Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____Робототехника и комплексная автоматизация_____

КАФЕДРА ____Компьютерные системы автоматизации производства___

_____

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к дипломному проекту на тему:

_____

_____

_____

Студент _____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Руководитель дипломного проекта _____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по
исследовательской части _____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по проектной части_____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по
организационно-экономической части ___ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по охране труда и экологии ___ _____ _____
(Подпись, дата) (И.О.Фамилия)

Москва, 2013

Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« ___ » _____ 20 __ г.

З А Д А Н И Е
на выполнение дипломного проекта

Студент _____
(Фамилия, имя, отчество)

_____
(Тема дипломного проекта)

_____

_____

Источник тематики (НИР кафедры, заказ организаций и т.п.) НИР кафедры

_____

_____

Тема дипломного проекта утверждена распоряжением по факультету № _____
от « 12 » марта 2013 г.

1. Исходные данные

_____

_____

_____

_____

_____

_____

2. Технико-экономическое обоснование

_____

_____

_____

_____

_____

(обзор и анализ альтернативных решений; выбор вариантов для сравнения;
конкретные улучшаемые характеристики или параметры; возможный технико-экономический эффект и т.п.)

3. Научно-исследовательская часть

_____

_____

_____

_____

_____

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

4. Техническое проектирование

_____
_____
_____
_____
_____
_____
_____
_____

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

5. Рабочее проектирование

_____
_____
_____
_____
_____

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

6. Организационно-экономическая часть

_____
_____
_____

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

7. Охрана труда и экология

_____
_____
_____

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

8. Оформление дипломного проекта

8.1. Расчетно-пояснительная записка на ___ листе формата А4.

8.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.) _____ _
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

В соответствии с учебным планом дипломный проект выполнить в полном объеме в срок до « ___ » _____ 20__ г.

Руководитель дипломного проекта _____ _____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

Примечание:

1. Задание оформляется в двух экземплярах; один выдаётся студенту, второй хранится на кафедре.

Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« ___ » _____ 20 __ г.

К А Л Е Н Д А Р Н Ы Й П Л А Н
выполнения дипломного проекта

Студент _____Романов Ярослав Андреевич_____
(Фамилия, имя, отчество)

_____ Разработка кроссплатформенной версии системы имитационного моделирования RAO-studio_____
(Тема дипломного проекта)

№ п/п	Наименование этапов дипломного проекта	Выполнение этапов		Примечание
		Срок	Объем, %	
1.	Предпроектное исследование		5%	
2.	Концептуальное проектирование		20%	
3.	Техническое задание		5%	
4.	Техническое проектирование		20%	
5.	Рабочее проектирование		20%	
6.	Научно-исследовательская часть		20%	
7.	Организационно-экономическая часть		5%	
8.	Охрана труда и экология		5%	

Руководитель дипломного проекта _____ _____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Отчет 124 с., 3 рис., 12 табл., 10 источников, 1 прил.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, РЕСУРС-ДЕЙСТВИЕ-ОПЕРАЦИЯ, МОДЕЛЬ, АКТИВНОСТЬ, СОБЫТИЕ, ПРОЦЕСС, ТИП РЕСУРСА, РЕСУРС.

Объектом разработки является система дискретного имитационного моделирования на основе РДО. Ресурс, действие, операция (РДО) – программный комплекс, предназначенный для имитационного моделирования сложных дискретных систем с целью проведения их анализа и синтеза.

Цель работы – разработка кроссплатформенной версии системы имитационного моделирования RAO-studio.

При создании системы проведены исследования, цель которых являлось разработать оптимальный алгоритм поиска ближайших слов при автозавершении, учитывающий большинство вводимых пользователем паттернов.

В результате работы создана новая версия системы RAO-studio, которая может работать под операционными системами Windows и Linux и разработана на основе современной кроссплатформенной библиотеки.

Эффективность созданной системы заключается в расширении возможности использования РДО под разными операционными системами, поддержке Unicode, а также подготовке базы для дальнейшей разработки системы.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, СИМВОЛОВ И СПЕЦИАЛЬНЫХ ТЕРМИНОВ С ИХ ОПРЕДЕЛЕНИЕМ

ARIS	Architecture of Integrated Information Systems (методология и программный продукт для моделирования бизнес-процессов компании)
OMG	Object Management Group (Группа управления объектами)
UML	Universal Modeling Language (Универсальный язык моделирования)
БД	База данных
БЗ	База знаний
ИМ	Имитационная модель
ОС	Операционная сиситема
ПП	Программный продукт
ПО	Программное обеспечение
ПЭВМ	Персональная электронно-вычислительная машина
РДО	Ресурс Действие Операция – система имитационного моделирования
СДС	Сложная дискретная система
ТЗ	Техническое задание
ЭП	Эскизный проект

ВВЕДЕНИЕ

Математическое моделирование является неотъемлемой частью современного мира информационных технологий. Эксперты все чаще прибегают к имитационному моделированию. Моделирование вобрало в себя весь арсенал новейших информационных технологий, включая развитые графические оболочки для целей конструирования моделей и интерпретации выходных результатов, мультимедийные средства и видео, поддерживающие анимацию в реальном масштабе времени, объектно-ориентированное программирование и др. В силу своей привлекательности и доступности эти технологии с легкостью покинули академические стены и сегодня осваиваются IT - специалистами в бизнесе.

Широкое использование ИМ в задачах анализа и синтеза систем объясняется сложностью (а иногда и невозможностью) применения строгих методов оптимизации, которая обусловлена размерностью решаемых задач и невозможностью формализации сложных систем.

Так как современные технологии стремительно развиваются, одной из основных задач разработчиков программного обеспечения является поддержка продукта «на волне» новых технологий, постоянное обновление и развитие функциональных возможностей продукта.

РДО – не исключение. Система активно развивается, расширяются ее функциональные возможности. Однако наступает момент, когда дальнейшее развитие сопровождается возрастающими трудностями из-за устаревших технологий, лежащих в основе продукта. Тогда встает вопрос перехода на современные, новейшие технологии, которые позволят и дальше развивать систему и поддерживать ее в актуальном состоянии.

Таким образом, разработка новой версии системы имитационного моделирования РДО на основе современных технологий обеспечит дальнейшее развитие функционала системы, расширение областей ее использования, позволит вывести ее на качественно более высокий уровень.

1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

1.1 Назначение программного комплекса РДО

Задачи системного анализа и синтеза объектов различной природы и назначения часто решаются с использованием имитационных моделей. Эти модели позволяют исследовать динамические аспекты поведения сложных дискретных систем и процессов. Имитация, в частности, позволяет выполнить анализ функционирования объекта, прогнозирование, организационное управление, поддержать принятие решений при проектировании и управлении.

RAO-studio является средством имитационного моделирования, позволяющим воспроизводить на ПЭВМ динамику объекта, принятие решений сложной системой управления, и даже моделировать деятельность человека при принятии решений. В основе имитатора лежит РДО-метод формализации знаний о дискретных системах и процессах. Знания представляются в форме модифицированных продукционных правил, событий и процессов. При этом сохраняются такие достоинства продукционных систем, как универсальность, гибкость и наличие формальных механизмов логического вывода. Традиционные продукционные правила являются частным случаем модифицированных, поэтому в имитационную модель легко могут быть включены, например, экспертные системы.

Язык описания объектов, алгоритмов управления и задач в RAO-studio – это по существу язык представления знаний. Он требует от пользователя лишь знаний в предметной области, а не в программировании. Пользователь описывает ресурсы, правила функционирования, требуемые показатели и анимационные кадры непосредственно в терминах предметной области, не прибегая при этом к представлению своей системы в терминах какого-либо известного метода (системы очередей, сети Петри, автоматы) или языка типа SLAM-II, ARENA, SIMPLE++ и других. Это резко повышает гибкость, мощность и наглядность модели. РДО – язык высокого уровня, использующий

символические имена, арифметические и логические выражения и функции, генераторы псевдослучайных чисел, модифицированные и простые продукции.

Основные элементы RAO-studio – это модифицированная продукционная система, аппарат событий, система трассировки, система построения графиков, система анимации.

Продукционная система, блок имитации событий совместно осуществляют построение имитационной модели системы. На основании анализа результатов имитации вычисляются требуемые показатели функционирования системы.

Система трассировки выводит подробную информацию о событиях в специальный файл, который затем обрабатывается для детального анализа работы модели. Система анимации отображает на экране во время имитации поведение моделируемого объекта.

RAO-studio может быть применен для создания имитационных моделей, систем планирования, игр и тренажеров, экспертных систем реального времени и гибридных систем, включающих экспертные системы, имитационные модели и алгоритмы оптимизации.

1.2 Функции программного комплекса

При выполнении работ, связанных с созданием и использованием ИМ в среде РДО, пользователь оперирует следующими основными понятиями [1]:

Модель – совокупность объектов языка РДО, описывающих какой-то реальный объект, собираемые в процессе имитации показатели, кадры анимации и графические элементы, используемые при анимации, результаты трассировки.

Прогон – это единая неделимая точка имитационного эксперимента. Он характеризуется совокупностью объектов, представляющих собой исходные данные и результаты, полученные при запуске имитатора с этими исходными данными.

Проект – один или более прогонов, объединенных какой-либо общей целью. Например, это может быть совокупность прогонов, которые направлены на исследование одного конкретного объекта или выполнение одного контракта на имитационные исследования по одному или нескольким объектам.

Объект – совокупность информации, предназначенной для определенных целей и имеющая смысл для имитационной программы. Состав объектов обусловлен РДО-методом, определяющим парадигму представления СДС на языке РДО.

Объектами исходных данных являются:

- типы ресурсов (с расширением .rtp);
- ресурсы (с расширением .rss);
- событий (с расширением .evn);
- образцы активностей (с расширением .pat);
- точки принятия решений (с расширением .dpt);
- процессы (с расширением .prc);
- константы, функции и последовательности(с расширением .fun);
- кадры анимации (с расширением .frm, .bmp);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr);
- проект (с расширением .rdox).

Объекты, создаваемые РДО-имитатором при выполнении прогона:

- результаты (с расширением .pmv);
- трассировка (с расширением .trc).

На данный момент РДО реализует следующие основные функции:

1. Создание модели на языке РДО:

- создание основных объектов (*.rtp, *.rss, *.evn, *.pat, *.dpt, *.prc, *.smr);

- создание объектов данных и функций (*.fun);

- создание объектов вывода (*.frm, *.pmd, *.bmp).

2. Проведение экспериментов:

- изменение параметров системы в процессе моделирования;

- генерация случайных чисел.

3. Вывод результатов моделирования:

- анимация объектов модели;

- вывод необходимых показателей;

- построение графиков;

- трассировка изменений объектов модели.

1.3 Постановка задачи

Система имитационного моделирования РДО используется для обучения имитационному моделированию в школах и университетах. Она разработана с использованием библиотеки MFC компании Microsoft. Данная библиотека накладывает ограничения на ОС, под которыми может работать РДО (на данный момент поддерживается работа только под ОС Windows). Более того, компания Microsoft практически прекратила развитие библиотеки MFC. Из-за этого затрудняется дальнейшее наращивание функциональных возможностей RAO-studio. Принимая во внимание эти факторы, было принято решение о разработке новой версии системы на основе более мощной библиотеки.

Выбранная библиотека должна удовлетворять ряду требований:

1. Кроссплатформенность;
2. Полнота документации;
3. Удобство использования;
4. Хорошие перспективы дальнейшего развития;
5. Адекватная обратная связь с разработчиками;
6. Свободно распространяемая (open source).

Таким образом, основная цель дипломного проекта – исправление вышеописанных недостатков путем разработки кроссплатформенной версии системы имитационного моделирования RAO-studio.

1.4 Выводы по предпроектному этапу

Рассмотренные выше проблемы представляют собой серьезное препятствие как для пользователя, который захочет работать с системой РДО в операционной системе, отличной от Windows, так и для разработчика в случае расширения функциональности системы. Пользователь, использующий Linux, не сможет запустить РДО и вынужден будет для его использования устанавливать операционную систему Windows, либо использовать средства эмуляции.

Для разработчика ситуация усложняется тем, что использование устаревшей и практически не развивающейся библиотеки сильно ограничивает возможности для развития системы.

Разработка новой версии РДО на основе современной кроссплатформенной библиотеки должна решить данную проблему.

2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Разработка технического задания на систему

Техническое задание разработано в соответствии с ГОСТ 19.201-78 (Единая система программной документации. Техническое задание. Требования к содержанию и оформлению) [7].

2.1.1 Основания для разработки

1) Разработка ведется на основании следующих документов:
   - Задание на выполнение дипломного проекта.
   - Календарный план на выполнение дипломного проекта.
2) Документы утверждены «12» марта 2013 года.
3) Тема дипломного проекта:

Разработка кроссплатформенной версии системы имитационного моделирования RAO-studio.

2.1.2 Назначение разработки

Основная цель данного дипломного проекта – разработать кроссплатформенную версию системы имитационного моделирования RAO-studio, позволяющую пользователям работать с программой в большинстве современных ОС.

2.1.3 Требования к программе или программному изделию

2.1.3.1 Требования к функциональным характеристикам

Требования к составу выполняемых функций реализуемой версии системы заключаются в следующем:

1. кроссплатформенность;
2. переход на Юникод (Unicode);
3. обновление документации РДО;
4. реализация функциональных и модульных тестов;
5. реализация основных функций РДО.

2.1.3.2 Требования к надежности

Действия пользователя не должны приводить к сбоям в работе программы.

Система должна обладать средствами самодиагностики и автоматической индикации типов ошибок.

Пользователю должен быть предоставлен интуитивно понятный интерфейс и система помощи.

2.1.3.3 Условия эксплуатации

Аппаратные средства должны эксплуатироваться в помещениях с выделенной розеточной электросетью 220В ±10%, 50 Гц с защитным заземлением при следующих климатических условиях:

- температура окружающей среды – от 15 до 30 градусов С;
- относительная влажность воздуха - от 30% до 80%;
- атмосферное давление - от 630 мм. р.с. до 800 мм. р.с.

2.1.3.4 Требования к составу и параметрам технических средств

Программный продукт должен работать на компьютерах со следующими характеристиками:

- объем ОЗУ не менее 256 Мб;
- объем жесткого диска не менее 20 Гб;
- микропроцессор с тактовой частотой не менее 400 МГц;
- монитор не менее 15” с разрешением от 800*600 и выше.

2.1.3.5 Требования к информационной и программной совместимости

Данная система должна работать под управлением операционных систем семейства Microsoft Windows для рабочих станций:

- Windows XP;
- Windows Vista;
- Windows 7;
- Windows 8;

а также для серверов:

- Windows Server 2003;

- Windows Server 2008;
- Windows Server 2008 R2;
- Windows Home Server 2011.

Данная система должна работать под управлением операционных систем семейства Linux.

2.1.3.6 Требования к маркировке и упаковке

Не предъявляются.

2.1.3.7 Требования к транспортированию и хранению

Не предъявляются.

2.1.3.8 Требования к программной документации

Не предъявляются.

2.1.4 Технико-экономические показатели

Расчет экономической эффективности разработанного приложения не является целью дипломного проектирования, однако возможный экономический эффект может быть достигнут за счет следующих преимуществ системы:

1) Работа под операционными системами Windows и Linux.
2) Открытие возможностей для дальнейшего развития системы.

2.1.5 Стадии и этапы разработки

Состав, содержание и сроки выполнения работ по созданию системы в соответствии с календарным планом на выполнение дипломного проекта.

2.1.6 Порядок контроля и приемки

Контроль и приемка приложения должны состоять из следующих этапов:

1. Запуск RAO-studio в операционной системе Windows и проверка неизменности работы на всех тестовых моделях;
2. Запуск RAO-studio в операционной системе Linux и проверка неизменности работы на всех тестовых моделях;
3. Проверка поддержки Unicode;

4. Проверка функциональности графического интерфейса пользователя RAO-studio.

2.1.7Приложения

Документы, используемые при разработке, приведены в списке использованных источников.

Используемое при разработке программное обеспечение:

- Операционная система Microsoft Windows 8 Professional;
- Среда разработки Microsoft Visual Studio 2008 Professional;
- Qt фреймворк;
- Среда разработки графических интерфейсов Qt Designer;
- Справочная система Qt Assistant;
- Централизованная система управления версиями Subversion;
- Система отслеживания ошибок Mantis Bug Tracker;
- Система документирования исходных текстов Doxygen;
- Исходные коды системы РДО.

3 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Цели разработки системы

Основной целью данного дипломного проекта является разработка новой версии системы с использованием современной библиотеки для разработки. Она состоит из подцелей:

1. сделать систему кроссплатформенной;
2. реализовать возможность дальнейшего наращивания функциональных возможностей системы;
3. упростить разработку и сопровождение системы;
4. перевести систему на Unicode;
5. провести функциональные и модульные тесты системы;
6. обновить документацию РДО.

3.2 Компоненты РДО

Система имитационного моделирования РДО безусловно является сложной и статически, и динамически. На это указывает сложная иерархическая структура системы со множеством различных связей между компонентами и ее сложное поведение во времени.

Ярко выраженная иерархическая структура и модульность системы определяют направление изучения системы сверху вниз. Т.е. принцип декомпозиции применяется до тех пор, пока не будет достигнут уровень абстракции, представление на котором нужных объектов не нуждается в дальнейшей детализации для решения данной задачи.

Базовый функционал основных компонентов РДО:

rdo_kernel реализует ядровые функции системы. Не изменяется при разработке системы.

RAO-studio.exe реализует графический интерфейс пользователя. Модернизируется при разработке системы.

rdo_repository реализует управление потоками данных внутри системы и отвечает за хранение и получение информации о модели. Не изменяется при разработке системы.

rdo_mbuilder реализует функционал, используемый для программного управления типами ресурсов и ресурсами модели. Не изменяется при разработке системы.

rdo_converter конвертирует модели созданные в старой версии РДО, производя резервное копирование файлов оригинальной модели. Благодаря нему обеспечивается обратная совместимость версий системы. Не изменяется при разработке системы.

rdo_simulator управляет процессом моделирования на всех его этапах. Он осуществляет координацию и управление компонентами rdo_runtime и rdo_parser. Не изменяется при разработке системы.

rdo_parser производит лексический и синтаксический разбор исходных текстов модели, написанной на языке РДО. Не изменяется при разработке системы.

Для отображения зависимости между компонентами системы РДО и выделения среди них модернизируемых служит соответствующая диаграмма в нотации UML. Так как модернизируется один компонент – RAO-studio.exe, то на диаграмме компонентов представлены зависимости внутри данного компонента.

3.3 Выбор библиотеки

При выборе библиотеки для разработки рассматривались три варианта:

1. wxWidgets;
2. GTK+;
3. Qt.

Библиотека wxWidgets [5] – это кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки

кроссплатформенных на уровне исходного кода приложений, в частности для построения графического интерфейса пользователя(GUI).

Основным преимуществом wxWidgets перед остальными библиотеками является то, что она использует родные (нативные) графические элементы интерфейса операционной системы. В результате они не только выглядят, как родные элементы ОС, они и в самом деле ими являются.

Однако у данной библиотеки есть существенный недостаток: отсутствие актуальной документации. По многим классам библиотеки в документации отсутствует справка вообще, либо ограничивается одной-двумя строками. Более того, некоторые технические нюансы вообще не освещены в документации.

Библиотека GTK+ [6] – кроссплатформенная библиотека элементов интерфейса. Она разработана на языке C. Поддержка остальных языков осуществляется через специально разработанные интерфейсы.

GTK+ реализует мощные средства для профессионального рисования и дизайна графических элементов. Кроме того, как следствие использования C, GTK+ – более оптимизированная, а также более портабельная на другие языки библиотека, поскольку сообщения, сгенерированные одним языком, могут быть обработаны в процедуре, написанной на другом.

GTK+ изначально разрабатывалась для создания графических элементов интерфейса. Возможность ее использования для других целей очень ограничена. А одной из наших целей является подготовка базы для дальнейшего развития системы РДО. Кроме того, документация по GTK+ плохо структурирована, и поиск нужной информации отнимает значительную часть времени, необходимого на знакомство с этой библиотекой.

Библиотека Qt [3] – это кроссплатформенный инструментарий разработки ПО на языке программирования C++.

