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:

Интегрированная среда разработки языка РДО

Студент _____ _____ К.Д. Александровский
(Подпись, дата) (И.О.Фамилия)

Руководитель дипломного проекта _____ _____ А.В. Урусов
(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, 2015

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)

« ___ » _____ 2015 г.

З А Д А Н И Е
на выполнение дипломного проекта

Студент Александровский Кирилл Дмитриевич
(Фамилия, имя, отчество)

Интегрированная среда разработки языка РДО
(Тема дипломного проекта)

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

Тема дипломного проекта утверждена распоряжением по факультету
№ _____ от « ___ » _____ 2015 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 _____ А.В. Урусов
(Подпись, дата) (И.О.Фамилия)

Студент _____ К.Д. Александровский
(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 Александровский Кирилл Дмитриевич
(Фамилия, имя, отчество)

Интегрированная среда разработки языка РДО
(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

Оглавление 5
Реферат 9
Перечень сокращений, символов и специальных терминов с их определением 10
Введение 11
1. Предпроектное исследование 13
1.1. Назначение программного комплекса РДО 13
1.2. Функции программного комплекса 14
1.3. Понятие интегрированной среды разработки 16
1.4. RAO-Studio 17
1.5. Постановка задачи 18
2. Техническое задание 20
2.1. Основания для разработки 20
2.2. Назначение разработки. 20
2.3. Требования к программе или программному изделию 20
2.3.1. Требования к функциональным характеристикам 20
2.3.6. Требования к маркировке и упаковке 22
2.3.7. Требования к транспортированию и хранению 22
2.3.8. Требования к программной документации 22
3. Концептуальный этап проектирования системы 24
3.1. Выбор платформы для разработки системы 24
3.1.1. RAO-Studio 24
3.1.2. Eclipse IDE 25
3.1.3. Библиотека Xtext 25
3.2. Задачи 26
3.2.1. Разработка грамматики языка 27
3.2.2. Библиотека языка РДО 27
3.2.3. Генератор кода 27
3.2.4. Управление работой имитационных моделей 28
4. Технический этап проектирования системы 29
4.1. Разработка грамматики языка РДО 29

4.2. Формирование компонентов Xtext 33
4.3. Структура библиотеки языка РДО 34
5. Рабочий этап проектирования системы 36
5.1. Библиотека языка РДО 36
5.1.1. Алгоритм поиска на графе состояний 36
5.2. Компилятор языка 37
5.3. Графический интерфейс пользователя 39
5.3.1. Консоль вывода 39
5.3.2. Окно отображения результатов 40
5.3.3. Подсистема анимации 40
5.3.4. Иерархическая структура модели 41
5.4. Управление процессом моделирования 42
5.5. Механизм обнаружения ошибок 43
5.6. Вывод 44
6. Исследовательская часть 45
6.1. Модель простейшей СМО 45
6.2. Модель «Пятнашек» 47
6.3. Выводы 48
7. Организационно-экономическая часть 49
7.1. Введение 49
7.2. Организация и планирование процесса разработки ПП 49
7.2.1. Расчет трудоемкости разработки технического задания 52
7.2.2. Расчет трудоемкости выполнения эскизного проекта 52
7.2.3. Расчет трудоемкости выполнения технического проекта 53
7.2.4. Расчет трудоемкости выполнения рабочего проекта 54
7.2.5. Расчет трудоемкости выполнения внедрения 56
7.2.6. Расчет суммарной трудоемкости 56
7.3. Определение стоимости разработки ПП 58
7.3.1. Расчет основной заработной платы 59
7.3.2. Расчет дополнительной заработной платы 59
7.3.3. Отчисления на социальное страхование 60
7.3.4. Накладные расходы 60
7.3.5. Расходы на оборудование (амортизация) 60

7.3.6. Результаты расчетов затрат на разработку программного продукта 61
7.4. Вывод 61
8. Мероприятия по охране труда и технике безопасности 62
8.1. Введение 62
8.2. Опасные и вредные факторы 62
8.2.1. Опасные факторы: 62
8.2.1.1. Физические 62
8.2.2. Вредные факторы: 62
8.2.2.1. Физические 62
8.2.2.2. Психофизиологические 63
8.3. Требования к помещениям для работы с ПЭВМ 63
8.3.1. Повышенное значение в электрической цепи, замыкание которой может произойти через тело человека 64
8.3.6. Повышенный уровень вибрации 69
8.3.9. Типовой расчет виброизоляции для системы кондиционирования 72
8.4. Утилизация ПЭВМ 73
8.4.2. Разборка изделий 74
8.4.2.1. Разборка персональных компьютеров (ПЭВМ), рабочих станций и серверов 74
8.4.2.2. Обеспечение комплексности технологии разборки 77
8.4.2.3. Извлечение вторичных чёрных металлов 78
8.4.2.4. Извлечение вторичных цветных металлов 78
8.4.3. Реализация партий 79
8.4.3.1. Классификация отходов 80
8.4.3.2. Классификация сырья вторичных драгоценных металлов 82
8.4.3.3. Сертификация партий 82
8.4.3.4. Основные требования к партиям электронного лома и упаковка 84
8.4.3.5. Сдача на склад 85
8.4.3.6. Заключение договора на реализацию 86
8.4.3.7. Транспортировка 86

8.4.3.8. Соблюдение требований безопасности при работе с вторичными драгоценными металлами 88
Заключение 92
Список литературы 93
Приложение 1 94
Приложение 2 104
Компилятор типов ресурсов 104
Компилятор точек принятия решений 116

Реферат

Отчет 121 стр., 10 рис., 13 табл., 16 ист., 2 прил.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ, РЕСУРС-ДЕЙСТВИЕ-ОПЕРАЦИЯ, МОДЕЛЬ, СОБЫТИЕ, ТИП РЕСУРСА, РЕСУРС.

