    - Изменен алгоритм обработки ситуации потери связи с датчиками.
    - Добавлен графический интерфейс для оповещения пользователя о выходе за допустимые пределы значений с датчиков, используемых в расчётах. Пределы значений задаются пользователем в отдельном окне настроек.
    - В графический интерфейс коррекции данных добавлен предпросмотр нового значения перед применением.

Подготовленная новая версия программы была отправлена на тестирование в полевых условиях.

Важно отметить, что в результате работы над внедрением данных изменений были внесены корректировки в проект технического задания разрабатываемой новой версии программы.

1.4 Цель работы и постановка задач

Целью данной работы является проектирование и реализация программного комплекса для приема данных с различных источников (блок СПД, специализированное ПО и т.д.), их обработки и последующей отправки в другие модули системы. Разрабатываемая программа должна предусматривать возможность быстрого внедрения регулярно возникающих потребностей пользователей и клиентов в изменении функциональности по сбору, обработке и визуальному представлению данных.

Для достижения поставленной цели сформулированы следующие задачи.

1. Определить всю существующую функциональность программы «GS Terminal».

2. Выполнить анализ исходного кода и архитектуры программы «GS Terminal» для выявления факторов, усложняющих внедрение запланированных изменений.
3. Составить и согласовать с руководством производственного центра «Геофит» техническое задание (ТЗ) по разработке новой версии программы «GS Terminal». ТЗ должно включать в себя функциональность, определенную в пункте 1, а также новые потребности заказчика.
4. Выполнить проектирование системы с учётом возможности запланированного расширения функциональности программы, используя диаграммы UML.
5. Спроектировать пользовательский интерфейс в эскизном виде, основываясь на обратной связи, полученной от пользователей.
6. Провести обзор доступных технических средств для разработки и выбрать наиболее подходящее.
7. Реализовать спроектированные программные модули.

Глава 2 Проектирование программной системы

2.1 Требования к системе

Требования к системе представлены в виде технического задания в приложении Б. Техническое задание было составлено на основе отзывов пользователей, описанных ранее, совещания с менеджером проекта телеметрических систем, руководством и сотрудниками отдела разработок, а также опыта автора, полученного в процессе анализа кода и внедрения изменений в существующую версию программы.

2.2 Видение системы

На рисунке 4 представлена упрощенная версия диаграммы IDEF0 исходной версии программы «GS Terminal». На данной диаграмме модули программы, посредством которых осуществляется обмен данными с конкретными внешними системами (оборудованием либо ПО), представлены функциональными блоками, реализующими протоколы обмена данными, применяемые в этих системах, или же реализующими алгоритмы по обработке данных.

Стоит отметить, что блоки «TCP клиент WITS СПД» и «TCP клиент WITS SibReceiver» обладают идентичной функциональностью: принимают поток данных по TCP и декодируют сообщения WITS, однако реализованы посредством разных классов с разной внутренней структурой. В частности, стоит отметить, что только блок «TCP клиент WITS СПД» из-за особенностей реализации его класса может передавать данные в блок «вычисление данных». Таким образом, представляется уместным создание для всех блоков, реализующих прием по протоколу WITS, общего класса с гибкой возможностью конфигурации параметров подключения. Такой подход также решает ранее озвученную задачу о расширении допустимого количества приемников данных по протоколу WITS. Диаграмма с объединенными блоками представлена на рисунке 5.

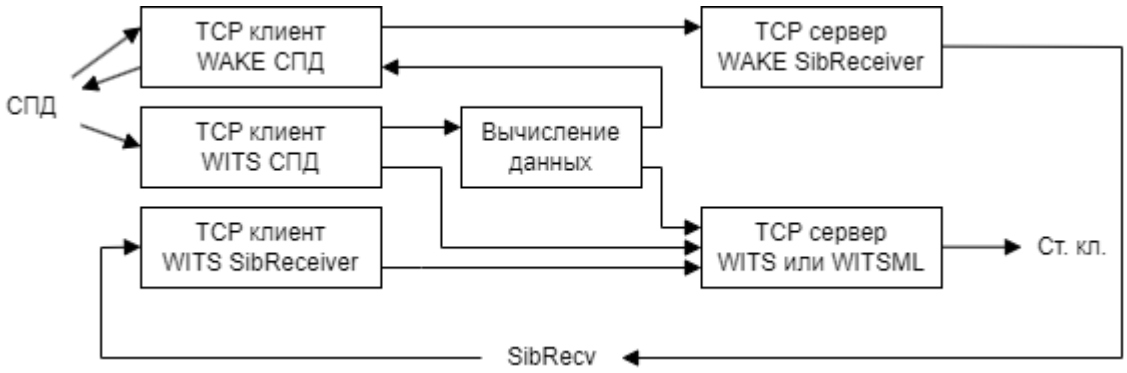


Рисунок 4 – Упрощенная диаграмма модулей «GS Terminal»

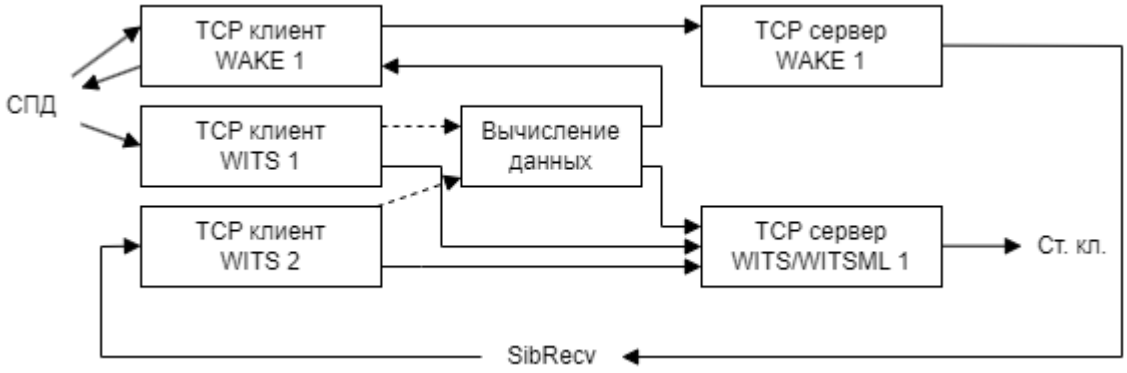


Рисунок 5 – Упрощенная диаграмма модулей «GS Terminal» с объединенными блоками

Однако, диаграмма на рисунке 5 не дает четкого понимания, какой из блоков приема данных WITS будет являться источником данных для блока вычислений. Следовательно, требуется разработать некую систему конфигурации связей между блоками, которая позволит в данном случае выбирать источник данных для вычислений. Для четкого определения вида связи между блоками необходимо ввести понятие «канал данных» – данные определенного типа, получаемые или передаваемые определенным блоком (например, азимут принятый от «SibReceiver», азимут для отправки через сервер WITS, давление от блока СПД, глубина забоя от блока СПД, глубина забоя от блока вычислений и т.п.). Каналы данных могут быть входными (получают данные из другого канала другого блока) или выходными (передают данные в другой канал другого блока). Очевидно, что для возможности однозначного определения источника входных данных, входной

канал может быть связан не более, чем с одним выходным каналом, но из одного выходного канала данные могут передаваться нескольким входным. Дополненная каналами диаграмма показана на рисунке 6.

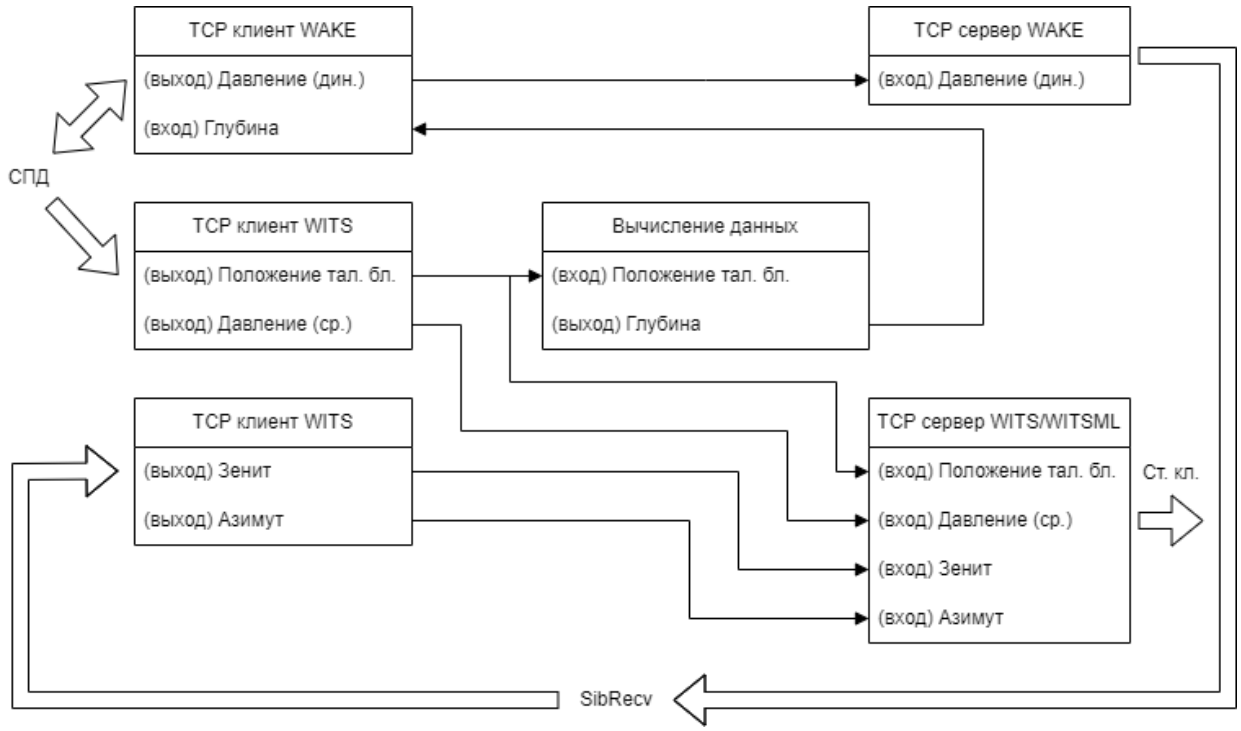


Рисунок 6 – Упрощенная диаграмма модулей «GS Terminal» с каналами

Набор каналов каждого блока зависит от его типа и может задаваться либо на этапе проектирования, либо пользователем во время работы программы. Некоторые блоки (например, для обмена по протоколу WITS) могут предоставлять пользовательский интерфейс для настройки каналов, а другие блоки (например, блок вычислений) имеют набор каналов, предопределенный в коде программы. Система конфигурации для каждого входного канала будет хранить информацию о его связи с выходным каналом другого блока. Для хранения конфигурации достаточно использования текстового файла (например, формата XML).

Блоки называются связанными, если у одного из блоков есть хотя бы один входной (выходной) канал, связанный с одним из выходных (входных) каналов другого блока. Для реализации обмена данными между связанными блоками возможны два подхода:

1. инициация обмена осуществляется блоком с входными каналами (англ. «pull»): блок с входными каналами при необходимости запрашивает последние данные у всех блоков, связанных с ним.
2. инициация обмена осуществляется блоком с выходными каналами (англ. «push»): при изменении значений выходных каналов в одном из блоков, он передает новые значения всем блокам, связанным с ним;

Подход «pull» может иметь несколько реализаций. Например, блок в случае необходимости использования данных запрашивает их у связанных с ним блоков. Каждый из этих связанных блоков, в свою очередь, может запросить данные у блоков, связанных с ним, что приводит к рекурсивному опросу большого количества блоков, даже если значения их выходных каналов не изменились. Это вызывает лишнюю нагрузку на систему.

В иной реализации подхода «pull» каждый блок с выходными каналами имеет внутри себя буфер, из которого любой связанный блок с входными каналами может запросить данные в любой момент времени, причем алгоритм блока, вычисляющий значения выходного буфера, не исполняется в момент запроса (иначе рассматриваемая реализация идентична предыдущей). Рассмотрим такой подход на следующем примере. Имеется блок с двумя входными каналами и одним выходным каналом, значением которого является сумма входных каналов. Очевидно, для запуска алгоритма обновления выходного буфера должен существовать некоторый триггер. Таким триггером может являться таймер, который будет с определенной периодичностью суммировать значения входных каналов и сохранять результат в выходной буфер. Однако, если временной интервал таймера больше интервала фактического обновления значений входных каналов, то часть информации будет утеряна или обработана с задержкой. Если временной интервал таймера меньше, то программа будет тратить процессорное время впустую на обновление неизменившихся данных. Этот подход вполне уместен в случаях, когда данные используются значительно реже, чем обновляются, и не

критично потерять информацию обо всех изменениях значений между запросами и самом времени возникновения изменения. Однако, в проектируемой системе требуется быстрая реакция на изменения значений из разных источников (датчиков, ПО и т.д.), и следовательно, данный подход не оптимален.

Второй подход «push» реализуется следующим образом. При изменении значений выходных каналов, блок передает изменившиеся значения всем связанным с ним блокам. Источником изменения может быть прием данных по сети, прием изменившихся данных от связанного блока или какое-то иное событие, специфичное для конкретного блока. В такой реализации связанные блоки оповещаются о каждом изменении почти в тот же момент, как они произошли, а значит вносимая задержка в обработку минимальна, и никакие данные не будут потеряны. Однако стоит отметить, что в случае, когда данные обновляются часто, но при этом используются в связанных с ними блоках значительно реже, происходит избыточная трата ресурсов компьютера. В проектируемой системе на текущий момент такая ситуация представляется маловероятной в силу того, что в большинстве случаев требуется быстрая реакция на события. Из рассмотренных подходов именно подход «push» более уместен в разрабатываемой системе с учетом предъявляемых к ней требований.

Передача данных между блоками будет осуществляться с использованием компонента, именуемого далее в работе «маршрутизатор данных». Каждый блок с выходными каналами может сгенерировать пакет данных, содержащий значения выходных каналов. Этот пакет данных передается в маршрутизатор данных. Маршрутизатор данных хранит в себе связи между каналами всех блоков, принимает пакеты данных от блоков, на их основе создает новые пакеты данных и передает их блокам, связанным с исходным. Ввод маршрутизатора данных позволяет каждому отдельному блоку не обменивается данными напрямую с другими, что убирает

зависимость между различными блоками и упрощает понимание исходного кода.

Нельзя исключать, что в будущем может потребоваться также реализация подхода «pull» для внедрения новой функциональности по требованию клиентов. В таком случае система может быть расширена без необходимости полной переработки путем добавления буфера данных в маршрутизатор для блоков, не реализующих подход «pull». Буфер данных будет обновляться при получении маршрутизатором пакета данных от блока. При запросе данных из блока, маршрутизатор вернет значения буфера.

Использование маршрутизатора данных также позволяет создавать блоки, не только связанные с приемом или передачей данных или вычислениями. Например, могут быть созданы блоки, проверяющие корректность проходящих через них данных; блоки, логирующие данные в файл или выводящие их на график. При всем этом, они будут независимы от конкретных протоколов обмена данными.

Так как слово «блок» без контекста не передает четкий смысл, в пользовательском интерфейсе блоки будут именоваться «терминалами». Пользовательский интерфейс программы для настройки и отображения связей между терминалами может быть реализован в виде ER-диаграммы. Так как у одного терминала может быть большое число каналов (более 30), связи между конкретными каналами могут отображаться или скрываться. Дополнительно пользователю предоставляется возможность просмотреть все связи одного выбранного канала, а также перейти в меню настройки терминала.

2.3 Функциональный анализ проектируемой системы

На рисунке 7 представлена контекстная диаграмма проектируемой системы в целом в нотации IDEF0. Проектируемая система эквивалентна в плане входов и выходов исходной системе, представленной на рисунке 2. На рисунке 8 изображена декомпозиция контекстной диаграммы IDEF0. Из диаграммы видно, что система состоит из блоков, отвечающих за прием,

передачу и обработку данных, как было описано в предыдущем разделе. Обмен данными происходит не напрямую между блоками, а через маршрутизатор данных, что является характерным отличием от предыдущей версии системы. Применение маршрутизатора данных позволяет конфигурировать связи между блоками конечному пользователю через соответствующий интерфейс программы. По сравнению с предыдущей версией программы, это делает настройку значительно более гибкой.

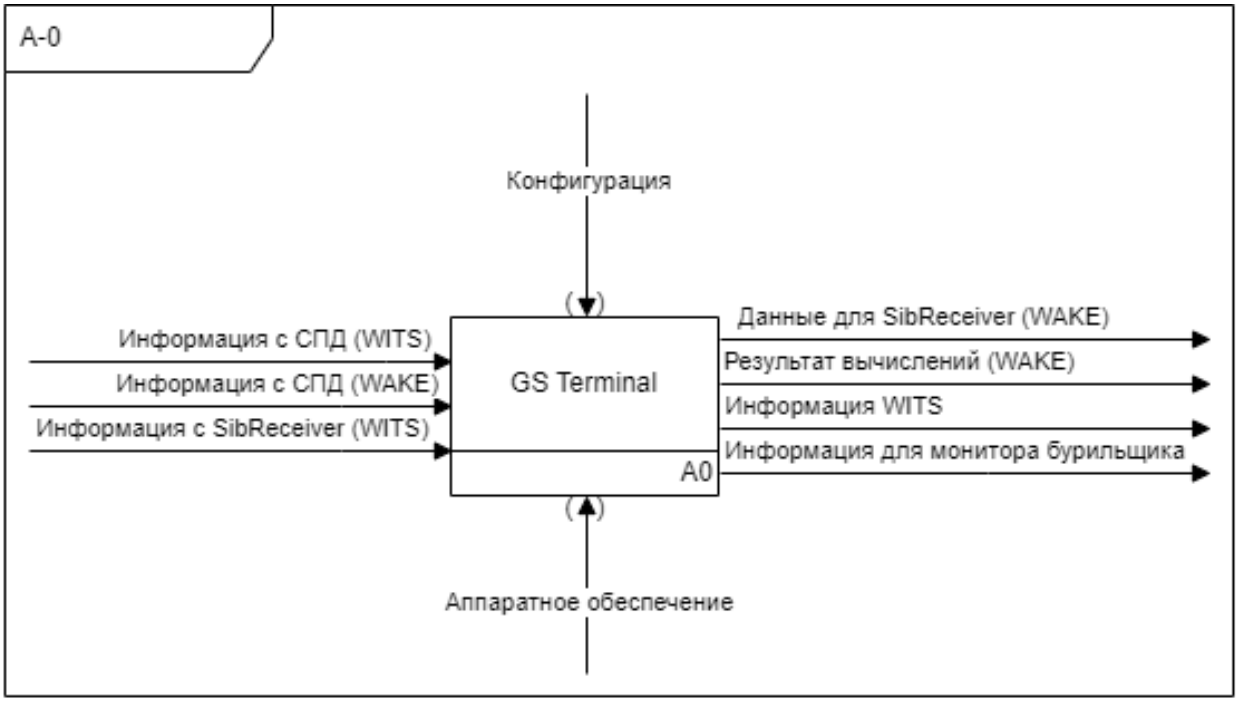


Рисунок 7 – Контекстная диаграмма IDEF0 для проектируемой системы

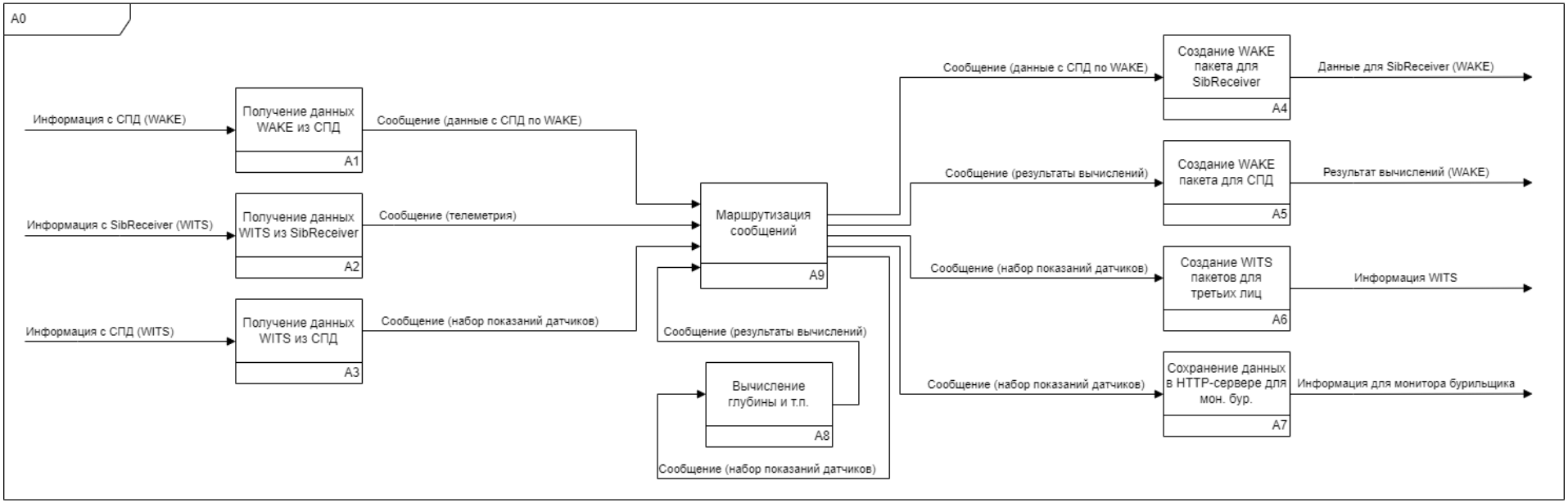


Рисунок 8 – Декомпозиция контекстной диаграммы IDEF0 для проектируемой системы

2.4 Обзор компонентов системы

В системе можно выделить следующие основные компоненты.

1. DataRouter – компонент, содержащий класс маршрутизатора данных. В нем также определены интерфейсы, реализуемые терминалами.
2. DataRouter.Factory – компонент, хранящий в себе список экземпляров фабрик отдельных типов терминалов. В нем определен интерфейс *INodeFactory*, который позволяет создавать экземпляры терминалов на основе конфигурационного файла. *INodeFactory* реализуется в компонентах отдельных терминалов.
3. DataRouter.Host – компонент, отвечающий за чтение конфигурационного файла; создание экземпляров терминалов, используя их фабрики; настройку маршрутизатора данных согласно конфигурационному файлу.

Определим следующие компоненты для работы с различными протоколами данных.

1. Net – компонент, реализующий классы, связанные с приемом и передачей данных по различным интерфейсам от внешних систем (TCP/IP, последовательный порт и т.д.).
2. Net.Wits – классы для работы с пакетами в формате WITS.
3. Net.Wake – классы для работы с пакетами в формате WAKE.

Определим компоненты, реализующие конкретные терминалы системы и фабрики для создания их экземпляров.

1. DataRouter.Math – терминалы для проведения математических операций.
2. DataRouter.Wits – терминал для приема и передачи данных по протоколу WITS.
3. DataRouter.Wake – терминал для приема и передачи данных по протоколу WAKE.

Компонент GSTerminal является основным исполняемым файлом и реализует графический интерфейс, с которым взаимодействует пользователь. Отметим, что код пользовательского интерфейса каждого конкретного

терминала реализовывается в компоненте конкретного терминала, а компонент GSTerminal только выводит его на экран.

Диаграмма компонентов и диаграмма зависимостей между основными компонентами системы представлены на рисунке 9 и рисунке 10, соответственно.