Программы, написанные на Qt, можно запустить на большинстве операционных систем путем простой компиляции программы для каждой ОС

без изменения исходного кода. Qt включает в себя все классы, необходимые для разработки современного прикладного ПО, начиная с элементов графического интерфейса и заканчивая работой с XML, базами данных и сетью. Qt является полностью объектно-ориентированной и легко расширяемой библиотекой. Qt комплектуется средой разработки графического интерфейса Qt Designer, справочной системой Qt Assistant, позволяющей писать кроссплатформенную справку для разрабатываемого на основе Qt ПО.

Кроме того, Qt обладает качественной документацией с удобной системой поиска. Статьи документации снабжены большим количеством примеров, что значительно упрощает освоение Qt.

Проведя сравнение данных библиотек, было принято решение выбрать библиотеку Qt, как наиболее мощное и современное средство разработки, удовлетворяющее поставленным требованиям (1.3).

3.4 Изменения в системе РДО

После решения поставленной задачи пользователь должен получить возможность:

1. использовать РДО под Windows или Linux;
2. хранить исходный код моделей в Unicode;
3. пользоваться обновленной, актуальной документацией.

Кроме того, разработчику станет более просто и удобно внедрять новый функционал.

3.5 Модифицируемые компоненты RAO-studio

Для перехода на новую версию системы должны быть модифицированы следующие компоненты:

- главное окно системы;
- окно редактора;
- окно графиков;

- окно анимации;
- окно трассировки;
- меню системы;
- всплывающие меню;
- диалоги (настройки, поиск/замена, настройки графиков, открыть модель, новая модель, перейти на строчку);

Для большинства этих элементов необходимо использовать только библиотеку Qt, однако окно редактора РДО использует библиотеку Scintilla [4] в качестве текстового редактора. Версия Scintilla, используемая в РДО, устарела, и возникла необходимость перейти на новую версию библиотеки прежде, чем продолжать разработку. Кроме того, новая версия Scintilla поддерживает работу с Qt, поэтому интеграция ее в новую версию системы избавит нас от необходимости самостоятельно реализовывать поддержку Qt в Scintilla.

Для диалогов должны быть спроектированы новые экранные формы на Qt.

Работа с настройками системы также должна осуществляться через механизмы Qt.

4 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

4.1 Перепроектирование редактора РДО на Qt

С точки зрения пользователя редактор РДО является основной подсистемой РДО, с которой ему предстоит работать. Обновленный текстовый редактор должен работать на современной версии Scintilla (3.3.0). Для этого необходимо рассмотреть все патчи, которые были применены к старой версии, и перенести необходимые изменения в новую версию для интеграции ее в систему.

Для реализации механизма автозавершения слов было решено разработать новый класс WordListUtil, являющийся оболочкой над классом WordList из Scintilla. В этом классе реализован механизм поиска ближайших слов.

4.2 Перепроектирование диалогов на Qt

4.2.1Диалог «Просмотр/Настройки» на Qt

Для того, чтобы работать с настройками системы, необходимо перевести диалог настроек на Qt. Данный диалог включает в себя 4 вкладки:

- Основные – общие настройки системы;
- Редактор – настройки окна редактора РДО;
- Табуляция – настройки табуляции;
- Стиль и цвет – настройки, позволяющие изменять стили и темы окон системы.

Функционал данных вкладок должен быть полностью перенесен в новую версию.

В старой версии РДО в данном диалоге присутствовала вкладка «Встраиваемые модули». Так как она не использовалась, было принято решение о ее удалении из диалога.

Для реализации этого диалога создан новый класс настроек ViewPreferences, который реализует методы интерфейса диалога настроек и обработку действий пользователя.

Помимо нового класса должна быть разработана новая экранная форма диалога на Qt.

4.2.2Диалог «Поиск/Найти» на Qt

Для возможности поиска слов в теле модели должен быть переведен диалог поиска. Новый диалог должен быть разработан на Qt как отдельный класс FindDialog. Он должен поддерживать поиск в обоих направлениях как вниз по тексту, так и вверх.

Помимо нового класса должна быть разработана новая экранная форма диалога на Qt.

4.2.3Диалог «Поиск/Найти в модели» на Qt

Диалог «Найти в модели» позволяет искать текст по всей модели и выводить найденные совпадения в отдельное окно поиска. Данный диалог представляет собой также класс FindDialog.

4.2.4Перевод диалога «Поиск/Заменить» на Qt

Диалог «Найти и заменить» должен реализовывать функционал поиска и замены слов. Должна быть реализована возможность поиска как вниз, так и вверх по тексту. Диалог должен поддерживать возможность замены сразу всех совпадений, найденных в модели, на новый текст (Заменить все). Диалог разработан как отдельный класс FindReplaceDialog.

Помимо нового класса должна быть разработана новая экранная форма диалога на Qt.

4.2.5Диалог «Поиск/Перейти на строчку» на Qt

Диалог должен обеспечивать возможность перехода по номеру введенной пользователем строки. Разработан как отдельный класс GoToLineDialog.

Помимо нового класса должна быть разработана новая экранная форма диалога на Qt.

4.2.6Диалог «График/Настройки» на Qt

Диалог настроек графика должен позволять изменять параметры визуального отображения графиков, такие, как количество значений по осям, цвет графиков, размер маркеров и т.д. Для этого был разработан новый класс ChartPreferences, реализующий интерфейс визуальных настроек графика и обработку действий пользователя.

Помимо нового класса должна быть разработана новая экранная форма диалога на Qt.

4.3 Перепроектирование пунктов меню на Qt

В системе RAO-studio присутствуют следующие пункты меню:

- Файл
- Правка
- Поиск
- Просмотр
- Вставка
- Модель
- График
- Окна
- Помощь

В рамках данного дипломного проекта мне необходимо перевести следующие пункты меню:

1. Правка:
    - Вырезать
    - Копировать
    - Вставить
    - Удалить
    - Выделить все
    - В нижний регистр
    - В верхний регистр
    - Закомментировать выделенное
    - Завершить слово
2. Поиск:
    - Найти
    - Найти следующий
    - Найти предыдущий
    - Заменить
    - Найти в модели
    - Поставить/Снять закладку
    - Перейти на строчку
3. Просмотр:
    - Свернуть/Развернуть все группы
    - Свернуть/Развернуть текущую группу
    - Увеличить масштаб
    - Уменьшить масштаб
    - Автоматический масштаб
    - Восстановить масштаб
    - Настройки
4. График:

- Создать
- Экспорт данных
- «Схлопнуть» время
- Настройки

Для реализации этих методов необходима модификация классов редактора, графиков, анимации.

4.4 Перепроектирование всплывающих меню на Qt

Необходимо перевести на Qt следующие всплывающие меню:

- меню окна редактора;
- меню окна компилятора;
- меню окна вывода;
- меню окна поиска;
- меню окна результатов;
- меню окна графиков.

Меню должны содержать только те пункты, которые доступны для текущего окна.

Меню реализовано как отдельный класс PopupMenu для окон компиляции, вывода, поиска и результатов. Для всплывающих меню остальных окон должны быть модифицированы классы редактора, графиков, трассировки.

4.5 Модификация классов стилей

В старой версии РДО понятие стиля было разбито на два: стиль и темы оформления.

Это разбиение вызывало множество затруднений, связанных как с читаемостью кода, так и с работой с ним, потому что приходилось оперировать одним понятием, разбитым на два отдельных класса.

В классах тем хранилась информация о настройках цветов и шрифтах, об отображении закладок и стиле групп (группы позволяют сворачивать/разворачивать участки кода, объединенные общим смысловым значением).

В классах стилей хранились атрибуты, отвечающие за такие параметры, как автозавершение, закладки, отображение номеров строк, символов табуляции. Помимо данных атрибутов, каждый класс стиля содержал внутри себя указатель на объект класса тем. При инициализации стилей производилась двойная работа: создавалось два объекта для одного элемента РДО – стиль и тема.

Было принято решение, что такой подход создает путаницу в определении стилей, и поэтому необходимо произвести слияние классов стилей и тем.

Таким образом, все необходимые методы и атрибуты тем были перенесены в классы соответствующих стилей, классы тем удалены из системы. Все настройки стилей теперь инициализируются в одном классе. Текущие взаимосвязи между классами стилей представлены на соответствующей диаграмме классов стилей.

4.6 Конвертация настроек

В процессе перехода на Qt необходимо перевести хранение настроек на QSettings. Возник вопрос сохранения настроек старой версии системы и их использовании при запуске новой версии системы.

Для того, чтобы при переходе на новую версию РДО сохранились старые пользовательские настройки, был разработан конвертор настроек. Настройки из старой версии конвертируются в новые. После успешной конвертации происходит удаление настроек старой версии РДО из операционной системы.

Конвертор разработан как класс Convertor.

Методы класса:

`rbool contains(const QString& name) const` - метод, проверяющий наличие старых настроек. В качестве атрибута выступает имя раздела настроек.

`void convert(const QString& from, const QString& to)` – метод, который непосредственно конвертирует настройки. Первый атрибут метода хранит путь к старым настройкам, второй атрибут – путь, куда должны быть сохранены новые настройки после конвертации.

`void setValue(const QString& to, const T& value)` – метод, сохраняющий значение value настроек в путь to. Другими словами, данный метод осуществляет запись настроек по указанному пути.

`void remove(const QString& name)` – метод, вызываемый после успешной конвертации настроек. Удаляет старые настройки из памяти операционной системы.

`static QVariant convertFrom(const QVariant& value)` – метод, преобразующий старые настройки в новый тип, с которым производится дальнейшая работа.

`static QVariant convertFrom<QColor>(const QVariant& value)` – перегруженый метод, используется для конвертации настроек цветов.

Данное решение связанно с разными способами хранения цвета. Старая версия РДО использует встроенные средства языка c++ для хранения цвета. Цвет в данном случае хранится в виде типа DWORD. Особенность его хранения в том, что при записи цвета в 16тиричном формате первая пара цифр хранит в себе синий цвет, вторая пара – зеленый, третья – красный.

В Qt цвет хранится в виде класса QColor. Данный класс использует стандартную модель представления цвета – RGB. Т.е., в отличие от предыдущего варианта, первая пара цифр хранит красный цвет, вторая – зеленый, третья – синий.

При обычной конвертации настроек происходило неявное инвертирование синих и красных цветов. Таким образом, мы видели на экране не тот цвет, который хотели. Для того, чтобы этого не происходило, метод конвертации был перегружен таким образом, что в настройки новой версии системы записывалось 16тиричное значение цвета с инвертированными парами, отвечающими за красный и синий цвета. Благодаря этому было достигнуто сохранение исходного цвета, определенного в старых настройках.

`static QVariant convertTo(const QVariant& value)` – метод, используемый для приведения значений настроек к типу, используемому в QSettings.

`static QVariant convertTo<QColor>(const QVariant& value)` – перегруженый метод, используется для записи QColor. Qt умеет работать с именами цветов, поэтому данный метод записывает имя цвета в настройки, а не сам объект QColor. Для стандартного набора цветов имя цвета представляет собой его английское название, для остальных цветов это 16тиричное значение в формате QString.

5 РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

5.1 Реализация диалога «Просмотр/Настройки»

Диалог настроек реализован в классе ViewPreferences.

```
1. rbool m_setup;
2. rbool m_checkInFuture;
3. rbool m_openLastProject;
4. rbool m_showFullName;
```

Данные атрибуты класса хранят в себе настройки для первой вкладки диалога «Основные».

Настройки вкладок «Редактор» и «Табуляция» хранятся в стиле редактора, поэтому для них не нужны дополнительные атрибуты.

```
1. rdo::gui::editor::ModelStyle      style_editor;
2. rdo::gui::editor::BuildStyle      style_build;
3. rdo::gui::editor::EditStyle       style_debug;
4. rdo::gui::editor::ResultsStyle    style_results;
5. rdo::gui::editor::FindStyle       style_find;
6. rdo::gui::tracer::LogStyle        style_trace;
7. rdo::gui::tracer::ChartViewStyle  style_chart;
8. rdo::gui::frame::FrameStyle       style_frame;
```

Данные атрибуты представляют собой настройки стилей РДО. При открытии диалога текущие настройки системы сохраняются внутри данных атрибутов. Это происходит в конструкторе диалога:

```
1.style_editor  = g_pApp->getStyle()->style_editor;
2.style_build   = g_pApp->getStyle()->style_build;
3.style_debug   = g_pApp->getStyle()->style_debug;
4.style_trace   = g_pApp->getStyle()->style_trace;
5.style_results = g_pApp->getStyle()->style_results;
6.style_find    = g_pApp->getStyle()->style_find;
7.style_chart   = g_pApp->getStyle()->style_chart;
8.style_frame   = g_pApp->getStyle()->style_frame;
```

После этого диалог работает с копией реальных настроек.

```
1.PTR(rdo::gui::editor::Model)        preview_editor;
2.PTR(rdo::gui::editor::Build)        preview_build;
3.PTR(rdo::gui::editor::Debug)        preview_debug;
```

```
4. PTR(rdo::gui::editor::Results)       preview_results;
5. PTR(rdo::gui::editor::Find)          preview_find;
6. PTR(rdo::gui::tracer::LogMainWnd)    preview_trace;
7. PTR(rdo::gui::tracer::ChartDoc)      preview_chart_doc;
8. PTR(rdo::gui::tracer::ChartView)     preview_chart;
9. std::vector<rdo::gui::tracer::Time>  preview_times;
10.    rdo::gui::tracer::LPSerie            preview_serie;
11.    PTR(rdo::gui::frame::OptionsView)    preview_frame;
```

Это элементы РДО, использующиеся в диалоге настроек для отображения предпросмотра стилей. При изменении стиля можно сразу увидеть результат в окне предпросмотра закладки «Стиль и цвет» без изменения реальных стилей системы.

Создание элементов предпросмотра осуществляется в методе createPreview:

```
void ViewPreferences::createPreview()

{

1. preview_editor = new Model(previewStackedWidget-
   >currentWidget(), previewStackedWidget->currentWidget());
2. ASSERT(preview_editor);
3. preview_editor->setEditorStyle(&style_editor);
4. preview_editor->setCanClearErrorLine(false);
5. preview_editor->appendText("{ comments }\n$Pattern
   pattern_name : operation trace\n$Relevant_resources\n
   rel_res2  : res_type2     Keep    Keep\n  rel_res1  :
   res_type1     Create  NoChange\n$Time = Abs(rel_res2.par1 -
   rel_res2.par3)\n{...}\n$End\n\ntext [ 10, 20, ... = 'text'
   ]\n\n$Re levant_resources");
6. preview_editor->scrollToLine(0);
7. preview_editor->setReadOnly(true);
8. preview_editor->bookmarkToggle();
9. preview_editor->setErrorLine(preview_editor->getLineCount() -
   1);
10.    previewStackedWidget->addWidget(preview_editor);
   …
}
```

Здесь приведен фрагмент этого метода. Строка 1 создает новый объект Model (окна редактора).

Строка 2 проверяет, смог ли объект правильно инициализироваться.

Строки 3-9 устанавливают параметры для предпросмотра.

В строке 10 объект добавляется на виджет, который его отображает.

Аналогичным образом в данном методе добавляются остальные элементы предпросмотра.

Для работы с настройками тем используются два внутренних класса:

```
class StyleProperty
     {
     public:
          StyleItem* item;
          int identificator;

          rdo::gui::style::StyleFont::style& font_style;

          QColor& fg_color;
          QColor& bg_color;
          QColor& fg_disable_color;
          QColor& bg_disable_color;

          StyleProperty(StyleItem* item, int identificator,
rdo::gui::style::StyleFont::style& font_style, QColor& fg_color,
QColor& bg_color, QColor& fg_disable_color = null_fg_color,
QColor& bg_disable_color = null_bg_color)
               : item(item)
               , identificator(identificator)
               , font_style(font_style)
               , fg_color(fg_color)
               , bg_color(bg_color)
               , fg_disable_color(fg_disable_color)
               , bg_disable_color(bg_disable_color)
          {}
     };
class StyleItem
     {
     public:
          ItemType                                type;
          int&                                    font_size;
          tstring&                                font_name;
          rbool&                                  wordwrap;
          rbool&                                  horzscrollbar;
          rbool&                                  warning;
          rdo::gui::editor::EditStyle::Bookmark&  bookmarkstyle;
          rdo::gui::editor::ModelStyle::Fold&     foldstyle;
          rbool&                                  commentfold;

          PropertyList properties;

          StyleItem(ItemType type, int& font_size, tstring&
font_name, rbool& wordwrap = null_wordwrap, rbool& horzscrollbar =
null_horzscrollbar, rdo::gui::editor::EditStyle::Bookmark&
```

```
bookmarkstyle = null_bookmarkstyle,
rdo::gui::editor::ModelStyle::Fold& foldstyle = null_foldstyle,
rbool& commentfold = null_commentfold, rbool& warning =
null_warning)
                : type(type)
                , font_size(font_size)
                , font_name(font_name)
                , wordwrap(wordwrap)
                , horzscrollbar(horzscrollbar)
                , warning(warning)
                , bookmarkstyle(bookmarkstyle)
                , foldstyle(foldstyle)
                , commentfold(commentfold)
          {}
     };
```

Класс StyleItem хранит в себе список элементов класса StyleProperty.

В диалоге хранится список:

```
StyleItemList style_list;
```

В него заносятся указатели на настройки каждого элемента окон редактора.

Это производится в методе createStyles:

```
void ViewPreferences::createStyles()
{
     StyleItem* item;
     item = new StyleItem(IT_ROOT, all_font_size, all_font_name);
     item->properties.push_back(new StyleProperty(item, IT_ROOT,
null_font_style, all_fg_color, all_bg_color));
     style_list.push_back(item);

     item = new StyleItem(IT_EDITOR, style_editor.font.size,
style_editor.font.name, style_editor.window.wordWrap,
style_editor.window.showHorzScrollBar, style_editor.bookmarkStyle,
style_editor.foldStyle, style_editor.commentFold);
     item->properties.push_back(new StyleProperty(item, IT_EDITOR,
style_editor.identifierStyle, style_editor.identifierColor,
style_editor.backgroundColor));
     item->properties.push_back(new StyleProperty(item,
IT_EDITOR_PLAINTEXT, style_editor.defaultStyle,
style_editor.defaultColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
     item->properties.push_back(new StyleProperty(item,
IT_EDITOR_IDENTIFICATOR, style_editor.identifierStyle,
style_editor.identifierColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
     item->properties.push_back(new StyleProperty(item,
IT_EDITOR_KEYWORD, style_editor.keywordStyle,
style_editor.keywordColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
```

```
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_FUNCTION, style_editor.functionsStyle,
style_editor.functionsColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_TRACE, style_editor.traceStyle, style_editor.traceColor,
null_bg_color, null_fg_color, style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_COLOR, style_editor.colorStyle, style_editor.colorColor,
null_bg_color, null_fg_color, style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_COMMENT, style_editor.commentStyle,
style_editor.commentColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_NUMBER, style_editor.numberStyle,
style_editor.numberColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_STRING, style_editor.stringStyle,
style_editor.stringColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_OPERATOR, style_editor.operatorStyle,
style_editor.operatorColor, null_bg_color, null_fg_color,
style_editor.backgroundColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_CARET, null_font_style, style_editor.caretColor,
null_bg_color));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_TEXTSELECTION, null_font_style, null_fg_color,
style_editor.selectionBgColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_BOOKMARK, null_font_style, style_editor.bookmarkFgColor,
style_editor.bookmarkBgColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_FOLD, null_font_style, style_editor.foldFgColor,
style_editor.foldBgColor));
    item->properties.push_back(new StyleProperty(item,
IT_EDITOR_ERROR, null_font_style, null_fg_color,
style_editor.errorBgColor));
    style_list.push_back(item);

  …

}
```

Для остальных настроек запись в список производится аналогичным образом.

Все изменения настроек, связанных с цветом и темами, производятся через этот список StyleItem. При изменении элемента из этого списка изменяется и стиль, которому этот элемент принадлежит.

5.1.1 Отслеживание действий пользователя

Для отслеживания действий пользователя в диалоге настроек используется механизм сигналов и слотов Qt. При нажатии на элемент графической формы диалог посылает сигнал с определенными параметрами, который обрабатывается слотом, принадлежащим диалогу. Соединение между сигналом и слотом устанавливается следующим образом:

```
    connect(setupCheckBox, SIGNAL(stateChanged(int)), this, 
SLOT(onSetup(int)));
```

Здесь видно, что сигнал stateChanged(int) объекта setupCheckBox подключается к слоту onSetup(int) диалога ViewPreferences (this). Таким образом, при нажатии пользователя на чекбокс setupCheckBox сигнал отправится и обработается соответствующим слотом:

```
void ViewPreferences::onSetup(int state)
{
     m_setup = state == Qt::Checked ? true : false;
     checkAllData();
}
```

По такому же механизму обрабатываются все остальные действия пользователя в диалоге.