Объектом разработки является программный продукт, интегрированная среда разработки языка РДО (ресурс, действие, операция), предназначенная для имитационного моделирования сложных дискретных систем с целью проведения их анализа и синтеза.

Цель работы – создание кроссплатформенной системы имитационного моделирования, соответствующей стандарту языка РДО, и обладающей всеми качествами современных интегрированных сред разработки (удобный редактор кода с подсветкой синтаксиса, инструменты, облегчающие процесс написания кода и т.д.).

При разработке системы проведены исследования ее производительности, то есть времени выполнения имитационных моделей, типичных для языка РДО.

В результате работы с нуля была разработана интегрированная среда разработки языка РДО, основанная на платформе Eclipse IDE. Выбранная платформа отличается гибкостью и обладает всеми необходимыми качествами, присущими современной среде разработки. Благодаря использованию языка Java среда способна работать на любой операционной системе, для которой существует виртуальная машина.

Эффективность созданной системы заключается в ее гибкости, высоком быстродействии, поддержке современных тенденций в разработке ПО (кроссплатформенность, использование Unicode, модульность и т.д.). Данная система может служить базой для дальнейших разработок и в скором времени быть использована в учебном процессе.

Перечень сокращений, символов и специальных терминов с их определением

UML	Universal Modeling Language (Универсальный язык моделирования)
IDE	Integrated Development Environment – Интегрированная среда разработки
ИСР	Интегрированная среда разработки
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
JRE	Java Runtime Environment
JDK	Java Development Kit

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.

Язык РДО 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 РДО – 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 РДО – 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.

РДО может быть применено 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;
- ресурсы;
- события;
- образцы активностей;
- точки принятия решений;
- константы, функции и последовательности;
- кадры анимации;
- собираемые показатели статистики;
- объект прогона;

На данный момент РДО реализует следующие основные функции:

1. Создание модели на языке РДО:

   - создание основных объектов;

   - создание объектов данных и функций;

   - создание объектов вывода.

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. Понятие интегрированной среды разработки

Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами. Это позволяет разработчику сделать меньше действий для переключения различных режимов, в отличие от дискретных программ разработки. Однако, так как ИСР является сложным программным комплексом, то лишь после долгого процесса обучения среда разработки сможет качественно ускорить процесс разработки ПО. Для уменьшения барьера вхождения многие достаточно интерактивны, а для облегчения перехода с одной на другую интерфейс у одного производителя максимально близок, вплоть до использования одной ИСР.

ИСР, обычно, представляет собой единственную программу, в которой проводилась вся разработка. Она, обычно, содержит много функций для создания, изменения, компилирования, развертывания и отладки программного обеспечения. Цель среды разработки заключается в том, чтобы абстрагировать конфигурацию, необходимую, чтобы объединить утилиты командной строки в одном модуле, который позволит уменьшить время, чтобы изучить язык, и повысить производительность разработчика. Также считается, что трудная интеграция задач разработки может далее повысить производительность. Например, ИСР позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках. [4, 6]

1.4. RAO-Studio

Программный комплекс RAO-Studio предназначен для разработки и отладки имитационных моделей на языке РДО. Основные цели данного комплекса - обеспечение пользователя легким в обращении, но достаточно мощным средством разработки текстов моделей на языке РДО, обладающим большинством функций по работе с текстами программ, характерных для сред программирования, а также средствами проведения и обработки результатов имитационных экспериментов.

В соответствии с основной целью программный комплекс решает следующие задачи:

- синтаксический разбор текста модели и настраиваемая подсветка синтаксических конструкций языка РДО;
- открытие и сохранение моделей;
- расширенные возможности для редактирования текстов моделей;
- автоматическое завершение ключевых слов языка;
- поиск и замена фрагментов текста внутри одного модуля модели;
- поиск интересующего фрагмента текста по всей модели;

- навигация по тексту моделей с помощью закладок;
- наличие нескольких буферов обмена для хранения фрагментов текста;
- вставка синтаксических конструкций языка и заготовок (шаблонов) для написания элементов модели;
- настройка отображения текста моделей, в т. ч. скрытие фрагментов текста и масштабирование;
- запуск и остановка процесса моделирования;
- изменение режима моделирования;
- изменение скорости работающей модели;
- переключение между кадрами анимации в процессе моделирования;
- отображение хода работы модели в режиме реального времени;
- построение графиков изменения интересующих разработчика характеристик в режиме реального времени;
- обработка синтаксических ошибок при запуске процесса моделирования;
- обработка ошибок во время выполнения модели. [2]

1.5. Постановка задачи

Таким образом, несмотря на то, что текущая версия среды разработки языка РДО RAO-Studio является ИСР, некоторые её возможности, в полном объёме предоставляемые современными ИСР крупнейших фирм-разработчиков программного обеспечения (такие, как статический анализ кода, автодополнение, навигация по исходному коду с помощью гиперссылок и т.д.), довольно сложны в разработке, а на обеспечение их корректной работы разработчику необходимо потратить большое количество времени.