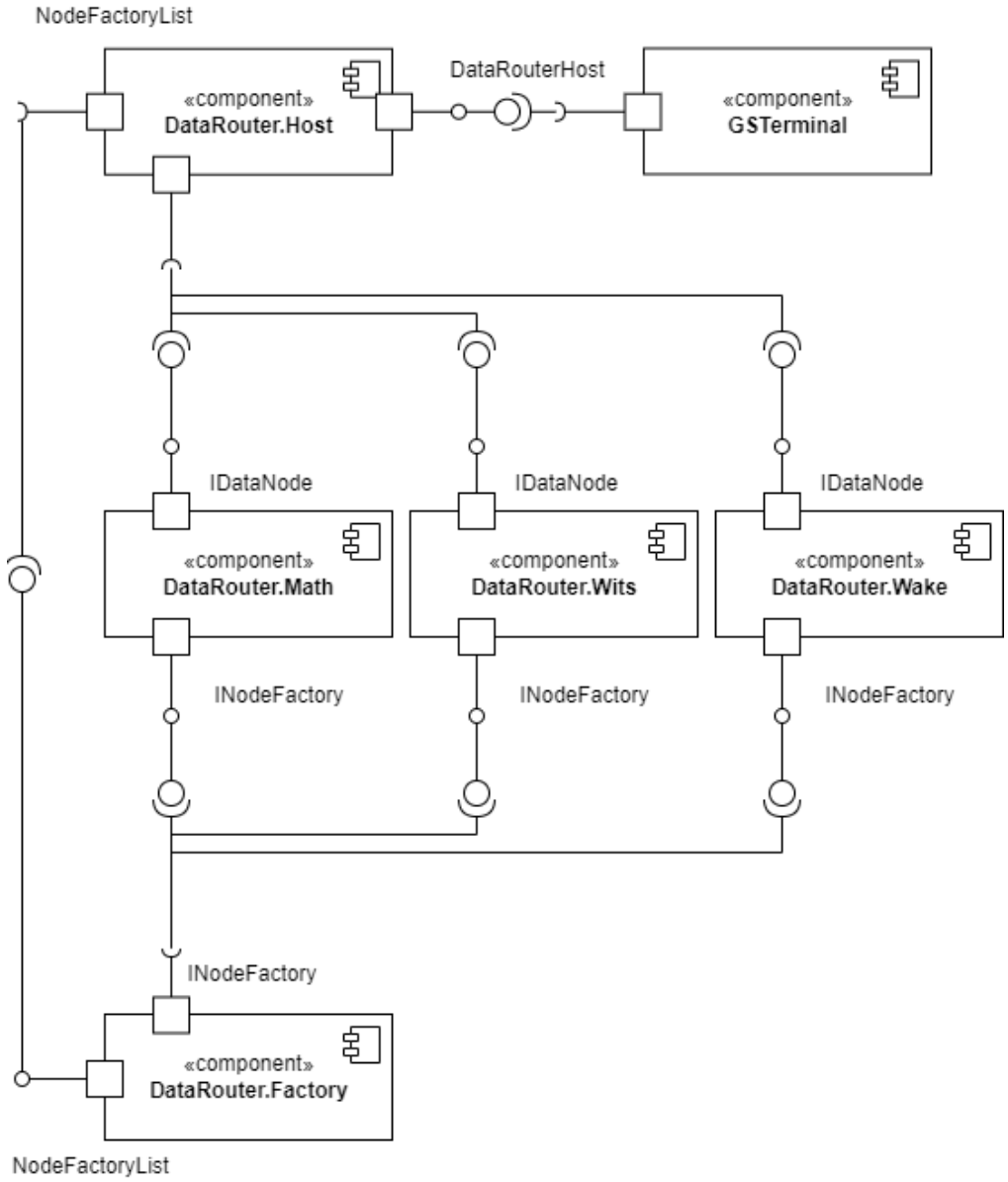


Рисунок 9 – Диаграмма компонентов

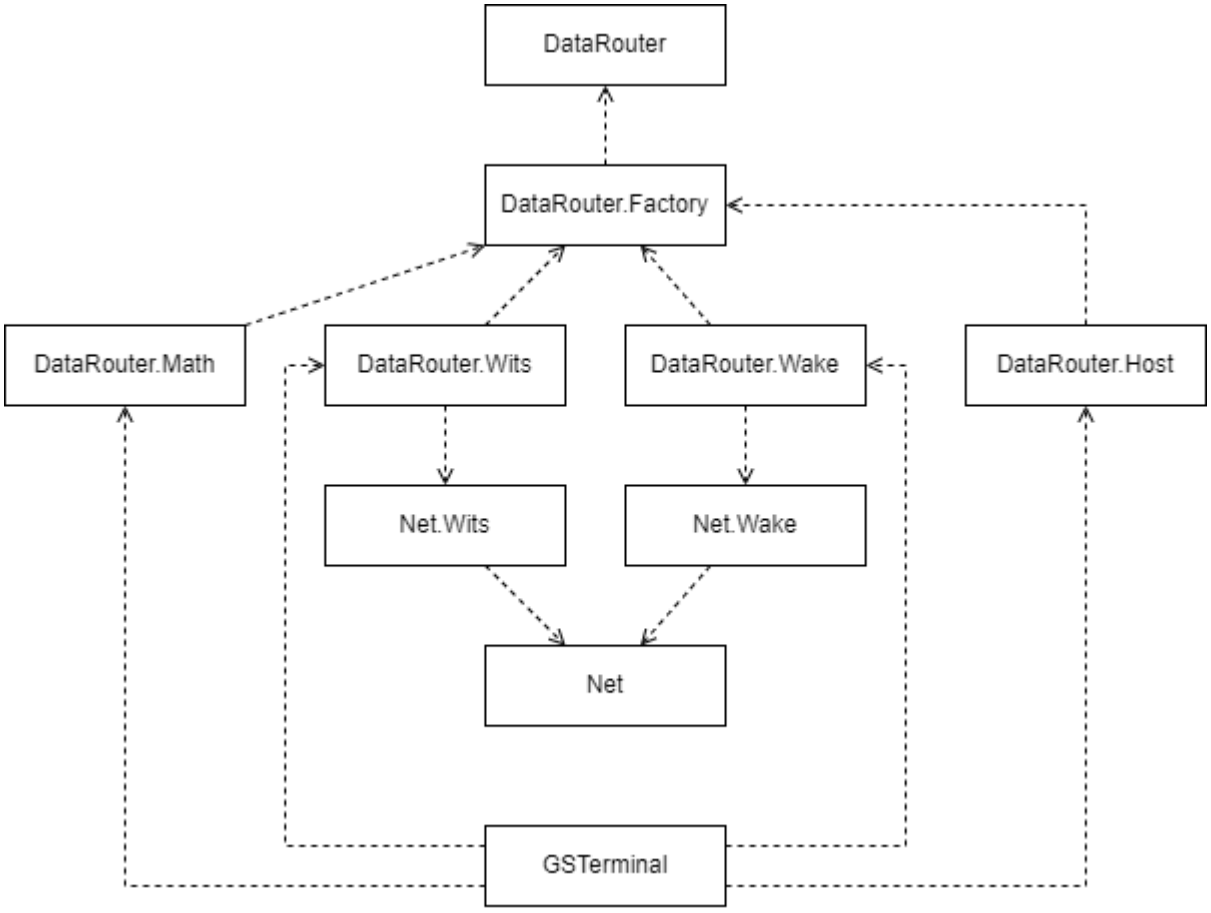


Рисунок 10 – Диаграмма зависимостей

2.5 Маршрутизатор данных

2.5.1 Компонент DataRouter

На рисунках 11-12 представлена диаграмма классов компонента *DataRouter*. Этот компонент описывает интерфейсы терминалов:

- *IDataNode* – общий интерфейс для всех терминалов;
- *IInputNode* – интерфейс для терминалов, имеющих входные каналы;
- *IOutputNode* – интерфейс для терминалов, имеющих выходные каналы.

Интерфейсы терминалов предоставляют методы для получения базовой информации о терминале, регистрации в определенном экземпляре маршрутизатора данных и обновления конфигурации. Метод *LoadConfig* позволяет уже после создания всех терминалов обновить конфигурацию без необходимости уничтожения старых экземпляров, что позволяет избежать закрытия сетевых соединений и возможной потери данных. Однако, не всегда

возможно использовать старый экземпляр для загрузки новой конфигурации (например, в новой конфигурации используется соединение по COM-порту вместо TCP-клиента). Для проверки возможности загрузки конфигурации в существующий экземпляр используется метод *CanLoadConfig*.

Основным классом является класс *DataRouter*, который хранит в себе связи между терминалами и их каналами и реализует логику маршрутизации пакетов. Пакеты представляет класс *DataPacket*. Он содержит время создания пакета и набор пар, где ключом выступает идентификатор канала, а значением – значение канала в формате числа с плавающей точкой двойной точности.

Для идентификации каналов используются классы *PortId, InputPortId* и *OutputPortId*. Они все содержат в себе целое число, которое идентифицирует канал в одном терминале. Использование различных типов с явными преобразованиями между ними позволяет отловить ошибки, когда идентификатор входного канала используется заместо идентификатора выходного канала на этапе компиляции программы, а не во время тестирования.

Дополнительно в этом компоненте определены классы *BaseNodeProperties* и *BasePortProperties*. Они используются не в самом компоненте, а в компонентах, реализующих терминалы. Эти классы хранят общую информацию о терминале и канале, соответственно, и предоставляют методы для ее загрузки и сохранения из/в конфигурационного файла.

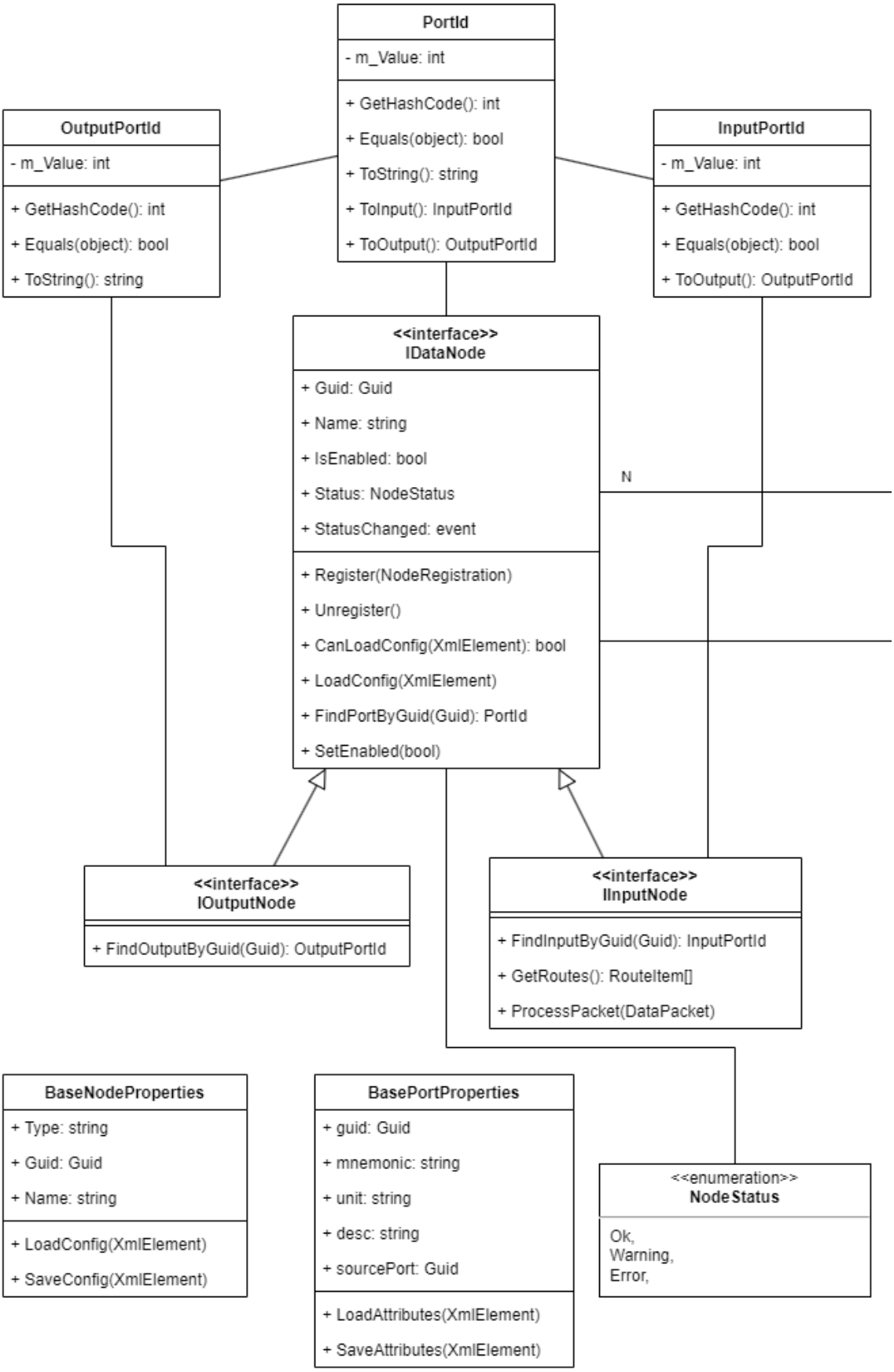


Рисунок 11 – Диаграмма классов компонента DataRouter (часть 1/2)

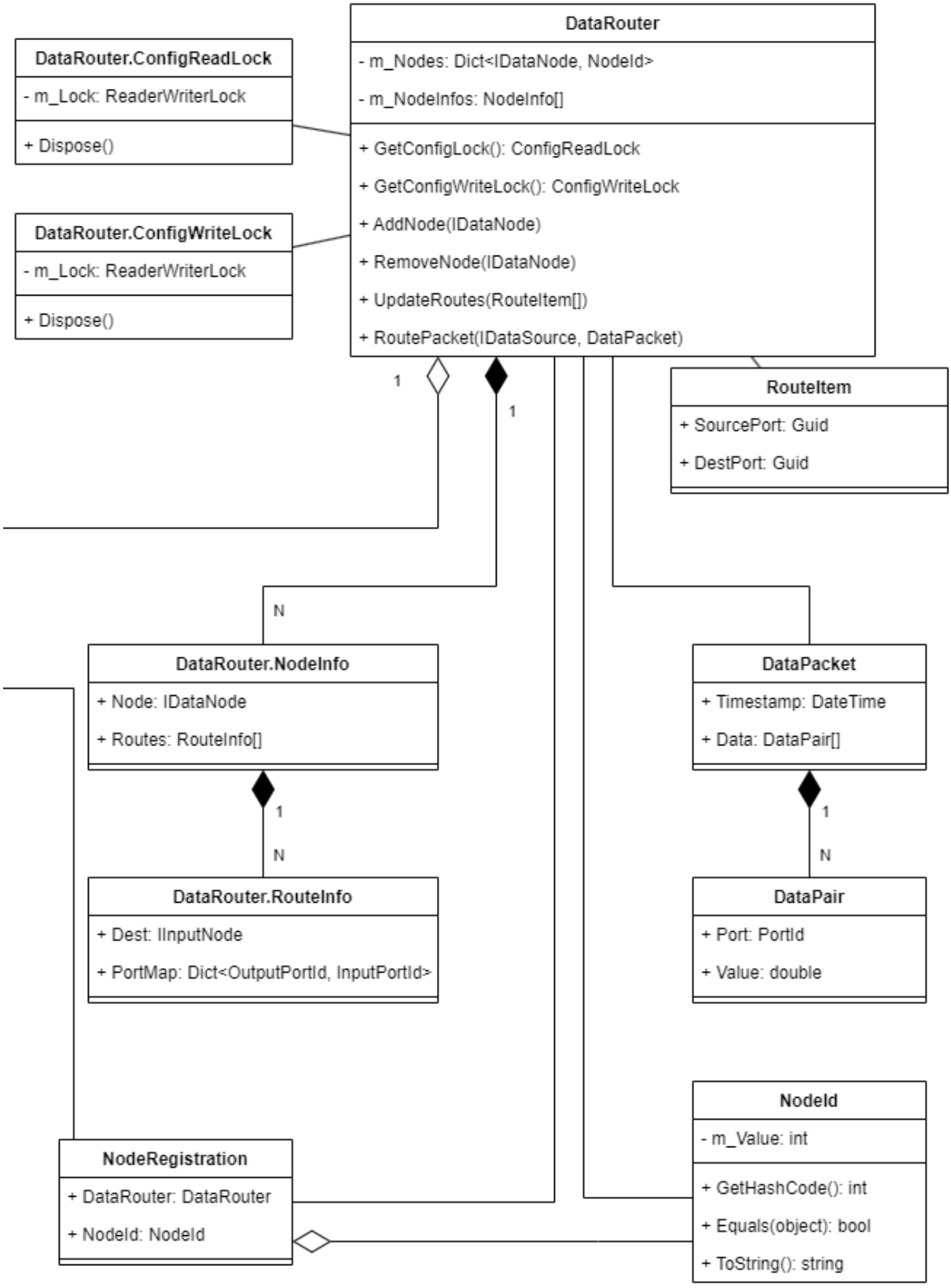


Рисунок 12 – Диаграмма классов компонента DataRouter (часть 2/2)

2.5.2 Синхронизация потоков

В зависимости от реализации конкретного терминала, допустимо создание потоков внутри терминала (например, для приема или передачи данных), поэтому методы обработки пакетов в маршрутизаторе данных должны быть потокобезопасными. Также требуется защитить терминалы от попыток обработки данных во время применения новой конфигурации. Для этого используется блокировка чтения-записи (readers-writer lock или shared mutex, в зависимости от языка программирования). Этот механизм синхронизации позволяет множеству потоков взять разделяемое владение над ресурсом, но только один поток может взять эксклюзивное владение [23].

Объект маршрутизатора данных содержит в себе экземпляр объекта блокировки чтения-записи. Когда терминал получает сигнал на обработку данных (например, сигнал от таймера или сигнал от сетевого сокета о получении данных), он в первую очередь берет блокировку маршрутизатора данных на чтение. Только затем он может считывать свою конфигурацию и рассылать пакеты. Блокировка на чтение предоставляется методом *GetConfigLock*. Когда требуется изменить конфигурацию маршрутизатора данных или отдельных терминалов, объект, отвечающий за изменение конфигурацию, берет блокировку на запись, используя метод *GetConfigWriteLock*.

Данное решение позволяет избежать гонки данных и потери пакетов при обновлении конфигурации.

2.5.3 Создание экземпляров терминалов

Для создания экземпляров терминалов используется шаблон проектирования «фабрика» [1]. Компонент DataRouter.Factory определяет интерфейс *INodeFactory*. Компоненты, содержащие в себе классы терминалов, должны содержать класс, реализующий этот интерфейс. Интерфейс предоставляет методы для создания экземпляра терминала на основе файла

конфигурации, а также модели, модели представления и представления для графического интерфейса программы.

Класс *NodeFactoryList* содержит в себе статический метод для получения экземпляра фабрики по названию типа терминала. Конкретная реализация этого метода будет зависеть от используемого языка программирования. Этот класс при необходимости может быть расширен методами для загрузки фабрик из внешних библиотек.

Диаграмма классов представлена на рисунке 13.

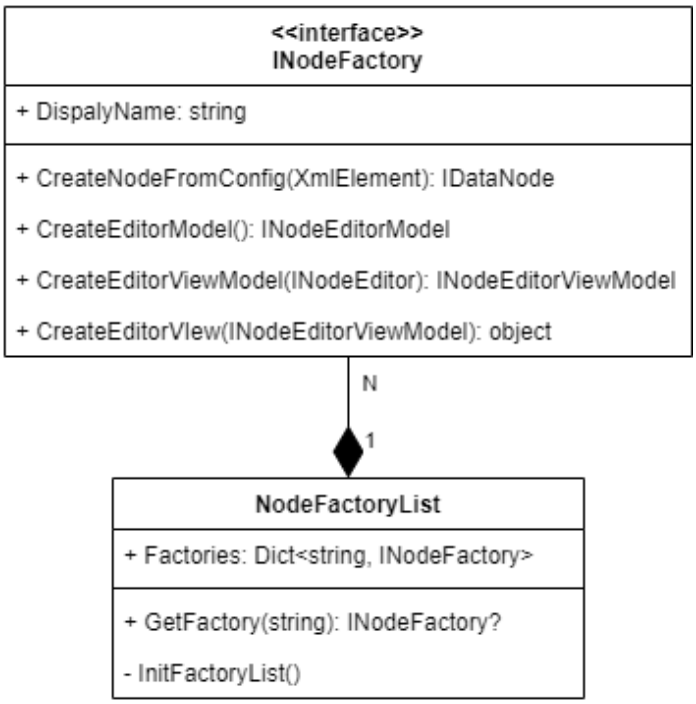


Рисунок 13 – Диаграмма классов компонента DataRouter.Factory

2.5.4 Инициализация маршрутизатора данных

За загрузку файла конфигурации и создание экземпляров терминалов (используя фабрики, описанные в предыдущем пункте) отвечает компонент DataRouter.Host. Он обладает следующей функциональностью:

1. создание экземпляров терминалов, описанных в файле конфигурации;
2. обновление конфигурации в процессе работы без обязательного пересоздания терминалов;
3. включение и отключение системы без удаления терминалов;

4. оповещение графического интерфейса о прогрессе операций.

Эта функциональность реализована в классе *DataRouterHost*. Он содержит методы для загрузки конфигурации, сброса состояния, запуска и остановки системы. Все эти методы являются блокирующими, поэтому должны вызываться из отдельного потока. Для оповещения интерфейса о текущем прогрессе они принимают функтор, который вызывается при каждом шаге операции. Функтор получает объект типа *ProgressEventArgs*, содержащий текущий этап операции и имя терминала, участвующего в текущем этапе. Диаграмма классов этого компонента представлена на рисунке 14.

Диаграмма последовательности инициализации системы представлена на рисунке 15. По команде пользователя вызывается метод *LoadConfig* у *DataRouterHost*. Он берет блокировку конфигурации, описанную в разделе 2.5.2, находит фабрики и вызывает метод *CreateNodeFromConfig* для каждого терминала в файле конфигурации. Фабрики, в свою очередь, создают экземпляры терминалов, загружает им конфигурацию и возвращают терминалы в виде интерфейса *IDataNode*. Затем *DataRouterHost* добавляет созданные терминалы в маршрутизатор данных, получает из них список маршрутов (связей между терминалов) и обновляет маршруты в маршрутизаторе данных. Затем вызывается метод *Start*, который включает все терминалы.

Во время исполнения этой последовательности могут возникнуть ошибки в методах *CreateNodeFromConfig, LoadConfig, SetEnable* интерфейса *INodeFactory,* а также *DataRouter.UpdateRoutes*. Все эти ошибки будут вызваны или ошибкой конфигурации, или ошибкой в логике программы. Так как конфигурация настраивается исключительно в графическом интерфейсе программы, ошибки конфигурации могут быть вызваны только отсутствием проверки ввода пользователя, что также является ошибкой логики программы. Поэтому наиболее подходящим способом обработки ошибок на этом этапе

является полный сброс объекта *DataRouterHost* (вызовом метода *Reset*) с уничтожением всех терминалов, исправление конфигурации пользователем и повторная инициализация.

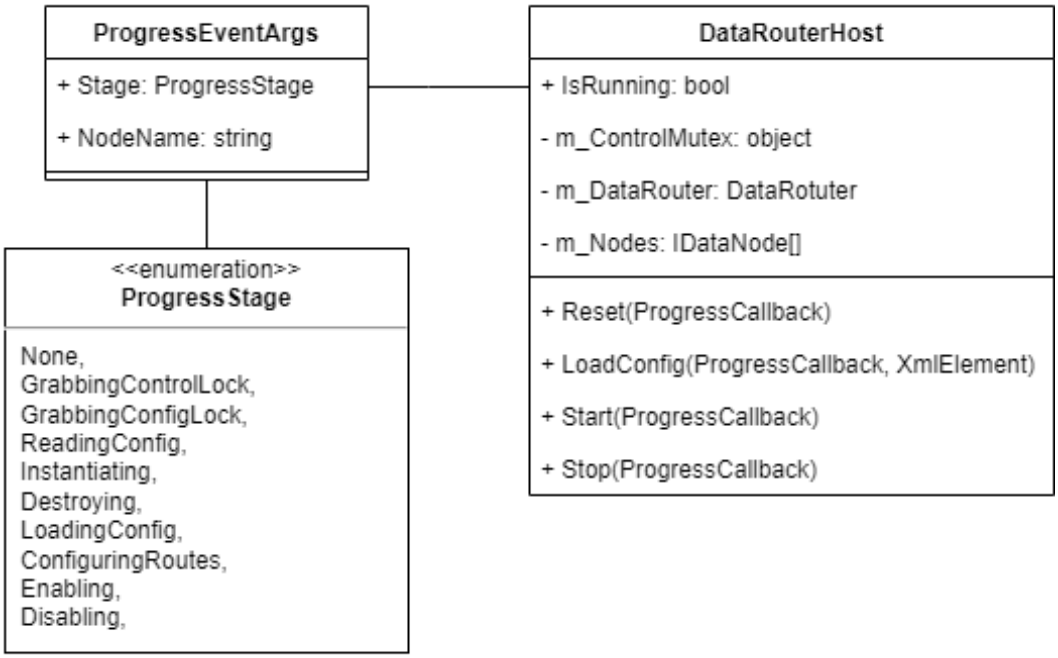


Рисунок 14 – Диаграмма классов компонента DataRouter.Host

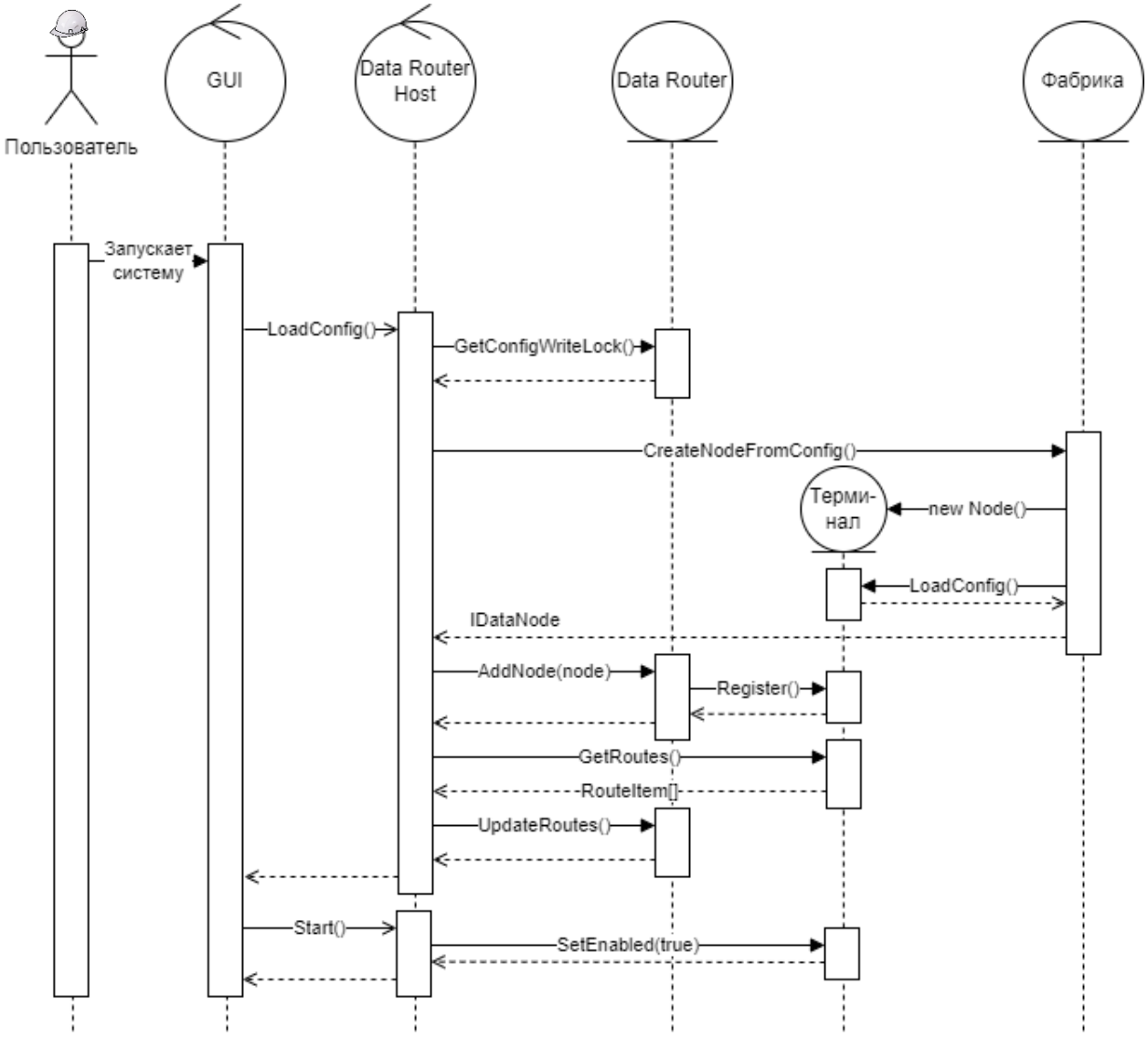


Рисунок 15 – Диаграмма последовательности инициализации

2.5.5 Статус терминалов

Во время работы системы могут происходить ошибки, которые невозможно определить на этапе конфигурирования: потеря связи с удаленным хостом, получены данные некорректного формате и т.п. Такие ошибки не должны останавливать работу системы или препятствовать инициализации. Вместо этого каждый терминал (*IDataNode*) имеет статус (свойство *Status*): ОК, предупреждение, ошибка. При изменении статуса по какой-либо он вызывает событие *StatusChanged* и пишет сообщение в лог. Это событие отлавливает *DataRouterHost* и передает его в графический интерфейс,

который может вывести окно с ошибкой, воспроизвести звук или привлечь внимание пользователя иным способом.