После записи нового значения атрибута в строке 2 вызывается метод checkAllData:

```
void ViewPreferences::checkAllData()
{
     rbool setupFlag = m_setup == g_pApp-
>getFileAssociationSetup() ? true : false;
     rbool checkInFutureFlag = m_checkInFuture == g_pApp-
>getFileAssociationCheckInFuture() ? true : false;
     rbool openLastProjectFlag = m_openLastProject == g_pApp-
>getOpenLastProject() ? true : false;
     rbool showFullNameFlag = m_showFullName == g_pApp-
>getShowCaptionFullName() ? true : false;
```

```
     rbool style_editor_flag = style_editor == g_pApp->getStyle()-
>style_editor ? true : false;
     rbool style_build_flag = style_build == g_pApp->getStyle()-
>style_build ? true : false;
     rbool style_debug_flag = style_debug == g_pApp->getStyle()-
>style_debug ? true : false;
     rbool style_trace_flag = style_trace == g_pApp->getStyle()-
>style_trace ? true : false;
     rbool style_results_flag = style_results == g_pApp-
>getStyle()->style_results ? true : false;
     rbool style_find_flag = style_find == g_pApp->getStyle()-
>style_find ? true : false;
     rbool style_chart_flag = style_chart == g_pApp->getStyle()-
>style_chart ? true : false;
     rbool style_frame_flag = style_frame == g_pApp->getStyle()-
>style_frame ? true : false;

     if(setupFlag
          && checkInFutureFlag
          && openLastProjectFlag
          && showFullNameFlag
          && style_editor_flag
          && style_build_flag
          && style_debug_flag
          && style_trace_flag
          && style_results_flag
          && style_find_flag
          && style_chart_flag
          && style_frame_flag)
          buttonApply->setEnabled(false);
     else
          buttonApply->setEnabled(true);
}
```

В данном методе проихсодит сравнение текущих настроек системы с настройками диалога (т.е. теми, которые пользователь мог изменить). Если настройки отличаются, то активируется кнопка «Применить».

При нажатии на кнопку «Применить» вызывается метод apply:

```
void ViewPreferences::onApplyButton()
{
     apply();
     buttonApply->setEnabled(false);
}
```

После применения настроек кнопка деактивируется.

```
Void ViewPreferences::apply()
{
     g_pApp->getStyle()->style_editor  = style_editor;
     g_pApp->getStyle()->style_build   = style_build;
     g_pApp->getStyle()->style_debug   = style_debug;
```

```
    g_pApp->getStyle()->style_trace   = style_trace;
    g_pApp->getStyle()->style_results = style_results;
    g_pApp->getStyle()->style_find    = style_find;
    g_pApp->getStyle()->style_frame   = style_frame;
    g_pApp->getStyle()->style_chart   = style_chart;
    g_pApp->setFileAssociationSetup(m_setup);
    g_pApp->setFileAssociationCheckInFuture(m_checkInFuture);
    g_pApp->setOpenLastProject(m_openLastProject);
    g_pApp->setShowCaptionFullName(m_showFullName);
    g_pApp->getStyle()->updateAllStyles();
}
```

В данном методе новые настройки применяются к стилям системы, и вызывается метод updateAllStyles, который обновляет эти стили, и пользователь видит новое оформление.

При нажатии на кнопку «ОК» происходит вызов метода apply и закрытие диалога:

```
void ViewPreferences::onOkButton()
{
    apply();
    done(Accepted);
}
```

Кнопка «Отмена» закрывает диалог:

```
void ViewPreferences::onCancelButton()
{
    done(Rejected);
}
```

5.2 Реализация диалога «График/Настройки»

Диалог изменяет настройки графика. Он содержит в себе атрибуты:

```
    int      m_valueCountX;
    int      m_valueCountY;
    Qstring  m_chartTitle;
```

Первые два отвечают за число значений по осям графика, последний атрибут отвечает за заголовок графика.

```
    Int      m_sizeMarker;
```

Данный атрибут определяет размер маркера на графике.

```
    Int      traceIndex;
```

Атрибут traceIndex определяет, какое значение откладывается по оси Y.

```
    PTR(ChartView)  m_pView;
    PTR(ChartSerie) m_pSerie;
```

Данные указатели указывают на объекты системы, отвечающие за отрисовку графиков.

Так же, как и в диалоге ViewPreferences, в данном диалоге реализованы методы onCheckAllData и apply:

```
void ChartPreferences::onCheckAllData()
{
    rbool legend = showLegendCheckBox->checkState() ==
Qt::Checked ? true : false;
    rbool showInLegend = showInLegendCheckBox->checkState() ==
Qt::Checked ? true : false;
    rbool showMarker = showMarkerCheckBox->checkState() ==
Qt::Checked ? true : false;
    rbool transparentMarker = transparentMarkerCheckBox-
>checkState() == Qt::Checked ? true : false;

    rbool titleFlag  = titleLineEdit->text() == m_pView-
>getDocument()->getTitle() ? true : false;
    rbool yValueFlag = yValueLineEdit->text().toInt() == m_pView-
>getValueCountX() ? true : false;
    rbool xValueFlag = xValueLineEdit->text().toInt() == m_pView-
>getValueCountY() ? true : false;
    rbool showLegendFlag = legend == m_pView->isDrawLegend();
    rbool showInLegendFlag = showInLegend == m_pSerie-
>options().showInLegend ? true : false;
    rbool markerSizeFlag = markerSizeLineEdit->text().toInt() ==
m_pSerie->options().markerSize ? true : false;
    rbool markerShowFlag = showMarker == m_pSerie-
>options().markerNeedDraw ? true : false;
    rbool markerTransparentFlag = transparentMarker == m_pSerie-
>options().markerTransparent ? true : false;
    rbool titleValueFlag = titleValueLineEdit->text() ==
m_pSerie->options().title;
    rbool colorFlag = colorComboBox->itemData(colorComboBox-
>currentIndex(), Qt::UserRole).value<Qcolor>() == m_pSerie-
>options().color ? true : false;
    rbool markerTypeFlag = markerComboBox-
>itemData(markerComboBox->currentIndex(), Qt::UserRole).toInt() ==
m_pSerie->options().markerType ? true : false;

    rbool valueFlag = traceIndex == yTraceComboBox-
>currentIndex();

    if (titleFlag
        && yValueFlag
```

```
          && xValueFlag
          && showLegendFlag
          && showInLegendFlag
          && markerSizeFlag
          && markerShowFlag
          && markerTransparentFlag
          && titleValueFlag
          && colorFlag
          && markerTypeFlag
          && valueFlag)
          applyButton->setEnabled(false);
     else
          applyButton->setEnabled(true);
}
```

Метод активирует/деактивирует кнопку «Применить».

```
Void ChartPreferences::apply()
{
     traceIndex = yTraceComboBox->currentIndex();

     m_pView->setYAxis(yTraceComboBox->currentIndex() != -1
          ? m_pView->getDocument()-
>getSerieList().at(yTraceComboBox->currentIndex())
          : NULL
     );
     m_pView->setDrawLegend(showLegendCheckBox->checkState() ==
Qt::Checked ? true : false);
     m_pView->setValueCountX(m_valueCountX);
     m_pView->setValueCountY(m_valueCountY);
     m_pView->getDocument()->setTitle(m_chartTitle);

     if (m_pSerie)
     {
          Qstring title = titleValueLineEdit->text();

          ChartSerie::Options options;
          options.title = title;
          options.color = colorComboBox->itemData(colorComboBox-
>currentIndex(), Qt::UserRole).value<Qcolor>();
          options.markerType =
static_cast<Serie::Marker>(markerComboBox-
>itemData(markerComboBox->currentIndex(), Qt::UserRole).toInt());
          options.markerSize = m_sizeMarker;
          options.markerNeedDraw = showMarkerCheckBox->checkState()
== Qt::Checked ? true : false;
          options.markerTransparent = transparentMarkerCheckBox-
>checkState() == Qt::Checked ? true : false;
          options.showInLegend = showInLegendCheckBox->checkState()
== Qt::Checked ? true : false;

          m_pSerie->setOptions(options);
     }
```

```
    m_pView->repaint();
}
```

Метод вызывается при нажатии кнопки «Применить» и сохраняет настройки, после чего вызывается repaint() m_pView, рисующий график с новыми настройками.

Отслеживание действий пользователя происходит механизму, аналогичному диалогу настроек редактора.

5.3 Реализация диалога «Поиск/Найти»

Диалог поиска был реализован как класс FindDialog. В нем хранится структура Settings:

```
    struct Settings
    {
        QString what;
        rbool   matchCase;
        rbool   matchWholeWord;
        rbool   searchDown;

        Settings();
        Settings(CREF(Settings) settings);
    };
```

Структура содержит в себе настройки поиска: слово для поиска, учет регистра букв, искать слово полностью или нет, направление поиска.

Диалог хранит внутри себя два указателя на функции:

```
    OnFindCallback   m_onFindCallback;
    OnCloseCallback  m_onCloseCallback;
```

```
typedef  boost::function<void (const Settings&)>  OnFindCallback;
typedef  boost::function<void ()>                 OnCloseCallback;
```

Данные функции вызываются при нажатии пользователем на кнопки поиска и закрытия диалога соответственно. В самом диалоге эти функции не определены. Инициализация функций производится в месте создания диалога в его конструкторе.

```
FindDialog::FindDialog(QWidget* pParent, const OnFindCallback&
onFindCallback, const OnCloseCallback& onCloseCallback)
    : QDialog(pParent, Qt::Dialog | Qt::CustomizeWindowHint |
Qt::WindowTitleHint | Qt::WindowCloseButtonHint)
    , m_onFindCallback (onFindCallback )
    , m_onCloseCallback(onCloseCallback)
{
    setupUi(this);
```

```
     layout()->setSizeConstraint(QLayout::SetFixedSize);

     connect(findButton,    SIGNAL(clicked(bool)),
this, SLOT(onFindButton()));
     connect(cancelButton,  SIGNAL(clicked(bool)),
this, SLOT(reject()));
     connect(lineEdit,      SIGNAL(textEdited(const QString&)),
this, SLOT(onWhatEdited(const QString&)));
     connect(matchCase,     SIGNAL(stateChanged(int)),
this, SLOT(onMatchCaseChanged(int)));
     connect(wholeWord,     SIGNAL(stateChanged(int)),
this, SLOT(onMatchWholeWordChanged(int)));
     connect(directionDown, SIGNAL(toggled(bool)),
this, SLOT(onDirectionDownToggled(bool)));

     setAttribute(Qt::WA_DeleteOnClose, true);
}
```

Это позволяет использовать для разных окон системы свои собственные функции поиска и сохраняет общий интерфейс диалога.

5.4 Реализация диалога «Поиск/Найти и заменить»

Диалог поиска и замены реализован в классе FindReplaceDialog. В нем также хранится структура настроек, которая отнаследована от настроек диалога поиска:

```
struct Settings : public FindDialog::Settings
{
     QString byWhat;

     Settings();
     Settings(CREF(Settings) settings);
};
```

Здесь к настройкам диалога поиска добавилось слово, на которое производится замена.

Так же, как и диалог поиска, диалог поиска и замены хранит в себе указатели на функции:

```
OnFindCallback   m_onFindCallback;
OnFindCallback   m_onReplaceCallback;
OnFindCallback   m_onReplaceAllCallback;
OnCloseCallback  m_onCloseCallback;
```

Эти указатели определяют фунции, вызываемые при нажатии на «Найти», «Заменить», «Заменить все», и «Отмена» и инициализируются в конструкторе диалога:

```
FindReplaceDialog::FindReplaceDialog(QWidget* pParent, const
OnFindCallback& onFindCallback, const OnFindCallback&
onReplaceCallback, const OnFindCallback& onReplaceAllCallback,
const OnCloseCallback& onCloseCallback)
     : QDialog(pParent, Qt::Dialog | Qt::CustomizeWindowHint |
Qt::WindowTitleHint | Qt::WindowCloseButtonHint)
     , m_onFindCallback      (onFindCallback      )
     , m_onCloseCallback     (onCloseCallback     )
     , m_onReplaceCallback   (onReplaceCallback   )
     , m_onReplaceAllCallback(onReplaceAllCallback)
{
     setupUi(this);

     layout()->setSizeConstraint(QLayout::SetFixedSize);

     connect(findButton,       SIGNAL(clicked()),
this, SLOT(onFindButton()));
     connect(replaceButton,    SIGNAL(clicked()),
this, SLOT(onReplaceButton()));
     connect(replaceAllButton, SIGNAL(clicked()),
this, SLOT(onReplaceAllButton()));
     connect(whatLineEdit,     SIGNAL(textEdited(const QString&)),
this, SLOT(onWhatEdited(const QString&)));
     connect(byWhatLineEdit,   SIGNAL(textEdited(const QString&)),
this, SLOT(onByWhatEdited(const QString&)));
     connect(matchCase,        SIGNAL(stateChanged(int)),
this, SLOT(onMatchCaseChanged(int)));
     connect(wholeWord,        SIGNAL(stateChanged(int)),
this, SLOT(onMatchWholeWordChanged(int)));
     connect(directionDown,    SIGNAL(toggled(bool)),
this, SLOT(onDirectionDownToggled(bool)));

     setAttribute(Qt::WA_DeleteOnClose, true);
}
```

5.5 Реализация диалога «Поиск/Перейти на строчку»

Диалог реализован в классе GoToLineDialog. Конструктор диалога:

```
GoToLineDialog::GoToLineDialog(QWidget* pParent, int line, int
lineCount)
     : QDialog(pParent, Qt::Dialog | Qt::CustomizeWindowHint |
Qt::WindowTitleHint | Qt::WindowCloseButtonHint)
     , m_line (line   )
{
     setupUi(this);

     layout()->setSizeConstraint(QLayout::SetFixedSize);

     label->setText(QString("Номер строки (1-
%1):").arg(lineCount));
```

```
    lineEdit->setValidator(new QIntValidator(1, lineCount,
this));
    lineEdit->setText(QString::number(m_line));
    lineEdit->setFocus();
    lineEdit->selectAll();

    connect(buttonOk, SIGNAL(clicked()), this,
SLOT(onOkButtonClicked()));
    connect(lineEdit, SIGNAL(textEdited(const QString&)), this,
SLOT(onCheckInput(const QString&)));
}
```

QIntValidator ограничивает возможность перехода по строчкам максимальным числом строк в тексте.

5.6 Реализация пунктов меню

Так как главное окно РДО было переведено с MFC на Qt, то изменился способ работы с пунктами меню, их активация/деактивация. Здесь работает механизм сигналов и слотов Qt. Для подключения пунктов меню к методам классов РДО используется следующая конструкция:

```
updateAction(
        1.    pMainWindow->actEditDel,
        2.    activated && !isReadOnly() && (getCurrentPos()
         != getLength() || isSelected()),
        3.    this, &Edit::onEditDel
);
```

В 1 строке указывается кнопка меню главного окна, которую мы хотим подключить.

Строка 2 содержит условие для ее активации, если оно true, кнопка активируется, если false – деактивируется.

В строке 3 указан объект и его метод, к которому будет привязана данная кнопка меню (this – указатель объекта на самого себя).

5.6.1 Реализация пунктов меню «Правка»

Реализация большинства пунктов данного меню при переходе на новую версию РДО не изменилась, только способ соединения, описанный выше.

Метод onEditCompleteWord, вызываемый при автозавершении слова, был обновлен. Теперь в нем используется WordListUtil. В связи с этим изменился способ сортировки слов для правильного отображения их на экране:

```
class compareStringScintilla {
public:
     bool operator()(tstring A, tstring B) {
          return CompareNCaseInsensitive(A.c_str(), B.c_str(),
A.length()) < 0;
     }
};

compareStringScintilla functor;
```

compareStringScintilla – это функтор, в котором определен operator(). Он используется при сравнении двух слов в процессе сортировки.

5.6.2 Реализация пунктов меню «Поиск»

Данные пункты меню используют новые диалоги поиска и замены.

Метод onSearchFind:

```
void Edit::onSearchFind()
{
  1. m_findSettings.what =
     QString::fromStdString(getCurrentOrSelectedWord());

  2. if (!m_pFindDialog)
  3. {
          m_pFindDialog = new FindDialog(
              this,
              boost::bind(&Edit::onFindDlgFind,  this, _1),
              boost::bind(&Edit::onFindDlgClose, this)
          );
  4. }

  5. if(m_pGroup)
  6. {
          m_findSettings.what.isEmpty()
              ? m_findSettings.what = m_pGroup->findStr
              : m_pGroup->findStr = m_findSettings.what;
          m_findReplaceSettings.searchDown = m_pGroup-
          >bSearchDown;
          m_findReplaceSettings.matchCase = m_pGroup->bMatchCase;
          m_findReplaceSettings.matchWholeWord = m_pGroup-
          >bMatchWholeWord;
  7. }

  8. m_pFindDialog->setSettings(m_findSettings);
```

```
   9. m_pFindDialog->show();
   10.     m_pFindDialog->raise();
   11.     m_pFindDialog->activateWindow();
}
```

В строке 1 метод получает слово для поиска, вызывая метод getCurrentOrSelectedWord. Если был выделен текст, то сохраняется он, если текст не был выделен, то берется слово, на котором стоит курсор.

В строках 2-4 создается новый объект диалога поиска. Задаются функции, выполняемые при нажатии на поиск и закрытии диалога.

```
void Edit::onFindDlgFind(CREF(FindDialog::Settings) settings)
{
     m_findSettings = settings;
     if (m_pGroup)
     {
          m_pGroup->findStr = m_findSettings.what;
     }
     onSearchFindNext();
     updateActionFind(isActivated());
}
```

Данный метод сохраняет последнее слово для поиска, и вызывает метод onSearchFindNext, который выполняет поиск. В конце обновляется состояние всех пунктов меню.

```
void Edit::onFindDlgClose()
{
     m_pFindDialog = NULL;
}
```

В даном методе происходит удаление диалога.

В строках 5-7 происходит обновление настроек диалога. Если раньше поиск уже произодился, последние настройки хранятся в m_pGroup.

В строках 8-11 диалог настраивается и ему передается команда на отрисовку.

Метод onSearchReplace вызывается при нажатии на пункт меню «Заменить»:

```
void Edit::onSearchReplace()
{
```

```
     m_findReplaceSettings.what =
QString::fromStdString(getCurrentOrSelectedWord());

     if (!m_pFindReplaceDialog)
     {
          m_pFindReplaceDialog = new FindReplaceDialog(
               this,
               boost::bind(&Edit::onFindReplaceDlgFind,       this,
_1),
               boost::bind(&Edit::onFindReplaceDlgReplace,    this,
_1),
               boost::bind(&Edit::onFindReplaceDlgReplaceAll, this,
_1),
               boost::bind(&Edit::onFindReplaceDlgClose,      this)
          );
     }

     if(m_pGroup)
     {
          m_findReplaceSettings.what.isEmpty()
               ? m_findReplaceSettings.what = m_pGroup->findStr
               : m_pGroup->findStr = m_findReplaceSettings.what;
          m_findReplaceSettings.searchDown = m_pGroup->bSearchDown;
          m_findReplaceSettings.matchCase = m_pGroup->bMatchCase;
          m_findReplaceSettings.matchWholeWord = m_pGroup-
>bMatchWholeWord;
     }

     m_pFindReplaceDialog->setSettings(m_findReplaceSettings);
     m_pFindReplaceDialog->show();
     m_pFindReplaceDialog->raise();
     m_pFindReplaceDialog->activateWindow();
    }
```

Его структура аналогична методу onSearchFind. Изменился тип диалога – FindReplaceDialog и настройки – m_findReplaceSettings.

Пункты меню «Найти следующий» и «Найти предыдущий» реализованы в виде фунций onSearchFindNext и onSearchFindPrevious:

```
void Edit::onSearchFindNext()
{
     findNext(
          m_pGroup->findStr.isEmpty()
               ? m_findSettings.what
               : m_pGroup->findStr,
          m_findSettings.searchDown,
          m_findSettings.matchCase,
          m_findSettings.matchWholeWord
     );
}
```

```
void Edit::onSearchFindPrevious()
{
     findNext(
          m_pGroup->findStr.isEmpty()
               ? m_findSettings.what
               : m_pGroup->findStr,
          !m_findSettings.searchDown,
          m_findSettings.matchCase,
          m_findSettings.matchWholeWord
     );
}
```

В них вызывается метод findNext, который и производит поиск.