В связи с этим, более оптимальным решением данной проблемы становится разработка ИСР на основе уже существующего программного

продукта с открытым исходным кодом, продукта, в котором уже реализованы все требуемые функции по ускорению и облегчению процесса разработки.

Описание требуемых от современной ИСР функций представлено на части листа А1 дипломного проекта «Постановка задачи».

2. Техническое задание

Техническое задание разработано в соответствии с ГОСТ 19.201-78 (Единая система программной документации. Техническое задание. Требования к содержанию и оформлению) [12].

2.1. 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 2015 года.
3) Тема дипломного проекта:

   Интегрированная среда разработки языка РДО.

2.2. Назначение разработки.

Основная цель данного дипломного проекта – создание интегрированной среды разработки языка РДО. Разработанная система должна обладать функционалом, присущим современным ИСР.

2.3. Требования к программе или программному изделию

2.3.1. Требования к функциональным характеристикам

В системе должны быть представлены следующие компоненты:

- Редактор кода имитационных моделей с подсветкой синтаксиса, автодополнением и проверкой ошибок;
- Компилятор языка РДО, преобразующий исходный код имитационных моделей в код на одном из языков программирования;
- Логика и алгоритмы языка РДО;

- Элементы пользовательского интерфейса, типичные для среды РДО.

2.3.2. **Требования к надёжности**

Основное требование к надёжности направлено на поддержание в исправном и работоспособном состоянии ЭВМ, на которой происходит использование программного комплекса.

2.3.3. **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.

2.3.4. **Требования к составу и параметрам технических средств**

Программный продукт должен работать на компьютерах со следующими характеристиками:

- объем ОЗУ не менее 1 Гб;
- объем жёсткого диска не менее 20 Гб;
- микропроцессор с тактовой частотой не менее 1ГГц;
- монитор с разрешением от 1024*768 и выше.

2.3.5. **Требования к информационной и программной совместимости**

Данная система должна работать под управлением операционных систем Windows 7, Windows 8, а также MacOS или Linux.

2.3.6. Требования к маркировке и упаковке

Не предъявляются.

2.3.7. Требования к транспортированию и хранению

Не предъявляются.

2.3.8. Требования к программной документации

Не предъявляются.

2.4. 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 операционной системой Linux.
2) Открытие возможностей для дальнейшего развития системы.

2.4.1. Стадии и этапы разработки

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

2.4.2. Порядок контроля и приемки

Контроль и приемка приложения должны состоять из следующих этапов:

1. Запуск интегрированной среды разработки языка РДО в операционной системе Linux и проверка основного функционала системы на тестовых имитационных моделях;
2. Запуск интегрированной среды разработки языка РДО в операционной системе Windows и проверка основного функционала системы на тестовых имитационных моделях;

2.4.3. **Приложения**

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

Используемое при разработке программное обеспечение:

- Операционная система Arch Linux;
- Операционная система Microsoft Windows 8;
- JDK 7 и JDK 8
- Среда разработки Eclipse Luna SR1;
- Библиотека Xtext;
- Система управления версиями Git;

3. Концептуальный этап проектирования системы

3.1. Выбор платформы для разработки системы

Первой и очень важной задачей при разработке ИСР становится выбор подходящей платформы для разработки. Существует множество различных библиотек, реализующих необходимые функции и позволяющих упростить процесс разработки системы. Так же плюсом является тесная интеграция используемых компонентов.

3.1.1. RAO-Studio

Подход, примененный при разработке RAO-Studio, заключается в следующем. ИСР разрабатывается с нуля средствами языка C++ с использованием следующих библиотек:

- GNU bison в качестве компилятора языка
- Qt в качестве библиотеки интерфейса пользователя
- Scintilla для подсветки синтаксиса
- и другие

Данный подход довольно и гибок и позволяет получить все нужные качества от разрабатываемой системы. Однако он также является самым трудозатратным. Это можно наглядно увидеть в разделе «О программе», взглянув на годы разработки.

Очевидно, что для решения данной задачи в рамках одного дипломного проекта требуется более эффективный подход к разработке ИСР.

3.1.2. Eclipse IDE

Программный комплекс Eclipse — это свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation.

Наиболее известные приложения, основанные на Eclipse Platform — различные «Eclipse IDE» для разработки ПО на множестве языков.

Eclipse служит в первую очередь платформой для разработки расширений, чем он и завоевал популярность: любой разработчик может расширить Eclipse своими модулями. Уже существуют Java Development Tools (JDT), C/C++ Development Tools (CDT), разрабатываемые IBM, и другие от различных разработчиков. Множество расширений дополняет среду Eclipse менеджерами для работы с базами данных, серверами приложений и др.

Гибкость Eclipse обеспечивается за счёт подключаемых модулей, благодаря чему возможна разработка не только на Java, но и на других языках. [6]

Основной каркас среды Eclipse, называемый Eclipse Platform, предоставляет все необходимые инструменты для быстрой и качественной разработки.

3.1.3. Библиотека Xtext

Библиотека Xtext служит для создания языков программирования и предметно-ориентированных языков (domain specific languages).