2.6 Связь со внешними устройствами

2.6.1 Абстрактный канал связи

Реализация обмена данными по любому высокоуровневому протоколу связи (WITS, WAKE и т.д.) требует некий объект, который принимает и отдает набор байт. Для TCP/IP таким объектом является сокет, но в требованиях к системе также требуется подключение по COM-порту. Чтобы избежать необходимости делать несколько реализаций одних и тех же протоколов, но с разным типом соединения, в компоненте Net определен интерфейс *INetChannel*. Он подобен TCP/IP сокету тем, что позволяет отсылать или принимать данные, но он также обладает следующими свойствами:

1. полностью абстрагирует особенности конкретных типов соединения;
2. использует события для приема данных; пользователю *INetChannel* не требуется создавать поток специально для приема данных;
3. при потере соединения он автоматически пытаться восстановить соединение.

Для *INetChannel* доступно три реализации для разных типов соединения:

1. клиент TCP (*TcpClientNetChannel*);
2. сервер TCP (*TcpServerNetChannel*);
3. COM-порт (*SerialNetChannel*).

Некоторые типы соединения (а именно, TCP-сервер) позволяют устанавливать несколько соединений. Поэтому *INetChannel* имеет свойство с числом клиентов и события, вызываемые при подключении или отключении клиентов. Типы соединения, которые подключаются только к одному удаленному хосту, эмулируют соединение с одним клиентом.

Диаграмма классов компонента Net изображена на рисунке 16.

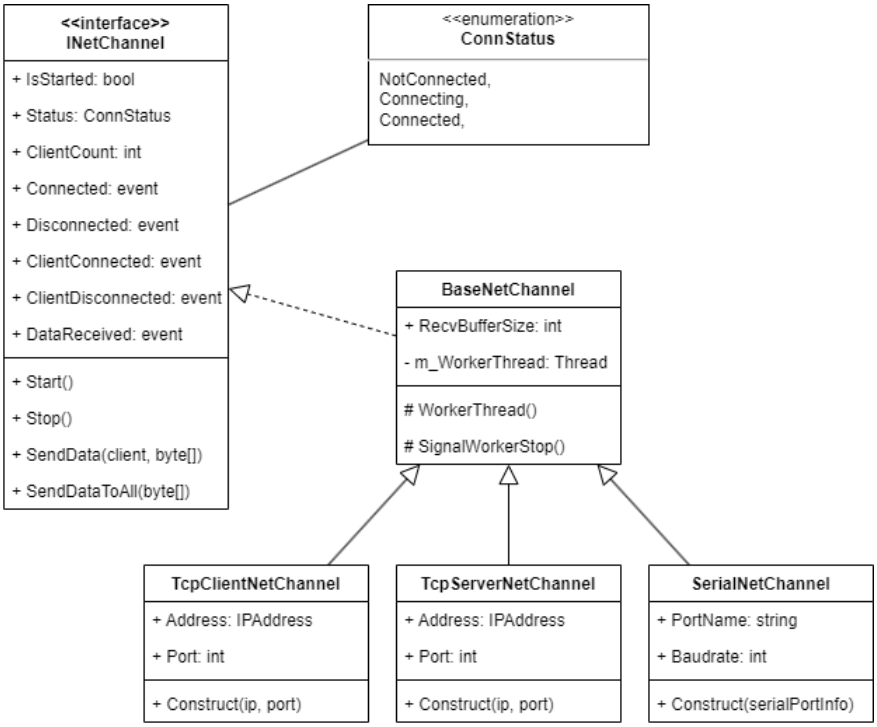


Рисунок 16 – Диаграмма классов компонента Net

2.6.2 Связь по протоколу WITS

Подробно протокол WITS описан в разделе 1.2. За работу с пакетами данных в формате WITS отвечает компонент Net.Wits, диаграмма классов представлена на рисунке 17. Класс *WitsItem* хранит в себе одну строку пакета. Класс *WitsPacket* представляет собой один пакет WITS (набор строк) и имеет метод для перевода его в текстовый вид. За разбор пакетов WITS в текстовом виде отвечает класс *WitsParser*. Он принимает на вход набор байт и для каждого сообщения в этом наборе байт вызывает событие *MessageReceived*. В случае ошибок разбора пакета вызывается событие *InvalidData*.

За интеграцию протокола WITS в маршрутизатор данных отвечает компонент DataRouter.Wits, диаграмма классов которого изображена на рисунке 18. Он содержит в себе классы терминалов *WitsRxNode* и *WitsTxNode*, принимающие и передающие данные по протоколу WITS, соответственно. Эти классы реализуют соответствующие интерфейсы и наследуются от класса *BaseWitsNode*. Класс *BaseWitsNode* отвечает за загрузку конфигурации и реализацию интерфейса *IDataNode*.

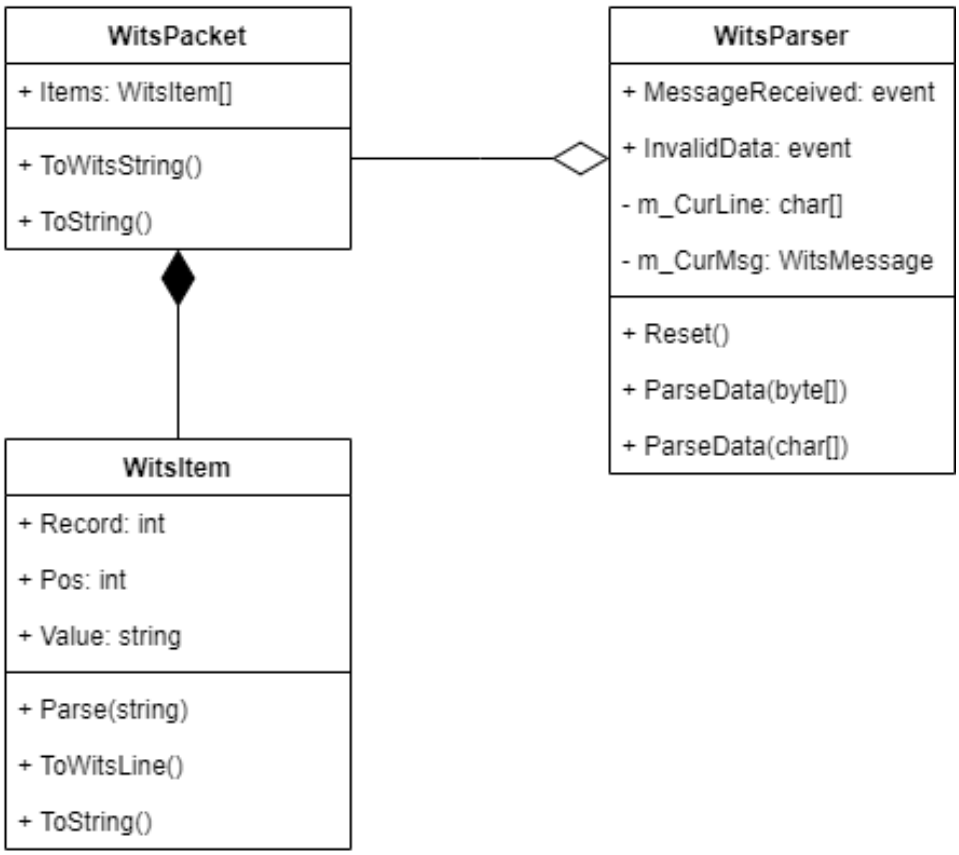


Рисунок 17 – Диаграмма классов компонента Net.Wits

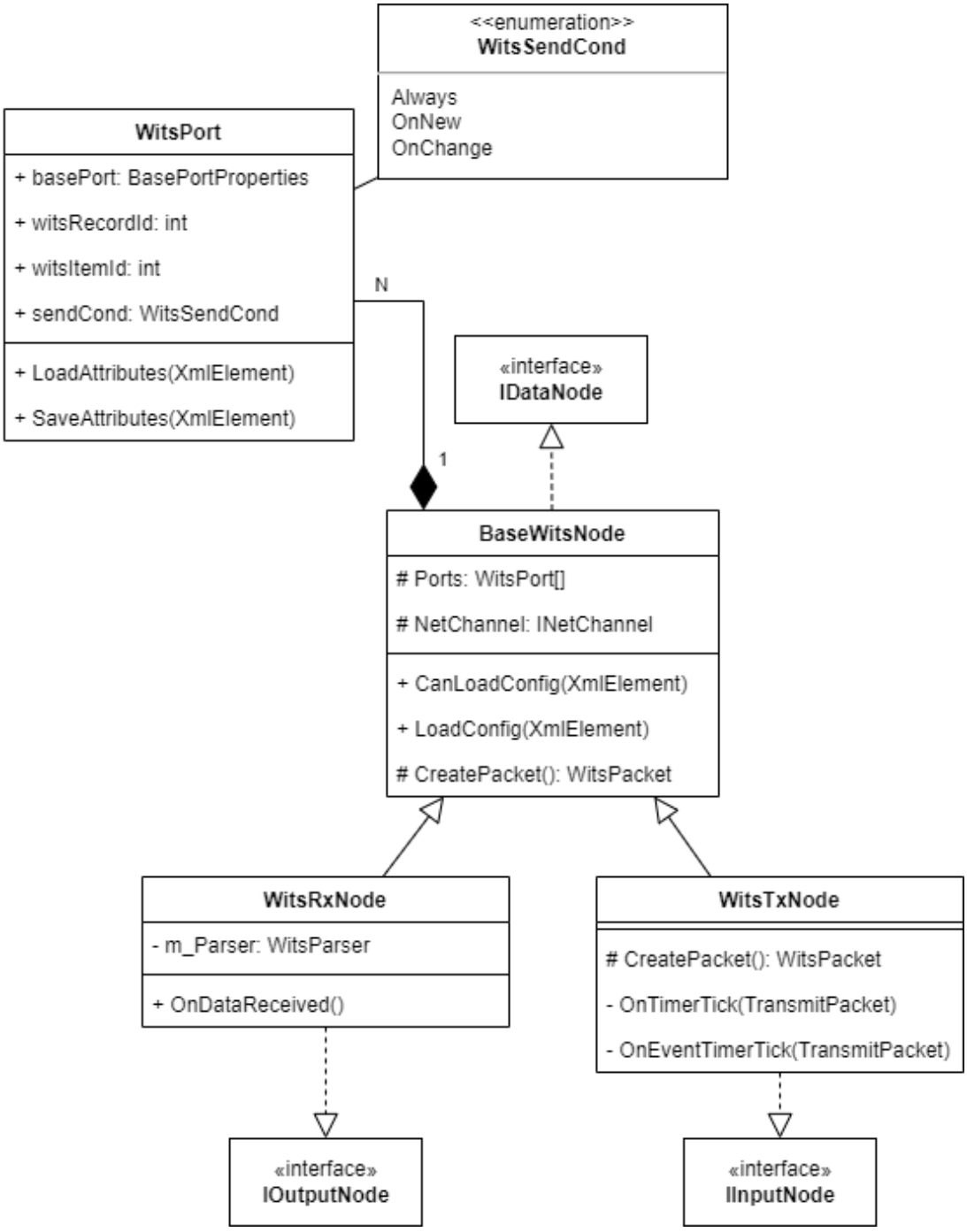


Рисунок 18 – Диаграмма классов компонента DataRouter.Wits

2.6.3 Связь по протоколу WAKE

Протокол WAKE описан в спецификации [5] и реализован в компоненте Net.Wake, диаграмма классов которого представлена на рисунке 19. Классы *WakeConst* и *WakeCrc* являются статическими и отвечают за описанный в спецификации константы и расчет контрольной суммы, соответственно. Классы *WakePacketRx* и *WakePacketTx* являются пакетами с

данными WAKE, полученными или переданными, соответственно. Такое разделение было выполнено, так как полученный пакет содержит дополнительную информацию (например, контрольную сумму из заголовка пакета и контрольную сумму рассчитанную), которая не имеет смысла в пакете, который только готовится к отправке.

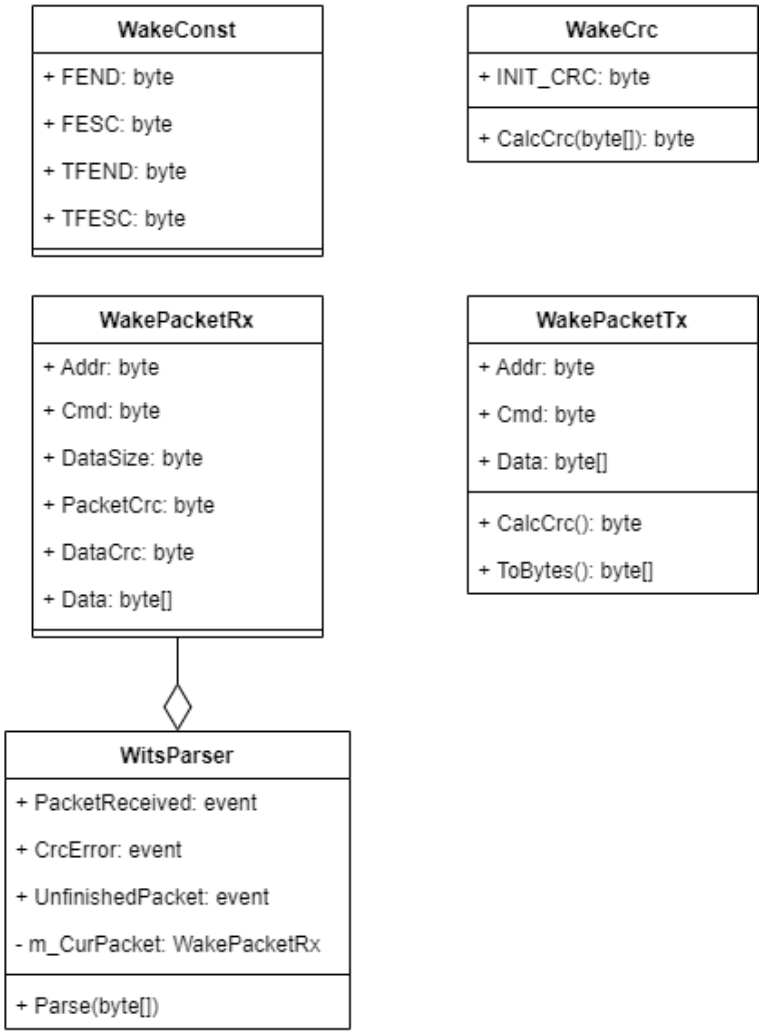


Рисунок 19 – Диаграмма классов компонента Net.Wake

За интеграцию протокола WAKE в маршрутизатор данных отвечает компонент DataRouter.Wake, диаграмма классов которого представлена на рисунке 20. В отличие от протокола WITS, имеющего определенную структуру, протокол WAKE предназначен для передачи двоичных данных, структура которых определяется приложением или устройством. Поэтому для обработки пакетов WAKE требуется знать точную структуру пакета.

Структуру можно описать набором полей, их отступом в пакет, длиной и типом данных. На данный момент поддерживаются следующие типы данных:

1. целое число;
2. вещественное число в формате IEEE-754;
3. вещественное число, линейно приведенное к целому (нормализованное целое).

Описание пакетов задается в формате XML и загружается в программу из файла. Набор пакетов, которые необходимо принимать или передавать, задается явно для каждого терминала. Пакет описывается классом *WakeNodePacket*. Для каждого элемента структуры пакета создается канал (*WakePort*), который затем может использоваться в системе.

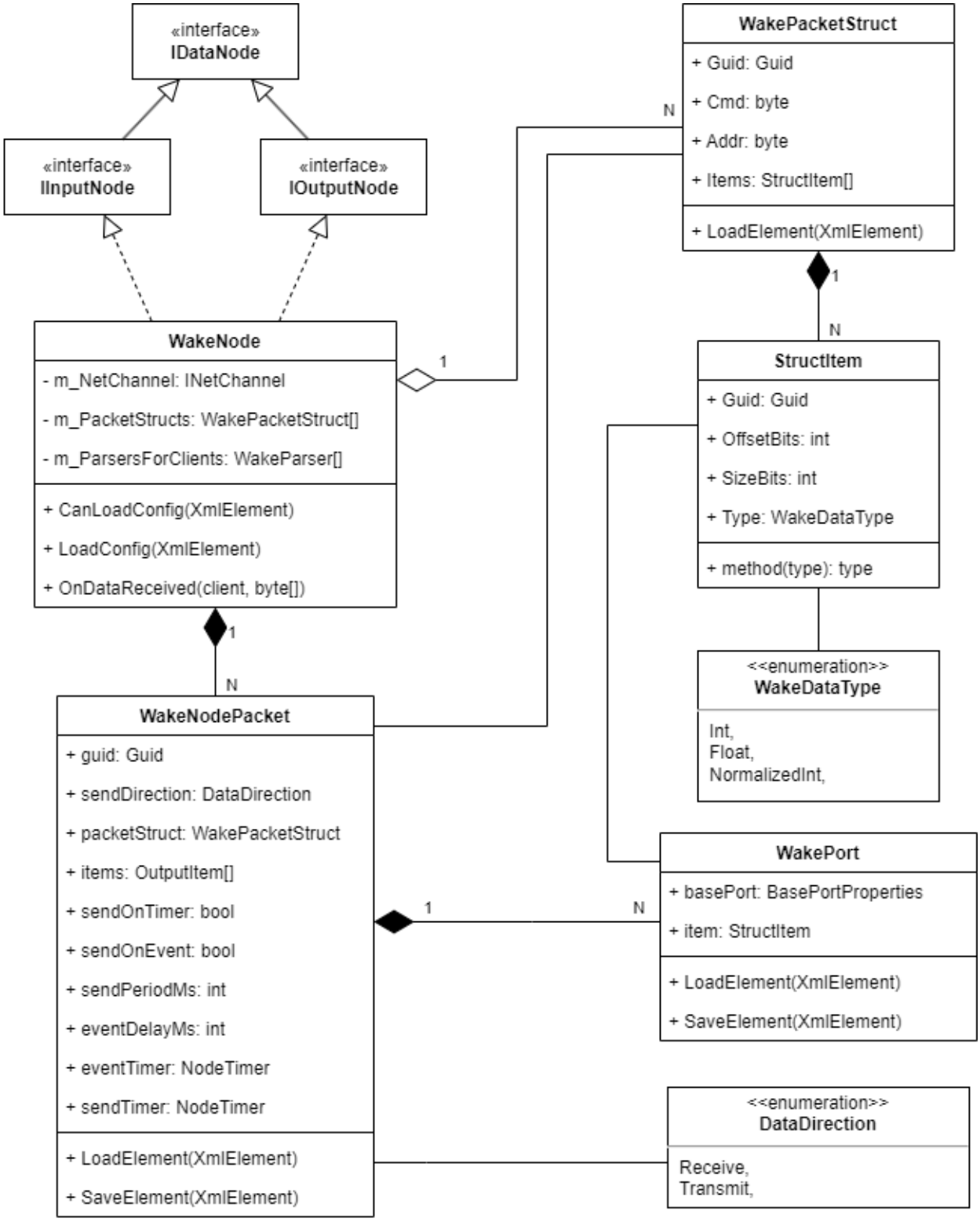


Рисунок 20 – Диаграмма классов компонента DataRouter.Wake

2.7 Проектирование графического интерфейса

2.7.1 ПО для проектирования графического интерфейса

Проектирование графического интерфейса было реализовано в программе Microsoft PowerPoint, так как на этот продукт уже имелась лицензия, а облачные сервисы могут не отвечать требованиям информационной безопасности.

2.7.2 Стартовое окно

Для удовлетворения требования INF-2 при запуске программы открывается стартовое окно, на котором пользователь выбирает, какой файл конфигурации требуется загрузить. Макет стартового окна представлен на рисунке 21.

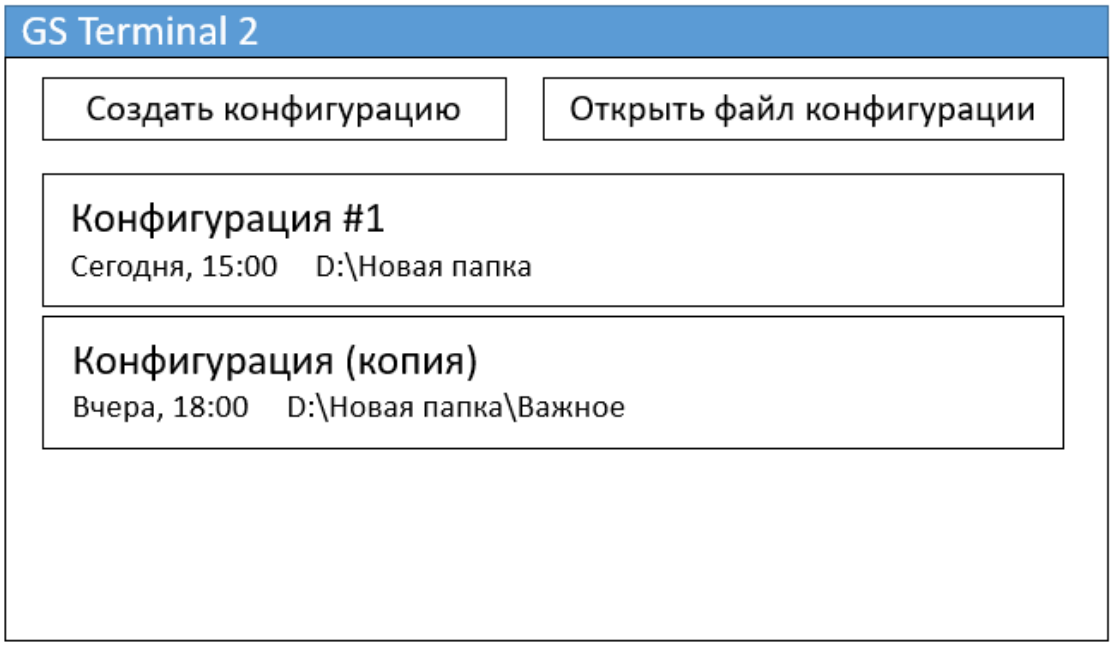


Рисунок 21 – Макет стартового окна

2.7.3 Главное окно

После создания новой или загрузки существующей конфигурации открывается главное окно программы, макет которого изображен на рисунке 22. Вверху окна находится панель меню с пунктами «Файл» и «Справка», а также информационное сообщение о статусе системы. Центральная часть окна содержит набор вкладок, которые будут рассмотрены в следующих пунктах

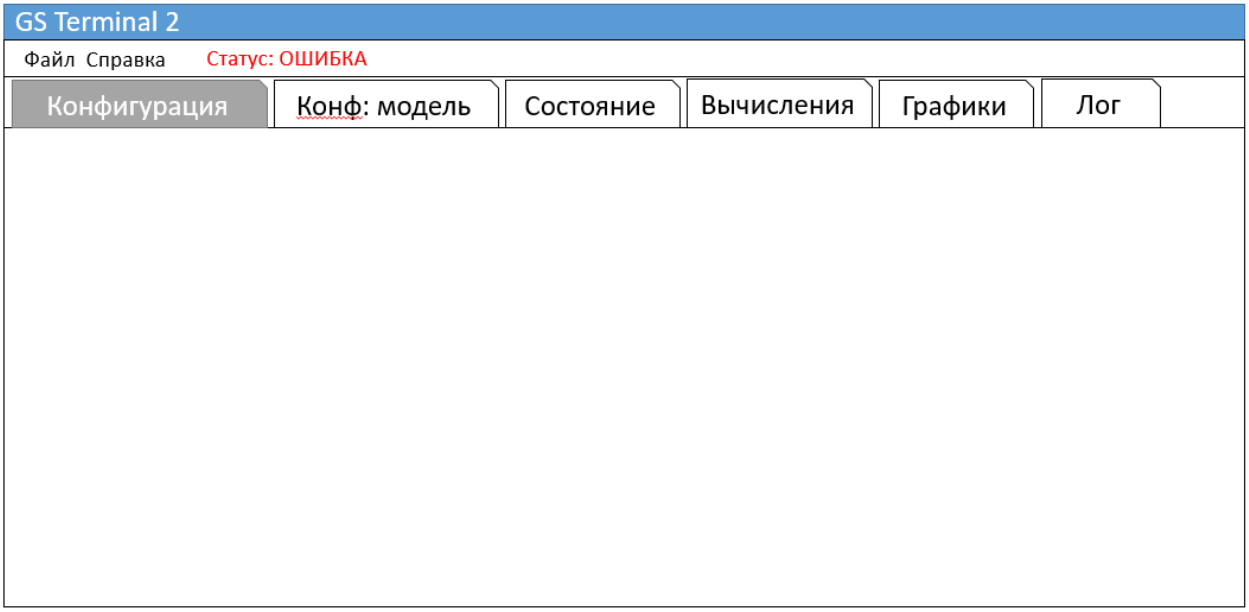


Рисунок 22 – Макет главного окна

2.7.4 Конфигурация терминалов

Интерфейс конфигурации терминалов, макет которого представлен на рисунке 23, разделен на две части. Левая часть содержит список терминалов и список найденных ошибок. Правая часть содержит настройки выбранного терминала и кнопки «Применить изменения» и «Отмена изменений». Содержимое настроек терминала определяется разработчиком терминалом и не зависит от интерфейса программы «GS Terminal».