Пункт меню «Перейти на строчку» вызывает диалог GoToLineDialog через метод onSearchGotoLine:

```
void Edit::onSearchGotoLine()
{
     GoToLineDialog dialog(this, getCurrentLineNumber() + 1,
getLineCount());
     if (dialog.exec() == QDialog::Accepted)
     {
          setCurrentPos(dialog.getLine() - 1, 0);
     }
}
```

5.6.3Реализация пунктов меню «Просмотр»

Реализация большинства пунктов данного меню при переходе на новую версию РДО не изменилась, только способ соединения, описанный выше.

Диалог настроек вызывается при нажатии пункта меню «Настройки».

```
connect(actViewSettings,&QAction::triggered,this,
     &MainWindow::onViewOptions);
```

Данная конструкция подключает сигнал нажатия кнопки меню к методу onViewOptions, который создает и открывает диалог настроек:

```
void MainWindow::onViewOptions()
{
     ViewPreferences dlg(this);
     dlg.exec();
}
```

5.6.4Реализация пунктов меню «График»

Реализация пункта меню «Создать» не потребовала изменений.

Пункт меню «Экспорт данных» вызывает метод onChartExport:

```
void ChartTree::onChartExport()
{
     if (!g_pTracer->getDrawTrace())
          return;

     LPChartTreeItem pItem = getIfItemIsDrawable(getSelected());
     if (!pItem)
          return;

     LPSerie pSerie = pItem.object_dynamic_cast<Serie>();
     ASSERT(pSerie);
     Serie::ExportData exportData = pSerie->exportData();
     if (exportData.empty())
          return;

     boost::filesystem::path path =
          boost::filesystem::path(g_pModel-
>getFullName().toStdString()).parent_path() /
          QString("%1.csv").arg(pSerie->getTitle()).toStdString();

     QString fileName = QFileDialog::getSaveFileName(
          this,
          "Сохранить",
          QString::fromStdString(path.string()),
          "csv-файл (*.csv);;Все файлы (*.*)"
     );
     if (fileName.isEmpty())
          return;

     QFile data(fileName);
     if (data.open(QIODevice::Text | QFile::WriteOnly |
QFile::Truncate))
     {
          QTextStream stream(&data);
          BOOST_FOREACH(CREF(Serie::ExportData::value_type)
exportItem, exportData)
          {
               stream << exportItem << endl;
          }
          data.close();
     }
}
```

Данный метод получает текущий график, экспортирует из него данные в вектор exportData. После этого открывается диалог сохранения, и данные из exportData переписываются в файл через потоковую запись.

Пункт меню «Схлопнуть время» вызывает метод onChartTimeWrap:

```
void ChartView::onChartTimeWrap()
{
     m_timeWrapFlag = !m_timeWrapFlag;
     recalcLayout();
     updateScrollBars();
     update();
     onUpdateActions(isActivated());
}
```

Данный метод меняет флаг m_timeWrapFlag и перерисовывает окно графиков.

Пункт меню «Настройки» открывает диалог настроек графиков через метод onChartOptions:

```
void ChartView::onChartOptions()
{
     ChartPreferences dlg(this);
     dlg.exec();
}
```

5.7 Реализация всплывающего меню

Всплывающее меню реализовано как отдельный класс PopupMenu. В его конструкторе создаются все необходимые элементы:

```
PopupMenu::PopupMenu(QWidget* pParent)
     : m_pPopupMenu(NULL)
{
     Ui::MainWindow* pMainWindow = g_pApp->getMainWndUI();
     ASSERT(pMainWindow);

     m_pPopupMenu = new QMenu(pParent);
     m_pPopupMenu->addAction(pMainWindow->actEditCopy);
     m_pPopupMenu->addAction(pMainWindow->actEditSelectAll);
     m_pPopupMenu->addSeparator();
     m_pPopupMenu->addAction(pMainWindow->actSearchFind);
     m_pPopupMenu->addAction(pMainWindow->actSearchFindNext);
     m_pPopupMenu->addAction(pMainWindow->actSearchFindPrevious);
     m_pPopupMenu->addSeparator();
```

```
     m_pPopupMenu->addAction(pMainWindow-
>actSearchBookmarksToggle);
     m_pPopupMenu->addAction(pMainWindow->actSearchBookmarkNext);
     m_pPopupMenu->addAction(pMainWindow->actSearchBookmarkPrev);
     m_pPopupMenu->addAction(pMainWindow-
>actSearchBookmarksClearAll);
}
```

5.8 Перевод Scintilla на версию 3.3.0

При переводе были перенесены необходимые патчи из старой версии.

5.8.1 Работа с ключевыми словами при автозавершении

Для перевода Scintilla был разработан класс-оболочка над WordList: WordListUtil. В нем реализован метод getNearestWords:

```
std::vector<std::string> WordListUtil::getNearestWords(const
std::string& userPattern) const
{
     struct PriorityResultItem
     {
          std::string  value;
          float        priority;

          PriorityResultItem()
               : priority(0.0)
          {}
          PriorityResultItem(const std::string& value, float
priority)
               : value   (value   )
               , priority(priority)
          {}

          rbool operator< (const PriorityResultItem& item) const
          {
               return priority == item.priority
                    ? value < item.value
                    : priority > item.priority;
          }
     };

     typedef  std::vector<std::string>  result_type;
     result_type result;

     if (wl.words == 0)
          return result;

     if (userPattern.empty())
     {
          for (int i = 0; i < wl.len; ++i)
```

```
          {
               result.push_back(wl.words[i]);
          }
          return result;
     }

     std::vector<PriorityResultItem> priorityResult;
     for (int i = 0; i < wl.len; ++i)
     {
          boost::iterator_range<char*> findPatternIt =
boost::ifind_first(wl.words[i], userPattern);
          if (!findPatternIt.empty())
          {
               boost::iterator_range<char*>
fullKeywordIt(wl.words[i], wl.words[i] + strlen(wl.words[i]));
               ruint position = findPatternIt.begin() -
fullKeywordIt.begin();
               ruint diff = position + (fullKeywordIt.end() -
findPatternIt.end());
               ruint wLen = fullKeywordIt.end() -
fullKeywordIt.begin();
               float positionPart = float(position) / float(wLen);
               float diffPart = float(diff) / float(wLen);
               float priority = 1 - (positionPart + diffPart) / 2;

     priorityResult.push_back(PriorityResultItem(wl.words[i],
priority));
          }
          else
          {

     priorityResult.push_back(PriorityResultItem(wl.words[i],
0.0));
          }
     }
     std::sort(priorityResult.begin(), priorityResult.end());
     float const minPriority = 0.3f;

     BOOST_FOREACH(const PriorityResultItem& item, priorityResult)
     {
          if(item.priority >= minPriority)
               result.push_back(item.value);
     }
     return result;
}
```

В данном методе создается структура PriorityResultItem, которая хранит пару value – ключевое слово, и priority – приоритет этого слова. В начале метода происходит заполение вектора ключевыми словами в особых случаях.

Далее происходит расчет приоритетов для всех ключевых слов в списке, и на основе расчета формируется результирующий список ключевых слов.

5.9 Изменения в классах стилей

Базовый класс стилей StyleBase больше не содержит указателя на тему:

```
OBJECT(StyleBase)
{
public:
     StyleBase();
     ~StyleBase();

     StyleBase& operator =(const StyleBase& style);
     rbool operator ==(const StyleBase& style) const;
     rbool operator !=(const StyleBase& style) const;

     StyleFont font;
     StyleFont::style defaultStyle;

     QColor defaultColor;
     QColor backgroundColor;
};
```

Атрибуты font и defaultStyle раньше хранились в классе темы.

Подобным изменениям подверглись все классы стилей.

5.10 Изменение способа сохранения и загрузки настроек в операционной системе

Старая версия РДО работала только под Windows, и все настройки системы хранились в реестре. Таким образом, система всегда знала, каким способом и куда сохранять настройки, и откуда загружать.

Стили и темы для сохранения и загрузки настроек использовали механизм виртуальных функций. При этом информация о том, куда сохраняться, содержалась внутри этих классов (т.е каждый класс сам устанавливал, куда ему нужно сохраняться). Такой подход плох тем, что в разных операционных системах свой способ хранения данных о программах.

После слияния классов стилей и тем было решено избавиться от механизма виртуальных функций в пользу механизма наследования. Используя механизм

наследования, программа знает, какие методы вызывать еще на этапе компиляции. Это позволяет однозначно определять цепочку сохранения и загрузки.

Для классов стилей были разработаны методы сохранения и загрузки. Данные методы основаны на записи данных в поток:

```
QSettings& operator<< (QSettings& settings, const StyleBase&
style);
QSettings& operator>> (QSettings& settings, StyleBase& style);
```

В качестве потока здесь выступает объект QSettings, отвечающий за хранение настроек в системе. Методы являются свободными (т.е не принадлежат ни к одному классу). Им просто передается объект, куда сохранять, и что сохранять. Поменяв объект сохранения, менять способ записи в него не нужно.

Метод сохранения StyleBase:

```
QSettings& operator<< (QSettings& settings, const StyleBase&
style)
{
     settings.beginGroup("font");
     settings << style.font;
     settings.endGroup();

     settings.beginGroup("theme");
     settings.setValue("default_color",
style.defaultColor.name());
     settings.setValue("background_color",
style.backgroundColor.name());
     settings.setValue("default_style", style.defaultStyle);
     settings.endGroup();

     return settings;
}
```

QSettings приходят извне, уже настроеные на запись. Далее в методе открываются группы (разделы), куда пишутся настройки.

В данных методах используется механизм наследования. Метод сохранения ModelStyle:

```
QSettings& operator<< (QSettings& settings, const ModelStyle&
style)
{
     settings << static_cast<const ParserStyle&>(style);

     settings.beginGroup("theme");
     settings.setValue("fold_fg_color", style.foldFgColor.name());
     settings.setValue("fold_bg_color", style.foldBgColor.name());
     settings.setValue("error_bg_color",
style.errorBgColor.name());
     settings.setValue("fold_style", style.foldStyle);
     settings.setValue("comment_fold", style.commentFold);
     settings.endGroup();

     settings.beginGroup("auto_complete");
     settings << style.autoComplete;
     settings.endGroup();

     settings.beginGroup("margin");
     settings << style.margin;
     settings.endGroup();


     return settings;
}
```

В 1 строке метода вызывается оператор записи в поток предка ModelStyle – ParserStyle:

```
QSettings& operator<< (QSettings& settings, const ParserStyle&
style)
{
     settings << static_cast<const EditStyle&>(style);

     settings.beginGroup("theme");
     settings.setValue("identifier_color",
style.identifierColor.name());
     settings.setValue("keyword_color",
style.keywordColor.name());
     settings.setValue("functions_color",
style.functionsColor.name());
     settings.setValue("trace_color", style.traceColor.name());
     settings.setValue("color_color", style.colorColor.name());
     settings.setValue("comment_color",
style.commentColor.name());
     settings.setValue("number_color", style.numberColor.name());
     settings.setValue("string_color", style.stringColor.name());
     settings.setValue("operator_color",
style.operatorColor.name());

     settings.setValue("identifier_style", style.identifierStyle);
     settings.setValue("keyword_style", style.keywordStyle);
```

```
    settings.setValue("functions_style", style.functionsStyle);
    settings.setValue("trace_style", style.traceStyle);
    settings.setValue("color_style", style.colorStyle);
    settings.setValue("comment_style", style.commentStyle);
    settings.setValue("number_style", style.numberStyle);
    settings.setValue("string_style", style.stringStyle);
    settings.setValue("operator_style", style.operatorStyle);
    settings.endGroup();

    return settings;
  }
```

Внутри него вызывается оператор записи в поток для предка EditStyle:

```
QSettings& operator<< (QSettings& settings, const EditStyle&
style)
{
    settings << static_cast<const StyleBase&>(style);

    settings.beginGroup("tab");
    settings << style.tab;
    settings.endGroup();
    settings.beginGroup("window");
    settings << style.window;
    settings.endGroup();

    settings.beginGroup("theme");
    settings.setValue("default_color",
style.defaultColor.name());
    settings.setValue("background_color",
style.backgroundColor.name());
    settings.setValue("caret_color", style.caretColor.name());
    settings.setValue("selection_bg_color",
style.selectionBgColor.name());
    settings.setValue("bookmark_fg_color",
style.bookmarkFgColor.name());
    settings.setValue("bookmark_bg_color",
style.bookmarkBgColor.name());
    settings.setValue("default_style", style.defaultStyle);
    settings.setValue("bookmark_style", style.bookmarkStyle);
    settings.endGroup();

    return settings;
}
```

И далее по цепочке оператор записи основного класса стилей – StyleBase, описанный выше.

Соответственно сперва сохраняются по цепочки все предки, начиная с базового, последним сохраняются атрибуты ModelStyle.

Подобный механизм позволил добиться независимости от способа хранения настроек в операционной системе.

6 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

6.1 Выбор критерия поиска ближайшего слова при автозавершении слов

В рамках данного дипломного проекта производился перевод текстового редактора на новую версию Scintilla. В процессе перехода был разработан новый алгоритм поиска ближайшего слова при автозавершении (autocomplete).

Автозавершение – функция, предусматривающая итеративный ввод текста по дополнению текста по уже введенной его части. В случае, если дополнение текста однозначно (т.е. существует единственный вариант слова, который может быть дополнен), текстовый редактор РДО автоматически дополняет слово. Если нет, выводится список слов, которые подходят под введенный пользователем текст.

Существует два варианта отображения списка: отображается полный список ключевых слов, доступных в РДО, или отображается список слов, ближайших по написанию с текстом, введенным пользователем. В первом случае все однозначно, во втором же возникает вопрос, какие слова считать ближайшими и по какому алгоритму их отбирать.

В рамках данной задачи был рассчитан критерий (приоритет), по которому выбираются ближайшие слова.

В первом случае критерий рассчитывается по близости вхождения текста, введенного пользователем, к началу ключевого слова.

$$priority = 1 - \frac{position}{wordLength},$$

где *priority* – приоритет, по которому отбираются слова,

$position$ – позиция вхождения текста пользователя в ключевое слово,

$wordLength$ - длина ключевого слова.

Однако при таком способе расчета возникли проблемы с определением ближайшего слова в случае, если ключевые слова имеют одинаковое начало, и

разница в их длине не так велика. Поэтому в список ближайших слов попадало много лишних слов. Помимо прочего, возникали ситуации, когда желаемого слова вообще не было в результирующем списке.

Во втором случае критерий определения ближайшего слова рассчитывается по формуле:

$$priority = 1 - \frac{(positionPart + diffPart)}{2},$$

где *priority* – приоритет, по которому отбираются слова,

$positionPart = position/wordLength$ - отношение первого вхождения введенного пользователем текста в проверяемое ключевое слово к длине слова,

$diffPart = diff/wordLength$ - отношение, определяющее количество символов в ключевом слове, отличных от введенного пользователем текста.

Данный критерий более полно учитывает соответствие ключевого слова пользовательскому тексту, так как учитывает не только позицию текста в слове, но и разницу в символах между текстом и словом.

Сравнив оба критерия, было решено использовать второй способ расчета, так как он показал более высокие результаты, чем первый.

6.2 Алгоритм поиска ближайших слов

Для поиска ближайших слов был разработан алгоритм (рис. 5.2.1). В нем используется рассмотренный ранее критерий для отбора слов из списка ключевых. Экспериментальным образом было установлено пороговое значение критерия отбора, равное 0.3. Все слова, которые после расчета приоритета, имеют меньший приоритет, отсеиваются.

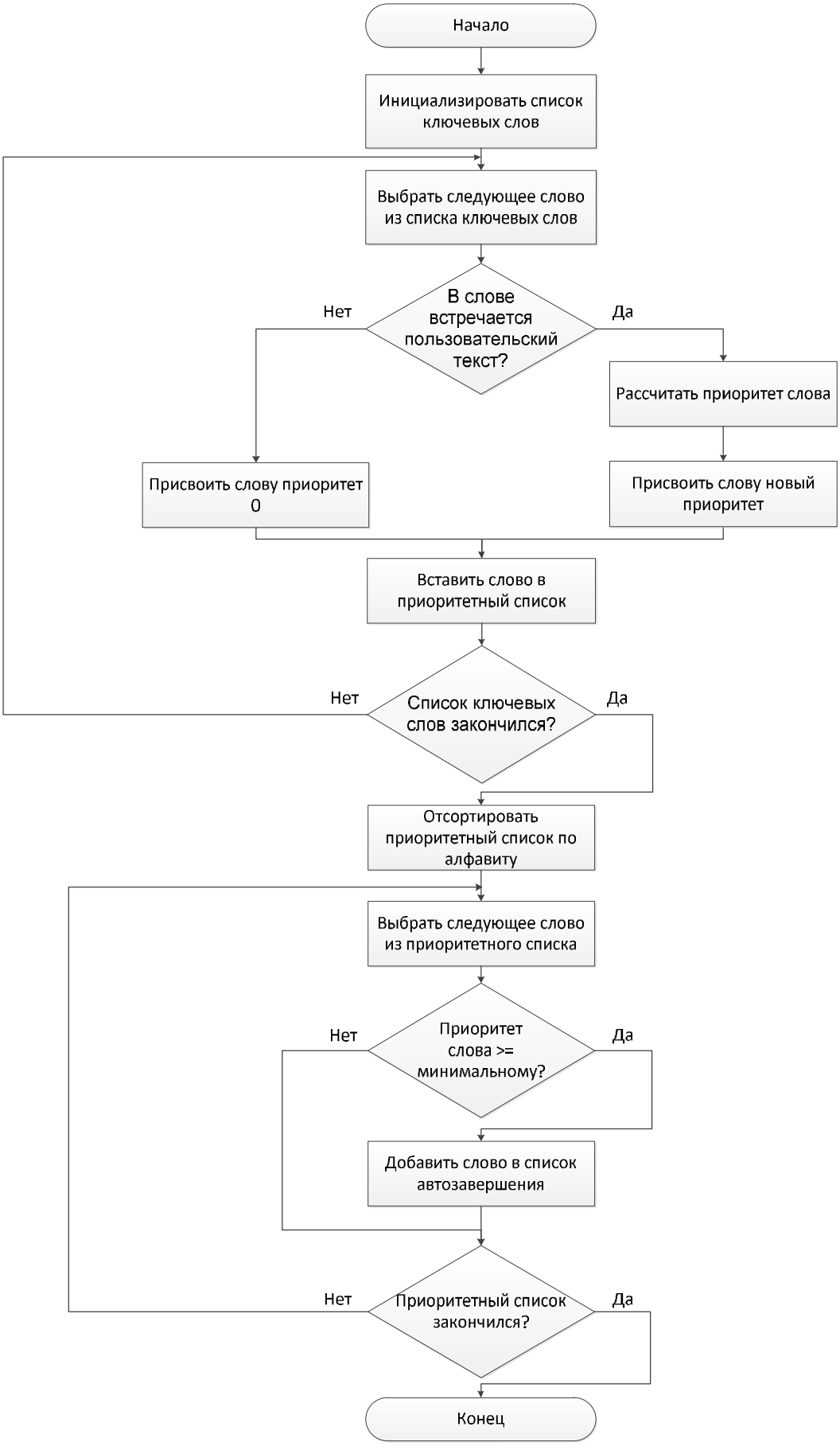

Рисунок 6.1 **Алгоритм поиска ближайших слов**

7 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

7.1 Введение

В этой главе выполнена организационно-экономическая часть дипломного проекта, выполнен расчет затрат на разработку кроссплатформенной версии системы имитационного моделирования RAO-studio. Для этого будут произведены следующие расчёты:

- Оценка трудоёмкости разработки и внедрения программного обеспечения;
- Оценка состава и численности разработчиков программного обеспечения;
- Оценка трудоёмкости работ каждого участника проектной группы;
- Оценка стоимости разработки кроссплатформенной версии системы имитационного моделирования RAO-studio.

Расчет действителен на 2-ой квартал 2013 года (цены на программное обеспечение, оборудование, расходные материалы, уровень заработной платы исполнителей и т.д.).

7.2 Организация и планирование процесса разработки ПП

В данном разделе дипломного проекта, посвященном экономическим вопросам, будет произведен расчет трудоемкости и стоимости разработки ПП. Целью данного расчета является определение затрат на разработку ПП.