Xtext покрывает все аспекты создания инфраструктуры разрабатываемого языка и предоставляет тесную интеграцию с Eclipse, а именно:

- Подсветка синтаксиса как на основе лексической, так и на основе семантической структуры языка;
- Автодополнение кода, как для ключевых слов, так для контекстно-зависимых конструкций языка;

- Проверка кода на наличие ошибок, производящаяся в фоновом режиме;
- Другие возможности по интеграции с компонентами среды Eclipse.

Все перечисленные выше пункты могут быть настроены под нужды разрабатываемого языка благодаря модульности реализующих их компонентов.

Исходный код на разрабатываемом языке преобразуется в код на языке Java с помощью правил генерации, определяемых разработчиком. Сгенерированный код компилируется и запускается в виртуальной машине Java.

Полученный проект Xtext экспортируется в виде расширения (plug-in) для Eclipse Platform, и работает независимо от модулей Xtext.

Процесс создания и запуска имитационной модели в разрабатываемой среде представлен на диаграмме активности, находящейся на листе А2 курсового проекта Процесс разработки имитационной модели.

3.2. Задачи

Согласно представленной выше информации, библиотека Xtext вкупе с Eclipse может служить отличным каркасом для разработки собственной ИСР для предметно-ориентированного языка – языка РДО.

Таким образом, для выполнения основной цели данного дипломного проекта, то есть разработки интегрированной среды, способной обрабатывать все конструкции языка и запускать имитационные модели, необходимо несколько задач.

3.2.1. Разработка грамматики языка

Самым первым шагом станет разработка грамматики языка. Грамматика должна быть основана на стандарте, изложенном в [1] и [2].

Повторное использование кода, написанного для генератора анализаторов GNU bison, невозможно в силу причин, которые будут описаны далее (разные подходы (алгоритмы) библиотек к синтаксическому анализу).

Таким образом, будет необходимо описать с нуля как все конструкции языка РДО (типы ресурсов, образцы точек принятия решений, собираемые показатели и т.д.), но и язык выражений, а также процедурный язык программирования. Данный функционал составляет неотъемлемую часть современной среды языка РДО, и реализован в полном объеме в RAO-Studio.

3.2.2. Библиотека языка РДО

Далее следует создать общую библиотеку языка РДО, которая будет содержать модули, необходимые для работы имитационных моделей – **rdo.lib**.

Так как изменяемой частью для каждой модели будет выступать генерируемый код на языке Java, любые классы, общие для всех имитационных моделей и составляющие их основу, должны быть вынесены в отдельную библиотеку для обеспечения простоты и чистоты разрабатываемого генератора кода.

3.2.3. Генератор кода

Каждая имитационная модель в разрабатываемой системе будет являться набором Java-классов, то есть скомпилированных файлов языка Java. Эта задача ложится на плечи генератора кода, который долен преобразовывать конструкции языка РДО, преобразованные в

специальное дерево объектов синтаксическим анализатором, построенным на основе грамматики языка, описанной ранее.

Этот генерируемый код должен использовать возможности разработанной библиотеки языка (**rdo.lib**).

3.2.4. Управление работой имитационных моделей

Сгенерированный код имитационных моделей необходимо запустить для осуществления самого процесса имитационного моделирования. Данный модуль должен управлять запуском, остановкой и работой имитационных моделей, полученных после кодогенерации на основе исходных кодов моделей.

3.2.5. Интерфейс пользователя

В RAO-Studio пользователю предлагается работа с широким набором средств, необходимых для взаимодействия с имитационными моделями. Среди них есть средства управления работой имитационных моделей, консоль вывода, окно отображения результатов моделирования, подсистема анимации и другие.

В графическом интерфейсе разрабатываемой системы необходимо реализовать аналоги этих компонентов, так как они являются устоявшимися и привычными для текущих пользователей

3.2.6. Дальнейшая разработка

Необходимо разрабатывать систему с учетом возможности дальнейшего наращивания ее функциональных возможностей.

4. Технический этап проектирования системы

4.1. Разработка грамматики языка РДО

В программировании для преобразования какого-либо языка в набор команд используются синтаксические и лексические анализаторы, т.е. программы, позволяющие превратить какой либо язык в удобный для машинной обработки вид с помощью формального описания этого языка – грамматики. [5]

В RAO-Studio для этой цели используется GNU bison – LR-анализатор, который выполняет разбор входного потока данных снизу вверх, преобразуя входные данные с код на языке C++.

LR-анализатор (англ. LR parser) — синтаксический анализатор для исходных кодов программ, написанных на некотором языке программирования, который читает входной поток слева (Left) направо и производит наиболее правую (Right) продукцию контекстно-свободной грамматики.

В библиотеке Xtext используется другой подход – в ней имеется собственный синтаксический анализатор, основанный на анализаторе ANTLR. Алгоритм этого анализатора, LL(1) – это нисходящий алгоритм синтаксического разбора с просмотром на один символ вперед. Этот алгоритм отличается очень высокой скоростью работы и позволяет писать грамматику языка в наиболее удобной понимания человеком форме. Однако, он не поддерживает левостороннюю рекурсию, характерную для грамматик, имеющих дело с языком выражений. В Xtext данное ограничение обходится путем процедуры, называемой «перезапись дерева». Для использования этого функционала требуется небольшое количество дополнительных инструкций в грамматике языка выражений.