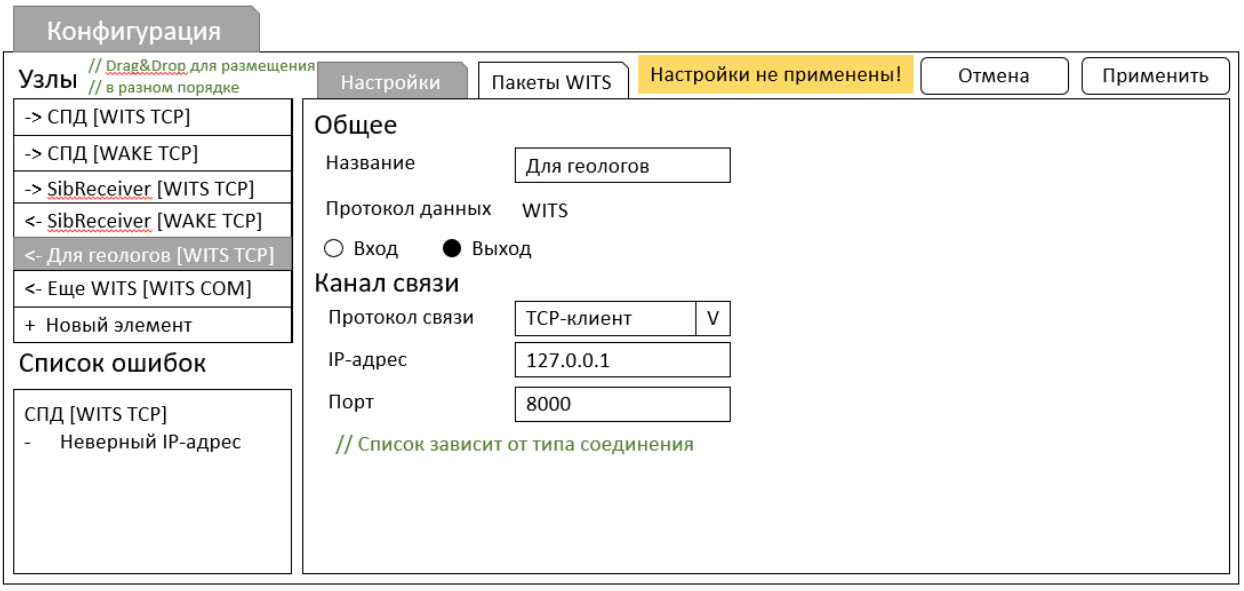


Рисунок 23 – Эскиз раздела конфигурации

2.7.5 Модель данных

Интерфейс модели данных отображает текущую конфигурацию в формате ER-диаграммы. Макет представлен на рисунке 24.

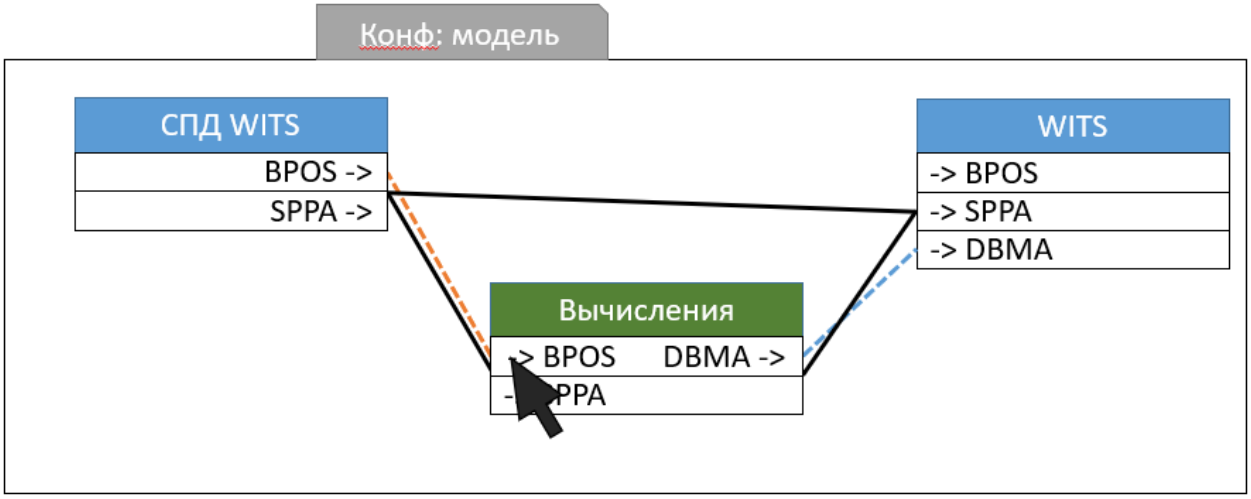


Рисунок 24 – Макет раздела модели данных

В этом разделе изображены терминалы вместе с их каналами. Так как у каждого терминала может быть большое количество каналов (несколько десятков), чтобы не выводить на экран большое количество линий, все связи между каналами двух терминалов могут объединяться в одну линию. При наведении на конкретный канал отобразятся новые линии, показывающие все связи этого канала.

Терминалы на экране можно произвольно размещать, передвигая их мышкой. Сам экран можно двигать и масштабировать.

2.7.6 Состояние системы

На рисунке 25 представлен макет интерфейса состояния системы, отображающего текущий статус узлов, текущие значения каналов и список ошибок за последнее время. Также здесь расположены кнопки для запуска и остановки системы.

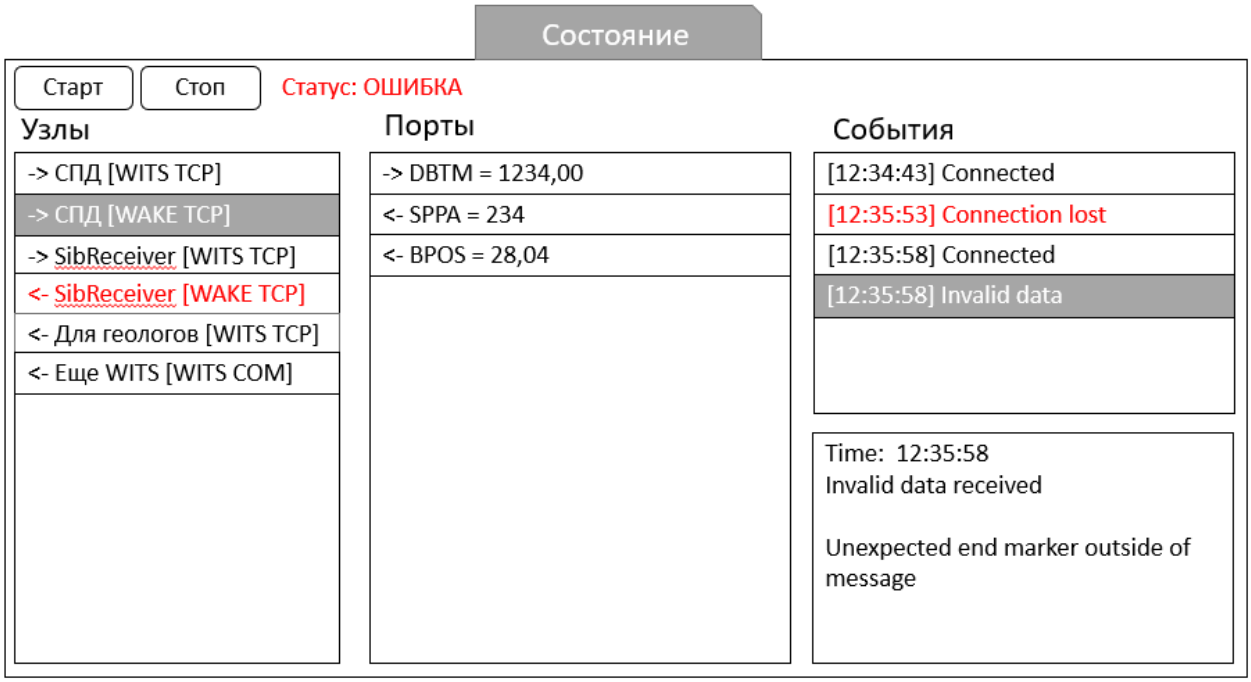


Рисунок 25 – Макет раздела состояния

2.8 Результаты работы

В ходе выполнения работы была спроектирована часть системы, отвечающая за обработку данных. Были построены макеты графического интерфейса. Результаты проектирования были доложены руководству и сотрудникам отдела разработок, и была одобрена и утверждена реализация системы, процесс которой описан в следующей главе.

Глава 3 Реализация программной системы

3.1 Обзор средств разработки графических приложений

Перед началом разработки приложения следует выбрать средство разработки программы (фреймворк). Фреймворк, в свою очередь, определит список поддерживаемых платформ, используемые язык программирования и шаблон проектирования графического интерфейса.

Для рассмотрения были выбраны следующие фреймворки:

1. Qt Widgets;
2. WinUI;
3. Windows Forms (WinForms);
4. Windows Presentation Foundation (WPF);
5. .NET Multi-Platform App UI (MAUI);
6. Electron.

Они будут сравниваться по следующим характеристикам.

Язык программирования. Требуется выбрать такой фреймворк, который использует язык программирования, который принято использовать у заказчика, чтобы в дальнейшем заказчик мог поддерживать продукт.

Поддерживаемые ОС. Обязательно требуется поддержка ОС Microsoft Windows как самой распространенной ОС. Дополнительно требуется поддержка различных дистрибутивов Linux, так как на данный момент доступ к Windows затруднен из-за санкций. Одним из способов добавить поддержку Linux к программе для Windows является использование транслятора Wine [30]. Он перехватывает вызовы Windows API и транслирует их в эквивалентные вызовы к API целевой системы.

Поддержка аппаратного ускорения. Различные фреймворки выводят элементы интерфейса на экран двумя разными способами. При отсутствии аппаратного ускорения, все операции отрисовки выполняются на центральном процессоре. В таком случае при выводе большого количества данных на экран (например, в большой таблице) возможны заметные зависания интерфейса

программы во время прокрутки или обновления экрана. При наличии аппаратного ускорения часть работы по отрисовке элементов перекладывается на графический ускоритель, который предназначен для работы с графикой. Это позволяет обновлять большую площадь экрана за раз [20].

Поддержка мониторов с высокой плотностью пикселей (HiDPI, High Dots Per Inch). На данный момент распространены мониторы с плотностью пикселей больше стандартных 96 DPI. Например, мониторы ноутбуков с разрешением 1920x1080 пикселей или более, мониторы настольных компьютеров с разрешением 4K (3840x2160). Фреймворки, которые используют для позиционирования и масштабирования элементов пиксели, на таких экранах будут выглядеть или мелко, если для них не включено масштабирование, или размыто, если программа рисуется в меньшем разрешении и масштабируется до большего [8].

Статус разработки. Важно учитывать, поддерживается ли фреймворк разработчиками. Здесь возможно три основных варианта:

1. активно развивается, то есть разработчиками добавляется новая функциональность;
2. поддерживается, то есть разработчики решают обнаруженные проблемы и поддерживают совместимость с современными ОС и библиотеками;
3. заброшен, то есть проблемы или не решаются, или решается самый их минимум, или планируется остановка поддержки в обозримом будущем.

Популярность фреймворка. От популярности фреймворка зависит количество информации о нем в сети Интернет, от чего зависит скорость и сложность решения возможных проблем.

Лицензия фреймворка. Она определяет, разрешено ли использовать фреймворк в коммерческом продукте и требуется ли раскрывать его исходный код. Часто используемыми лицензиями являются лицензия MIT [18], GNU General Public License (GPL) [12], GNU Library/Lesser General Public License [13]. Лицензия MIT позволяет изменять код библиотеки любым способом и не

требует распространять изменения в коде или же весь связанный код, программы, использующей библиотеку. Требуется только указать автора оригинального кода. GPL же, напротив, требует распространения исходного кода любой программы, использующей библиотеку, распространяемую по GPL, и связанного с ней кода. LGPL ослабляет это ограничение, требуя только предоставлять конечному пользователю возможность заменить LGPL код на свой. Для языков C/C++ это означает, что библиотека должна использоваться с применением динамической компоновки, т.е. в формате DLL (Dynamic Link Library) или SO (Shared Object).

Влияние санкций. В условиях санкций против РФ используемый фреймворк не должен требовать для работы компоненты, импорт которых на территорию РФ запрещен или затруднен, а также средства разработки, доступ к которым может быть остановлен.

3.1.1 Qt Widgets

Qt – это фреймворк для разработки кросс-платформенных приложений, разрабатываемый компанией The Qt Company [22]. Компонентом, отвечающим за его графический интерфейс, является Qt Widgets. Основным языком программирования является C++. В данный момент Qt Widgets не находится в состоянии активного развития, но все еще поддерживается разработчиком. От разработчиков нет никаких рекомендаций не использовать Qt Widgets для новых разработок. Для разработки программ на Qt используется Qt Creator, который свободно доступен всем желающим. Пакет Qt распространяется под коммерческой лицензией, которая в данный момент недоступна на территории РФ, или под лицензией LGPL (с некоторыми компонентами под GPL). Компонент Qt Widgets не поддерживает аппаратное ускорение графики, но поддерживает HiDPI мониторы. В Интернете доступно множество информации по Qt Widgets.

3.1.2 WinUI

WinUI 3 – это библиотека компонентов графического интерфейса, разработанная корпорацией Microsoft [31][32]. Она была представлена в 2021 году как часть Windows App SDK [28]. Единственными официально поддерживаемыми платформами являются Windows 10 и 11. Библиотека доступна в языках C++ и C#, написание графического интерфейса осуществляется через язык разметки XAML. Windows App SDK распространяется по лицензии MIT. Для разработки рекомендуется использовать среду Microsoft Visual Studio 2022, которая может быть недоступна в РФ, но при этом разработка также возможна в любой другой среде. Так как это современная библиотека, она поддерживает аппаратное ускорение графического интерфейса и HiDPI мониторы. Но, в связи с этим же, информации в Сети по ней на данный момент мало.

3.1.3 Windows Forms

Windows Forms – это библиотека для языка C#, предоставляющая классические элементы управления, доступные в Windows API [29]. Она разработана корпорацией Microsoft в 2001 году [26] и включена в состав среды исполнения .NET. Так как она реализована на основе встроенных в Windows элементов управления, она не поддерживает ни аппаратное ускорение, ни HiDPI мониторы. Активная разработка не ведется. Библиотека не имеет какого-либо языка разметки, структура графического интерфейса описывается явно в файле исходного кода [26]. Microsoft Visual Studio содержит дизайнер графического интерфейса, который обязателен для разработки, так как ручное написание кода для расположения элементов является очень трудоемким и долгим процессом. Windows Forms и весь .NET распространяются по лицензии MIT.

3.1.4 Windows Presentation Foundation

Windows Presentation Foundation (WPF) – это библиотека для языка C#, аналогично Windows Forms включенная в состав .NET и выпущенная в 2006

году. Она использует язык разметки XAML для описания элементов графического интерфейса. Последняя версия библиотеки поддерживает аппаратное ускорение и HiDPI мониторы, а в Интернете доступно большое количество информации по ее использованию. WPF и весь .NET распространяются по лицензии MIT. Аналогично WinUI, для разработки рекомендуется использовать среду Microsoft Visual Studio 2022, но при этом разработка также возможна в любом текстовом редакторе.

3.1.5 Multi-platform App UI

Multi-platform App UI (MAUI) это выпущенный корпорацией Microsoft фреймворк для разработки кросс-платформенных приложений на языке C# [1]. Он поддерживает Windows (используя WinUI), Android и iOS. Официальной поддержки Linux нет. Аналогично WinUI и WPF, для описания элементов интерфейса используется язык XAML. Это новая библиотека, поэтому на данный момент информации в Интернете по ней мало.

3.1.6 Electron

Electron – это фреймворк для разработки графический приложений на основе веб-технологий [10]. Он представляет собой веб-браузер на основе Chromium без рамки окна, вкладок и иной функциональности, специфичной для веб-браузера. Для программирования используется язык JavaScript, а для описания интерфейса – HTML и CSS. Electron поддерживает Windows и Linux, а также может запускаться в веб-браузере. Главным недостатком Electron является его тяжесть (как по занимаемому пространству диска, так и используемым ресурсам) – ему требуется распространять с собой и запускать веб-браузер Google Chrome. Он распространяется по лицензии MIT, а его использование похоже на обычную веб-разработку, поэтому в Интернете доступно огромное количество информации.

3.1.7 Сравнение фреймворков

Сводная таблица сравнения фреймворков представлена на таблице 1.

Таблица 1 – Сравнение фреймворков.

	Qt Widgets	WinUI	WinForms	WPF	MAUI	Electron
Язык	C++	C++/C#	C#	C#	C#	JavaScript
ОС	Windows, Linux	Windows, Wine	Windows, Wine	Windows, Wine	Windows, Wine	Windows, Linux
Аппаратное ускорение		+		+	+	+
HiDPI мониторы	+	+		+	+	+
Статус разработки	Поддерж.	Активная	Поддерж.	Поддерж.	Активная	Активная
Информация в Интернете	Много	Мало	Много	Много	Мало	Много
Влияние санкций: можно ли использовать	Да	Да	Да	Да	Да	Да
Влияние санкций на удобство разработки	Нет	Незнач.	Значител.	Незнач.	Незнач.	Нет

Из рассмотренных фреймворков был выбран Windows Presentation Foundation. Язык программирования C# уже используется у заказчика, а значит заказчик может в дальнейшем поддерживать продукт своими силами. Этот язык, по сравнению с другими рассмотренными языками, обладает полезным при разработке приложений функциями: сборщик мусора, упрощающий работу с памятью; встроенные события; произвольные атрибуты для классов, полей и методов.

3.2 Паттерн MVVM

Windows Presentation Foundation использует шаблон Model-View-ViewModel (MVVM) для реализации графического интерфейса пользователя [34]. Этот паттерн разделяет систему на три части:

1. представление (view) – элементы управления программой, с которыми взаимодействует пользователь;
2. модель данных (model) – отвечает за хранение и обработку данных без привязки к графическому интерфейсу;
3. модель представления (view-model) – обрабатывает действия пользователя, проверяет пользовательский ввод и связывает модель данных с представлением.

Согласно концепциям шаблона MVVM, модель представления не должна содержать элементов бизнес-логики. Она должна только вызывать методы модели данных по событиям от элементов интерфейса, проверять пользовательский ввод и предоставлять информацию представлению в необходимом для отображения виде. Модель данных не должна быть явно связана с моделью представления, ведь она может использоваться вне графического интерфейса (например, в автоматических тестах). Модель представления не должна быть привязана к конкретному представлению, потому что она может использоваться в разных представлениях.

Связь представления и модели представления в WPF осуществляется через «привязку данных» (англ. «data binding») [7]. Модель представления имеет набор свойств языка C# с геттерами и сеттерами и реализует интерфейс *INotifyPropertyChanged*. Этот интерфейс предоставляет событие *PropertyChanged*, которое позволяет объекту уведомить слушателей об изменении свойства. WPF, используя рефлексию, получает свойства из объекта модели представления, выводит в представлении текущее значение и при изменении свойства (вследствие вызова события *PropertyChanged*) обновляет представление.

Для упрощения реализации модели представления разработчиками из Microsoft была создана библиотека Prism [21]. Она добавляет класс *BindableBase*, который должен использоваться в качестве базового класса для всех моделей представления. Он имеет метод *SetProperty*, который принимает

на вход ссылку на переменную, а которой хранится значение свойства, и новое его значение. Если новое значение отличается от старого, метод обновит свойство и сам вызовет событие *PropertyChanged*. Пример использования *BindableBase* и *SetProperty* показан в листинге 2.

Листинг 2 – Пример использования BindableBase

```
public class ExampleViewModel : BindableBase
{
    public int Example
    {
        get => m_Example;
        set => SetProperty(ref m_Example, value);
    }

    private int m_Example = 0;
}
```

3.3 Обработка ошибок

Обработка ошибок осуществляется в модели представления, реализуя интерфейс *INotifyDataErrorInfo*. Ошибки выявляются в сеттерах свойств, и при выявлении ошибки вызывается событие *ErrorsChanged*. Элементы представления в такой ситуации вызывают у объекта метод *GetErrors* с названием свойства и выводят ошибку пользователю.

Для упрощения проверки ошибок был создан базовый класс для моделей представления *ValidationBase*. Диаграмма классов проверки ошибок представлена на рисунке 26. *ValidationBase* хранит в себе словарь правил (*ValidationRule*) для каждого свойства объекта. При вызове метода *SetProperty* осуществляется обход всех правил для свойства и проверка нового значения на ошибки. Обнаруженные ошибки сохраняются в словарь ошибок и вызывается свойство *ErrorsChanged*. Добавление правил осуществляется методом *AddValidationRule*.

При таком подходе список правил будет дублироваться у каждого экземпляра модели представления, которых может быть создано большое количество (например, для каждого элемента списка). К тому же, в большинстве случаев правила не зависят от конкретного экземпляра. Было бы удобно задавать их не в конструкторе объекта вызовами *AddValidationRule*, а

атрибутами у свойств. Для этого был создан класс *ValidationBase<T>*, принимающий тип модели представления в аргументе обобщения (generic argument). Он хранит все правила, заданные атрибутами, в статическом свойстве, тем самым избегая дублирование данных.

Пример использования атрибутов для обработки ошибок представлен на листинге 3. Свойству *Name* указано правило, что строка не должна быть пустой. Свойству *PostCode* указано правило, что число должно быть от 1 до 100.