Организация и планирование процесса разработки ПП предусматривает выполнение следующих работ:

- формирование состава выполняемых работ и группировка их по стадиям разработки;
- расчет трудоемкости выполнения работ;
- установление профессионального состава и расчет количества исполнителей;
- определение продолжительности выполнения отдельных этапов разработки;
- построение календарного графика выполнения разработки;
- контроль выполнения календарного графика.

Трудоемкость разработки ПП зависит от ряда факторов, рассмотрим эти факторы применительно к данной системе.

По степени новизны разрабатываемый программный продукт может быть отнесен к группе новизны «Б» (разработка ПП, не имеющей аналогов, в том числе разработка пакетов ПП).

По степени сложности алгоритма функционирования - к 1 группе сложности (реализующие оптимизационные и моделирующие алгоритмы)

По виду представления исходной информации ПП относится к группе 11 – исходная информация представлена в форме документов, имеющих различный формат и структуру.

По структуре выходной информации относим ПП к группе 22 - требуется вывод на печать одинаковых документов, вывод информационных носителей на машинные носители.

Ниже приведен перечень стадий и состава работ (Таблица 7.1) при разработке программной продукции.

Таблица 7.1. **Перечень стадий и состав работ**

Стадия разработки программного продукта	Состав выполняемых работ
Техническое задание	Постановка задач, выбор критериев эффективности. Разработка технико-экономического обоснования разработки. Определение состава пакета прикладных программ, состава и структуры информационной базы. Предварительный выбор методов выполнения работы. Разработка календарного плана выполнения работ.
Эскизный проект	Предварительная разработка структуры входных и выходных данных. Разработка общего описания алгоритмов реализации решения задач. Разработка пояснительной записки. Консультации разработчиков постановки задач. Согласование и утверждение эскизного проекта.
Технический проект	Разработка алгоритмов решения задач. Разработка пояснительной записки. Согласование и утверждение технического проекта. Разработка структуры программы. Разработка программной документации и передача ее для включения в технический проект. Уточнение структуры, анализ и определение формы представления входных и выходных данных. Выбор конфигурации технических средств.
Рабочий проект	Комплексная отладка задач и сдача в опытную эксплуатацию. Разработка проектной документации. Программирование и отладка программ. Описание контрольного примера. Разработка программной документации. Разработка, согласование программы и методики испытаний. Предварительное проведение всех видов испытаний.
Внедрение	Подготовка и передача программной документации для сопровождения с оформлением соответствующего Акта приема-сдачи работ. Проверка алгоритмов и программ решения задач, корректировка документации после опытной эксплуатации программного продукта.

7.2.1Расчет трудоемкости разработки технического задания

Трудоемкость разработки технического задания рассчитывается по формуле: $t_{ТЗ,} = t_{РЗ,} + t_{РП}$,

где $t_{РЗ,}$ - затраты времени разработчика постановки задачи на разработку ТЗ, чел.-дн.;

$t_{РП}$ - затраты времени разработчика ПП на разработку ТЗ, чел.-дн.

$t_{РЗ,} = t_З \cdot K_{РЗ,}$

$t_{РП} = t_З \cdot K_{РП}$,

где $t_З$ - норма времени на разработку ТЗ на ПП, чел.-дн.

Исходя из рекомендаций [8] в данном случае $t_З = 57$ чел.-дн. (Группа новизны ПП - Б; функциональное назначение ПП – реализует оптимизационные и моделирующие алгоритмы);

$K_{РЗ}$ - коэффициент, учитывающий удельный вес трудоёмкости работ, выполняемых разработчиком постановки задачи на стадии ТЗ. В случае совместной с разработчиком ПО разработки: $K_{РЗ} = 0,65$;

$K_{РП}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком ПП на стадии ТЗ. В случае совместной с разработчиком ПО разработки: $K_{РП} = 0,35$.

$t_{ТЗ} = 57 \cdot 0,65 + 57 \cdot 0,35 = 57$ чел.-дн.

7.2.2Расчет трудоемкости выполнения эскизного проекта

Трудоемкость разработки эскизного проекта $t_{ЭП}$ рассчитывают по формуле: $t_{ЭП} = t_{РЗ} + t_{РП}$, где:

$t_{РЗ}$ - затраты времени разработчика постановки задач на разработку ЭП, чел.-дни;

$t_{РП}$ - затраты времени разработчика ПП на разработку ЭП, чел.-дни;

$t_{РЗ} = t_Э \cdot K_{РЗ}$,

$t_{РП} = t_Э \cdot K_{РП}$

где:

$t_Э$ - норма времени на разработку ЭП программного продукта, чел.-дни.

$t_Э = 117$ чел.-дней;

$K_{РЗ}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ЭП . В случае совместной с разработчиком ПО разработки ЭП: $K_{РЗ} = 0,7$;

$K_{РП}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком ПП на стадии ЭП. В случае совместной с разработчиком постановки задач разработки ЭП: $K_{РП} = 0,3$;

$t_{ЭП} = 117 \cdot 0,7 + 117 \cdot 0,3 = 117$ чел.-дней.

7.2.3Расчет трудоемкости выполнения технического проекта

Трудоемкость разработки технического проекта зависит от функционального назначения ПП, количества разновидностей форм входной и выходной информации и определяется как сумма времени, затраченного разработчиком постановки задач и разработчиком ПП, и определяется по формуле: $t_{ТП} = (t_{РЗ} + t_{РП}) \cdot K_В \cdot K_Р$, где:

$t_{РЗ}$, $t_{РП}$ - норма времени, затрачиваемого на разработку ТП разработчиком постановки задач и разработчиком ПО:

$t_{РЗ} = 86$ ч. ,

$t_{РП} = 23$ ч.

$K_Р$ - коэффициент учёта режима обработки информации:

$K_Р = 1.45$

$K_В$ - коэффициент учёта вида используемой информации, определяется из выражения:

$$K_В = \frac{K_П \cdot N_П + K_{НС} \cdot N_{НС} + K_Б \cdot N_Б}{N_П + N_{НС} + N_Б},$$

где:

$K_П$, $K_{НС}$, $K_Б$ - значения коэффициентов учёта вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно;

$N_П$, $N_{НС}$, $N_Б$ - количество наборов данных для переменной, нормативно-справочной информации и баз данных соответственно.

Для группы новизны Б:

$K_П = 1,2$ $N_П = 2$ j

$K_{НС} = 1,08$ $N_{НС} = 1$

$K_Б = 3,12$ $N_Б = 0$

Таким образом $K_В = \frac{1,2 \cdot 2 + 1,08 \cdot 1 + 3,12 \cdot 0}{2 + 1 + 0} = 1,16$

$t_{ТП} = (86 + 23) \cdot 1,16 \cdot 1,45 = 183,34$ чел.-дней

7.2.4Расчет трудоемкости выполнения рабочего проекта

Трудоёмкость разработки рабочего проекта зависит от функционального назначения ПП, количества разновидностей форм входной и выходной информации, сложности алгоритма функционирования, сложности контроля информации, степени использования готовых программных модулей, уровня алгоритмического языка программирования и определяется по формуле:

$$t_{РП} = K_К \cdot K_Р \cdot K_Я \cdot K_З \cdot K_{ИА} \cdot (t_{РЗ} + t_{РП}),$$

где:

$K_Р$ - коэффициент учёта режима обработки информации = 1.45;

K_K - коэффициент учёта сложности контроля информации = 1,07 ;

$K_Я$ - коэффициент учёта уровня алгоритмического языка программирования =1,0 ;

$K_З$ - коэффициент учёта степени использования готовых программных модулей – 0.8;

(на 20-25% использования готовых программных модулей);

$K_{ИА}$- коэффициент учёта вида используемой информации и сложности алгоритма ПП.

Значение коэффициента $K_{ИА}$ определяют из выражения:

$$K_{ИА} = \frac{K'_П \cdot N_П + K'_{НС} \cdot N_{НС} + K'_Б \cdot N_Б}{N_П + N_{НС} + N_Б},$$

где:

$K'_П, K'_{НС}, K'_Б$ - значения коэффициентов учёта сложности алгоритма ПП и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

В нашем случае присутствует только один вид используемой информации – переменной; соответствующее группе новизны Б значение коэффициента $K'_П = 1.62$ $K_{НС} = 0,97$ $K'_Б = 0.81$

(группа новизны - Б)

$$K_{ИА} = \frac{1,62 \cdot 2 + 0,97 \cdot 1 + 0,81 \cdot 0}{2 + 1 + 0} = 1,403$$

$t_{РЗ}$, $t_{РП}$ - норма времени, затраченного на разработку РП на алгоритмическом языке высокого уровня разработчиком постановки задач и разработчиком ПП.

$t_{РЗ} = 28$

$t_{РП} = 156$

$t_{РП} = 1{,}16 \cdot 1{,}45 \cdot 1{,}0 \cdot 0{,}8 \cdot 1{,}403 \cdot (28 + 156) = 347{,}37$ чел.-дней.

7.2.5Расчет трудоемкости выполнения внедрения

Трудоёмкость выполнения стадии внедрения может быть рассчитана по формуле:

$$t_B = (t_{РЗ} + t_{РП}) \cdot K_K \cdot K_P \cdot K_З,$$

где:

$t_{РЗ}$, $t_{РП}$ - норма времени, затраченного разработчиком постановки задач и разработчиком ПО на выполнение процедур внедрения ПП.

$t_{РЗ} = 29$

$t_{РП} = 33$

$t_B = (29 + 33) \cdot 1{,}16 \cdot 1.45 \cdot 0.8 = 83{,}42$ чел.-дней.

7.2.6Расчет суммарной трудоемкости

Таким образом, суммарная трудоемкость разработки программной продукции:

$t_{ПП} = 57 + 117 + 183{,}34 + 347{,}37 + 83{,}42 = 788{,}13$ чел.-дней.

Трудоемкость этапов разработки программного продукта (Таблица 7.2.):

Таблица 7.2.

№пп	**Стадия разработки**	**Трудоемкость чел.-дни**
1	Техническое задание	57
2	Эскизный проект	117
3	Технический проект	183,34
4	Рабочий проект	347,37
5	Внедрение	83,42
Итого:		788,13

Для целей контроля и планирования выполнения работ в данном случае используем ленточный график, потому что разработку осуществляет небольшой, стабильный по составу коллектив исполнителей . Для построения ленточного графика необходимо знать срок начала работ, срок окончания работ и количество работников, участвующих на каждом этапе разработки.

Продолжительность выполнения всех работ по этапам разработки программного продукта определяется из выражения

$$T_i = \frac{\tau_i + Q}{n_i}$$, где

τ_i - трудоемкость i-й работы, чел.-дни;

Q - трудоемкость дополнительных работ, выполняемых исполнителем, чел.-дни;

n_i - количество исполнителей, выполняющих i-ю работу, чел.

Пусть разработка системы на стадии разработки технического задания ведется одним специалистом, не привлекаемым к дополнительным работам, на стадии разработки эскизного проекта – тремя специалистами, на стадии разработки технического и рабочего проекта - тремя специалистами, а на стадии внедрения

– двумя специалистами, то продолжительность разработки программного продукта:

$$\mathrm{T} = \frac{57}{1} + \frac{117}{3} + \frac{183,34}{3} + \frac{347,37}{3} + \frac{83,42}{2} = 314,61 \text{раб.дня.}$$

Для построения календарного плана необходимо перевести рабочие дни в календарные. Для этого длительность каждого этапа нужно разделить на поправочный коэффициент К=0.7. В результате получим:

$$T = \frac{314,61}{0.7} = 449,44 \text{ кал.дн} \qquad T_{ТЗ} = \frac{57}{0.7} = 81,42 \text{ кал.дн}$$

$$Тэп = \frac{117}{0,7 \cdot 3} = 55,71 \text{ кал.дн} \qquad T_{ТП} = \frac{183,34}{0,7 \cdot 3} = 87,3 \text{ кал.дн}$$

$$T_{РП} = \frac{347,37}{0,7 \cdot 3} = 165,41 \text{ кал.дн} \qquad Тв = \frac{83,42}{0,7 \cdot 2} = 59,58 \text{ кал.дн}$$

7.3 Определение стоимости разработки ПП

Для определения стоимости работ, необходимо на основании плановых сроков выполнения работ и численности исполнителей, рассчитать общую сумму затрат на разработку программного продукта.

Себестоимость ПП представляет собой стоимостную оценку используемых в процессе производства продукции работ, услуг, природных ресурсов, сырья материалов, топлива, энергии, основных фондов, трудовых ресурсов, а также других затрат на ее производство и реализацию.

Затраты, образующие себестоимость ПП, группируются в соответствии с их экономическим содержанием по следующим элементам:

- Расчёт основной заработной платы;
- Расчёт дополнительной заработной платы;

- Амортизация оборудования
- Отчисления в социальные фонды;
- Накладные расходы.

7.3.1Расчет основной заработной платы

В статью включается основная заработная плата всех исполнителей, непосредственно занятых разработкой данного ПП, с учётом их должностного оклада и времени участия в разработке. Расчёт ведётся по формуле:

$$C_{зо} = \sum_i \frac{З_i}{d} \cdot t_i,$$

где $З_i$ - среднемесячный оклад i-го исполнителя, руб.; d – среднее количество рабочих дней в месяце; t_i - трудоемкость работ, выполняемых i-м исполнителем, чел.-дни (определяется из календарного плана-графика).

$З_i = 35000\,руб.$;

$d = 21$;

$t_i = 788{,}13$

$$C_{зо} = \frac{35000}{21} \cdot 788{,}13 = 1313550\ руб.$$

7.3.2Расчет дополнительной заработной платы

В статье учитываются все выплаты непосредственным исполнителям за время (установленное законодательством), непроработанное на производстве, в том числе: оплата очередных отпусков, компенсация за неиспользованный отпуск, оплата льготных часов подросткам и др. Расчет ведётся по формуле:

$$C_{зд} = C_{зо} \cdot a_д,$$

где $a_д$ - коэффициент на дополнительную заработную плату.

$a_д = 0{,}2$

$C_{зд} = 1313550 \cdot 0{,}2 = 262710\ руб.$

7.3.3Отчисления на социальное страхование

В статье учитываются отчисления в бюджет социального страхования по установленному законодательному тарифу от суммы основной и дополнительной заработной платы, т.е.

$C_{сс} = a_{сс} \cdot (C_{зо} + C_{зд})$

$a_{д} = 0,30 + 0,002 = 0,302$

$C_{зд} = (1313550 + 262710) \cdot 0,302 = 476031$ *руб.*

7.3.4Накладные расходы

В статье учитываются затраты на общехозяйственные расходы, непроизводительные расходы и расходы на управление. Накладные расходы определяют в процентном отношении к основной заработной плате, т.е.

$C_{н} = C_{зо} \cdot a_{н}$,

$a_{н} = 1,8$

$C_{зд} = 1313550 \cdot 1,8 = 2364390$ *руб.*

7.3.5Расходы на оборудование (амортизация)

В статье учитываются суммарные затраты на приобретение или проектирование специального оборудования:

$C_{СО} = \sum_{i} \frac{Ц_{Б\,i} \cdot a_i}{100 \cdot F_{Д}} t_i$, где

$Ц_{Б\,i}$ - балансовая цена i-го вида оборудования, руб.;

a_i - норма годовых амортизационных отчислений для оборудования i-го вида, %;

t_i - время использования i-го вида оборудования при выполнении данной разработки, ч.

$F_д$ - действительный годовой фонд рабочего времени сотрудника, ч. (5-ти дневная неделя, 8-и часовой рабочий день – 2080 ч.);

Табл. **7.3.** Специальное оборудование и ПО, используемое при разработке

№ п/п	Наименование	Единица измерения	Количество	Цена за единицу	Сумма, руб
1	ПЭВМ	шт	1	20000	40000

Затраты на амортизацию оборудования (ПЭВМ):

$C_{оо}$ = 40 000 · 20 · 449,44 · 8 / (100 · 2080) = 13829 руб.

7.3.6 Результаты расчетов затрат на разработку программного продукта

№ пп	Наименование статьи	Сметная стоимость, руб.
1	основная заработная плата	1313550
2	Дополнительная заработная плата	262710
3	Отчисления на социальное страхование	476031
4	Накладные расходы	2364390
5	Расходы на использование оборудования	13829
	Итого	4430510

7.4 Вывод

Произведенный расчет показал:

- Суммарная трудоемкость продукта составляет 788,13 чел.дней. Поэтому на каждом этапе необходимо привлечение нескольких специалистов.

- Длительность разработки программного продукта составляет 449,44 календарный день.
- Себестоимость программного продукта составляет 4430510 руб.

8 МЕРОПРИЯТИЯ ПО ОХРАНЕ ТРУДА И ТЕХНИКЕ БЕЗОПАСНОСТИ

8.1 Введение

В данном разделе дипломного проекта рассматриваются и анализируются опасные и вредные факторы при использовании программного продукта RAO-studio в рамках комплекса лабораторных занятий. Описываются мероприятия по обеспечению безопасности и безвредных условий труда, приводятся рекомендации по способам и методам ограничения действия вредных и исключению воздействия опасных факторов на студентов и преподавательский состав, проходящих и проводящих занятия в компьютерной сфере. При этом на человека, работающего в зоне действия ПЭВМ, влияет большое количество вредных факторов, состояние которых необходимо контролировать.

8.2 Опасные и вредные факторы

8.2.1Опасные факторы:

8.2.1.1 Физические

- пожарная опасность, обусловленная наличием на рабочем месте мощного источника энергии;
- поражение электротоком, обусловленное повышенным значением напряжения в электрической цепи, замыкание которой может произойти через тело человека.

8.2.2Вредные факторы:

8.2.2.1 Физические

- повышенные уровни электромагнитного излучения;
- повышенный уровень статического электричества;
- повышенные уровни запыленности воздуха рабочей зоны;
- пониженная или повышенная влажность воздуха рабочей зоны;
- пониженная или повышенная подвижность воздуха рабочей зоны;

- повышенный уровень шума;
- повышенный уровень вибрации;
- повышенный или пониженный уровень освещенности;
- повышенный уровень прямой блесткости;
- повышенный уровень отраженной блесткости;
- неравномерность распределения яркости в поле зрения;
- повышенная яркость светового изображения;
- повышенный уровень пульсации светового потока;

8.2.2.2 Химические

- повышенное содержание в воздухе рабочей зоны двуокиси углерода, вредных выхлопов и газов;

8.2.2.3 Психофизиологические

- напряжение зрения;
- напряжение внимания;
- интеллектуальные нагрузки;
- эмоциональные нагрузки;
- длительные статические нагрузки;
- монотонность труда;
- большой объем информации обрабатываемой в единицу времени;
- нерациональная организация рабочего места.

8.2.2.4 Биологические

- повышенное содержание в воздухе рабочей зоны микроорганизмов.

8.3 Требования к помещениям для работы с ПЭВМ

Помещения для эксплуатации ПЭВМ должны иметь естественное и искусственное освещение. Эксплуатация ПЭВМ в помещениях без естественного освещения допускается только при соответствующем обосновании и наличии положительного санитарно-эпидемиологического заключения, выданного в установленном порядке.

Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток. Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электроннолучевой трубки (ЭЛТ) должна составлять не менее 6 м2, в помещениях культурно-развлекательных учреждений и с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 м2. При использовании ПВЭМ с ВДТ на базе ЭЛТ (без вспомогательных устройств - принтер, сканер и др.), отвечающих требованиям международных стандартов безопасности компьютеров, с продолжительностью работы менее 4-х часов в день допускается минимальная площадь 4,5 м2 на одно рабочее место пользователя (взрослого и учащегося высшего профессионального образования).

Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно-отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5.

Полимерные материалы используются для внутренней отделки интерьера помещений с ПЭВМ при наличии санитарно-эпидемиологического заключения.

Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

8.3.1 Требования к микроклимату, вредных химических веществ в воздухе на рабочих местах, оборудованных ПЭВМ

В производственных помещениях, в которых работа с использованием ПЭВМ является основной и связана с нервно-эмоциональным напряжением, должны обеспечиваться оптимальные параметры микроклимата для категории работ 1а в соответствии с действующими санитарно-эпидемиологическими нормативами микроклимата производственных помещений. На других рабочих местах следует поддерживать параметры микроклимата на допустимом уровне, соответствующем требованиям указанных выше нормативов.