Грамматика пишется разработчиком в нотации, похожей на нотацию EBNF (расширенная форма Бэкуса-Наура). РБНФ – формальная система определения синтаксиса, в которой одни синтаксические категории последовательно определяются через другие. Используется для описания контекстно-свободных формальных грамматик.

Как и в БНФ, описание грамматики в РБНФ представляет собой набор правил, определяющих отношения между терминальными символами (терминалами) и нетерминальными символами (нетерминалами).

- Терминальные символы — это минимальные элементы грамматики, не имеющие собственной грамматической структуры. В РБНФ терминальные символы — это либо предопределённые идентификаторы (имена, считающиеся заданными для данного описания грамматики), либо цепочки — последовательности символов в кавычках или апострофах.
- Нетерминальные символы — это элементы грамматики, имеющие собственные имена и структуру. Каждый нетерминальный символ состоит из одного или более терминальных и/или нетерминальных символов, сочетание которых определяется правилами грамматики. В РБНФ каждый нетерминальный символ имеет имя, которое представляет собой строку символов [5].

Однако в Xtext помимо самого описания символов возможно присваивание им внутренних идентификаторов, таким образом, конструкция

```
ResultDeclaration:
      name = ID ':' type = ResultType;
```

означает, что для описания собираемого показателя необходим идентификатор (ID), который будет записан в переменную name, символ двоеточия, и составной символ ResultType, который, в свою очередь, будет записан в переменную type.

```
PatternChoiceMethod
   : first ?= 'first'
   | withtype = PatternChoiceWithType
                    expression = ExpressionOr
;
```

В этом примере метод выбора релевантных ресурсов может состоять либо из ключевого слова first, либо из типа выбора релевантных ресурсов (PatternChoiceWithType) и выражения (ExpressionOr). Если написано ключевое слово first, в одноименную переменную будет записано true, а в переменные из альтернативной ветки будет записано нулевое значение. В противном случае надо будет описать конструкции из правой части выражения согласно их правилам.

Также имеется возможность присваивать переменным массивы элементов:

```
    ...
'$Relevant_resources'
    relevantresources += EventRelevantResource+
'$Body'
    ...
```

здесь в переменную relevantresources будут записаны 1 и более объектов EventRelevantResource.

После разбора исходного файла (модели) вместо исходного кода на каком-либо языке программирования анализатор Xtext на основе описанных выше правил создает синтаксическое дерево (с ним можно работать в дальнейшем в модуле, именуемом генератором кода).

Вершины этого дерева – структуры, полями которых являются элементарные типы данных, ссылки на другие вершины, и даже массивы таких ссылок в случаях, когда требуется описание неопределенного количества однотипных элементов.

Таким образом, на основании описания языка РДО, представленного в документации [1], была составлена его грамматика в нотации Xtext. Синтаксические диаграммы, соответствующие грамматике, представлены на трех листах А1 курсового проекта «Синтаксическая диаграмма языка РДО».

Помимо этого, был написан язык выражений (expressions) и команд (statement), то есть объектов, присущих всем процедурным языкам программирования. Процедурный язык также является неотъемлемой частью РДО на текущем этапе его развития.

Также для поддержки Unicode в именах идентификаторах была переопределена часть стандартной грамматики, отвечающая за терминальный символ идентификатора.

Пример нетерминального символа, отвечающего за точку принятия решений типа “some”:

```
DecisionPointSome:
    '$Decision_point' name = ID ':' 'some'
   ('$Parent' parent = [DecisionPoint|FQN])?
   ('$Condition' (condition = ExpressionOr | 'NoCheck'))?
    '$Activities'
         activities += DecisionPointActivity+
    '$End'
;
```

Язык выражений и команд, не вошедший в листы курсового проекта, представлен в Приложении А.

4.2. Формирование компонентов Xtext

После написания грамматики Xtext генерирует начальные версии всех своих компонентов на ее основе. В их число входят следующие пакеты:

- `ru.bmstu.rk9.rdo.formatting`

  Отвечает за автоматическое выравнивание кода при редактировании, а также позволяет определить свои правила форматирования.

- `ru.bmstu.rk9.rdo.scoping`

  Отвечает за область видимости для терминальных символов, являющихся ссылками на описанные в модели нетерминальные символы.

- `ru.bmstu.rk9.rdo.validation`

  Предоставляет механизм проверки кода модели на ошибки согласно логике, описанной в файле `RDOValidator.xtend`

- `ru.bmstu.rk9.rdo.generator`

  Содержит код, в результате выполнения которого в папке проекта создаются сгенерированные файлы с исходным кодом на языке Java. Программный код для такого преобразования пишется разработчиком, и подразумевает использование синтаксического дерева, получаемого на этапе синтаксического разбора.

- `ru.bmstu.rk9.rdo.ui.contentassist`

  В данном пакете находятся правила, изменяющие поведение стандартного меню автодополнения для каких либо объектов грамматики языка.

- `ru.bmstu.rk9.rdo.ui.labeling`

  Отвечает за текстовое представление имен объектов грамматики.

- `ru.bmstu.rk9.rdo.ui.outline`

  Содержит правила, позволяющие изменить стандартный механизм построения дерева иерархической структуры.

- `ru.bmstu.rk9.rdo.ui.quickfix`

  Правила, изменяющие и расширяющие меню подсказок при возникновении каких-либо ошибок в коде модели.