Листинг 3 – Проверка свойств на ошибки, используя атрибуты

```
public class DataInputSubViewModel :
    ValidationBase<DataInputSubViewModel>
{
    [NotEmptyValidationRule]
    public string Name
    {
        get => m_Model.name;
        set => SetProperty(ref m_Model.name, value);
    }

    [RangeValidationRule<int>(1, 100)]
    public int PostCode
    {
        get => m_Model.postCode;
        set => SetProperty(ref m_Model.postCode, value);
    }

    private readonly DataInputModel m_Model;

    public DataInputSubViewModel(DataInputModel model)
    {
        m_Model = model;
        RefreshValidation();
    }
}
```

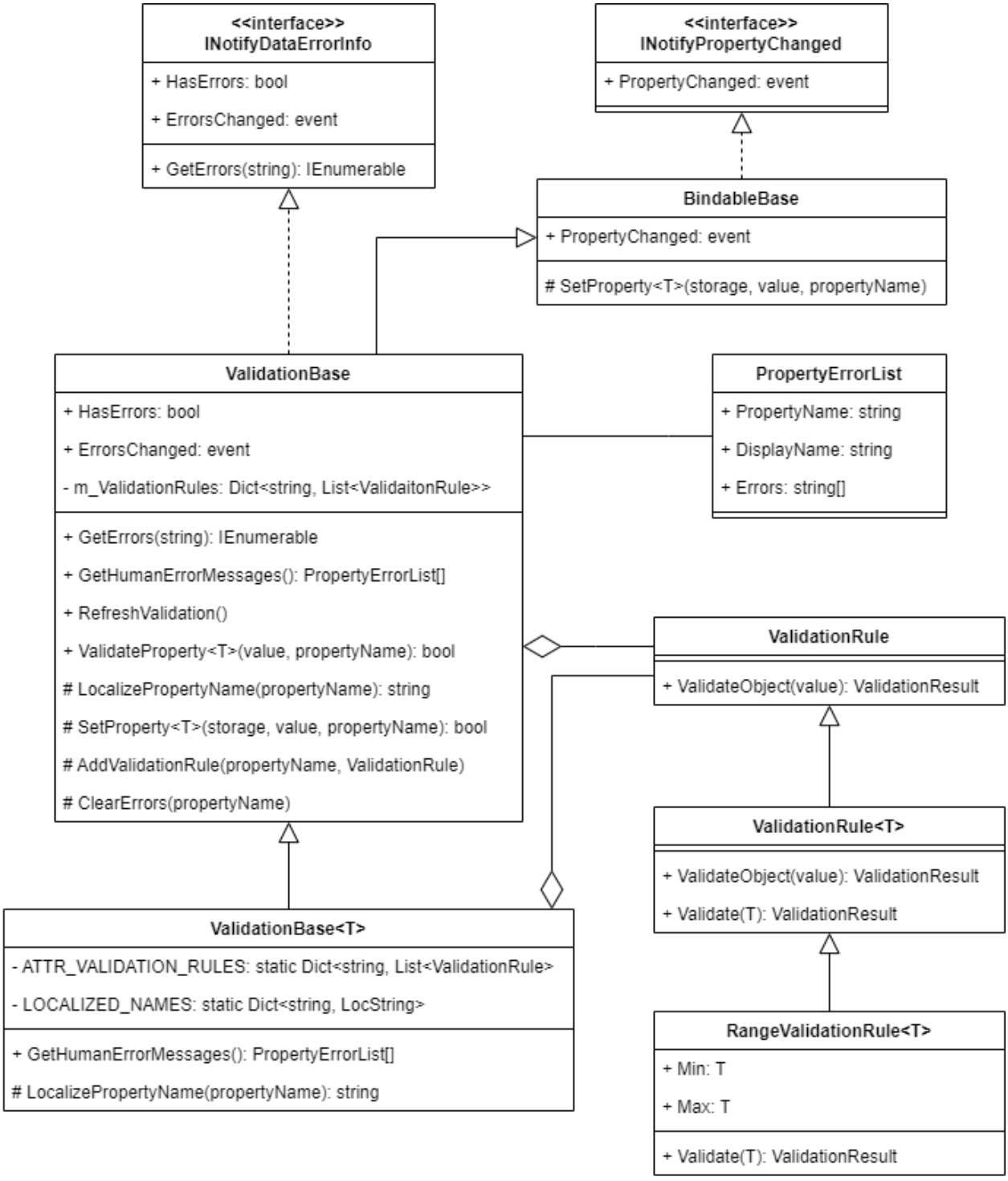


Рисунок 26 – Диаграмма классов системы валидации данных

3.4 Локализация

WPF имеет встроенную систему локализации для представлений [11]. Она позволяет генерировать файлы локализации для разметки на XAML, которые затем можно передать переводчикам. Однако, в WPF нет встроенной системы для локализации строк, использующихся в программном коде.

Microsoft предлагают использовать для этого систему словарей ресурсов [14]. Для упрощения работы с ней был создан класс *LocString*. Он принимает в конструкторе токен локализации – строку, начинающуюся с символа @ и определяющую ключ в словаре ресурсов. Если класс найдет эту строку в загруженных словарях ресурсов, он сохранит ее в свойстве *Value*. Иначе же, в *Value* будет сохранен токен локализации и будет выведена ошибка в лог программы. В листинге 4 представлен пример словаря ресурсов для локализации, а в листинге 5 – пример использования *LocString* в коде.

Листинг 4 – Словарь строк для локализации

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    xml:lang="ru-RU">

    <sys:String x:Key="loc_notEmptyValidationRule_text">
        Значение не может быть пустым
    </sys:String>

    <sys:String x:Key="loc_rangeValidationRule_min">
        Значение должно быть больше или равно {0}
    </sys:String>
    <sys:String x:Key="loc_rangeValidationRule_max">
        Значение должно быть меньше или равно {0}
    </sys:String>
</ResourceDictionary>
```

Листинг 5 – Пример использования *LocString*

```
private static LocString ErrorText { get; } =
    new("@notEmptyValidationRule_text");

public override ValidationResult Validate(string value, CultureInfo cultureInfo)
{
    if (string.IsNullOrWhiteSpace(value))
       return new ValidationResult(false, ErrorText.Value);