В производственных помещениях, в которых работа с использованием ПЭВМ является основной, величины показателей микроклимата на рабочих местах должны соответствовать действующим санитарным нормам (СанПиН 2.2.4.548-96 Гигиенические требования к микроклимату производственных помещений). При использовании данной подсистемы характер труда оператора ПЭВМ относиться к категории 1а, т.к. работы производятся сидя и не требуют физического напряжения, расход энергии составляет до 120 ккал/ч, следовательно, величины показателей микроклимата на рабочих местах в ПЭО должны соответствовать оптимальным величинам показателей для категории работ 1а (см. Таблица 8.1).

Таблица 8.1. **Оптимальные величины показателей микроклимата**.

Период года	Категория работ	Температура воздуха, ° С не более	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Легкая – Ia	22-24	40-60	0,1
Теплый	Легкая – Ia	23-25	40-60	0,1

Содержание вредных химических веществ в воздухе производственных помещений, в которых работа с использованием ПЭВМ является основной, не должно превышать предельно допустимых концентраций вредных веществ в воздухе рабочей зоны в соответствии с действующими гигиеническими

нормативами (Гигиенические нормативы ГН 2.1.6.13 1338-03 Предельно допустимые концентрации (ПДК) загрязняющих веществ в атмосферном воздухе населенных мест). ПДК среднесуточная: окись углерода – 3,0 мг/м^3, окислы азота – 0,04 мг/м^3.

Данный микроклимат может обеспечиваться системой общеобменной приточно-вытяжной вентиляции. В приточной ветви нужно предусмотреть использование фильтров очистки воздуха, для удаления из воздуха вредных веществ перед его подачей на рабочие места. Для повышения влажности воздуха в помещениях с ВДТ и ПЭВМ следует применять увлажнители воздуха, заправляемые ежедневно дистиллированной или прокипяченной питьевой водой. Также в помещениях, оборудованных ПЭВМ, должна проводиться ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

Для поддержания необходимого уровня ионизации необходимо применять ионизаторы воздуха (например, ионизатор «Люстра Чижевского»).

8.3.2 Требования к уровням шума и вибрации на рабочих местах, оборудованных ПЭВМ

В производственных помещениях при выполнении основных или вспомогательных работ с использованием ПЭВМ уровни шума на рабочих местах не должны превышать предельно допустимых значений, установленных для данных видов работ в соответствии с действующими санитарно-эпидемиологическими нормативами (*СанПиН 2.2.2/2.4.1340-03* **Гигиенические требования к персональным электронно-вычислительным машинам и организации работы**) (см. Таблица 8.2).

Таблица 8.2. **Допустимые значения уровней шума**.

Уровни звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц									Уровни звука в дБА
31,5	63	125	250	500	1000	2000	4000	8000	
86	71	61	54	49	45	42	40	38	50

Снизить уровень шума в помещениях с ВДТ и ПЭВМ можно использованием подвесных акустических звукопоглощающих панелей с максимальными коэффициентами звукопоглощения в области частот 250-1000 Гц (источниками шума являются операторы ПЭВМ). Дополнительным звукопоглощением служат однотонные занавеси из плотной ткани, гармонирующие с окраской стен и подвешенные в складку на расстоянии 15 - 20 см от ограждения. Ширина занавеси должна быть в 2 раза больше ширины окна.

Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

В помещениях, в которых работа с ПЭВМ является основной, уровень вибрации на рабочих местах не должен превышать допустимых значений для жилых и общественных зданий в соответствии с действующими санитарно-эпидемиологическими нормативами (Санитарные нормы СН 2.2.4/2.1.8.566-96 Производственная вибрация, вибрация в помещениях жилых и общественных зданий) (см. Таблица 8.3).

Таблица 8.3. **Допустимые значения уровней вибрации**.

Среднегеометрические частоты октавных полос, Гц	Допустимые значения оси X, Y			
	по виброускорению		по виброскорости	
	$м/с^2 \times 10^{-3}$	дБ	$м/с \times 10^{-3}$	дБ
2	10	80	0,79	84
4	11	81	0,45	79
8	14	83	0,28	75
16	28	89	0,28	75
31,5	56	95	0,28	75
63	110	101	0,28	75
Корректированные значения и их уровни	10	80	0,28	75

Для снижения вибрации в помещениях оборудование, аппараты и приборы, являющиеся источниками вибрации, необходимо устанавливать на

амортизирующие прокладки. Также могут быть использованы средства индивидуальной защиты.

8.3.3 Требования к освещению на рабочих местах, оборудованных ПЭВМ

Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, и естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 - 500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м^2.

Следует ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м^2 и яркость потолка не должна превышать 200 кд/м^2.

Показатель дискомфорта в помещениях – не более 15.

Яркость светильников общего освещения в зоне углов излучения от 50 до 90 градусов с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м^2, защитный угол светильников должен быть не менее 40 градусов.

Светильники местного освещения должны иметь не просвечивающий отражатель с защитным углом не менее 40 градусов.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1 - 5:1, а между рабочими поверхностями и поверхностями стен и оборудования – 10:1.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). В светильниках местного освещения допускается применение ламп накаливания, в том числе галогенных.

Для освещения помещений с ПЭВМ следует применять светильники с зеркальными параболическими решетками, укомплектованными электронными пускорегулирующими аппаратами (ЭПРА). Допускается использование многоламповых светильников с электромагнитными пускорегулирующими аппаратами (ЭПРА), состоящими из равного числа опережающих и отстающих ветвей.

Применение светильников без рассеивателей и экранирующих решеток не допускается.

При отсутствии светильников с ЭПРА лампы многоламповых светильников или рядом расположенные светильники общего освещения следует включать на разные фазы трехфазной сети.

Общее освещение при использовании люминесцентных светильников следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении видеодисплейных терминалов. При периметральном расположении компьютеров линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору.

Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1,4.

Коэффициент пульсации не должен превышать 5%.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

8.3.4 Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах пользователей, представлены ниже (см. Таблица 8.4).

Таблица 8.4.

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах.

Наименование параметров		ВДУ
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Напряженность электростатического поля		15 кВ/м

Основным источником электромагнитного излучения на рабочем месте оператора компьютера является монитор. Величина излучения монитора во-многом зависит от его модели, среди наиболее безопасных необходимо выделить мониторы с маркировкой “Low Radiation”, компьютеры с жидко-кристаллическим экраном и самые безопасные – мониторы с установленной защитой по методу замкнутого металлического экрана. Рекомендуется использовать подобные видеотерминалы.

В случае использования видеотерминалов с ЭЛТ для уменьшения воздействия электромагнитных полей на человека устанавливаются поглощающие или отражающие экраны перед видеомонитором. Современный качественный защитный экран позволяет уменьшить уровень

электромагнитного излучения и электростатического поля в 10-15 раз и более. Кроме того, существуют специальные очки для защиты глаз от электромагнитного излучения.

8.3.5 Требования к визуальным параметрам ВДТ, контролируемым на рабочих местах

Предельно допустимые значения визуальных параметров ВДТ, контролируемые на рабочих местах, представлены в .Таблица 8.5

Таблица 8.5.

N	Параметры	Допустимые значения
1	Яркость белого поля	Не менее 35 кд/кв.м
2	Неравномерность яркости рабочего поля	Не более +-20%
3	Контрастность (для монохромного режима)	Не менее 3:1
4	Временная нестабильность изображения (мелькания)	Не должна фиксироваться
5	Пространственная нестабильность изображения (дрожание)	Не более 2 x 10(-4L), где L - проектное расстояние наблюдения, мм

Для дисплеев на ЭЛТ частота обновления изображения должна быть не менее 75 Гц при всех режимах разрешения экрана, гарантируемых нормативной документацией на конкретный тип дисплея, и не менее 60 Гц для дисплеев на плоских дискретных экранах (жидкокристаллических, плазменных и т.п.).

Обеспечить выполнение этих требований можно соответствующим образом выбирая компьютерную технику, а также используя различные защитные экраны.

8.3.6 Общие требования к организации рабочих мест пользователей ПЭВМ

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного

видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5 - 2,0 м.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

8.3.7 Требования к организации и оборудованию рабочих мест с ПЭВМ для взрослых пользователей

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1000, 1200 и 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм.

Конструкция рабочего стула должна обеспечивать:

- ширину и глубину поверхности сиденья не менее 400 мм;
- поверхность сиденья с закругленным передним краем;
- регулировку высоты поверхности сиденья в пределах 400 - 550 мм и углам наклона вперед до 15 град, и назад до 5 град.;
- высоту опорной поверхности спинки 300 +-20 мм, ширину - не менее 380 мм и радиус кривизны горизонтальной плоскости - 400 мм;
- угол наклона спинки в вертикальной плоскости в пределах +-30 градусов;
- регулировку расстояния спинки от переднего края сиденья в пределах 260 - 400 мм;
- стационарные или съемные подлокотники длиной не менее 250 мм и шириной - 50 - 70 мм;
- регулировку подлокотников по высоте над сиденьем в пределах 230 +-30 мм и внутреннего расстояния между подлокотниками в пределах 350 - 500 мм.

Рабочее место пользователя ПЭВМ следует оборудовать подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20°. Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

8.3.8Электробезопасность рабочего помещения

Работа с разработанным ПО должна проводиться в офисных помещениях, относящихся, согласно ПУЭ (правилам устройства электроустановок), к классу помещений «без повышенной опасности», т.к. в помещении нет условий повышенной и особой опасности

Питание устройств ПЭВМ осуществляется от однофазной сети переменного тока с частотой 50 Гц, и напряжением 220 В.

В целях обеспечения необходимой электробезопасности при проведении работ в помещениях с ПЭВМ, необходимо выполнять следующие требования:

- для обеспечения работы операторов ПЭВМ необходимо исключить возможность случайного соприкосновения людей с токонесущими частями оборудования. Это достигается путем изоляции токоведущих частей ЭВМ и приборов и размещения их в недоступных зонах;
- не оставлять ЭВМ и другое оборудование под напряжением без наблюдения;
- не подключать разъёмы кабелей ЭВМ при включении напряжения в сети;

- сеть электропитания технических средств должна обеспечивать защитное заземление технических средств, предусмотренное их технической документацией.

8.3.9 Требования пожарной безопасности

Помещение, где функционирует разработанная подсистема, относится к категории В — пожароопасные помещения, в котором имеются твердые сгораемые вещества, способные только гореть, но не взрываться при контакте с кислородом воздуха. Наиболее вероятной причиной пожара является неисправность электрооборудования и электросетей. При эксплуатации ЭВМ возможны возникновения следующих аварийных ситуаций: короткие замыкания, перегрузки, повышение переходных сопротивлений в электрических контактах, перенапряжение, возникновение токов утечки. При возникновении аварийных ситуаций происходит резкое выделение тепловой энергии, которая может явиться причиной возникновения пожара. Требования к пожаробезопасности зданий и сооружений определяются согласно СНиП 21.01-97 [10].

Для снижения вероятности возникновения пожара необходимо проводить различные профилактические мероприятия:

- Организационные – правильная эксплуатация электрооборудования, правильное содержание зданий и помещений;
- Технические — соблюдение противопожарных правил и норм, норм при проектировании зданий, при устройстве отопления, вентиляции освещения, правильное размещение оборудования;
- Мероприятия режимного характера – запрещение курения в неустановленных местах и т.д.;
- Эксплуатационные — своевременные профилактические осмотры и ремонт неисправного электрооборудования.

Для снижения вероятности возникновения и распространения пожара на ранней стадии необходимо:

- установить пожарную сигнализацию с системой оповещения работников, дежурного по объекту и, желательно, автоматическое оповещение противопожарных служб;
- иметь в наличии несколько ручных углекислотных огнетушителей (например, огнетушитель марки ОУ-3);
- помещение должно быть планом эвакуации при пожаре и аптечкой первой помощи. В помещении должен быть ответственный за пожарную безопасность.

8.3.10 Требования к организации медицинского обслуживания пользователей ПЭВМ

Лица, работающие с ПЭВМ более 50% рабочего времени (профессионально связанные с эксплуатацией ПЭВМ), должны проходить обязательные предварительные при поступлении на работу и периодические медицинские осмотры в установленном порядке.

Женщины со времени установления беременности переводятся на работы, не связанные с использованием ПЭВМ, или для них ограничивается время работы с ПЭВМ (не более 3-х часов за рабочую смену) при условии соблюдения гигиенических требований, установленных *СанПиН 2.2.2/2.4.1340-03* **Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.** Трудоустройство беременных женщин следует осуществлять в соответствии с законодательством Российской Федерации.

8.3.11 Требования к проведению государственного санитарно-эпидемиологического надзора и производственного контроля

Государственный санитарно-эпидемиологический надзор за производством и эксплуатацией ПЭВМ осуществляется в соответствии с *СанПиН 2.2.2/2.4.1340-03* **Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.**

Не допускается реализация и эксплуатация на территории Российской Федерации типов ПЭВМ, не имеющих санитарно-эпидемиологического заключения.

Инструментальный контроль за соблюдением требований *СанПиН 2.2.2/2.4.1340-03* осуществляется в соответствии с действующей нормативной документацией.

Производственный контроль за соблюдением санитарных правил осуществляется производителем и поставщиком ПЭВМ, а также предприятиями и организациями, эксплуатирующими ПЭВМ в установленном порядке, в соответствии с действующими санитарными правилами и другими нормативными документами.

8.3.12 Типовой расчет виброизоляции для системы кондиционирования

В ходе обследования помещения, где происходит эксплуатация рассматриваемой системы, было обнаружено оборудование, производящее вибрации – кондиционер с частотой вращения вентилятора 720 оборотов в минуту. Проведем расчет жесткости виброизоляторов и их количества.

Оптимальное соотношение частоты вращения вентилятора в кондиционере и его собственной частоты определяется формулой (согласно ГОСТ 12.4.093-80): $\frac{f}{f_0} = 3$

Частота вращения вентилятора, рассчитывается по формуле: $f = \frac{n}{60} = \frac{720}{60} = 12 Гц$

Таким образом, собственная частота: $f_0 = \frac{f}{3} = \frac{12}{3} = 4 Гц$

С другой стороны, собственная частота рассчитывается по формуле:

$f_0 = \frac{1}{2 \cdot p} \cdot \sqrt{\frac{q_1 \cdot N}{m}}$, где

q_1 - вертикальная жесткость виброизолятора, Н/см

N - число виброизоляторов, штук

m - масса виброизолятора, кг

Выразим и рассчитаем соотношение $\frac{q_1 \cdot N}{m}$:

$$\frac{q_1 \cdot N}{m} = (2 \cdot p \cdot f_0)^2 = (2 \cdot 3.1415 \cdot 4)^2 = 631.66$$

Выбираем виброизолятор ДО-38:

Таблица 8.6. Параметры виброизолятора ДО-38

Обозначение	Нагрузка Р, Н		Вертикальная жесткость, Н/см	Высота в свободном состоянии	Осадка пружины под нагрузкой, мм		Число рабочих витков	Масса, кг
	Рабочая (Рраб.)	Предельная (Рпр.)			Рраб.	Рпр.		
ДО38	122	152	45	72	27	33,7	5,6	0,3

Для 4 опор-виброизоляторов, согласно таблице 7.2, выражение $\frac{q_1 \cdot N}{m} = \frac{45 \cdot 4}{0,3} = 600$, собственная частота $f_0 = \frac{1}{2 \cdot p} \cdot \sqrt{\frac{q_1 \cdot N}{m}} = \frac{1}{2 \cdot 3.14} \cdot \sqrt{600} = 3.89 Гц$

Таким образом, соотношение частоты вращения вентилятора в кондиционере и его собственной частоты будет равно:

$$\frac{f}{f_0} = \frac{12}{3.89} = 3.08$$

Это соотношение оптимально.

Таким образом, для устранения в помещении вибраций, создаваемых кондиционером, необходимо установить его на 4 опоры-виброизоляторы марки ДО-38.

8.4 Утилизация ПЭВМ

Извлечение драгоценных металлов из вторичного сырья является частью проблемы использования возвратных ресурсов, которая включает в себя следующие аспекты: нормативно-правовой, организационный, сертификационный, технологический, экологический, экономико-финансовый. Проблема использования вторичного сырья, содержащего драгоценные материалы из компьютеров, периферийного оборудования и иных средств вычислительной техники (СВТ) актуальна в связи с техническим перевооружением отраслей промышленности.

К драгоценным металлам относятся: золото, серебро, платина, палладий, родий, иридий, рутений, осмий, а также любые химические соединения и сплавы каждого из этих металлов. Статья 2 п. 4 "Федерального закона о драгоценных металлах и драгоценных камнях" от 26 марта 1998 года №1463 гласит: "Лом и отходы драгоценных металлов подлежат сбору во всех организациях, в которых образуются указанные лом и отходы. Собранные лом и отходы подлежат обязательному учёту и могут перерабатываться собирающими их организациями для вторичного использования или реализовываться организациям, имеющим лицензии на данный вид деятельности, для дальнейшего производства и аффинажа драгоценных металлов".

Порядок учёта, хранения, транспортировки, инвентаризации, сбор и сдача отходов драгоценных металлов из СВТ, деталей и узлов, содержащих в своём составе драгоценные металлы для предприятия, учреждения и организации (далее - предприятие), независимо от форм собственности, установлен инструкцией Министерства финансов Российской Федерации от 4 августа 1992 года №67. Все виды работ с драгоценными металлами строго регламентированы нормативно-правовыми документами, перечень которых представлен в Приложении 1.

8.4.1 Разборка изделий

Последовательность разборки определяется типом изделия СВТ, его конструкционными особенностями и комплектацией.

Как правило, процесс разборки должен выполняется в последовательности, обратной процессу сборки изделия. Основные направления деятельности на этапе «Разборка»

8.4.1.1 Разборка персональных компьютеров (ПЭВМ), рабочих станций и серверов

Технологии разборки ПЭВМ, рабочих станций, серверов и информационно-вычислительных систем едины поскольку состав их модулей стандартный. Он содержит системный блок и комплект периферийных устройств.

Разборку ПЭВМ и составных модулей целесообразно осуществлять по технологической схеме представленной на рисунке (см. Рисунок 8.1).

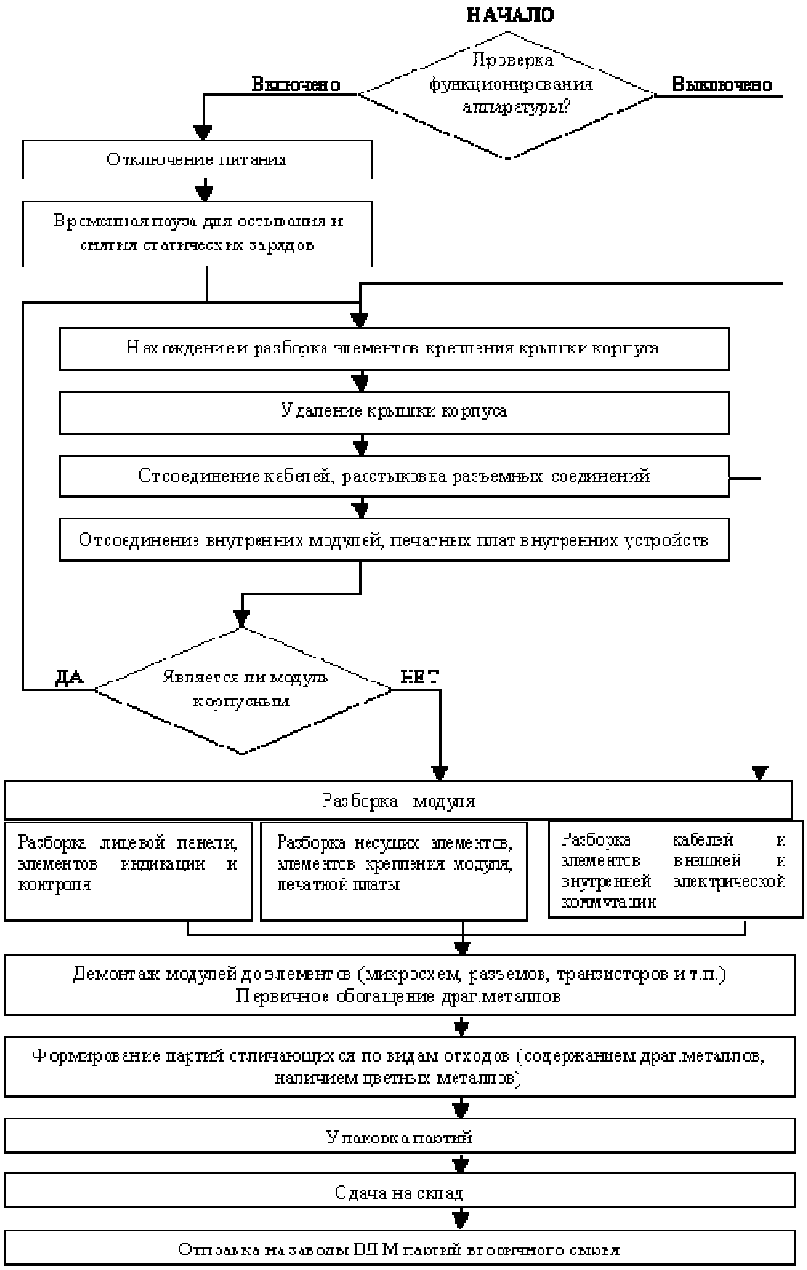

Рис. 7. Технологическая схема разборки ПЭВМ

Рисунок 8.2 **Технологическая схема разборки ПЭВМ**

Порядок разборки системного блока:

Выключить компьютер и отсоединить шнур питания от розетки и системного блока. Отсоединить переходной шнур питания от системного блока к монитору.