Каждый из этих модулей предназначен для расширения и улучшения базового набора возможностей, предоставляемых библиотекой Xtext на основе грамматики.

Также существует большое количество модулей, не создаваемых при генерации проектов из грамматики, расширением которых можно добиться изменения поведения Xtext практически в любом аспекте его работы.

4.3. Структура библиотеки языка РДО

Библиотека языка РДО, используемая во всех имитационных моделях, код которых будет генерироваться, должна включать в себя следующие элементы:

- Симулятор

  Задача симулятора – непосредственно процесс моделирования, обработка событий и точек принятия решений. Так же к симулятору можно отнести различные классы, соответствующие объектам языка РДО – события (класс Event), точки принятия решений (DecisionPoint DecisionPointPrior и DecisionPointSearch) и так далее

- База данных

  База данных (класс Database) занимается учетом всех изменений, произошедших в системе, таких как произошедшие события, выполненные активности, изменившиеся параметры ресурсов. Различные вспомогательные классы базы данных также должны быть разработаны.

Также в библиотеке языка РДО представлены интерфейсы, стандартизующие разработку изменяемой части Java-кода имитационной модели.

Диаграммы классов языка UML, отображающие проектные решения по симулятору языка, представлены на листах «Диаграмма классов библиотеки симулятора РДО» (лист 1 и 2).

5. Рабочий этап проектирования системы

5.1. Библиотека языка РДО

Так как главным элементом библиотеки симулятора является собственно класс Simulator, необходимо реализовать алгоритм его работы. Данный алгоритм должен работать согласно стандарту на язык РДО, и реализовывать событийный подход и характерный для языка РДО подход сканирования активностей.

Укрупненный алгоритм симулятора представлен на листе А2 «Алгоритм работы симулятора» дипломного проекта.

Алгоритм проверки точек принятия решений, реализующий подход сканирования активностей, представлен на листе А2 дипломного проекта «Алгоритм проверки точек принятия решений».

5.1.1. Алгоритм поиска на графе состояний

РДО также реализует алгоритм поиска решения на графе состояний, именуемый также алгоритмом A*.

Поиск A* (произносится «А звезда» или «А стар», от англ. A star) — в информатике и математике, алгоритм поиска по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной).

Порядок обхода вершин определяется эвристической функцией «расстояние + стоимость» (обычно обозначаемой как f(x)). Эта функция — сумма двух других: функции стоимости достижения рассматриваемой вершины (x) из начальной (обычно обозначается как g(x) и может быть как эвристической, так и нет) и эвристической оценкой расстояния от рассматриваемой вершины к конечной (обозначается как h(x)).

Функция h(x) должна быть допустимой эвристической оценкой, то есть не должна переоценивать расстояния к целевой вершине. Например, для задачи маршрутизации h(x) может представлять собой расстояние до цели по прямой линии, так как это физически наименьшее возможное расстояние между двумя точками.

Этот алгоритм был впервые описан в 1968 году Питером Хартом, Нильсом Нильсоном и Бертрамом Рафаэлем. Это по сути было расширение алгоритма Дейкстры, созданного в 1959 году. Новый алгоритм достигал более высокой производительности (по времени) с помощью эвристики. В их работе он упоминается как «алгоритм A». Но так как он вычисляет лучший маршрут для заданной эвристики, он был назван A* [11].

Данный алгоритм в полной мере реализован в симуляторе языка РДО. Вершинами графа в данном случае являются состояния всей системы имитационной модели. Для быстрой работы данного алгоритма в той части библиотеке симулятора, которая отвечает за работу с ресурсами имитационной модели, был реализован механизм Copy-on-Write, позволяющий копировать в памяти лишь те данные, которые изменяются в процессе поиска решения.

Алгоритм самого поиска A* описан на листе А2 дипломного проекта «Алгоритм поиска на графе состояний».

5.2. Компилятор языка

Правила генерации Java-кода из объектов синтаксического дерева необходимо описать в файлах модуля кодогенератора. Точкой входа в этом случае является файл `RDOGenerator.xtend`, в котором описан одноименный класс. В этом файле (и тех, на которые он ссылается) находится наибольший объем логики и алгоритмов разрабатываемой системы.

Кодогенерация представляет собой процесс записи в файловую систему текстовых файлов, созданных на основе объектов синтаксического дерева, построенного на основе грамматики языка и исходного кода имитационных моделей. В процессе кодогенерации для каждого объекта языка РДО в папке проекта создается свой Java-класс, описывающий его поведение и так или иначе использующий функции из библиотеки **rdo.lib**. Далее полученные классы компилируются Java-модулем среды Eclipse, после чего все готово для запуска имитационной модели.

Для описания правил генерации фирмой Itemis (разработчик Xtext) был написан специальный Java-совместимый язык Xtend, в котором сильно упрощен процесс формирования строк кода, путем встраивания в них языка шаблонов, аналогичного стандарту JSP и аналогичным технологиям из области web-программирования.