    return ValidationResult.ValidResult;
}
```

3.5 Логирование

Для логирования информационных сообщений и ошибок в разрабатываемой системе используется интерфейс *ILogger* из пакета Microsoft.Extensions.Logging [16][17]. Для этого был создан глобальный экземпляр фабрики *ILoggerFactory*, который используется для инициализации

статических полей типа *ILogger* в классах, использующих систему логирования.

Логируемые сообщения выводятся в одной из вкладок в главном окне программы (рисунок 27). Для вывода сообщений был создан класс *WpfLoggerProvider*, реализующий интерфейс *ILoggerProvider*. Он хранит в себе коллекцию всех сообщений для отображения в графическом интерфейсе и создает экземпляры *WpfLogger* (реализует *ILogger*). При появлении нового сообщения система логирования вызовет метод *Log* у *WpfLogger*, который передаст сообщение в *WpfLoggerProvider*. Он, в свою очередь, добавит сообщение в коллекцию, если оно поступило из основного потока графического интерфейса, или же сохранит его в очередь и создаст запрос на перенос сообщений из очереди в коллекцию для основного потока, используя *Dispatcher* из WPF.

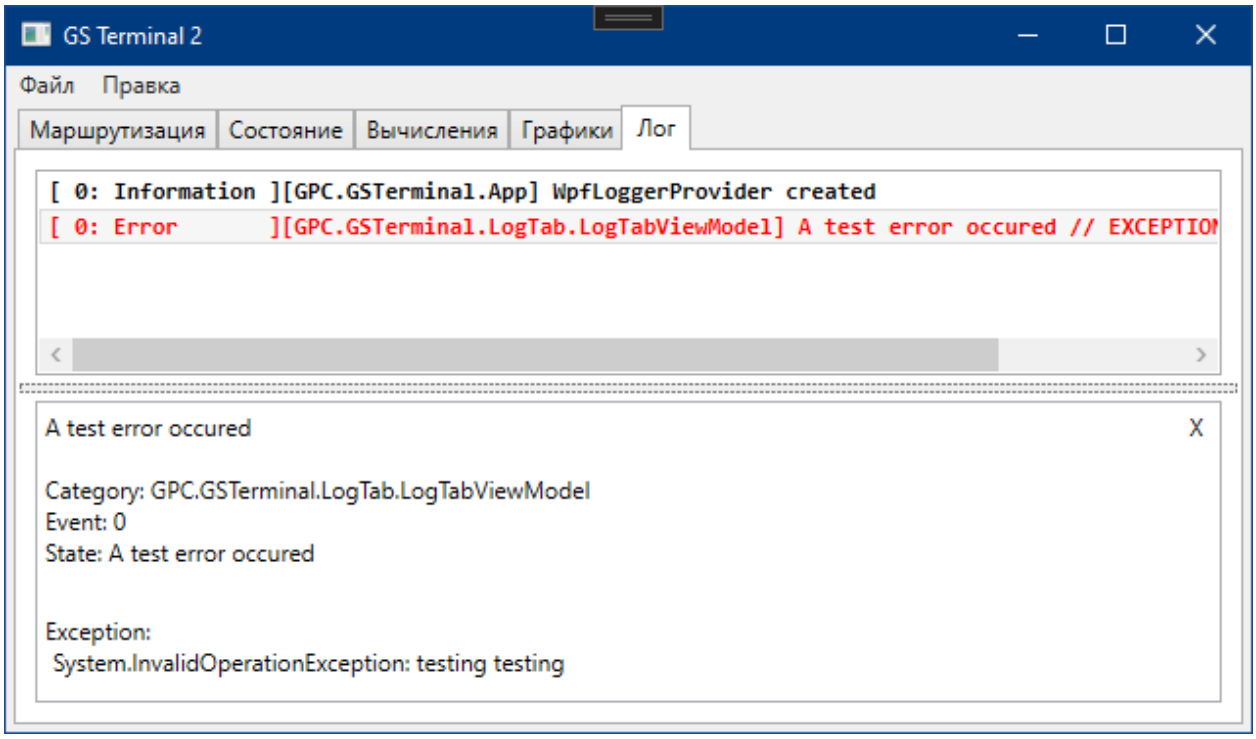


Рисунок 27 – Вкладка с сообщениями из лога программы

3.6 Диалог конфигурации

Одним из основных пользовательских элементов является диалог конфигурации (рисунок 30). Он располагается в первой вкладке в основном окне программы. Он должен обладать следующей функциональностью:

1. создание и удаление терминалов;
2. применение и сброс изменений;
3. редактирование параметров терминала;
4. проверка ввода пользователя на ошибки.

Распределим задачи между моделью и моделью представления.

Модель:

1. хранит XML-элемент конфигурации всех терминалов системы;
2. хранит список всех терминалов и XML-элементы конфигурации отдельных узлов;
3. загружает конфигурацию из XML-элемента в переменные и свойства объекта;
4. предоставляет базовую информацию о терминалах (название, список каналов и т.д.).

Модель представления:

1. обрабатывает запросы пользователя на применение или сброс конфигурации;
2. обрабатывает ошибки пользовательского ввода;
3. предоставляет представлению статус конфигурации: есть ли изменения, есть ли ошибки, какие ошибки.

Вид экрана настроек и набор опций зависит от типа конкретного терминала, поэтому для каждого типа терминала созданы свои классы представления, модели представления и модели. Модель и модель представления должны реализовать интерфейсы *INodeEditorModel* и *INodeEditorViewModel*, соответственно. Эти интерфейсы показаны на диаграмме классов на рисунке 28. Классами модели и модели представления

диалога конфигурации являются *RouterSettingsModel* и *RouterSettingsViewModel*, соответственно. Они хранят информацию о каждом терминале в классах *ItemModel* и *ItemViewModel*. Создание экземпляров модели, модели представления и представления терминала реализовано через фабрику, описанную в разделе 2.5.3. Диаграмма классов экрана настроек показана на рисунке 29.

Иногда возможна ситуация, когда в файле конфигурации записан терминал неизвестного типа. В таком случае для него не будет найдена фабрика и, следовательно, невозможно создать объекты для экрана настроек. Тогда будет использоваться модель представления *InvalidNodeEditorViewModel*, которая сообщит пользователю об ошибке на месте элементов управления конфигурацией.

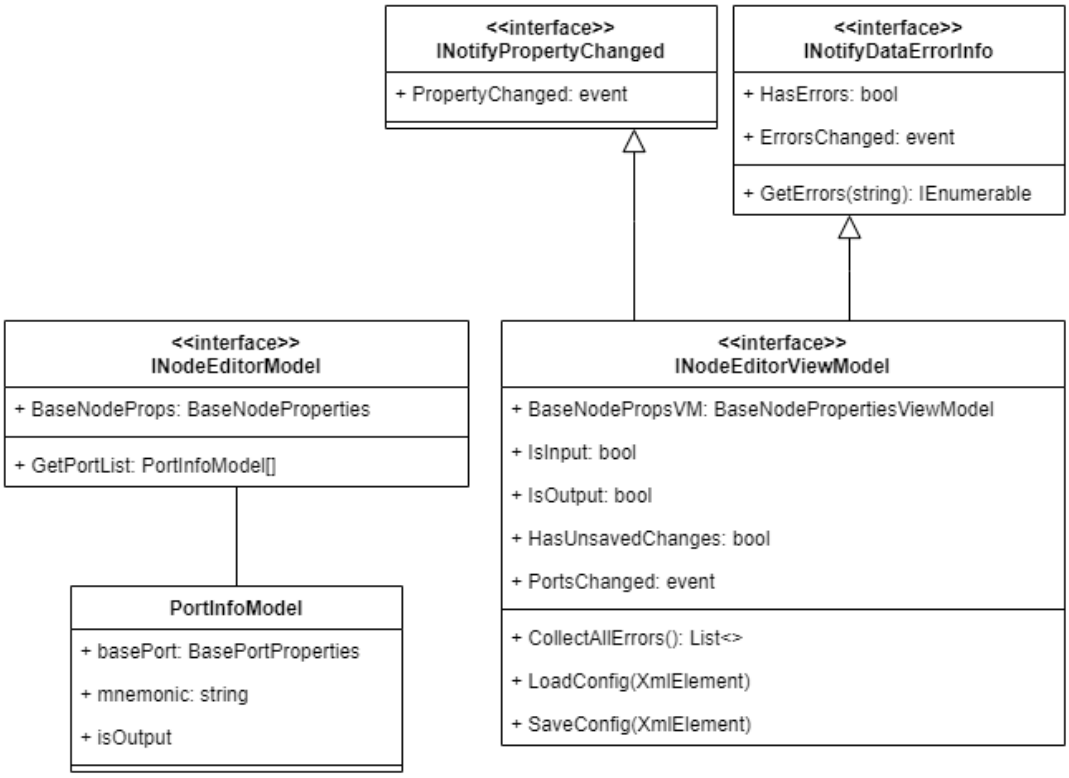


Рисунок 28 – Диаграмма классов модели и модели представления конфигурацторов узлов

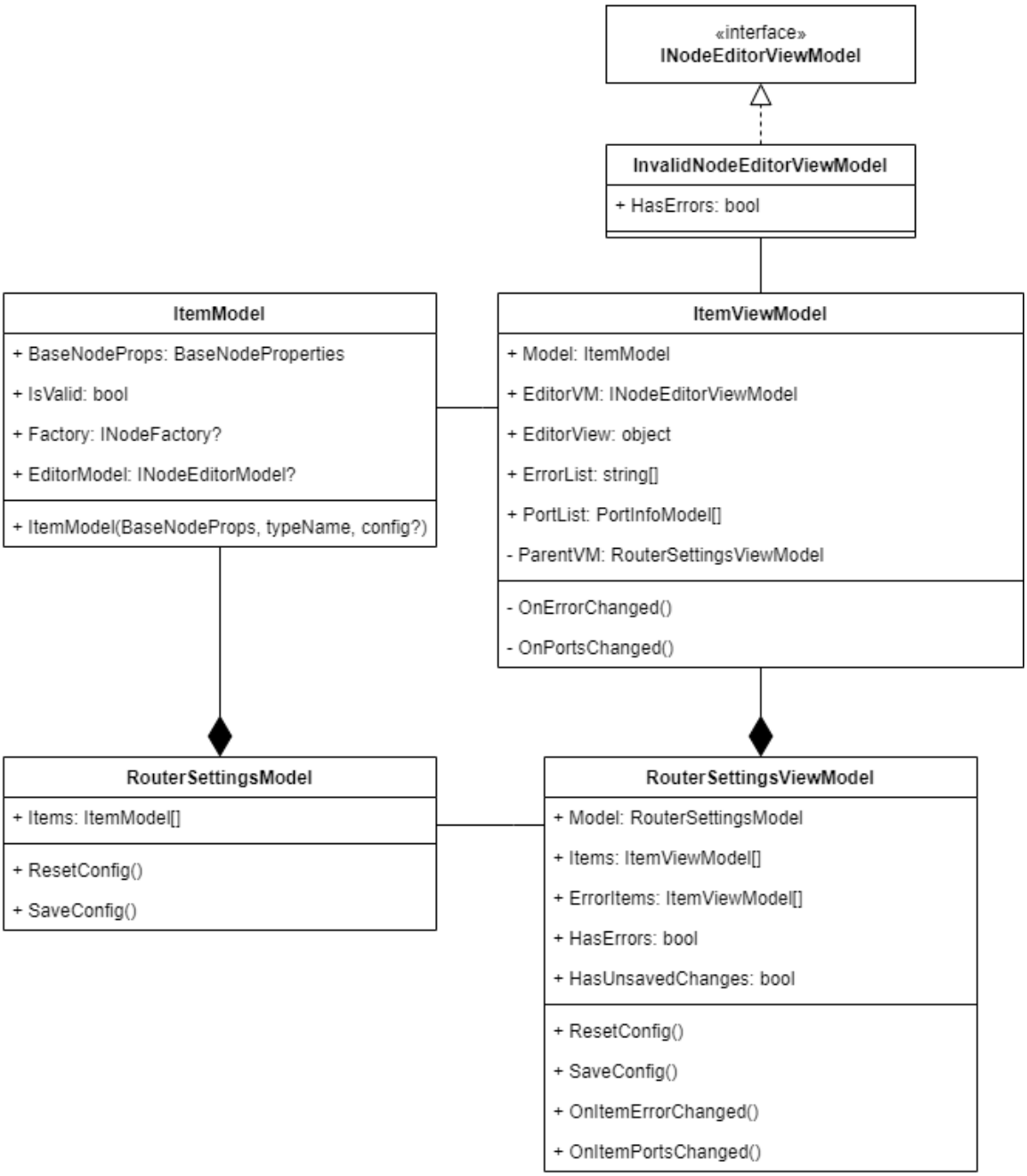


Рисунок 29 – Диаграмма классов модели и модели представления экрана конфигурации

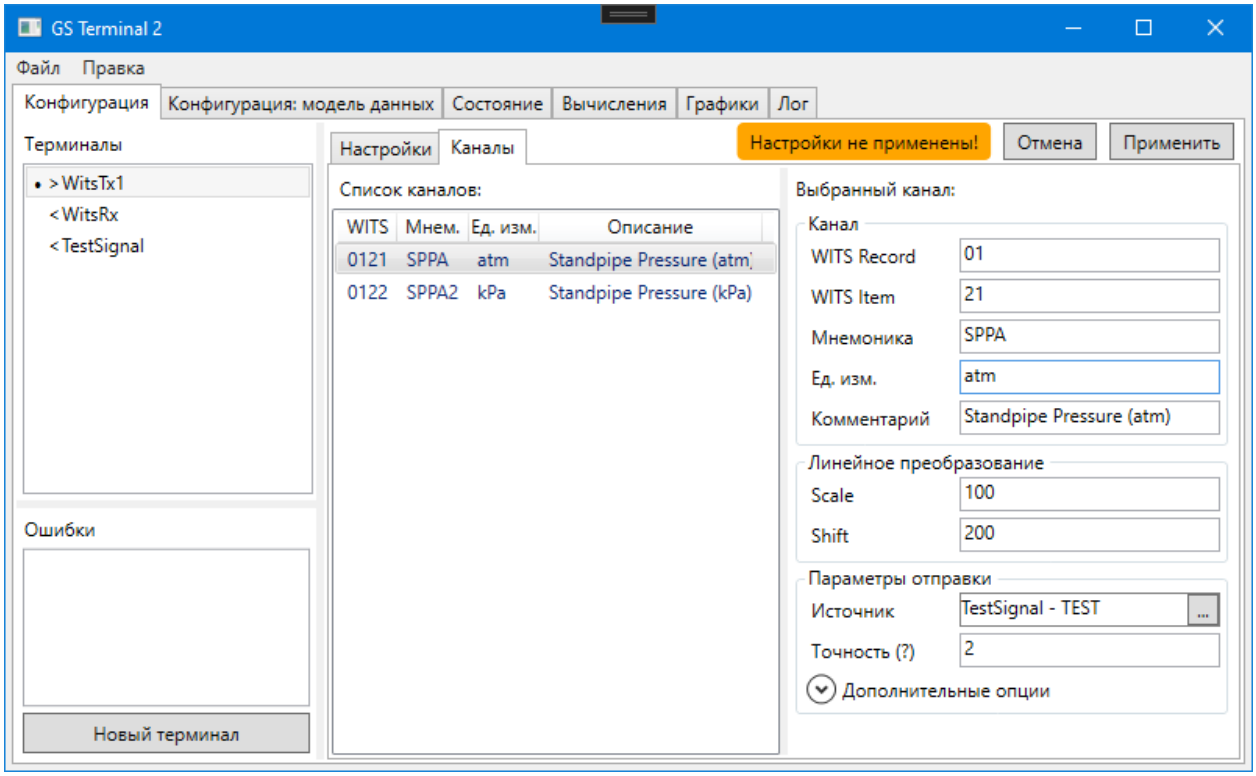


Рисунок 30 – Экран конфигурации

3.7 Представление конфигурации в графическом виде

Для наглядного представления конфигурации системы пользователю было выбрано представление в виде ER-диаграммы (рисунок 31). На диаграмме отображаются терминалы, каналы и связи между ними. Пользователь имеет возможность перемещать терминалы курсором мыши. Для реализации этого представления был создан элемент графического интерфейса под названием *DragCanvas*, содержащий в себе *ItemsControl* из WPF. Он выводит элементы коллекции в виде блоков, состоящих из заголовка и тела. Пользователь может перемещать элементы, наведясь и зажав левую кнопку мыши на заголовке. Заголовок и тело элемента задается в XAML-коде, используя *DataTemplate*, аналогично использованию *ItemsControl*.

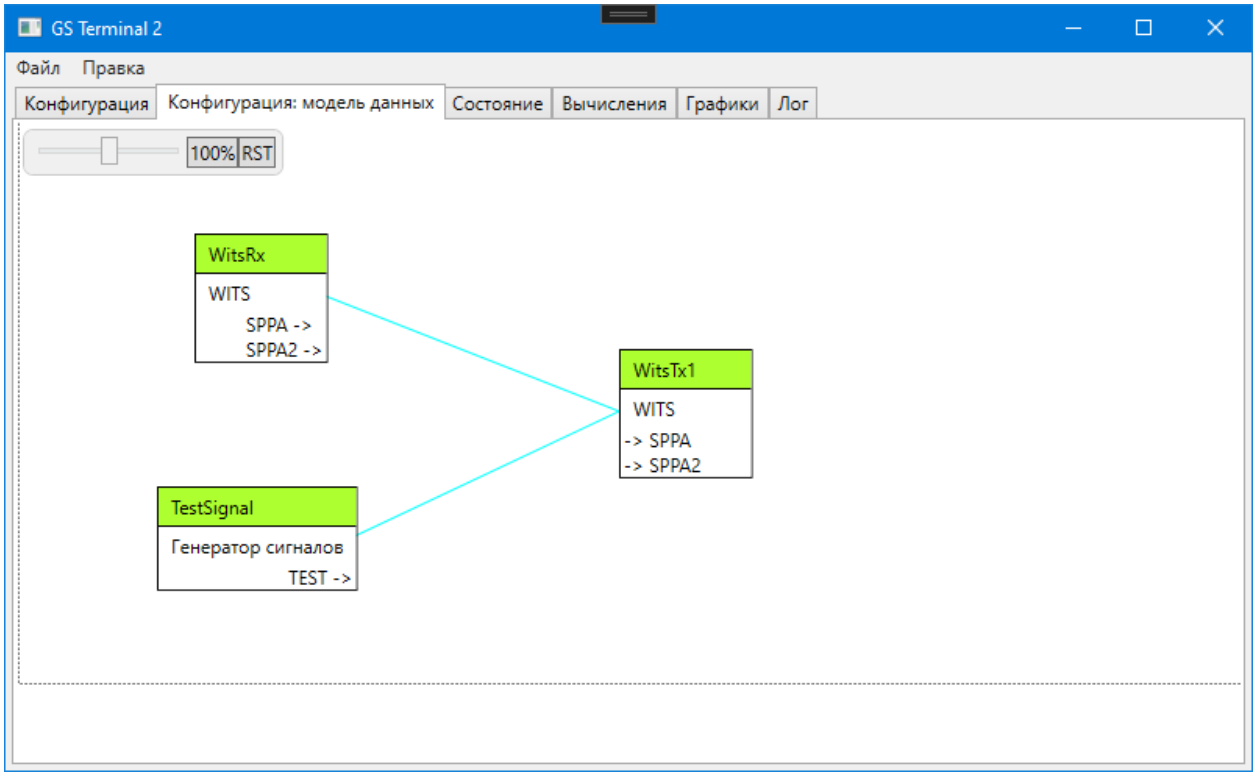


Рисунок 31 – Экран с ER-диаграммой

3.8 Диалог состояния системы

Диалог состояния системы (рисунок 32) состоит из трех вертикальных столбцов: списка терминалов, списка каналов в выбранном терминале и лога выбранного терминала. Список текущих терминалов доступен в классе *DataRouterHost*, рассмотренном в разделе 2.5.4. Для отображения текущей информации о каналах графический интерфейс периодически получает последние значения каналов, вызывая метод *GetPortValues* у терминалов (*IDataNode*). Сбор сообщений для лога каждого терминала осуществляется путем создания отдельного экземпляра фабрики *ILoggerFactory* со своими экземплярами *WpfLoggerProvider* (рассмотренными в разделе 2.5.3).

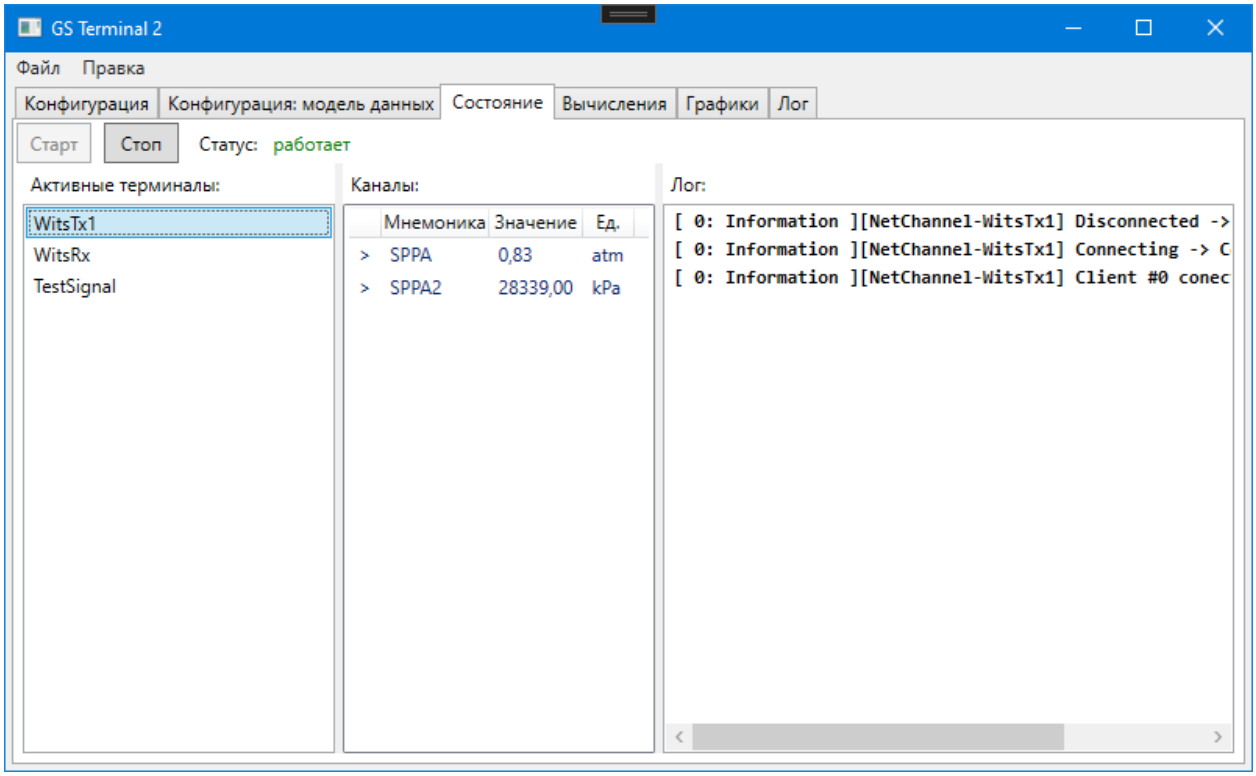


Рисунок 32 – Диалог состояния системы

3.9 Выводы по главе

В данной главе был описан процесс разработки графического интерфейса программы. Был выбран инструмент разработки графического приложения Windows Presentation Foundation и был описан способ его применения для разработки графического интерфейса программы.

Глава 4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

4.1 Введение

В данной главе проанализированы трудовые и денежные затраты, необходимые для проектирования и разработки программного комплекса для обработки данных на буровой установке.

4.2 Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

4.2.1 Потенциальные потребители результатов исследования

Разрабатываемое программное обеспечение предназначено для использования вместе с уже существующим оборудованием, производимым на инженерно-производственном центре «Геофит». Следовательно, потенциальными потребителями являются нынешние и будущие клиенты производственного центра.

4.2.2 Анализ конкурентных решений

Разрабатываемое программное обеспечение является заменой уже существующей программы. Подробнее о причинах необходимости разработки новой программы описано в главе 1. В текущем разделе разработанное решение будет сравнено с оригинальной версией программы, которую оно заменяет. Для сравнения используется оценочная карта, представленная в таблице 2.

Таблица 2 – Оценочная карта для сравнения старой и новой разработки

Критерии оценки	**Вес кри-терия**	**Баллы**		**Конкуренто-способность**	
		Нов.	**Ст.**	**Нов.**	**Ст.**
1	2	3	4	5	6
Технические критерии оценки реусурсоэффективности					
1. Функциональная мощность	0,10	5	3	0,5	0,3
2. Удобство в эксплуатации	0,15	4	3	0,6	0,45

Критерии оценки	Вес критерия	Баллы		Конкурентоспособность	
		Нов.	Ст.	Нов.	Ст.
1	2	3	4	5	6
3. Производительность	0,10	4	3	0,4	0,3
4. Возможность настройки в полевых условиях	0,20	5	2	1	0,4
5. Качество внутренней документации	0,15	4	2	0,6	0,3
Экономические критерии оценки эффективности					
1. Простота решения технических проблем	0,15	5	1	0,75	0,15
2. Простота расширения функционала	0,15	5	1	0,75	0,15
Итого	1	32	15	4,6	2,05

На основание приведенных в таблице 2 расчетов можно сделать вывод, что разработанная система имеет высокий уровень конкурентоспособности по сравнению со старой системой. Наиболее важным критерием была выбрана возможность настройки в полевых условиях. Следующими по важности критериями являются простота расширения функциональности программы и решения технических проблем, а также связанное с ними качество внутренней документации. Это обосновывается необходимостью изменения функциональности программы по требованиям со стороны клиентов.

4.2.3 Технология QuaD

Для анализа целесообразности разработки системы используется технология Quality Advisor (QuaD). Оценочная карта для технологии QuaD представлена в таблице 3.

Оценка качества и перспективности по технологии QuaD:

$$П_{ср} = \sum В_i Б_i \cdot 100 = 89,$$

где $П_{ср}$ – средневзвешенное значение показателей качества и перспективности научной разработки;

$В_i$ – вес показателя (в долях единицы);

$Б_i$ – средневзвешенное значение i-го показателя.

Таблица 3 – Оценочная карта для анализа технологией QuaD

Критерии оценки	**Вес критерия**	**Баллы**	**Максимальный балл**	**Относительное значение (3/4)**	**Средневзвешенное значение (5х2)**
1	2	3	4	5	6
Технические критерии оценки реусурсоэффективности					
1. Функциональная мощность	0,10	90	100	0,9	0,09
2. Удобство в эксплуатации	0,15	80	100	0,8	0,12
3. Производительность	0,10	80	100	0,8	0,08
4. Возможность настройки в полевых условиях	0,20	95	100	0,95	0,19
5. Качество документации	0,15	85	100	0,85	0,1275
Экономические критерии оценки эффективности					
1. Простота решения технических проблем	0,15	90	100	0,9	0,135
2. Простота расширения функционала	0,15	95	100	0,95	0,1425
Итого	1	615	700	6,15	0,89

Из результатов проведенного анализа следует, что разработанная система считается перспективной по технологии QuaD.

4.2.4 SWOT-анализ

SWOT-анализ является одним из самых часто используемых методов анализа в менеджменте и маркетинге. Он дает ясное представление о текущей ситуации, а также помогает понять, какие действия необходимо предпринять для максимизации возможностей проекта и нейтрализации слабых сторон и угроз. Матрица SWOT-анализа представлена в таблице 4.

Таблица 4 – SWOT-анализ научно-исследовательского проекта

	Сильные стороны С1. Удобство в эксплуатации. С2. Функциональная мощность. С3. Наличие хорошей документации на исходный код.	**Слабые стороны** Сл1. Затраты на реализацию. Сл2. Отсутствие опыта использования в полевых условиях.
Возможности В1. Поддержка новых протоколов передачи данных. В2. Анализ ошибок путем повторного воспроизведения всех операций из лог-файла.	– Спроектированная модульная архитектура программы и наличие документации на код позволяют расширить его новыми протоколами.	– Анализ лог-файла поможет выявить проблемы, которые трудно выявить даже при тщательном тестировании.
Угрозы У1. Потеря доступа к среде разработки в связи с санкциями. У2. «Feature creep» - постоянное и излишнее наращивание функциональности, ведущее к ошибкам в работе.	– Программа использует фреймворк WPF, который не подвержен влиянию санкций. – Модульная архитектура программы позволяет реализовывать различную функциональность отдельно друг от друга, уменьшая влияние новой функциональности на работу существующей.	– «Feature creep» ведет к повышению стоимости разработки и снижению качества продукта.

4.3 Планирование научно-исследовательских работ

4.3.1 Структура работ в рамках научного исследования

В данной части составлен перечень этапов и работ в рамках выполнения выпускной квалификационной работы с распределением исполнителей по видам работ. В соответствии с видами работ участниками планирования были выбраны:

- научный руководитель (НР);
- научный консультант от компании (НК);
- студент (С).

Результат планирования представлен таблице 5.

Таблица 5 – Перечень этапов, работ и распределение исполнителей

Основные этапы	№ раб.	Содержание работ	Должность исполнителя
Подготовительный	1	Выбор темы ВКР	НР, НК, С
	2	Анализ исходной программы	С
	3	Составление технического задания	НР, НК, С
Основной	4	Проектирование системы	С
	5	Проектирование графического интерфейса	С
	6	Утверждение результатов проектирования	НР, НК, С
	7	Разработка системы	С
Составление отчета	8	Подготовка главы «Социальная ответственность»	С
	9	Подготовка главы «Финансовый менеджмент»	С
	10	Составление пояснительной записки к ВКР	НР, С

4.3.2 Определение трудоемкости выполнения работ

В большинстве случаев основную часть стоимости разработки образуют трудовые затраты, поэтому важным моментом является определение трудоемкости работ каждого из участников. Ожидаемая трудоемкость находится по формуле

$$t_{\text{ож}i} = \frac{3t_{\min i} + 2t_{\max i}}{5},$$

где $t_{\text{ож}i}$ – ожидаемая трудоемкость выполнения i-ой работы, чел.-дн.;

$t_{\min i}$ – минимально возможная трудоемкость выполнения заданной i-ой работы (оптимистическая оценка), чел.-дн.;

$t_{\max i}$ – максимально возможная трудоемкость выполнения заданной i-ой работы (пессимистичная оценка), чел.-дн.

Продолжительность каждой работы в днях рассчитывается по формуле

$$T_{p_i} = \frac{t_{\text{ож}i}}{\text{Ч}_i},$$

где T_{p_i} – продолжительность одной работы, раб. дн.;

$t_{\text{ож}i}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.;

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

Результаты расчетов трудоемкости представлены в таблице 6.

Таблица 6 – Временные показатели для научного исследования

№ раб.	Содержание работ	Трудоемкость работ								
		$t_{min i}$			$t_{max i}$			$t_{ожi}$		
		НР	НК	С	НР	НК	С	НР	НК	С
1	Выбор темы ВКР	1	1	2	2	2	4	1,4	1,4	2,8
2	Анализ исходной программы			18			24			20,4
3	Составление технического задания	1	2	3	2	3	6	1,4	2,4	4,2
4	Проектирование системы			14			21			16,8
5	Проектирование графического интерфейса			3			7			4,6
6	Утверждение результатов проектирования	1	1	1	1	2	2	1	1,4	1,4
7	Разработка системы			21			28			23,8
8	Подготовка главы «Социальная ответственность»			1			2			1,4
9	Подготовка главы «Финансовый менеджмент»			1			2			1,4
10	Составление пояснительной записки к ВКР	2		3	3		5	2,4		3,8

4.3.3 Разработка графика проведения научного исследования

При выполнении краткосрочных работ, вида выпускной квалификационной работы, наиболее удобным и наглядным видом представления графика проведения работы является диаграмма Ганта. Для удобства ее построения следует перевести длительность каждого из этапов работ в календарные дни по формуле

$$T_{k_i} = T_{p_i} \cdot k_{\text{кал}},$$

где T_{k_i} – продолжительность выполнения i-й работы в календарных днях;

T_{p_i} – продолжительность одной работы в днях;

$k_{\text{кал}}$ – коэффициент календарности.

Коэффициент календарности определяется по формуле

$$k_{\text{кал}} = \frac{T_{\text{кал}}}{T_{\text{кал}} - T_{\text{вых}} - T_{\text{пр}}},$$

где $T_{\text{кал}}$ – количество календарных дней в году;

$T_{\text{вых}}$ – количество выходных дней в году;

$T_{\text{пр}}$ – количество праздничных дней в году.

В 2023 году при пятидневной рабочей неделе 365 календарных дней и 118 выходных/праздничных, следовательно для 2023 года $k_{\text{кал}} = 1{,}478$.

Результаты расчетов трудоемкости представлены в таблице 7. Диаграмма Ганта представлена в таблице 8.

Таблица 7 – Временные показатели для научного исследования

№ раб.	**Содержание работ**	**Длит. в раб. днях (T_{p_i})**			**Длит. в кал. Днях (T_{k_i})**		
		НР	**НК**	**С**	**НР**	**НК**	**С**
1	Выбор темы ВКР	0,467	0,467	0,933	1	1	2

№ раб.	Содержание работ	Длит. в раб. днях (T_{p_i})			Длит. в кал. Днях (T_{k_i})		
		НР	НК	С	НР	НК	С
2	Анализ исходной программы			20,4			31
3	Составление технического задания	0,467	0,8	1,4	1	2	3
4	Проектирование системы			16,8			25
5	Проектирование графического интерфейса			4,6			7
6	Утверждение результатов проектирования	0,333	0,467	0,467	1	1	1
7	Разработка системы			23,8			36
8	Подготовка главы «Социальная ответственность»			1,4			3
9	Подготовка главы «Финансовый менеджмент»			1,4			3
10	Составление пояснительной записки к ВКР	1,2		1,9	2		3

Таблица 8 – Диаграмма Ганта

№ раб.	Содержание работ	Испол-нители	T_{k_i}	Июнь				Июль				Март				Апрель				Май			
1	Выбор темы ВКР	НР	1																				
		НК	1																				

№ раб.	Содержание работ	Испол-нители	T_{k_i}	Июнь				Июль				Март				Апрель				Май			
		С	2																				
2	Анализ исходной программы	С	31																				
3	Составление технического задания	НР	1																				
		НК	2																				
		С	3																				
4	Проектирование системы	С	25																				
5	Проектирование графического интерфейса	С	7																				
6	Утверждение результатов проектирования	НР	1																				
		НК	1																				
		С	1																				
7	Разработка системы	С	36																				
8	Подготовка главы «Социальная ответственность»	С	3																				
9	Подготовка главы «Финансовый менеджмент»	С	3																				
10	Составление пояснительной записки к ВКР	НР	2																				
		С	3																				

4.3.4 Бюджет научно-технического исследования (НТИ)

В состав затрат на разработку программного обеспечения включается стоимость всех расходов, необходимых для реализации работ, составляющих содержание данной разработки. Возможные расходы разделяются на следующие категории:

- материальные затраты;
- затраты на специальное оборудование;
- основная заработная плата исполнителей;
- дополнительная заработная плата исполнителей;
- отчисления во внебюджетные фонды;

- затраты на научные и производственные командировки;
- контрагентные затраты;
- накладные расходы.

4.3.4.1 Расчет материальных затрат НТИ

В материальных затратах учтены только затраты на канцелярские товары. Материальные затраты представлены в таблице 9. Затраты на оборудование указаны в следующем разделе.

Таблица 9 – Материальные затраты

Наименование	**Количество**	**Цена за ед., руб**	**Затраты, руб.**
Шариковая ручка	3	60	180
Бумага А4, 500 листов	1	400	400
		Итого:	580

4.3.4.2 Расчет затрат на специальное оборудование

Специальное оборудование и ПО состоит из следующих пунктов:

- ПК и периферийные устройства;
- лицензия на ПО для разработки.

Производственный центр уже имел в наличии свободный ПК с периферийными устройствами, однако для реализации проекта были закуплены обновленные комплектующие к нему. Также требуется закупить лицензию на среду разработки Microsoft Visual Studio 2022 Professional, которая выдается конкретному пользователю.

Амортизационные отчисления для рассматриваемого проекта включают в себя амортизацию используемого оборудования и ПО за время использования. Амортизационные отчисления для ПК рассчитываются по формуле:

$$C_{\text{ам}} = \frac{\text{Н}_\text{А}\text{Ц}_\text{ОБ}}{F_\text{Д}} \cdot t_{\text{рф}} n,$$

Где $Н_А$ – норма амортизации (величина, обратная сроку амортизации оборудования);

$Ц_{ОБ}$ – цена оборудования;

$F_Д$ – действительный годовой фонд рабочего времени;

$t_{рф}$ – время работы компьютерной техники;

n – число задействованных единиц техники.

Согласно постановлению Правительства РФ от 01.01.2002 N 1 «О Классификации основных средств, включаемых в амортизационные группы», срок амортизации персонального компьютера составляет 3 года, следовательно для ПК $Н_А = 0{,}33$.

Примем, что ПК активно используется только $3/4$ рабочего дня, а остальное время на нем открыты только легковесные программы, не нагружающие его.

Среда разработки Microsoft Visual Studio 2022 Professional, в отличие от оборудования, предоставляется по подписке и имеет четкий срок окончания действия, вне зависимости от действительного времени использования. Тогда амортизационные отчисления для единицы ПО, предоставляемого по подписке, рассчитываются по формуле:

$$C_{ам} = Н_А Ц_{ОБ}.$$

В случае с годовой подпиской на среду разработки, $Н_А = 1$.

Расчет затрат на амортизационные отчисления представлен в таблице 10.

Таблица 10 – Расчет амортизационных затрат

Наименование	$Н_А$	**Стоимость**	**Время работы техники, ч.**	**Действ. год. фонд. р. вр., ч.**	**Аморт. отчисл., руб.**
Накопитель данных SSD, 500 ГБ	0,33	5 000	1482	1976	1238

Наименование	$Н_А$	Стоимость	Время работы техники, ч.	Действ. год. фонд. р. вр., ч.	Аморт. отчисл., руб.
Оперативная память, 32 ГБ	0,33	7 000	1482	1976	1733
Microsoft Visual Studio 2022 Professional, подписка на 1 год	1	17 792	-	-	17792
				Итого:	**20762**

4.3.4.3 Основная заработная плата исполнителей

Расчет основной заработной платы выполняется на основе трудоемкости выполнения каждого этапа и величины месячного оклада (МО) исполнителя. В расчетах, представленных в данной работе, МО научного руководителя составляет 30 000 руб. МО научного консультанта от организации составляет 50 000 руб. МО студента, работающего в организации, составляет 30 000 руб. Политика компании запрещает разглашение данных о доходах сотрудников, поэтому были выбраны произвольные значения для демонстрации процесса расчета.

Основная заработная плата рассчитывается по формуле:

$$З_{осн} = \frac{З_м М}{F_д} \cdot T_р,$$

где $З_{осн}$ – основная заработная плата, руб.;

$З_м$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года;

$F_д$ – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн.;

$T_р$ – продолжительность работ, выполняемых научно-техническим работником, раб. дн.

Для пятидневной рабочей недели и отпуске в 24 раб. дн, $М = 11{,}2$ месяца. $F_д$ в 2023 году для пятидневной рабочей недели равен 247 дням.

Месячный должностной оклад сотрудника рассчитывается по формуле:

$$З_м = З_{тс} \cdot (1 + k_{пр} + k_д) \cdot k_р,$$

где $З_{тс}$ – заработная плата по тарифной ставке, руб;

$k_{пр}$ – премиальный коэффициент, равный 0,3;

$k_д$ – коэффициент доплат и надбавок;

$k_р$ – районный коэффициент, равный 1,3 для Томска.

Расчет основной заработной платы представлен в таблице 11.

Таблица 11 – Расчет основной заработной платы

Исполнители	$З_{тс}$	$k_{пр}$	$k_д$	$k_р$	$З_м$	$T_р$	$З_{осн}$
Науч. рук.	15385	0,3	0,2	1,3	30000	2,47	3355
Науч. конс.	29586	0,3	0	1,3	50000	1,73	3930
Студент	17752	0,3	0	1,3	30000	73,10	99440
						Итого:	106725

4.3.4.4 Дополнительная заработная плата

Дополнительная заработная плата включает заработную плату не за отработанное рабочее время, а заработную плату, гарантированную действующим законодательством. Дополнительная заработная плата рассчитывается по формуле:

$$З_{доп} = k_{доп} \cdot З_{осн},$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,12).

Расчет дополнительной заработной платы представлен в таблице 12.

Таблица 12 – Расчет дополнительной заработной платы

Исполнители	$З_{осн}$	$k_{доп}$	$З_{доп}$
Науч. рук.	3355	0,12	403
Науч. конс.	3930	0,12	472
Студент	99440	0,12	11933
		Итого:	12807

4.3.4.5 Отчисления во внебюджетные фонды

Величина отчислений во внебюджетные фонды рассчитывается по формуле:

$$З_{\text{внеб}} = k_{\text{внеб}} \cdot (З_{\text{осн}} + З_{\text{доп}}),$$

где $k_{\text{внеб}}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.). На 2023 год этот коэффициент составляет 30%.

Расчет отчислений во внебюджетные фонды представлен в таблице 13.

Таблица 13 – Расчет дополнительной заработной платы

Исполнитель	**Основная заработная плата, руб**	**Дополнительная заработная плата, руб**
Науч. рук.	3355	403
Науч. конс.	3930	472
Студент	99440	11933
Коэфф. Отчислений	30%	
Итого		
Науч. рук.	1127	
Науч. конс.	1320	
Студент	33412	

4.3.4.6 Расчет затрат на научные и производственные командировки

Необходимости в научных и производственных командировках, а следовательно, и в расчете затрат на них, нет.

4.3.4.7 Контрагентные расходы

К работе контрагенты не привлекались.

4.3.4.8 Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов, оплата услуг связи, электроэнергии и т.д. Расчет накладных расходов рассчитывается по формуле:

$$З_{\text{накл}} = \sum З_i k_{\text{нр}},$$

где $З_{накл}$ – накладные расходы, руб.;

$З_i$ – расходы на пункт i этого раздела, руб.;

$k_{нр}$ – коэффициент, учитывающий накладные расходы. В данной работе он берется в размере 16%.

$$З_{накл} = 0{,}16 \cdot (580 + 20762 + 106725 + 12807 + 33412) = 27886 \text{ руб}$$

4.3.4.9 Формирование бюджета затрат проекта

Рассчитанная величина затрат научно-исследовательской работы является основой для формирования бюджета затрат проекта. Определение бюджета затрат на научно-исследовательский проект по каждому варианту исполнения приведен в таблице 14.

Таблица 14 – Расчет бюджета затрат НТИ

Наименование статьи	Сумма, руб.	Примечание
1. Материальные затраты	580	Пункт 2.4.1
2. Специальное оборудование	20762	Пункт 2.4.2
3. Основная заработная плата	106725	Пункт 2.4.3
4. Дополнительная заработная плата	12807	Пункт 2.4.4
5. Отчисления во внебюджетные фонды	33412	Пункт 2.4.5
6. Накладные расходы	27886	Пункт 2.4.8
7. Бюджет затрат НТИ	202172	

4.3.5 Определение ресурсной (ресурсосберегающей) финансовой, бюджетной, социальной и экономической эффективности исследования

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный финансовый показатель разработки определяется как:

$$I_{финр}^{исп.i} = \frac{Ф_{pi}}{Ф_{max}},$$

где $I_{\text{финр}}^{\text{исп.}i}$ – интегральный финансовый показатель разработки;

$\Phi_{\text{р}i}$ – стоимость i-го варианта исполнения;

Φ_{max} – максимальная стоимость исполнения научно-исследовательского проекта (в т.ч. аналоги).

Интегральный показатель ресурсоэффективности вариантов исполнения объекта исследования можно определить следующим образом:

$$I_{\text{р}-\text{исп}i} = \sum a_i b_i \,,$$

где $I_{\text{р}-\text{исп}i}$ - интегральный показатель ресурсоэффективности для i-го варианта исполнения;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b_i^a, $b_i^{\text{р}}$ – бальная оценка i-го варианта исполнения разработки, устанавливающаяся экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Для сравнения были выбраны варианты исполнения на основе различных языков программирования и фреймворков для проектирования графического интерфейса:

1. C#, Windows Presentation Foundation;
2. C#, Windows Forms;
3. C++; Qt Widgets.

Расчет интегрального показателя ресурсоэффективности представлен в таблице 15.

Таблица 15 – Расчет интегрального показателя ресурсоэффективности

Критерии оценки	**Вес критерия**	**ВИ 1**	**ВИ 2**	**ВИ 3**
1. Удобство разработки	0,50	5	3	4
2. Производительность	0,15	4	3	4
3. Независимость от иностранного ПО	0,20	4	1	5
4. Кроссплатформенность	0,15	4	3	5
Интегральный коэффициент ресурсоэффективности		4,5	2,6	4,35

Интегральный показатель эффективности вариантов исполнения разработки определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формуле:

$$I_{\text{исп.}i} = \frac{I_{\text{р}-\text{исп}i}}{I_{\text{финр}}^{\text{исп.}i}}.$$

Сравнение интегрального показателя эффективности вариантов исполнения разработки позволит определить сравнительную эффективность проекта и выбрать наиболее целесообразный вариант из предложенных. Сравнительная эффективность проекта ($Э_{\text{ср}}$):

$$Э_{\text{ср}} = \frac{I_{\text{исп.}i}}{I_{\text{исп.}j}}.$$

Разработка всех трех вариантов исполнения займет одинаковое время, поэтому $I_{\text{финр}}^{\text{исп.}i} = 1$ для каждого варианта исполнения.

Сравнительная эффективность разработки представлена в таблице 16.

Таблица 16 - Сравнительная эффективность разработки

№	Показатели	ВИ 1	ВИ 2	ВИ 3
1	Интегральный финансовый показатель разработки	1	1	1
2	Интегральный показатель ресурсоэффективности разработки	4,5	2,6	4,35
3	Интегральный показатель эффективности	4,5	2,6	4,35
4	Сравнительная эффективность вариантов исполнения	1	0,578	0,967

Из результатов расчетов следует, что наиболее эффективным вариантом решения поставленной задачи с позиции финансовой и ресурсной эффективности является вариант 1 (Windows Presentation Foundation).

4.4 Вывод по разделу

В результате выполнения работы по разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» был подсчитан теоретический бюджет проекта, равный 202 172 руб. Более половины расходов приходится на одну лишь оплату труда. Следующая по стоимости статья –

отчисления во внебюджетные фонды, что также связано с оплатой труда. Наибольшей статьей расходов, не связанной напрямую с оплатой труда, являются накладные расходы, а за ними следует закупка оборудования и ПО.

Была определена ресурсная и финансовая эффективность проекта. Были сравнены три варианта исполнения, реализация которых занимает одно и тоже время и которые стоят одинаковы. Был получен результат, что выбранный вариант исполнения является самым эффективным.

Глава 5 Социальная ответственность

5.1 Введение

Проектирование и разработка программных систем, как и любая другая трудовая деятельность, подвержена влиянию различных вредных и опасных производственных факторов. В данном разделе рассматриваются вопросы выполнения требований к безопасности и гигиене труда, к охране окружающей среды и ресурсосбережению.

Темой выпускной квалификационной работы является разработка программного комплекса для обработки данных на буровой установке. В задачи программного комплекса входит прием данных с телеметрической системы, проведение расчетов и передача данных во внешние системы. Разрабатываемое программное обеспечение предназначено для использования вместе с уже существующим оборудованием, производимым на инженерно-производственный центре «Геофит». Следовательно, потенциальными пользователями являются нынешние и будущие клиенты производственного центра, использующие выпускаемые производственным центром системы для проведения направленного бурения скважин и геофизических исследований в процессе бурения на буровой установке.

Целью раздела является выявление и анализ вредных и опасных факторов, которые могут повлиять на здоровье и общее самочувствие студента при выполнении выпускной квалификационной работы.

Реализация проекта по разработке программного комплекса для обработки данных на буровой установке проводилась в помещении офисного типа на территории инженерно-производственного центра «Геофит».

Характеристика рабочего места:

- размеры: длина – 5 м, ширина – 5 м, высота – 3 м;
- площадь помещения – 25 м^2;
- объем помещения – 75 м^3;

- установлен кондиционер, имеется естественная вентиляция в виде двух открываемых окон;
- установлено искусственное освещение, имеется естественное освещение.

Помещение оборудовано тремя рабочими местами с компьютером и двумя мониторами каждое. Максимальное количество постоянных сотрудников – 3.

5.2 Правовые и организационные вопросы обеспечения безопасности

На территории Российской Федерации трудовая деятельность регулируется Трудовым Кодексом. Статья 91 ТК РФ указывает, что нормальная продолжительность рабочего времени не может превышать 40 часов в неделю. В статье 108 ТК РФ указано, что работнику должен предоставляться перерыв от 30 минут до 2 часов для отдыха и приема пищи в течение рабочего дня. Статья 163 ТК РФ указывает, что работодатель обязан обеспечить нормальные условия для выполнения работы. Помещения и оборудование обязаны находиться в исправном состоянии, а условия труда обязаны соответствовать требованиям охраны труда и безопасности производства.

Обязательные требования к обеспечению безопасных для человека условий труда указаны в СП 2.2.3670-20 «Санитарно-эпидемиологические требования к условиям труда». К офисному рабочему месту оператора ПЭВМ применимы следующие разделы:

- VI «Требования к организации технологических процессов и рабочих мест»;
- XXII «Требования к организации работ с персональными электронными вычислительными машинами и копировально-множительной техникой».

Пункт 6.3 главы VI требует на сидячем рабочем месте пространство для размещения ног высотой не менее 600 мм, глубиной не менее 500 мм на уровне

колен и 600 мм на уровне стоп. Шириной не менее 500 мм. Рассматриваемое рабочее место соответствует этому требованию.

Пункт 249 главы XXII регламентирует минимальную площадь на одно постоянное рабочее место на базе плоских жидкокристаллических экранов не менее 4.5 м^2. Для рассматриваемого рабочего места этот показатель составляет 8,3 м^2. Пункт 250 требует установку жалюзи на окна, которые в рассматриваемом помещении имеются.

Требования к сидящему рабочему месту устанавливаются ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования».

Конструкция рабочего места должна обеспечивать оптимальное положение работающего, что достигается регулированием высоты рабочей поверхности, сиденья и пространства для ног. Требования к регулируемому креслу описаны в ГОСТ 21889-76 «Система человек-машина. Кресло человека-оператора. Общие эргономические требования».

Органы управления (в данном случае, клавиатура и мышка) должны быть размещены так, чтобы при их использовании не было перекрещивания рук. Классическое расположение клавиатуры посередине рабочего места и мышки сбоку от нее удовлетворяет этому требованию.

Основные средства отображения информации (основной монитор) следует располагать в вертикальной плоскости под углом ±15° от нормальной линии взгляда и в горизонтальной плоскости под углом ±15° от сагиттальной плоскости. Часто используемые средства отображения информации (второй монитор) следует аналогичным требованиям, но с углом ±30°. Рассматриваемое рабочее место удовлетворяет этим требованиям.

5.3 Производственная безопасность

На оператора ПЭВМ по время исполнения своих трудовых обязанностей влияют различные факторы, перечисленные в ГОСТ 12.0.003-

2015 «Опасные и вредные производственные факторы. Классификация» и представленные в таблице 17.

Таблица 17 – Возможные опасные и вредные факторы на рабочем месте оператора ПЭВМ

Факторы (ГОСТ 12.0.003-2015)	**Нормативные документы**
Повышенный уровень шума	Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда», пункт 5.4. «Виброакустические факторы» СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
Наличие электромагнитных полей промышленных и радиочастот	Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда», пункт 5.7. «Неионизирующие электромагнитные поля и излучения» ГОСТ 12.1.006-84 «ССБТ. Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению» СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
Отсутствие или недостаток необходимого искусственного освещения	СП 52.13330.2016 «Естественное и искусственное освещение»
Повышенная пульсация светового потока	

Факторы (ГОСТ 12.0.003-2015)	Нормативные документы
Нервно-психические перегрузки, связанные с напряженностью трудового процесса	Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда», пункт 5.10. «Тяжесть и напряженность трудового процесса»
Факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов	ГОСТ 12.1.019-2017 «ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты» ГОСТ 12.1.030-81 «ССБТ. Электробезопасность. Защитное заземление. Зануление»

5.3.1 Повышенный уровень шума

Источников повышенного уровня шума на рассматриваемом рабочем месте возможно несколько:

- шум систем охлаждения рабочих компьютеров при выполнении ресурсоемких задач (например, компиляция программного кода);
- шум включенного кондиционера;
- шум погрузочных машин на территории предприятия при открытом окне.

Повышенный уровень шума приводит к снижению внимания, повышению числа ошибок при выполнении работы и оказывает влияние на сбор информации и аналитические процессы, являющиеся основной деятельностью программиста. Длительное влияние интенсивного шума оказывает влияние на весь организм и может привести к возникновению сердечно-сосудистых заболеваний и потере слуха.

Согласно СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания», пункт 35, нормальным эквивалентным уровнем звука (уровнем воздействия на работающего за рабочую смену) является уровень 80 дБА.

Для снижения уровня шума возможно выполнить следующие действия.

- Заменить систему охлаждения ПК на систему с радиатором и вентилятором большего размера. При одинаковом воздушном потоке больший вентилятор крутится с меньшей скоростью и создает меньше шума.
- Закрыть окно на время погрузочных операций.
- Оборудовать помещение звукопоглощающими занавесями.

По субъективным ощущениям в худшей ситуации (все три компьютера загружены на полную и включен кондиционер) дискомфорта от уровня шума не наблюдается и принимается, что рабочее место соответствует норме.

5.3.2 Наличие электромагнитных полей промышленных и радиочастот

Основным источником электромагнитных полей вблизи работника являются компьютер и мониторы, обладающие импульсными блоками питания и полупроводниковыми элементами, работающими на частотах в несколько ГГц. Длительное воздействие электромагнитных полей приводит к головным болям, снижению памяти, нарушению ритма и замедлению частоты сердечных сокращений.

Согласно СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания», пункт 38, предельно допустимый уровень напряженности электромагнитного поля промышленной частоты 50 Гц на рабочем месте в течение всей смены устанавливается равным 5 кВ/м. Так как все оборудование питается от сети 220 В 50 Гц, напряженность является ниже 5 кВ/м, и следовательно находится в допустимых пределах. Все оборудование обладает металлических корпусом или экраном внутри пластикового корпуса, поэтому электромагнитные поля радиочастотного диапазона не оказывают влияния на оператора ПЭВМ.

5.3.3 Отсутствие или недостаток необходимого искусственного освещения

Недостаточная освещенность при работе с ПЭВМ приводит к повышенной нагрузке зрительного аппарата и головным болям. Продолжительная работа в условиях недостаточной освещенности приводит к проблемам со зрением.

Требования к естественному и искусственному освещению указаны в СП 52.13330.2016 «Естественное и искусственное освещение». Работу программиста можно отнести к IV-в разряду зрительной работы: размер объекта от 0,5 до 1,0 мм (размер нескольких пикселей экрана), контраст и фон средний. Требования к освещению при работе этого типа представлены в таблице 18.

Таблица 18 – Требования к освещению помещений промышленных предприятий разряда IV-в

Освещенность, лк			
При системе комбинированного освещения		При системе общего освещения	Коэффициент пульсации, не более
Всего	В том числе от общего		
400	200	200	20%

При использовании компьютера дополнительно требуется, чтобы яркость экрана и уровень освещения были близки, так как яркий свет в области периферийного зрения вызывает утомляемость глаз. Во время начала работы с ПК была произведена настройка яркости мониторов, чтобы при отображении белого изображения их яркость совпадала с уровнем освещения помещения.

5.3.4 Повышенная пульсация светового потока

Повышенная пульсация светового потока оказывает негативное влияние на человека. Она приводит к повышенной нагрузке зрительного аппарата, головным болям и снижению работоспособности человека.

В СП 52.13330.2016 «Естественное и искусственное освещение» указано, что в помещениях разряда IV-в максимально допустима пульсация 20% (таблица 3). Коэффициент пульсации зависит от используемого источника освещения. Высокий коэффициент пульсации имеют люминесцентные лампы с электромагнитными пускорегулирующими аппаратами, пульсирующие с частотой 100 Гц, и некоторые регулируемые светодиодные осветительные приборы.

В СП 52.13330.2016 «Естественное и искусственное освещение» указано, что пульсация освещенности при частоте выше 300 Гц не оказывает влияния на общую и зрительную работоспособность. Для снижения влияния пульсации светового потока на деятельность человека необходимо использовать осветительные приборы с коэффициентом пульсации ниже 20% или с частотой пульсации выше 300 Гц. В случае с люминесцентными лампами следует использовать электронные пускорегулирующие автоматы, а для светодиодных источников освещения – регулировку яркости с частотой 300 Гц или выше. Освещение на рассматриваемом рабочем месте удовлетворяет этим требованиям.

5.3.5 Нервно-психические перегрузки, связанные с напряженностью трудового процесса

Напряженность трудового процесса характеризуется эмоциональной нагрузкой на организм при труде, требующем преимущественно интенсивной работы мозга по получению и переработке информации. Длительная работа в условиях нервно-психических перегрузок может привести к стрессу и сердечно-сосудистым заболеваниям.

Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда» в пункте 5.10 «Тяжесть и напряженность трудового процесса» разделяет труд по показателям напряженности трудового процесса на классы. Проектирование и разработка программного обеспечения по содержанию работы относится к напряженному труду первой (решение сложных задач с выбором по известным алгоритмам) и второй (эвристическая/творческая деятельность, требующая поиска алгоритма) степеням в зависимости от текущей работы. Для избегания перегрузок требуется построение производственного процесса определенным образом. Планирование задач должно выполняться отдельно от работы по разработке системы. Процесс разработки разделяется на проектирование и разработку. В процессе проектирования система разбивается на простые модули, информацию об одном из которых можно держать в голове.

5.3.6 Факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов

Компьютер и периферийные устройства во время работы подключены к электросети 220 В, что создает риск поражения электрическим током, приводящий к смерти. Для избегания этого требуется соблюдать технику электробезопасности, регламентируемую ГОСТ 12.1.019-2017 «ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты».

Наиболее вероятно поражение током при контакте оголенных токоведущих частей с телом человека, поэтому все металлические части устройств (например, корпуса) должны быть соединены с защитным заземлением, как указано в ГОСТ 12.1.030-81 «ССБТ. Электробезопасность. Защитное заземление. Зануление». Провода и удлинители перед использованием должны проверяться на предмет наличия повреждений в изоляции.

5.4 Экологическая безопасность

В процессе разработки и использования программного обеспечения самого по себе не производятся отходы. В этом разделе работы будет рассмотрена только утилизация компьютерной техники. Процесс утилизации современной компьютерной техники описан в ГОСТ Р 55102-2012 «Обращение с отходами. Руководство по безопасному сбору, хранению, транспортированию и разборке отработавшего электротехнического и электронного оборудования, за исключением ртутьсодержащих устройств и приборов». Предпочтительным способом утилизации является разборка и повторное использование, а сжигание должно рассматриваться в последнюю очередь. Неверная утилизация компьютерной техники (например, захоронение на свалках или сжигание в неподготовленном месте) приводит к загрязнению атмосферы, гидросферы и литосферы.

Основным источником загрязнения атмосферы являются вещества, выделяемые при сжигании пластиков, которые используется повсеместно для корпусов компьютерной техники. Для предотвращения этого требуется следовать требованиям по переработке техники и ни в коем случае не сжигать отходы.

Загрязнение гидросферы и литосферы происходит в результате захоронения или сжигания техники без извлечения компонентов, содержащие опасные вещества. Старые компьютерные мониторы используют люминесцентные лампы для подсветки экрана, которые содержат в себе ртуть, и потому относятся к первому классу отходов, самому опасному. Современные ноутбуки и иная портативная техника содержат литий-ионные аккумуляторы, которые относятся ко второму классу опасности отходов. Старая техника может содержать никель-кадмиевые аккумуляторы, относящиеся к первому классу отходов. Для предотвращения загрязнения окружающей среды требуется отделять опасные отходы от техники и перерабатывать отдельно, согласно соответствующим руководствам.

5.5 Безопасность в чрезвычайных ситуациях

Процесс разработки программного обеспечения осуществлялся в офисном помещении на территории завода. Для офисов характерными чрезвычайными ситуациями являются:

- природные (землетрясения, ураганы и т.д.);