Отсоединить от компьютера клавиатуру, монитор, манипулятор "мышь", принтер, сканер и иные внешние устройства.

Найти элементы крепления крышки корпуса (винты, шурупы, пружинные защелки и т.д.). Освободить крышку от элемента крепления.

Снять крышку.

Отсоединить внутренние кабели и плоские шлейфы.

Найти элементы крепления дисководов (НМД, НГМД) в отсеке для дисководов (винты, шурупы, саморезные винты, пружинные защелки и др.). Освободить дисководы и извлечь их из дискового отсека.

Освободить от крепёжных элементов периферийные платы. Извлечь из разъёмов непосредственного контактирования все периферийные платы.

Найти элементы крепления системной платы к корпусу (винты, шурупы). Освободить элементы крепления и извлечь системную плату из корпуса.

Извлечь модули памяти из разъёмов системной платы.

Найти элементы крепления блока питания к корпусу (винты, шурупы, саморезные винты, пружинные защелки и пр.). Освободить элементы крепления и извлечь блок питания.

Разобрать блок питания и извлечь высоковольтные конденсаторы содержащие тантал.

Разобрать ПП и модули памяти до компонентов (микросхем, транзисторов, разъёмов и т.п.).

Произвести сортировку компонентов и сформировать партии электронного лома.

Упаковать партии, составить опись, произвести расчёт (анализ) драгметаллов и передать их на склад.

Провести сортировку цветных и чёрных металлов, пластмасс, сформировать партии и передать их на склад или на переработку.

При оценке содержания драгоценных металлов в партии электронного лома отечественных ПЭВМ необходимо руководствоваться паспортными данными. При оценке ПЭВМ импортного производства необходимо провести ориентировочные расчёты по отечественным аналогам.

8.4.1.2 Обеспечение комплексности технологии разборки

При разборке изделий СВТ образуются материалы и изделия, которые имеют материальную ценность и подлежат реализации.

Примерный перечень материалов представлен ниже см. Таблица 8..

Таблица 8.7. **Перечень материалов, подлежащих утилизации.**

Вид материалов или изделий	Характеристика
Печатные платы, разъемы и соединители, микросхемы	вторичные драгоценные металлы
Электрические провода и кабели, соединители	вторичная медь и её сплавы
Свинец и олово из печатных плат	вторичные припойные пасты (олово и свинец)
Танталовые конденсаторы К-53-1	вторичный танталл
Некоторые корпуса компьютеров, дисковод и т.д.	алюминиевые сплавы
Корпуса стоек, ячеек, шкафов, компьютеров	сталь
Крепежные изделия	болты, гайки, винты
Вентиляторы и электромоторы	по паспорту СВТ
Пластиковая "фракция"	стеклотекстолит, пластмасса разъёмов и соединителей
Экраны компьютеров	стеклофаза, содержащая Pв, Cd, CdS, редкоземельные металлы

Таким образом, можно следать вывод о целесообразности извлечения вторичных чёрных и цветных металлов, пластмасс, стекла, крепёжных изделий, вентиляторов и электромоторов.

8.4.1.3 Извлечение вторичных чёрных металлов

Отечественная практика показывает, что на 1 г извлекаемого золота приходится около 1 кг лома чёрных металлов. В связи с высокой стоимостью транспортно-погрузочных работ рекомендуется производить отгрузку предприятиям-покупателям партий лома чёрных металлов весом не менее 10 тонн. Блоки, панели, съёмные кожухи, рамы, каркасы шкафов и стоек

стационарных ЭВМ, изготовленные из стального нормализованного профиля или листа, подвергаются сортировке, набираются в партии и реализуются.

Предпочтительно заключение договоров при условии, когда предприятие-покупатель своим транспортом вывозит вторичные металлы с территории предприятия-продавца.

Крепёжные изделия, заготовки стального профиля, листов, вентиляторы, электропускатели, кнопки, электрический кабель направляются на реализацию непосредственно в торговую сеть.

Опыт показывает, что денежные средства от реализации этих изделий не превышают 0,6 % от общей суммы.

8.4.1.4 Извлечение вторичных цветных металлов

В процессе разборки изделий СВТ образуется лом (содержащий медь) классификация которого должна проводиться по ГОСТ 1639.

В соответствии с ГОСТ 1639 медные шины целесообразно относить к классу А, группам I и II; латунь - к группам IV-VIII; бронзу - к группам XI-XII; отходы кабеля и проводов ПП следует относить к классу Г, группа XIII.

Все виды ломов необходимо сортировать по классам и группам, формировать в партии и реализовывать.

В процессе разборки изделий СВТ алюминий и его сплавы обычно содержатся в типовых конструкциях изделий. По ГОСТ 1639 их следует относить к классам А3 и Б5.

Все виды отходов необходимо сортировать, формировать в партии и реализовывать.

Свинцово-оловянные припои содержатся в печатных платах и их количество превышает количество золота в десятки раз.

Припои регенерируются при переработке печатных плат.

При разборке СВТ танталовые конденсаторы необходимо складировать отдельно для последующей реализации.

Переработка изделий из пластмасс

Пластмассы следует сортировать по видам.

Переработке подлежат термопласты: поливинилхлорид, полиэтилен, полистирол и т.п.

Стёкла люминесцентных экранов электронно-лучевых трубок следует использовать в производстве керамики и в качестве сырья при производстве новых люминесцентных трубок.

8.4.2 Реализация партий

На рисунке ниже (см. Рисунок 8) представлены основные направления деятельности на этапе "Реализация партий".

Основные действия на этапе "Реализация партий" представляют собой последовательность действий, создающих основу для успешного выполнения процедур завершающего этапа утилизации СВТ.

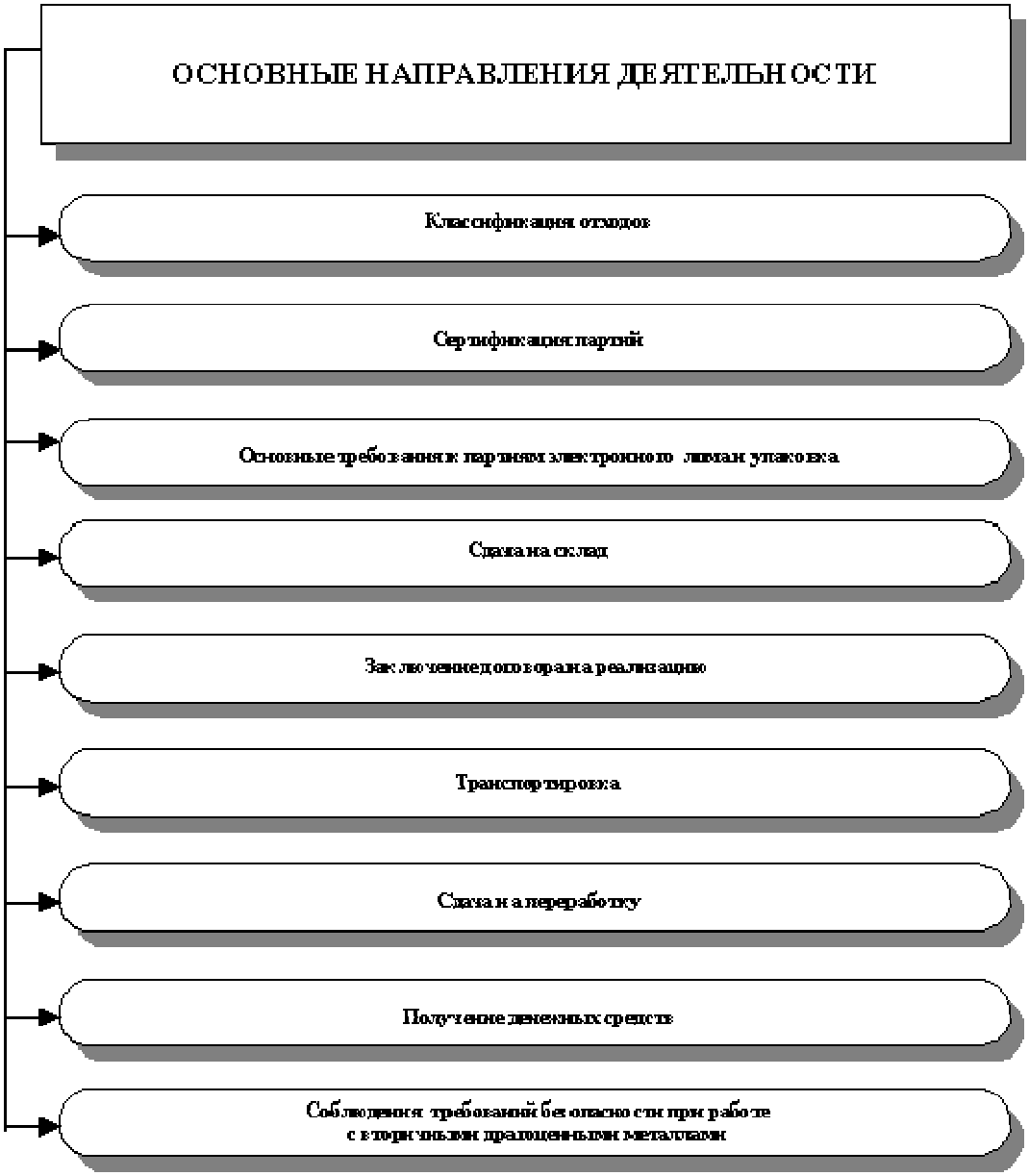

Рис. 8. Основные направления деятельности на этапе «Реализация партий»

Рисунок 8.2 **Основные направления деятельности на этапе «Реализация партий»**

8.4.2.1 Классификация отходов

В настоящее время в России и за рубежом не существует единой классификации вторичного сырья, содержащего драгоценные металлы. Поэтому, возможно разделение вторичного сырья по следующим признакам.

По содержанию драгоценных металлов:

- бедное (менее 1 % золота, 5 % серебра и 1 % металлов платиновой группы);

- богатое (более 1 % золота, 5 % серебра и 1 % металлов платиновой группы).

По составу материала основания:

- на металлической основе;
- на органической (пластиковой) основе;
- на керамической основе;
- на комбинированной основе.

По физическим признакам:

- твёрдые компактные отходы;
- сыпучие (порошки);
- жидкие.

Возможна классификация вторичного сырья в зависимости от сферы производства в:

- ювелирной промышленности;
- химической промышленности;
- электронной, электрохимической, оборонной, радиопромышленности (радиолампы, разъёмы, контакты, контактные устройства, платы на органической основе, микросхемы, радиодетали, кабели и провода, лента, высечка, вырубка, аккумуляторы, элементы питания, прочие отходы);
- бытовых отходах (лом бытовой радиоэлектронной аппаратуры, бытовой стеклянный и фарфоровый бой, лом ювелирных украшений и т.д.).

Отходы классифицируются по элементному составу.

При этом электронный лом отличается особым многообразием состава. Например, современный компьютерный лом содержит несколько десятков видов деталей, содержащих благородные, цветные и чёрные металлы.

8.4.2.2 Классификация сырья вторичных драгоценных металлов

Согласно методике опробования электронного лома и отходов, содержащих драгоценные металлы, разработанной ИАСЦ ГНЦ "ГИРЕДМЕТ", сырьё вторичных драгоценных металлов следует рассортировывать на классы, приведённые в табл.8.8.

Таблица 8.8. **Классификация электронного лома и отходов, содержащих драгоценные металлы (по видам)**

Номер класса	**Вид сырья**
1.	Микросхемы, транзисторы, диоды россыпью
2.	Конденсаторы россыпью
3.	Ножки разъёмов позолоченные и посеребренные
4.	Контакты разделанные
5.	Платы, содержащие элементы классов 1 и 2
6.	Разъёмы с позолоченными и посеребренными ножками
7.	Реле, переключатели, тумблеры
8.	Радиолампы
9.	Платы, содержащие элементы классов 1, 2, 6, 7 и 8
10.	Крупногабаритные детали (волноводы и т.п.)с покрытием из драгметаллами
11.	Элементы питания, аккумуляторы, ампульные батареи
12.	Сыпучие материалы (шихта катализаторов, зола фотоматериалов, шлам фиксажный и т.п.)

8.4.2.3 Сертификация партий

В целях обеспечения строгого учёта, сохранности, сокращения потерь и эффективности использования драгоценных металлов, содержащихся в электронном ломе и отходах, а также для обеспечения единства и требуемой точности измерений при опробовании и проведении анализов химического состава, необходимо руководствоваться нормативными документами утверждёнными Комитетом Российской Федерации по драгоценным металлам и драгоценным камням, Комитетом Российской Федерации по стандартизации, метрологии и сертификации: "Порядком выдачи сертификатов химического состава на партии электронного лома и отходов, содержащих драгоценные металлы", "Временной методикой опробования электронного лома и отходов, содержащих драгоценные металлы".

Указанные документы определяют порядок проведения работ и требования по опробованию и сертификации химического состава партий электронного лома и отходов.

Сертификация химического состава электронного лома и отходов, содержащих драгоценные металлы, включает следующие работы:

- оформление и представление заявки в соответствующий орган по сертификации;
- создание комиссии по апробированию;
- апробирование, оформление документов по результатам апробирования, передача пробы на анализ;
- собственно анализ пробы и оформление количественного химического анализа;
- оформление сертификата.

Выполнение измерений химического состава проб (анализ) электронного лома и отходов, содержащих драгоценные металлы, следует производить методами количественного химического анализа по аттестованным методикам, Приложение 2.

Анализ проб осуществляется аналитическими лабораториями, которые аккредитованы Комитетом Российской Федерации по стандартизации, метрологии и сертификации, а также рекомендованными организациями и органами по сертификации, Приложение 3.

По результатам анализа аккредитованная лаборатория оформляет протокол измерений химического состава пробы электронного лома или отходов по установленной форме.

По результатам процедур опробования и анализа химического состава электронного лома или отходов, орган по сертификации оформляет и выдает заявителю Сертификат химического состава на содержание драгоценных металлов установленной формы.

Сертификат химического состава и комплекс документов по опробованию сертифицируемой партии электронного лома включается в состав сопроводительной документации при передаче партии вторичного сырья от сдатчика заготовителю или переработчику, а также при вывозе за границу для переработки.

При возникновении разногласий, в процессе передачи сертифицированных партий электронного лома и отходов от сдатчика заготовителю или переработчику, производится арбитраж. В этом случае партия не может быть передана переработчику до получения заключения арбитражной лаборатории, аккредитованной Комитетом Российской Федерации по стандартизации, метрологии и сертификации.

8.4.2.4 Основные требования к партиям электронного лома и упаковка

Утверждённые технические требования к партиям электронного лома, в частности его упаковке, до настоящего времени отсутствуют.

В связи с этим рекомендуется придерживаться следующих правил.

Не допускается смешивание различных классов лома.

В ломе и отходах драгоценных металлов не допускаются посторонние предметы, не относящиеся к естественной засорённости.

Лом и отходы драгоценных металлов должны храниться в специально предназначенной для этого таре: в пакетах из полиэтиленовой плёнки по ГОСТ 10354, в фанерных ящиках по ГОСТ 9396, в металлических ящиках с замками собственного изготовления или мешках из мешочной бумаги по ГОСТ 2226.

Ящики внутри должны быть выложены упаковочной бумагой по ГОСТ 515 или каким-либо плёночным материалом. Пакеты и мешки, предназначенные для хранения и отправки лома и отходов, должны изготавливаться с вывернутыми внутрь двумя боковыми швами без нижнего шва.

Мешки или пакеты, уложенные в деревянные или металлические ящики, допускается заклеивать какой-либо клеевой лентой по ГОСТ 18351.

Опечатывание отходов в таре производится после прошивания мешков и пакетов шпагатом по ГОСТ 17308 или заваривания открытых краев полиэтиленовых пакетов с последующим складыванием краёв "гармошкой" и прошиванием её двумя концами шпагата.

Партия лома и отходов должна состоять из одного или нескольких мест в каждом из которых вложена одна или несколько позиций различающихся по составу компонентов, конфигураций, габаритным размерам и другими признаками, не меняющих принципиально сущности последующего опробования на перерабатывающих заводах.

Взвешивание и упаковка отходов производится с участием материально ответственных лиц подразделений предприятия сдатчика.

После контрольного взвешивания каждое место должно быть опломбировано или опечатано сургучной печатью сдатчика.

В сопроводительных документах указывается описание пломбы или печати.

8.4.2.5 Сдача на склад

Передача партии электронного лома из производственного подразделения на склад драгоценных металлов предприятия осуществляется на основе

приёмно-передаточного акта, акта изъятия узлов и изделий техники и других нормативных документов.

8.4.2.6 Заключение договора на реализацию

Между продавцом и покупателем заключается типовой договор, предметом которого является полученная партия электронного лома.

В договоре указываются обязанности сторон, порядок подготовки, отправки лома и отходов драгоценных металлов, порядок приёмки, опробования лома и отходов, порядок расчётов и ответственность сторон.

8.4.2.7 Транспортировка

Транспортирование лома и отходов драгоценных металлов с содержанием золота и металлов платиновой группы более 5 % должно производиться через специальную связь в соответствии с инструкцией Министерства связи Российской Федерации о перевозке ценностей.

Лом и отходы с содержанием золота и металлов платиновой группы менее 5 %, а также отходы серебра отправляются на перерабатывающие заводы почтовыми посылками, багажом по железной дороге или другим видом транспорта с оценочной стоимостью отгружаемого груза.

Сдача на переработку.

Порядок сдачи партии на переработку определяется условиями договора. Типичные условия договора для завода ВДМ следующие.

Вскрытие посылок (мест) производится на заводе приёмной комиссией, которая взвешивает и сверяет фактическое наличие сырья и его качественный состав по каждому виду сырья с данными продавца. По результатам приёмки сырья покупатель высылает продавцу приёмный акт в течение 15 дней от даты поступления сырья.

При доставке сырья транспортом продавца, приёмный акт на количество мест выдаётся на руки уполномоченному представителю продавца в день сдачи сырья.

Представителю продавца необходимо иметь копию описи сдаваемого сырья.

В случае нарушения упаковки или целостности печати материально ответственный работник покупателя в акте на приём отходов отмечает все нарушения.

При расхождении фактически установленных данных при приёмке сырья с данными, значившимися в сопроводительных документах продавца, а также при отсутствии сопроводительных документов, окончательными результатами приёмки является масса брутто, нетто сырья, установленные приёмной комиссией покупателя.

Взвешивание, опробование, пробоотбор и химический анализ проб каждой партии производится в соответствии с нормативно-технической документацией и по технологии, применяемой покупателем.

Однотипные позиции партии подлежат объединению и апробированию по единой технологической схеме.

По результатам опробования сырья на содержание драгоценных металлов, которое производится в течение 60 дней со дня его поступления, покупатель высылает продавцу паспорт с указанием количества драгоценного металла с учетом процента выхода чистого металла в готовую продукцию.

Паспорт подписывается руководителем предприятия-покупателя, главным бухгалтером или их заместителями и скрепляется печатью покупателя.

Порядок получения денежных средств зависит от условия договора. Типичные условия завода ВДМ следующие.

Стоимость поставленного продавцом сырья определяется паспортом покупателя, составленным на основании прейскуранта завода, по мировым ценам на продукцию, получаемую из сырья на день, предшествующий выписке паспорта и пересчитанным в рубли по курсу, установленному Центральным Банком Российской Федерации на день выписки паспорта.