Пример процедуры, создающей класс константы языка:

Вызов:

```
fsa.generateFile(filename+"/"+e.name+".java", e.compileConstant(filename))
```

Тело функции:

```
def compileConstant(ConstantDeclaration con, String filename)
{
	'''
	package «filename»;

	public class «con.name»
	{
		public static final «con.type.compileType» value = «
			IF con.type.compileType.endsWith("_enum")»«
			con.value.compileExpressionContext(
			(new LocalContext).populateWithEnums(
				con.type.resolveAllSuchAs as RDOEnum)).value»«
			ELSE»«con.value.compileExpression.value»«ENDIF»;
		«IF con.type instanceof RDOEnum»

			public enum «(con.type as RDOEnum)
				.getEnumParentName(false)»_enum
```

```
            {
                «(con.type as RDOEnum).makeEnumBody»
            }«
        ENDIF»
    }
    '''
}
```

Текст с серым фоном – статическая часть, которая будет превращена в код в неизменном виде. Текст в кавычках («...») – код, который будет обработан и превратится в строки на этапе генерации модели.

Общий объем кода, написанного для модуля генератора языка, составляет несколько тысяч строк, но, к сожалению, трудно формализуется и не может быть превращен ни в одну из существующих нотаций без серьезных смысловых потерь.

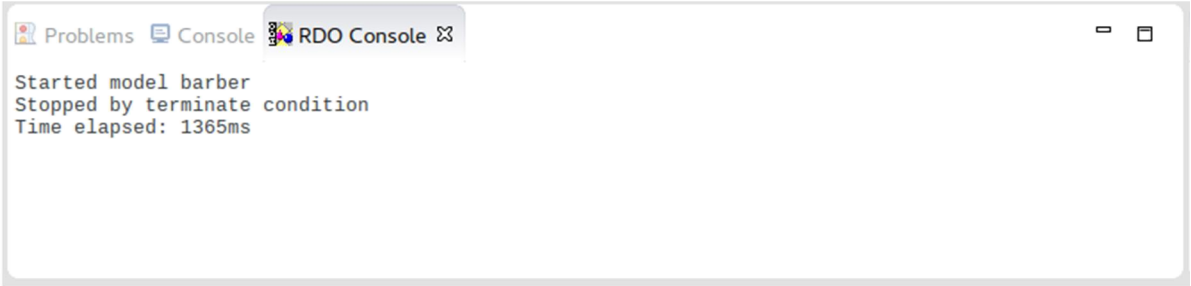
Таким образом, большая часть работы не может быть графически отображена на листах дипломного проекта. В связи с этим некоторые конструкции из модуля генератора приведены в Приложении Б.

5.3. Графический интерфейс пользователя

Графический интерфейс пользователя во многом повторяет таковой в RAO-Studio. Для этого были написаны следующие компоненты.

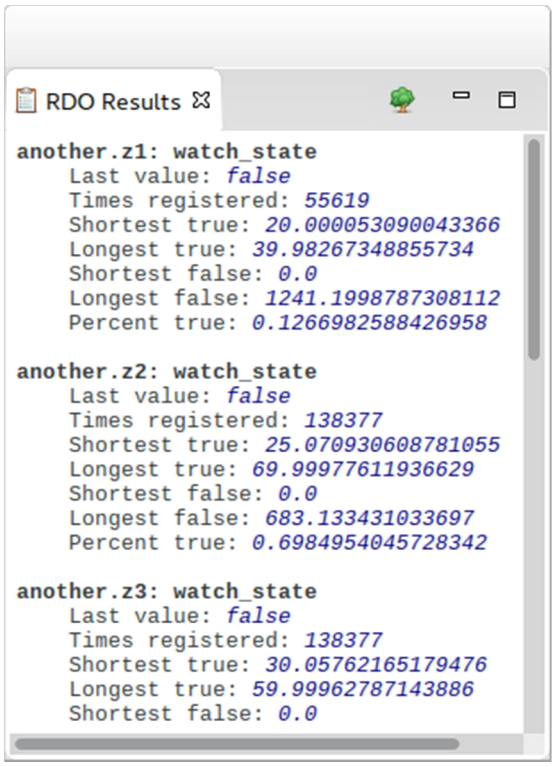
5.3.1. Консоль вывода

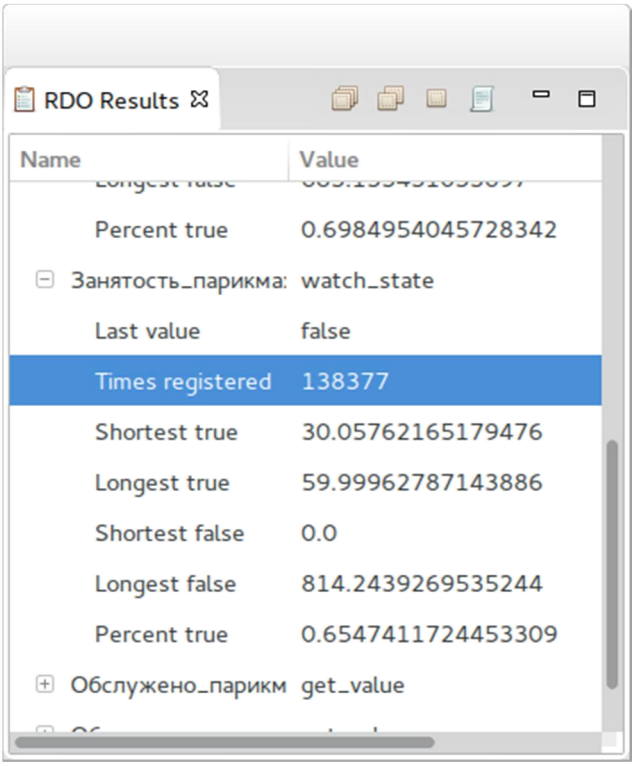
Консоль вывода – самый простой из компонентов. Она позволяет выводить пользователю произвольную информацию во время выполнения имитационной модели.