- техногенные (транспортные аварии, пожары и т.д.);
- биолого-социальные (эпидемии).

Наиболее характерной ЧС является пожар вследствие электрической неисправности. Согласно Федеральному закону от 22.07.2008 N 123-ФЗ (ред. От 30.04.2021) «Технический регламент о требованиях пожарной безопасности» такой пожар будет иметь класс А - пожары твердых горючих веществ и материалов, так как материалом горения будет являться пластик и дерево, из которого изготовлена техника и мебель. Здание обладает следующими средствами первичного пожаротушения: огнетушители и пожарные краны на каждом этаже.

Следующие основные причины могут привести к пожару:

- короткое замыкание;
- перегрузка электросетей, приводящая к нагреву проводки и возгоранию изоляции или короткому замыканию;
- запуск неисправного оборудования.

Для предупреждения пожара и избегания жертв в случае ЧС требуется предпринять следующие меры:

- проводить инструктаж по технике безопасности;
- проводить учебные тревоги, чтобы сотрудники могли сориентироваться в случае ЧС;
- обеспечить наличие плана эвакуации для каждого офисного помещения;
- обеспечить наличие огнетушителей в близкой доступности от каждого офисного помещения.

В случае возникновения пожара необходимо вызвать пожарную службу по телефону 101 или нажатием кнопки пожарной тревоги. При отсутствии прямой угрозы жизни можно попытаться ликвидировать возгорание самому, используя огнетушитель. Иначе следует следовать плану эвакуации из здания.

5.6 Вывод по разделу

В результате проведенного анализа был рассмотрен процесс проектирования и разработки программной системы с правовой и экологической точек зрения, также со стороны обеспечения безопасности при чрезвычайных ситуациях. Было выявлено соответствие фактических значений потенциально возможных вредных и опасных производственных факторов нормативным значениям.

Согласно правилам устройства электроустановок рассматриваемое помещение соответствует категории помещений без повышенной опасности. В помещении производится только работа с ПЭВМ, поэтому персонал относится к группе II по электробезопасности. Осуществляемая деятельность по СанПиН 1.2.3685-21 относится к категории работ Iа (энергозатраты до 139 Вт). По взрывопожарной и пожарной опасности рассматриваемое помещение относится к категории В1 (пожароопасность) в связи с наличием горючих материалов (мебель и техника).

Согласно критериям, утвержденным постановлением Правительства РФ от 31.12.2020 N 2398, рассматриваемая ПЭВМ относится к IV категории (объекты, оказывающие минимальное негативное воздействие на окружающую среду).

Заключение

На начальном этапе работы был проведен анализ существующей версии программы «GS Terminal», предназначенной для сбора данных от разных источников и обмена данными между различными подразделениями на буровой установке. Был выполнен анализ исходного кода программы, а также построены диаграммы, описывающие внутреннее устройство программы. В силу бизнес-необходимости, в существующую версию программы был внесен ряд изменений по требованиям клиентов; подготовленная новая версия программы в настоящее время проходит тестирование в полевых условиях.

В процессе анализа исходного кода программы были выявлены особенности в архитектуре программы, усложняющие внедрение новой функциональности. Результаты анализа были доложены техническим экспертам и руководству инженерно-производственного центра «Геофит», и после совместного обсуждения было принято решение спроектировать новую архитектуру программы.

На этапе проектирования на основе анализа, проведенного на начальном этапе, было составлено техническое задание на разработку новой версии программного комплекса для обработки данных на буровой установке. На основании составленного технического задания автором был представлен проект архитектуры новой версии программы в виде диаграмм IDEF0, диаграмм классов UML, диаграмм последовательности UML. Проект был утвержден для реализации.

На этапе реализации был выбран инструмент разработки графического интерфейса Windows Presentation Foundation и язык программирования C# для разработки центральных компонентов комплекса «GS Terminal 2». Были реализованы ядро системы, часть модулей системы (в частности, взаимодействие по протоколам WITS и WAKE), значительная часть

пользовательского интерфейса для конфигурирования системы в целом и отдельных модулей.

На этапе финансового анализа был рассчитан бюджет проекта и была определена ресурсная и финансовая эффективность проекта.

На этапе анализа социальной ответственности было отмечено отсутствие нарушений при выполнении выпускной квалификационной работы с правовой и экологической точек зрения, также со стороны обеспечения безопасности при чрезвычайных ситуациях.

Таким образом, в результате выполнения выпускной квалификационной работы автором была исследована предметная область обработки данных на буровой установке при использовании системы телеметрического сопровождения процесса бурения, а также разработан программный комплекс для обмена данными о процессе бурения. Данный программный комплекс создан для приема данных от множества различных систем и перенаправления их экспертам различных подразделений на буровой с использованием различных протоколов. причем то, какие именно системы участвуют в обмене данными и какие протоколы будут использоваться, может задаваться в пользовательском интерфейсе программы «GS Terminal». В разработке новой версии системы был поставлен акцент на упрощении процесса внедрения новой функциональности, регулярно возникающей на основании анализа обратной связи от пользователей и бизнес-потребностей.

Список использованных источников и литературы

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. — СПб.: Питер, 2020. — 448 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-4461-1595-2
2. Дмитриев А.Ю. Основы технологии бурения скважин: учебное пособие. – Томск: Изд-во ТПУ, 2008. – 216 с.
3. Кульчицкий В.В., Ларионов А.С., Архипов А.И. Учебное пособие «Применение технических средств контроля процессов бурения нефтегазовых скважин». М.: Издательский центр РГУ нефти и газа имени И.М. Губкина, 2010 – 151 с. 130 илл
4. Ron Baker. A primer of oilwell drilling: a basic text of oil and gas drilling, 6th ed. – Petroleum Extension Service, The University of Texas at Austin, 2001.
5. Спецификация протокола WAKE [Электронный ресурс]. URL: http://www.leoniv.diod.club/articles/wake/wake.html (дата обращения: 07.06.2023).
6. CSS Snapshot 2022 [Электронный ресурс]. URL: https://www.w3.org/TR/css-2022/ (дата обращения: 07.06.2023).
7. Data binding overview (WPF .NET) [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-7.0 (дата обращения: 07.06.2023).
8. DPI and device-independent pixels [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/windows/win32/learnwin32/dpi-and-device-independent-pixels (дата обращения: 07.06.2023).
9. ECMA-404. The JSON data interchange syntax [Электронный ресурс]. URL: https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf (дата обращения: 07.06.2023).
10. Electron [Электронный ресурс]. URL: https://www.electronjs.org/ (дата обращения: 07.06.2023).

11. Globalization for WPF [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/advanced/globalization-for-wpf?view=netframeworkdesktop-4.8 (дата обращения: 07.06.2023).
12. GNU General Public License [Электронный ресурс]. URL: https://www.gnu.org/licenses/gpl-3.0-standalone.html (дата обращения: 07.06.2023).
13. GNU Lesser General Public License [Электронный ресурс]. URL: https://www.gnu.org/licenses/lgpl-3.0-standalone.html (дата обращения: 07.06.2023).
14. How to: Use a ResourceDictionary to Manage Localizable String Resources [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/advanced/how-to-use-a-resourcedictionary-to-manage-localizable-string-resources?view=netframeworkdesktop-4.8 (дата обращения: 07.06.2023).
15. HTML. Living standard [Электронный ресурс]. URL: https://html.spec.whatwg.org/ (дата обращения: 07.06.2023).
16. Logging in C# and .NET [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/core/extensions/logging (дата обращения: 07.06.2023).
17. Microsoft.Extensions.Logging – NuGet Gallery [Электронный ресурс]. URL: https://www.nuget.org/packages/Microsoft.Extensions.Logging (дата обращения: 07.06.2023).
18. MIT License [Электронный ресурс]. URL: https://spdx.org/licenses/MIT.html (дата обращения: 07.06.2023).
19. .NET Multi-platform App UI (.NET MAUI) [Электронный ресурс]. URL: https://github.com/dotnet/maui (дата обращения: 07.06.2023).
20. Optimizing Performance: Taking Advantage of Hardware [Электронный ресурс]. URL: https://learn.microsoft.com/en-

us/dotnet/desktop/wpf/advanced/optimizing-performance-taking-advantage-of-hardware?view=netframeworkdesktop-4.8 (дата обращения: 07.06.2023).

21. Prism [Электронный ресурс]. URL: https://prismlibrary.com/index.html (дата обращения: 07.06.2023).
22. Qt [Электронный ресурс]. URL: https://www.qt.io/ (дата обращения: 07.06.2023).
23. ReaderWriterLock [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/api/system.threading.readerwriterlock?view=net-7.0 (дата обращения: 07.06.2023).
24. RFC 1055. A nonstandard for transmission of IP datagrams over serial lines: SLIP [Электронный ресурс]. 1988. URL: https://datatracker.ietf.org/doc/html/rfc1055 (дата обращения: 07.06.2023).
25. RFC 9110. HTTP Semantics [Электронный ресурс]. URL: https://www.rfc-editor.org/rfc/rfc9110.html (дата обращения: 07.06.2023).
26. State of the Windows Forms Designer for .NET Applications [Электронный ресурс]. URL: https://devblogs.microsoft.com/dotnet/state-of-the-windows-forms-designer-for-net-applications/ (дата обращения: 07.06.2023).
27. Wellsite Information Transfer Specification [Электронный ресурс] // Geoservices, A Schlumberger Company. Дата обновления: 03.02.2015. URL: http://www.petrospec-technologies.com/resource/wits_doc.htm (дата обращения: 07.06.2023).
28. Windows App SDK Releases [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/windows/apps/windows-app-sdk/downloads (дата обращения: 07.06.2023).
29. Windows Forms [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/ (дата обращения: 07.06.2023).
30. Wine [Электронный ресурс]. URL: https://www.winehq.org/ (дата обращения: 07.06.2023).

31. WinUI [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/windows/apps/winui/ (дата обращения: 07.06.2023).
32. WinUI 3 [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/windows/apps/winui/winui3/ (дата обращения: 07.06.2023).
33. WITSML Data Standards [Электронный ресурс] // Energistics Consortium. URL: https://www.energistics.org/witsml-data-standards/ (дата обращения: 07.06.2023).
34. WPF Apps With The Model-View-ViewModel Design Pattern [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern (дата обращения: 07.06.2023).

Приложение А. Снимки экрана для графического интерфейса существующей версии «GS Terminal»

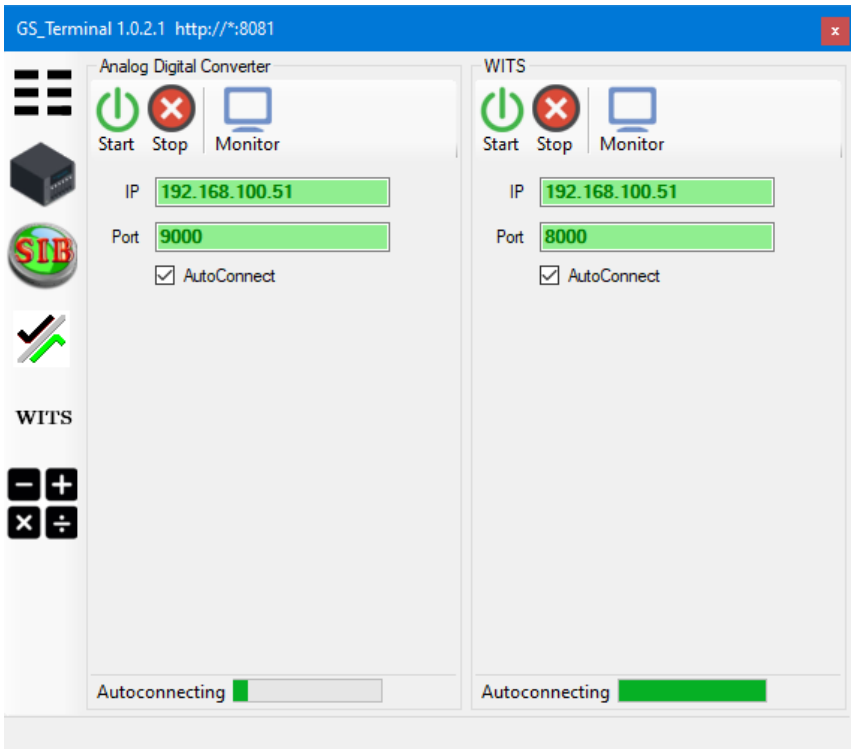


Рисунок А.1 – настройки соединения с КСПД по протоколам WAKE и WITS.

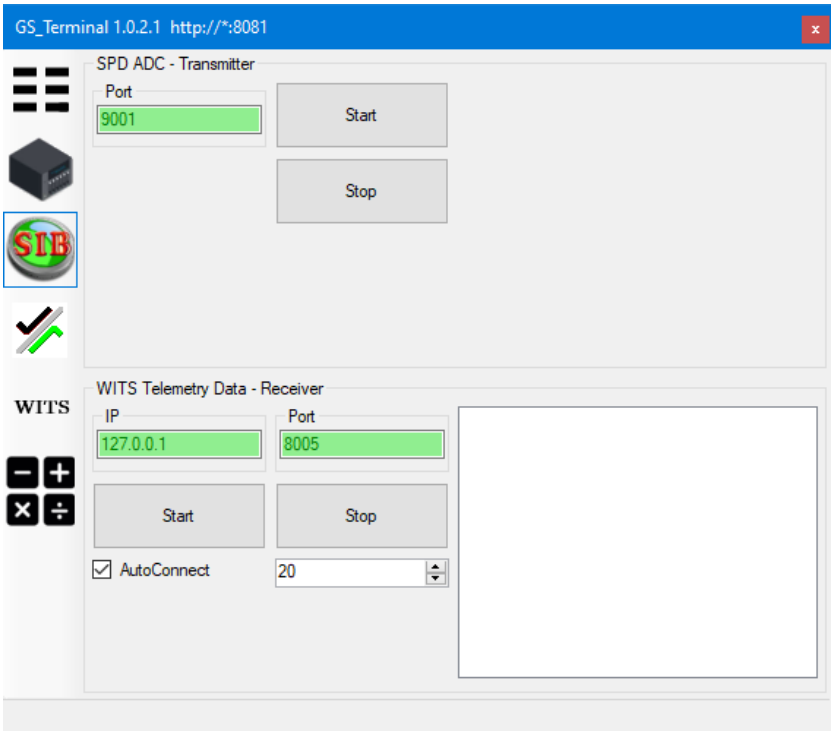


Рисунок А.2 – настройки соединения с SibReceiver по протоколам WAKE и WITS.

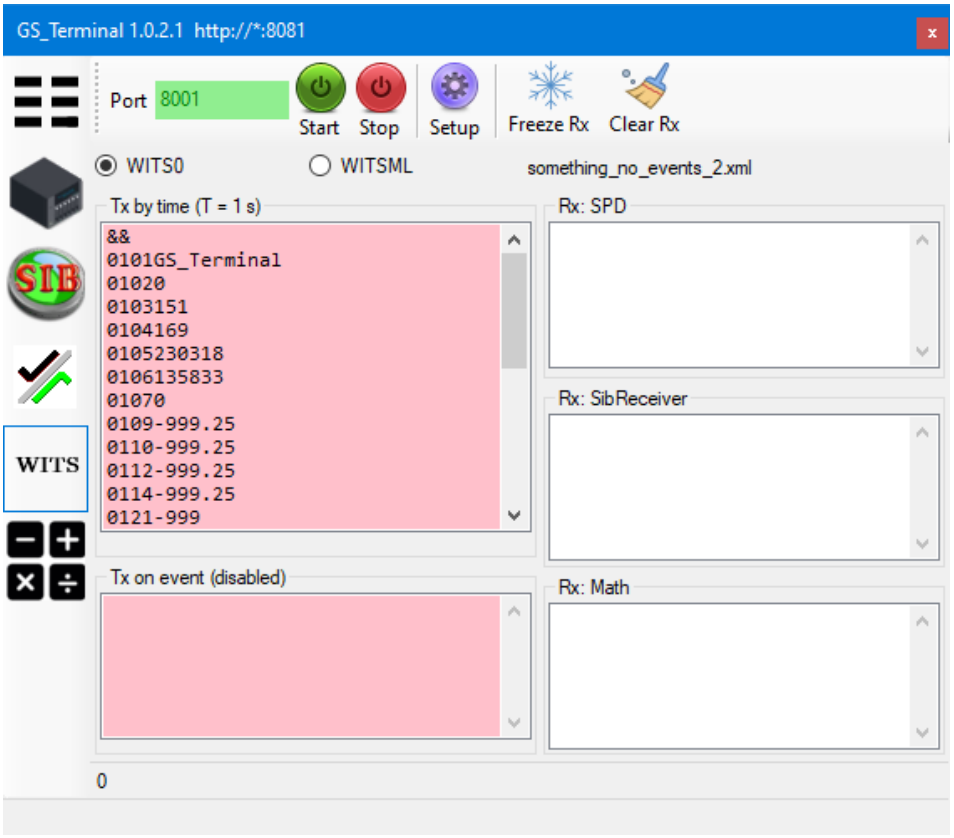
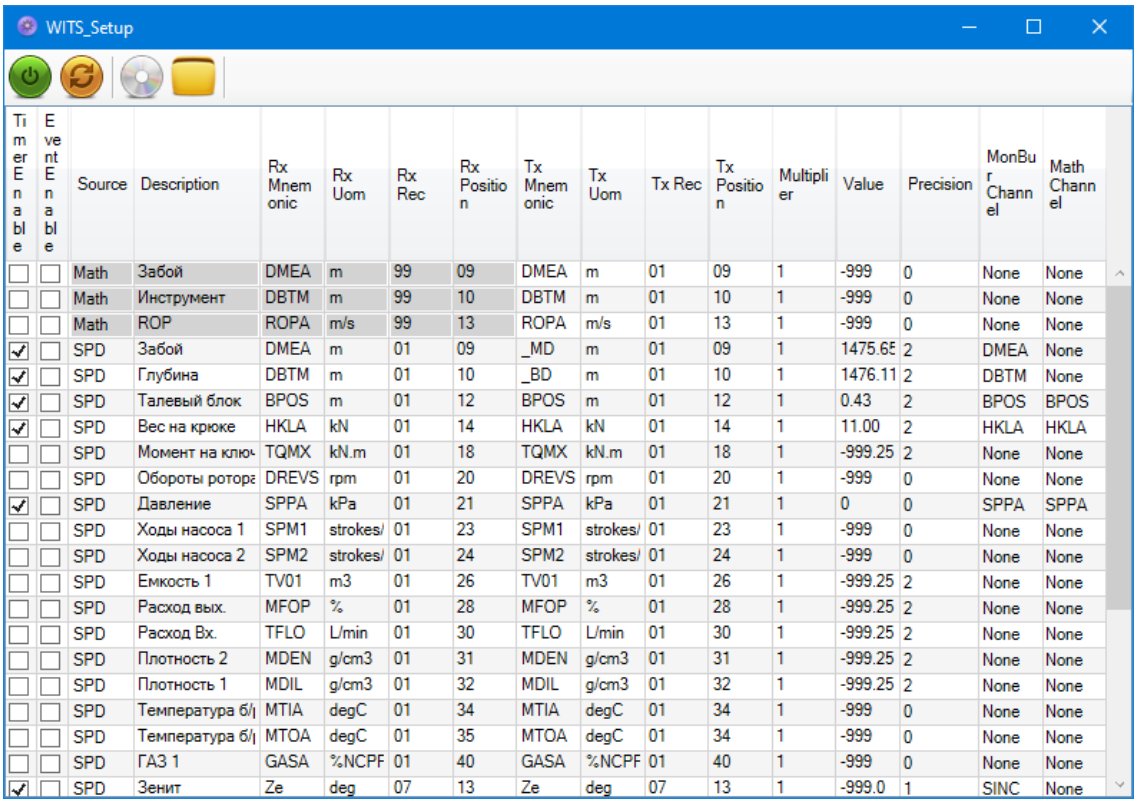


Рисунок А.3 – Текущее состояние приемника и передатчика данных по протоколу WITS.



TimerEnable	EventEnable	Source	Description	Rx Mnemonic	Rx Uom	Rx Rec	Rx Position	Tx Mnemonic	Tx Uom	Tx Rec	Tx Position	Multiplier	Value	Precision	MonBur Channel	Math Channel
☐	☐	Math	Забой	DMEA	m	99	09	DMEA	m	01	09	1	-999	0	None	None
☐	☐	Math	Инструмент	DBTM	m	99	10	DBTM	m	01	10	1	-999	0	None	None
☐	☐	Math	ROP	ROPA	m/s	99	13	ROPA	m/s	01	13	1	-999	0	None	None
☑	☐	SPD	Забой	DMEA	m	01	09	_MD	m	01	09	1	1475.65	2	DMEA	None
☑	☐	SPD	Глубина	DBTM	m	01	10	_BD	m	01	10	1	1476.11	2	DBTM	None
☑	☐	SPD	Талевый блок	BPOS	m	01	12	BPOS	m	01	12	1	0.43	2	BPOS	BPOS
☑	☐	SPD	Вес на крюке	HKLA	kN	01	14	HKLA	kN	01	14	1	11.00	2	HKLA	HKLA
☐	☐	SPD	Момент на ключ	TQMX	kN.m	01	18	TQMX	kN.m	01	18	1	-999.25	2	None	None
☐	☐	SPD	Обороты ротора	DREVS	rpm	01	20	DREVS	rpm	01	20	1	-999	0	None	None
☑	☐	SPD	Давление	SPPA	kPa	01	21	SPPA	kPa	01	21	1	0	0	SPPA	SPPA
☐	☐	SPD	Ходы насоса 1	SPM1	strokes/	01	23	SPM1	strokes/	01	23	1	-999	0	None	None
☐	☐	SPD	Ходы насоса 2	SPM2	strokes/	01	24	SPM2	strokes/	01	24	1	-999	0	None	None
☐	☐	SPD	Емкость 1	TV01	m3	01	26	TV01	m3	01	26	1	-999.25	2	None	None
☐	☐	SPD	Расход вых.	MFOP	%	01	28	MFOP	%	01	28	1	-999.25	2	None	None
☐	☐	SPD	Расход Вх.	TFLO	L/min	01	30	TFLO	L/min	01	30	1	-999.25	2	None	None
☐	☐	SPD	Плотность 2	MDEN	g/cm3	01	31	MDEN	g/cm3	01	31	1	-999.25	2	None	None
☐	☐	SPD	Плотность 1	MDIL	g/cm3	01	32	MDIL	g/cm3	01	32	1	-999.25	2	None	None
☐	☐	SPD	Температура б/р	MTIA	degC	01	34	MTIA	degC	01	34	1	-999	0	None	None
☐	☐	SPD	Температура б/р	MTOA	degC	01	35	MTOA	degC	01	34	1	-999	0	None	None
☐	☐	SPD	ГАЗ 1	GASA	%NCPF	01	40	GASA	%NCPF	01	40	1	-999	0	None	None
☑	☐	SPD	Зенит	Ze	deg	07	13	Ze	deg	07	13	1	-999.0	1	SINC	None

Рисунок А.4 – Окно конфигурации приемопередатчика WITS.

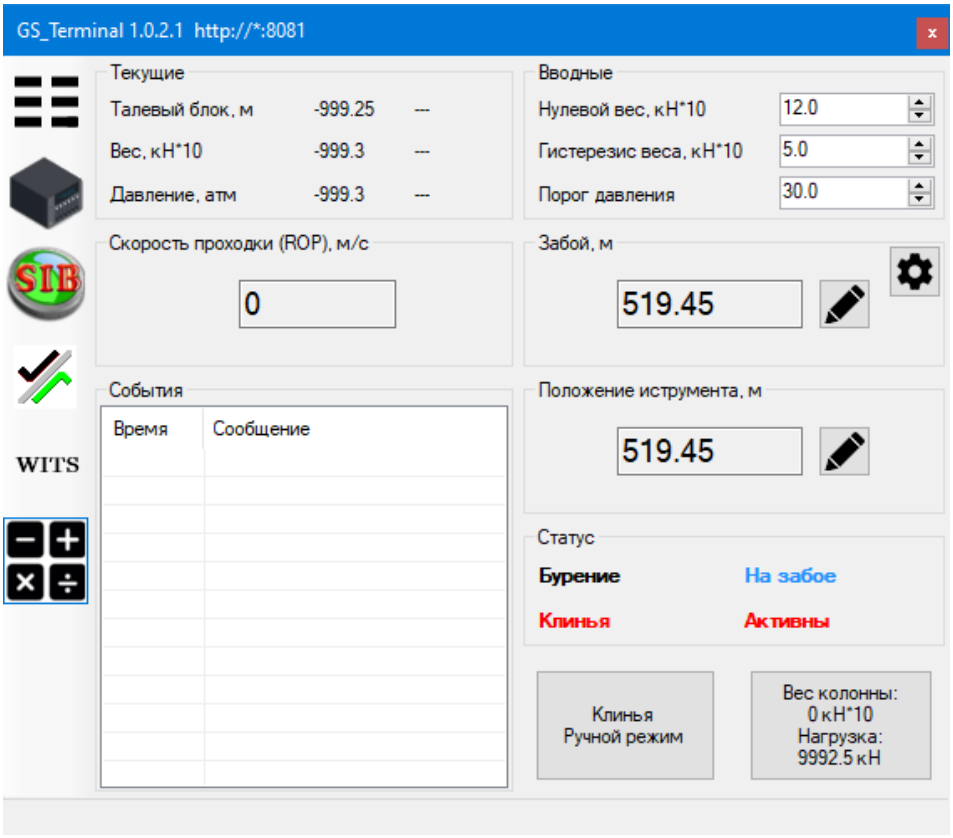


Рисунок А.5 – Панель вычислений.

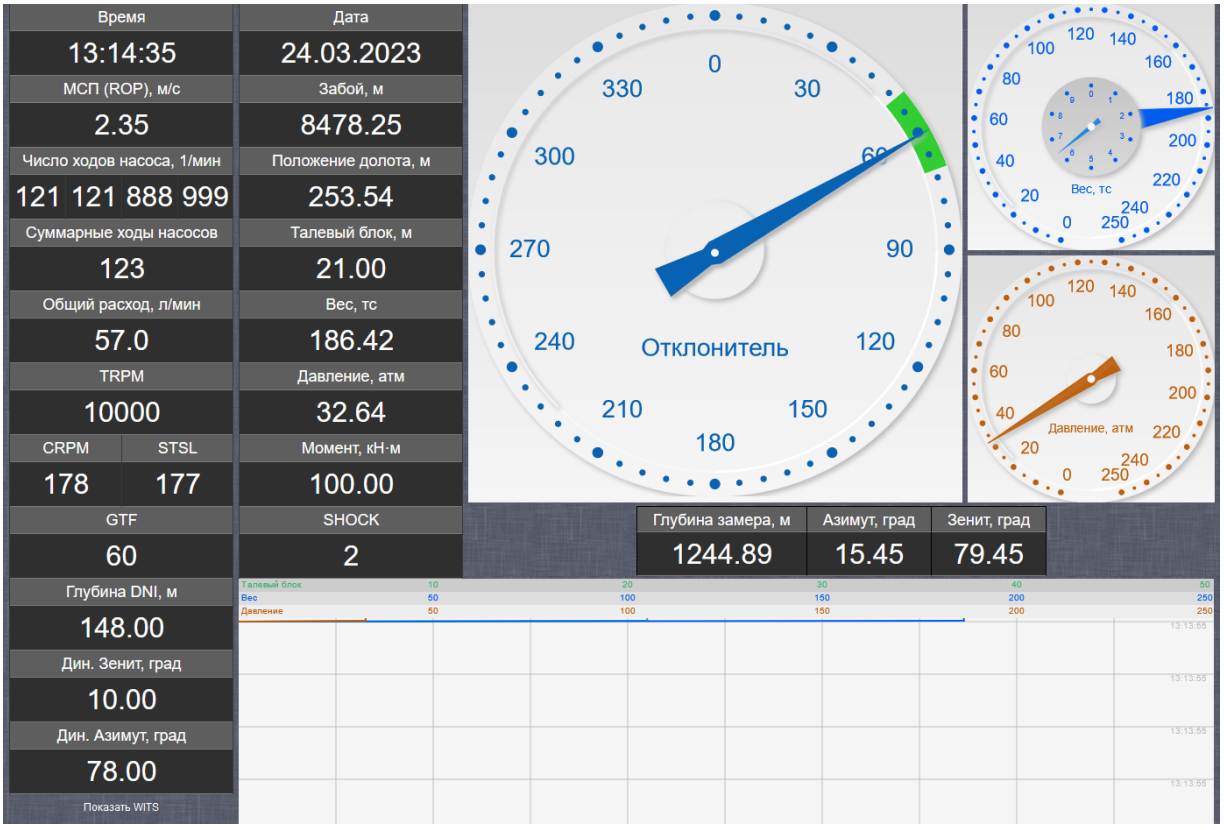


Рисунок А.6 – «Монитор бурильщика».

Приложение Б. Техническое задание

ГОСТ 34.602-2020

1 Общие сведения

1.1 Наименование системы

Программный комплекс для обработки данных на буровой установке.

1.2 Разработчик

Туев Дмитрий Васильевич

1.3 Заказчик

ООО «Технологическая Компания Шлюмберже»

1.4 Плановые сроки работы

01.02.2023 – 24.05.2023

2 Цели и назначение создания системы

2.1 Цели создания системы

Целью создания системы является обмен данными между программным и аппаратным обеспечением Заказчика и сторонних подрядчиков в процессе бурения нефтяных скважин.

2.2 Назначение системы

Назначением системы является автоматизация телеметрического сопровождения процесса бурения нефтяных скважин.

3 Характеристики объекта автоматизации

Объект автоматизации – система телеметрического сопровождения процесса бурения. В системе за опрос датчиков и передачи их показаний на компьютер отвечает блок сбора и подготовки данных (СПД). На компьютере работает программа Заказчика SibReceiver, отвечающая за декодирование данных. Декодированные данные передаются по протоколу WITS сторонним подрядчикам для проведения геофизических исследований скважин.

4 Требования к системе

4.1 Требования к структуре АС в целом

4.1.1 Требования к режимам функционирования системы

ОМ-1. Система должна функционировать 7 дней без требования перезагрузки.

ОМ-2. Система должна поддерживать произвольное отключение соединений со смежными системами.

ОМ-3. Система должна автоматически осуществлять попытку переподключения к сетевым узлам при потере соединения.

4.1.2 Требования к характеристикам взаимосвязей создаваемой АС со смежными АС, требования к интероперабельности, требования к ее совместимости, в том числе указания о способах обмена информацией

ІС-1. Система должна принимать показания датчиков с КСПД по протоколу WITS.

ІС-2. Система должна принимать информацию с АЦП СПД по протоколу WAKE.

ІС-3. Система должна передавать информацию, полученную с АЦП СПД, в программу SibReceiver по протоколу WAKE.

ІС-4. Система должна передавать результаты вычислений в КСПД по протоколу WAKE.

ІС-5.Система должна принимать декодированные данные из программы SibReceiver по протоколу WITS.

ІС-6. Система должна отправлять произвольные данные по протоколу WITS.

ІС-7. Система должна поддерживать подключение по TCP для приема и передачи данных.

ІС-8. Система должна поддерживать подключение по COM-порту для приема и передачи данных.

IC-9. Система должна иметь возможность указания коэффициента усиления и значения сдвига для каждого принимаемого значения.

IC-10. Система должна иметь возможность указания коэффициента усиления и значения сдвига для каждого отправляемого значения.

IC-11. Отправка значений по протоколу WITS должна настраиваться по пакетам. Пакетом является набор адресов каналов с их значениями, отправляемый при одном из заданных условий.

IC-12. Пакеты WITS могут отправляться по таймеру.

IC-13. Пакеты WITS могут отправляться при обновлении одного из пересылаемых значений («по событию»).

IC-14. Отправка отдельных значений каналов по протоколу WITS должна иметь следующие условия отправки:

1. каждый раз, когда отправляется пакет;
2. только если с момента отправки пришло новое значение;
3. только если с момента отправки пришлое новое значение, и оно отличается от последнего отправленного.

IC-15. Отправка отдельных значений каналов по протоколу WITS, если задано условие отправки и значение не требуется отправлять, должна иметь возможность отправки нулевого значения, определенного спецификацией протокола.

4.1.3 Требования по диагностированию системы

DIAG-1. Система должна иметь возможность генерации синусоидальных сигналов (далее – генерация сигналов) заместо произвольных выходных данных.

DIAG-2. Генерация сигналов должна иметь настраиваемую частоту и/или период.

DIAG-3. Генерация сигналов должна иметь настраиваемый фазовый сдвиг относительно других генерируемых синусоидальных сигналов.

DIAG-4. Генерация сигналов должна иметь настраиваемую амплитуду.

DIAG-5. Генерация сигналов должна иметь настраиваемую постоянную составляющую.

DIAG-6. Система должна иметь опцию записи всего сетевого трафика.

DIAG-7. Система должна сохранять все зафиксированные ошибки в лог-файл.

DIAG-8. Система должна отображать все последние полученные значения данных в графическом интерфейсе.

DIAG-9. Система должна отображать все последние отправленные значения данных в графическом интерфейсе.

4.1.4 Перспективы развития и модернизации АС

MOD-1. Система должна предусматривать возможность добавления иных протоколов связи (например, WITSML).

MOD-2. Система должна предусматривать возможность добавления описания математических вычислений в виде алгоритмов, не требующих перекомпиляции программы.

MOD-3. Система должна предусматривать возможность использования типов числовых данных, отличных от числа с плавающей точкой двойной точности.

MOD-4. Система должна предусматривать возможность локализации на языки, отличные от русского.

4.2 Требования к функциям (задачам), выполняемым АС

4.2.1 Требования к приему и передаче данных

IO-1. Система должна принимать показания датчиков с КСПД.

IO-2. Система должна принимать информацию с АЦП СПД.

IO-3. Система должна передавать информацию, полученную с АЦП СПД, в программу SibReceiver.

IO-4. Система должна иметь возможность приема произвольных данных по протоколам, описанным в разделе 4.1.2.

IO-5. Система должна иметь возможность отправки произвольных данных по протоколам, описанным в разделе 4.1.2.

IO-6. Система должна предоставлять выбор источника данных для каждой единицы отправляемых данных.

IO-7. Источником данных может быть:

1. КСПД;
2. SibReceiver;
3. произвольные приемники данных.

4.2.2 Требования к процессу конфигурирования системы

CFG-1. Процесс конфигурации системы не должен влиять на работающую систему.

CFG-2. Система должна применять измененную конфигурацию по команде пользователя.

CFG-3. Процесс конфигурации должен оповещать пользователя об ошибках, которые могут быть определены до применения конфигурации.

CFG-4. Система не должна принимать конфигурацию, содержащую ошибки, определенные в предыдущем пункте.

4.3 Требования к видам обеспечения АС

4.3.1 Требования к информационному обеспечению

INF-1. Система должна хранить конфигурацию в текстовом машиночитаемом формате.

INF-2. Система не должна использовать одну глобальную конфигурацию на компьютер.

INF-3. Система должна поддерживать импорт конфигураций отдельных протоколов приема/передачи данных в текстовом формате машиночитаемом формате.

INF-4. Система должна поддерживать экспорт конфигураций отдельных протоколов приема/передачи данных в текстовом формате машиночитаемом формате.

INF-5. Конфигурирование системы перед вводом в эксплуатацию выполняет оператор системы.

4.3.2 Требования к лингвистическому обеспечению

LING-1. Графический интерфейс системы должен быть на русском языке.

LING-2. Графический интерфейс системы может использовать английский язык для технических сообщений (например, записей в журнале ошибок).

4.4 Общие технические требования к АС

4.4.1 Требования к показателям назначения

NF-1. Система должна обрабатывать пакеты по протоколу WITS с частотой не менее 100 Гц.

NF-2. Система должна обрабатывать пакеты по протоколу WITSML с частотой не менее 100 Гц.

NF-3. Система должна обрабатывать пакеты по протоколу WAKE с частотой не менее 1000 Гц.

NF-4. Система должная обрабатывать данные с задержкой не более 200 мс.

4.4.2 Требования к надежности

REL-1. Программа должна функционировать с коэффициентом готовности не ниже 99,9% без учета аппаратных проблем.

4.4.3 Требования к эргономике и технической эстетике

UI-1. Графический интерфейс системы должен представлять текущую конфигурацию в процессе ее редактирования в виде ER-диаграммы.

UI-2. Графический интерфейс системы должен представлять используемую на данный момент конфигурацию в виде ER-диаграммы.

UI-3. Графический интерфейс системы должен отображать текущее состояние соединений со смежными системами.

UI-4. Графический интерфейс системы должен оповещать пользователя о наличии проблем, если они были обнаружены.

5 Порядок контроля и приемки системы

1. Предварительные автономные испытания частей системы.
    a. Unit-тестирование программного кода.
    b. Тестирование модулей отдельно друг от друга.
2. Предварительные автономные испытания системы в целом.
3. Предварительные комплексные испытания.
4. Опытная эксплуатация.
5. Приемочные испытания.