Денежные средства перечисляются на расчётный счет продавца в течение трёх банковских дней со дня получения подтверждающего документа о поступлении денежных средств на расчётный счёт покупателя.

Продавец производит оплату с каждой поставленной партии за опробование.

Все платежи по договору должны производиться в валюте Российской Федерации в безналичной форме.

8.4.2.8 Соблюдение требований безопасности при работе с вторичными драгоценными металлами

Выполнение работ по разборке списанных СВТ предполагает соблюдение общих правил, изложенных в инструкциях по охране труда для слесаря механо-сборочных работ и лиц, работающих с ручным электроинструментом.

Специальные требования техники безопасности при работе с вторичными драгоценными металлами следующие.

Не допускается сбор, заготовка и переработка радиоактивного лома и отходов драгоценных металлов.

Степень воздействия на организм человека вредных веществ, выделяющихся в процессе заготовки и переработки лома и отходов драгоценных металлов, класс опасности и их предельно-допустимая концентрация (ПДК) в воздухе рабочей зоны установлены по ГОСТ 12.1.005 и ГОСТ 12.1.007, табл.8.9.

Таблица 8.9. **Степень воздействия драгоценных металлов на организм человека**

Наименование металла	**Характер действия на организм человека**	**Пути проникновения**	**Класс опасности**	**ПДК вредных веществ в воздухе рабочей зоны, мг/м^3**
Серебро и его	Отходы могут оказывать раздражающее действие на	Органы	11	0,5-1

соединения	слизистую оболочку носа и дыхательных путей.	дыхания		
Золото, платина и его соединения	При длительном контакте могут вызывать аллергические дерматиты и экземы.	Открытые участки кожи	-	-
Рутений		Органы дыхания	11	-
Родий	Оказывает раздражающее действие на слизистую оболочку носа и дыхательных путей. У рабочих, занятых очисткой родия, иногда развивается сверхчувствительность	-	-	-

Контроль за содержанием вредных веществ в воздухе рабочей зоны проводится в соответствии с требованиями ГОСТ 12.1.005 и ГОСТ 12.1.007. Анализ проб воздуха проводится в соответствии с нормативно-технической документацией, утверждённой Минздравом Российской Федерации.

Производственные помещения в местах образования вредных веществ и пыли должны быть оборудованы вентиляцией согласно ГОСТ 12.4.021 с обеспечением санитарно-гигиенических требований к воздуху рабочей зоны.

Для снятия статического электричества пылеприёмники, воздуховоды вентиляционных установок должны иметь заземление, выполненное и обозначенное в соответствии с ГОСТ 12.2.007.0, ГОСТ 12.2.007.14 и ГОСТ 21130.

Для предотвращения попадания пыли, твёрдых веществ на слизистую оболочку глаз необходимо пользоваться защитными очками типа ПО-2, ПО-3 согласно ГОСТ 12.4.013.

При работе с пылящими отходами необходимо пользоваться фильтрующими респираторами РУ-60 и РУ-60му по ГОСТ 17269 и респираторами "Лепесток" по ГОСТ 12.4.028.

При этом респираторы должны периодически подвергаться промывке.

Средства индивидуальной защиты работающих с ломом и отходами драгоценных металлов и сплавов должны соответствовать типовым отраслевым нормам бесплатной выдачи рабочим и служащим металлургических производств. Спецодежда должна соответствовать ГОСТ 29057 и ГОСТ 29058.

Помещения в местах выгрузки и загрузки лома и отходов, оказывающих вредное воздействие на организм человека, должны быть оборудованы местными отсосами согласно ГОСТ 12.4.021.

Производственные помещения должны соответствовать требованиям "Санитарных норм проектирования промышленных предприятий СН 245-71".

Метеорологические условия производственных помещений должны соответствовать санитарным нормам проектирования промышленных предприятий по ГОСТ 12.1.005.

Требования безопасности при погрузочно-разгрузочных работах лома и отходов драгоценных металлов и сплавов должны соответствовать ГОСТ 12.3.009.

Требования по обеспечению взрывобезопасности.

Предприятия и организации, заготавливающие и перерабатывающие лом и отходы драгоценных металлов сплавов, должны проверять весь лом и отходы драгоценных металлов на взрывобезопасность.

Из лома необходимо отобрать и удалить взрывоопасные предметы, материалы, в том числе электронно-вакуумные трубки дисплеев.

Замкнутые сосуды, резервуары и другие полые предметы (баллоны, цилиндры, сосуды, электровакуумные изделия и т.д.) разгерметизируются и освобождаются от содержимого (газов или жидкостей).

Разгерметизацию должны производить рабочие, прошедшие специальное обучение, которые перед началом работы инструктируются в установленном порядке о мерах предосторожности.

ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта были получены следующие результаты:

Проведено предпроектное исследование системы имитационного моделирования РДО, обоснована необходимость разработки кроссплатформенной версии системы.

На этапе концептуального проектирования системы описаны основные внутренние компоненты РДО и с помощью диаграммы компонентов нотации UML показано укрупненное устройство компонентов, которые потребуют внесения изменений в ходе работы.

На этапе технического проектирования разработаны новые диалоги, используемые системой, с помощью диаграммы классов. С помощью диаграммы классов разработана архитектура новой системы. С помощью блок-схем разработаны новые алгоритмы, реализующие новый механизм автозавершения и работы новых диалогов. Разработан конвертор настроек, работа которого показана на диаграмме последовательностей.

На этапе рабочего проектирования написан программный код для спроектированных ранее алгоритмов работы и архитектуры компонентов RAO-studio.exe. Проведены отладка и тестирование системы, в ходе которого исправлялись найденные ошибки. Работоспособность системы была проверена в операционных системах Windows и Linux на написанных ранее моделях.

Таким образом, поставленная цель дипломного проекта достигнута в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. Материалы по системе имитационного моделирования РДО. [В Интернете] http://rdo.rk9.bmstu.ru/forum/viewforum.php?f=4.

2. **Емельянов В.В., Ясиновский С.И.** *Введение в интелектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО.* Москва : Анвик, 1998.

3. Документация по Qt [В Интернете] http://qt-project.org/

4. Документация по Scintilla [В Интернете] http://www.scintilla.org/ScintillaDoc.html

5. Информация о функциональных возможностях wxWidgets [В Интернете] http://www.wxwidgets.org/

6. Информация о функиональных возможностях GTK+ [В Интернете] http://www.gtk.org/

7. *Единая система программной документации. Техническое задание.*

8. **Сажин Ю.Б., Самохин С.В.** *Выполнение организационно -экономической части дипломного проекта по разработке и использованию программного продукта : Методическое пособие.* – М .: Изд-во МГТУ им. Н.Э . Баумана, 2006. – 60 с .: ил.

9. *Сборник типовых расчетов по курсу «Охрана труда» для студентов.*

МВТУ : б.н., 1984 г.

10. СНиП 21.01-97. *Нормы пожарной безопасности «Определение категорий»*

ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ФАЙЛА ОПИСАНИЯ КЛАССА ДИАЛОГА НАСТРОЕК

```
/*!
  \copyright (c) RDO-Team, 2003-2012
  \file      app/rdo_studio/src/view_preferences.h
  \author    Романов Ярослав (robot.xet@gmail.com)
  \date      27.11.2012
  \brief
  \indent    4T
*/

#ifndef _RDO_STUDIO_VIEW_PREFERENCES_H_
#define _RDO_STUDIO_VIEW_PREFERENCES_H_

// ----------------------------------------------------------------
-------- INCLUDES
#include "utils/warning_disable.h"
#include <QDialog>
#include <QColorDialog>
#include "ui_view_preferenses.h"
#include "utils/warning_enable.h"
// ----------------------------------------------------------------
-------- SYNOPSIS
#include "app/rdo_studio/src/editor/model_edit.h"
#include "app/rdo_studio/src/editor/build_edit.h"
#include "app/rdo_studio/src/editor/debug_edit.h"
#include "app/rdo_studio/src/editor/find_edit.h"
#include "app/rdo_studio/src/editor/model_edit_style.h"
#include "app/rdo_studio/src/editor/results_edit_style.h"
#include "app/rdo_studio/src/editor/build_edit_style.h"
#include "app/rdo_studio/src/editor/find_edit_style.h"
#include "app/rdo_studio/src/editor/results_edit.h"
#include
"app/rdo_studio/src/tracer/logger/tracer_logger_main_wnd.h"
#include "app/rdo_studio/src/frame/frame_view.h"
#include "app/rdo_studio/src/frame/frame_options_view.h"
#include "app/rdo_studio/src/tracer/chart/chart_view_style.h"
#include "app/rdo_studio/src/frame/frame_style.h"
#include "app/rdo_studio/src/tracer/chart/chart_view.h"
// ----------------------------------------------------------------
-----------------

class ViewPreferences
     : public QDialog
     , private Ui::ViewPreferencesDialog
{
Q_OBJECT
public:
     explicit ViewPreferences(QWidget* pParent = NULL);
```

```
private slots:
     void onOkButton();
     void onCancelButton();
     void onApplyButton();

     void onCheckInput(const QString& text);

     void onSetup(int state);
     void onCheckInFuture(int state);
     void onOpenLastProject(int state);
     void onShowFullName(int state);

     void onCodeCompUse(int state);
     void onCodeCompShowFullList(bool state);
     void onMarginFold(int state);
     void onMarginBookmark(int state);
     void onMarginLineNumber(int state);

     void onUseTabSymbol(int state);
     void onIndentAsTab(int state);
     void onAutoIndent(int state);
     void onEraseWithTab(bool state);
     void onTabSize(const QString& text);
     void onIndentSize(const QString& text);

     void onTreeWidgetItemActivated(QTreeWidgetItem* item, int
column);
     void onSwitchPreviewComboBox(int index);
     void onFontSize(int index);
     void onFontType(int index);
     void onFontBold(int state);
     void onFontItalic(int state);
     void onFontUnderline(int state);
     void onHorzScroll(int state);
     void onWordWrap(int state);
     void onBookmark(int index);
     void onFold(int index);
     void onComment(int state);
     void onWarning(int state);
     void onHorzIndent(const QString& text);
     void onVertIndent(const QString& text);
     void onFgColor(int index);
     void onBgColor(int index);
     void onFgColorDialog();
     void onBgColorDialog();
     void onFgColorSelected(const QColor& color);
     void onBgColorSelected(const QColor& color);
     void onTitleSize(int index);
     void onLegendSize(int index);
     void onTickWidth(const QString& text);
     void onThemeComboBox(int index);

     void onHelpContext();
```

```
private:
     enum StyleType
     {
          ST_CURRENT = 0,
          ST_DEFAULT,
          ST_CPP,
          ST_PASCAL,
          ST_HTML,
          ST_CLASSIC,
          ST_TWILIGHT,
          ST_OCEAN
     };

     enum ItemType
     {
          IT_ROOT = 0,
          IT_EDITOR,
          IT_BUILD,
          IT_DEBUG,
          IT_LOG,
          IT_RESULT,
          IT_FIND,
          IT_CHART,
          IT_FRAME,
          IT_EDITOR_PLAINTEXT,
          IT_EDITOR_IDENTIFICATOR,
          IT_EDITOR_KEYWORD,
          IT_EDITOR_FUNCTION,
          IT_EDITOR_TRACE,
          IT_EDITOR_COLOR,
          IT_EDITOR_COMMENT,
          IT_EDITOR_NUMBER,
          IT_EDITOR_STRING,
          IT_EDITOR_OPERATOR,
          IT_EDITOR_CARET,
          IT_EDITOR_TEXTSELECTION,
          IT_EDITOR_BOOKMARK,
          IT_EDITOR_FOLD,
          IT_EDITOR_ERROR,
          IT_BUILD_TEXT,
          IT_BUILD_SELECTEDLINE,
          IT_LOG_ES,
          IT_LOG_EB,
          IT_LOG_EF,
          IT_LOG_EI,
          IT_LOG_ER,
          IT_LOG_RC,
          IT_LOG_RE,
          IT_LOG_RK,
          IT_LOG_V,
          IT_LOG_STATUS,
          IT_LOG_DPS,
```

```
            IT_LOG_SB,
            IT_LOG_SO,
            IT_LOG_STN,
            IT_LOG_STD,
            IT_LOG_STR,
            IT_LOG_SRC,
            IT_LOG_SRE,
            IT_LOG_SRK,
            IT_LOG_SD,
            IT_LOG_SES,
            IT_LOG_SEN,
            IT_LOG_SEM,
            IT_LOG_SEF,
            IT_LOG_SEU,
            IT_FIND_SEARCHTEXT,
            IT_CHART_AXIS,
            IT_CHART_TITLE,
            IT_CHART_LEGEND,
            IT_CHART_CHART,
            IT_CHART_TIME,
            IT_FRAME_BORDER,
            IT_FRAME_BACKGROUND
      };

      class StyleItem;

      class StyleProperty
      {
      public:
            StyleItem* item;
            int identificator;

            rdo::gui::style::StyleFont::style& font_style;

            QColor& fg_color;
            QColor& bg_color;
            QColor& fg_disable_color;
            QColor& bg_disable_color;

            StyleProperty(StyleItem* item, int identificator,
rdo::gui::style::StyleFont::style& font_style, QColor& fg_color,
QColor& bg_color, QColor& fg_disable_color = null_fg_color,
QColor& bg_disable_color = null_bg_color)
                  : item(item)
                  , identificator(identificator)
                  , font_style(font_style)
                  , fg_color(fg_color)
                  , bg_color(bg_color)
                  , fg_disable_color(fg_disable_color)
                  , bg_disable_color(bg_disable_color)
            {}
      };
```

```
    typedef std::list<PTR(StyleProperty)> PropertyList;

    class StyleItem
    {
    public:
        ItemType                                    type;
        int&                                        font_size;
        tstring&                                    font_name;
        rbool&                                      wordwrap;
        rbool&                                      horzscrollbar;
        rbool&                                      warning;
        rdo::gui::editor::EditStyle::Bookmark&      bookmarkstyle;
        rdo::gui::editor::ModelStyle::Fold&         foldstyle;
        rbool&                                      commentfold;

        PropertyList properties;

        StyleItem(ItemType type, int& font_size, tstring&
font_name, rbool& wordwrap = null_wordwrap, rbool& horzscrollbar =
null_horzscrollbar, rdo::gui::editor::EditStyle::Bookmark&
bookmarkstyle = null_bookmarkstyle,
rdo::gui::editor::ModelStyle::Fold& foldstyle = null_foldstyle,
rbool& commentfold = null_commentfold, rbool& warning =
null_warning)
            : type(type)
            , font_size(font_size)
            , font_name(font_name)
            , wordwrap(wordwrap)
            , horzscrollbar(horzscrollbar)
            , warning(warning)
            , bookmarkstyle(bookmarkstyle)
            , foldstyle(foldstyle)
            , commentfold(commentfold)
        {}
    };

    typedef std::list<PTR(StyleItem)> StyleItemList;

    StyleItemList style_list;

    int         all_font_size;
    tstring     all_font_name;
    QColor      all_fg_color;
    QColor      all_bg_color;

    rdo::gui::style::StyleFont::style
null_font_style;
    static rbool                                    null_wordwrap;
    static rbool
null_horzscrollbar;
    static rbool                                    null_warning;
    static rbool
null_commentfold;
```

```
    static rdo::gui::editor::EditStyle::Bookmark
null_bookmarkstyle;
    static rdo::gui::editor::ModelStyle::Fold    null_foldstyle;
    static QColor                                null_fg_color;
    static QColor                                null_bg_color;

    rbool m_setup;
    rbool m_checkInFuture;
    rbool m_openLastProject;
    rbool m_showFullName;

    rdo::gui::editor::ModelStyle      style_editor;
    rdo::gui::editor::BuildStyle      style_build;
    rdo::gui::editor::EditStyle       style_debug;
    rdo::gui::editor::ResultsStyle    style_results;
    rdo::gui::editor::FindStyle       style_find;
    rdo::gui::tracer::LogStyle        style_trace;
    rdo::gui::tracer::ChartViewStyle  style_chart;
    rdo::gui::frame::FrameStyle       style_frame;

    PTR(rdo::gui::editor::Model)         preview_editor;
    PTR(rdo::gui::editor::Build)         preview_build;
    PTR(rdo::gui::editor::Debug)         preview_debug;
    PTR(rdo::gui::editor::Results)       preview_results;
    PTR(rdo::gui::editor::Find)          preview_find;
    PTR(rdo::gui::tracer::LogMainWnd)    preview_trace;
    PTR(rdo::gui::tracer::ChartDoc)      preview_chart_doc;
    PTR(rdo::gui::tracer::ChartView)     preview_chart;
    std::vector<rdo::gui::tracer::Time>  preview_times;
    rdo::gui::tracer::LPSerie            preview_serie;
    PTR(rdo::gui::frame::OptionsView)    preview_frame;

    typedef PTR(QTreeWidgetItem) treeItem;

    //Все окна
    treeItem m_pRoot;
    treeItem m_pText;
    treeItem m_pCompile;
    treeItem m_pDebug;
    treeItem m_pTrace;
    treeItem m_pResult;
    treeItem m_pSearch;
    treeItem m_pChart;
    treeItem m_pAnimation;

    //Исходный текст
    treeItem m_pPlainText;
    treeItem m_pVariable;
    treeItem m_pKeyword;
    treeItem m_pFunction;
    treeItem m_pTraceText;
    treeItem m_pColor;
    treeItem m_pComment;
```

```
treeItem m_pNumber;
treeItem m_pString;
treeItem m_pOperator;
treeItem m_pCaret;
treeItem m_pSelection;
treeItem m_pBookmark;
treeItem m_pGroup;
treeItem m_pError;

//Окно компиляции
treeItem m_pTextCompile;
treeItem m_pSelectedString;
treeItem m_pCaretCompile;
treeItem m_pSelectionCompile;
treeItem m_pBookmarkCompile;

//Окно отладки
treeItem m_pTextDebug;
treeItem m_pCaretDebug;
treeItem m_pSelectionDebug;
treeItem m_pBookmarkDebug;

//Окно трассировки
treeItem m_pES;
treeItem m_pEB;
treeItem m_pEF;
treeItem m_pEI;
treeItem m_pER;
treeItem m_pRC;
treeItem m_pRE;
treeItem m_pRK;
treeItem m_pV;
treeItem m_pStatus;
treeItem m_pDPS;
treeItem m_pSB;
treeItem m_pSO;
treeItem m_pSTN;
treeItem m_pSTD;
treeItem m_pSTR;
treeItem m_pSRC;
treeItem m_pSRE;
treeItem m_pSRK;
treeItem m_pSD;
treeItem m_pSES;
treeItem m_pSEN;
treeItem m_pSEM;
treeItem m_pSEF;
treeItem m_pSEU;

//Окно результатов
treeItem m_pPlainTextResult;
treeItem m_pVariableResult;
treeItem m_pKeywordResult;
```

```
    treeItem m_pNumberResult;
    treeItem m_pStringResult;
    treeItem m_pOperatorResult;
    treeItem m_pCaretResult;
    treeItem m_pSelectionResult;
    treeItem m_pBookmarkResult;

    //Окно поиска
    treeItem m_pTextSearch;
    treeItem m_pStringSearch;
    treeItem m_pSelectedStringSearch;
    treeItem m_pCaretSearch;
    treeItem m_pSelectionSearch;
    treeItem m_pBookmarkSearch;

    //Окно графиков
    treeItem m_pAxis;
    treeItem m_pTitle;
    treeItem m_pLegend;
    treeItem m_pGraph;
    treeItem m_pTime;

    //Окно анимации
    treeItem m_pEdgingColor;
    treeItem m_pBackgroundColor;

    PTR(QColorDialog) fgColorDlg;
    PTR(QColorDialog) bgColorDlg;

    void createStyles();
    void createPreview();
    void createTree();
    void insertColors(QComboBox* colorBox);
    void insertColor (const QColor& color, const QString&
colorName, QComboBox* colorBox);
    PTR(StyleProperty) getStyleProperty();
    PTR(StyleItem) getStyleItem();
    PTR(QTreeWidgetItem) createTreeItem (PTR(QTreeWidgetItem)
parent, CREF(QString) name, ItemType itemType);

    void apply();
    void checkAllData();

    void keyPressEvent(QKeyEvent* pEvent);

    void updateDialog();
    void updatePreview();
    void updateStyleTab();
    void updateThemeComboBox(PTR(StyleProperty) prop);
    void updateTheme();
};

#endif // _RDO_STUDIO_VIEW_PREFERENCES_H_
```