5.3.2. Окно отображения результатов

Помимо сбора статистики на протяжении работы имитационной модели, пользователю также требуется видеть результаты ее работы. В RAO-Studio такое окно было представлено в виде, аналогичном консоли вывода. В разработанной системе было решено расширить данный компонент, предоставив пользователю выбор между классическим отображением в виде текста (при этом было добавлено форматирование текста для лучшей его читаемости) и между отображением в виде дерева.





5.3.3. Подсистема анимации

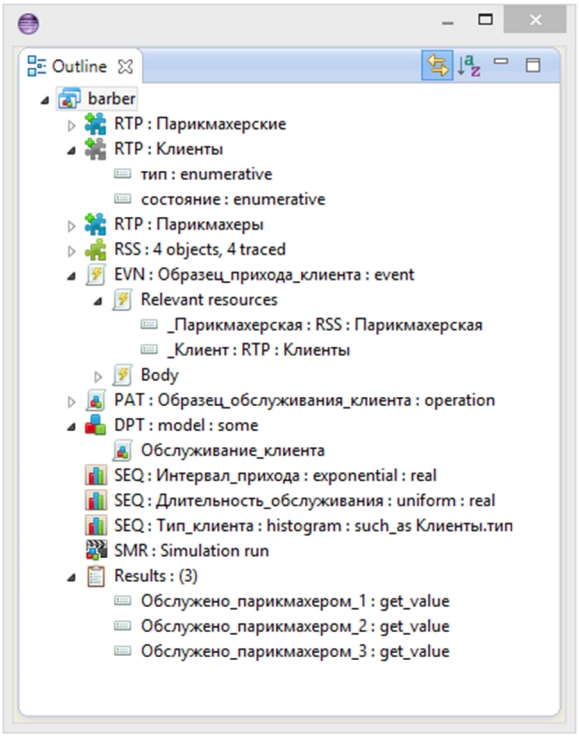
Подсистема анимации позволяет пользователю внести интерактивность в имитационную модель, наглядно визуализировать происходящие в ней процессы. Кадры анимации являются частью языка РДО, поэтому должны в полной мере поддерживаться в разрабатываемой системе.

Для этого была создана система анимации, состоящая из двух частей – интерфейса анимации (`AnimationContext`), и его реализаций.

Интерфейс описывает методы, не зависящие от конкретной реализации и используемой графической библиотеки, например drawCircle или drawText. Реализация же для каждого из описанных методов предоставляет код, взаимодействующий с конкретной графической библиотекой. В данном случае была написана реализация для библиотеки SWT, которая лежит в основе всех графических компонентов среды Eclipse. Также в будущем возможно написать реализацию для других библиотек, например, JavaFX, и позволить пользователю переключаться между ними.

5.3.4. Иерархическая структура модели

Окно структуры модели является особенностью среды Eclipse, и не имеет аналога в RAO-Studio. Внешний вид его определяется грамматикой языка, а также изменениями, описываемыми в файлах `RDOOutlineTreeProvider.xtend` и `RDOLabelProvider.xtend` пакетов `ru.bmstu.rk9.rdo.outline` и `ru.bmstu.rk9.rdo.labeling` соответственно. Данные изменения позволяют запретить элементам иерархии порождать дочерние вершины, а также определяют имена и значки для каждого типа элемента. На рисунке представлен внешний вид иерархии для модели парикмахерской:



Пример кода из файлов, отвечающих за расширение и изменения, вносимые в данный компонент:

RDOLabelProvider:

```
def image(ConstantDeclaration c) { "constant2.gif" }
def  text(ConstantDeclaration c) {c.name + c.type.typeGenericLabel}

// Sequence
def  text(Sequence seq) { "SEQ : " + seq.name + " : " + (
     if(seq.type instanceof EnumerativeSequence) "enumerative"
     else "" + if(seq.type instanceof RegularSequence)
          (seq.type as RegularSequence).type.literal else "" +
     if(seq.type instanceof HistogramSequence) "histogram"
          else "" ) + seq.returntype.typeGenericLabel }
def image(Sequence seq) { "chart.gif" }
```

RDOOutlineTreeProvider:

```
// Functions
def _createChildren(IOutlineNode parentNode, Function fun)
{
     for(e : fun.eAllContents.toIterable.filter
          (typeof(FunctionParameter)))
     {
          createNode(parentNode, e)
     }
}
def _isLeaf(FunctionParameter p) { true }
def image(ConstantDeclaration
```

5.4. Управление процессом моделирования

После генерации кода из имитационной модели его необходимо запустить. Для этого разработаны классы в пакете `ru.bmstu.rk9.rdo.ui.runtime`. Код в этих классах загружает скомпилированные файлы имитационной модели в текущую виртуальную машину, в которой работает Eclipse, а также обеспечивает взаимодействие пользователя и различных модулей с запущенной имитационной моделью.

Синхронизация процесса моделирования также важна для среды, реализующей возможности языка РДО. Для этого существует класс `SimulationSynchronizer`, обеспечивающий скорость работы, соответствующую заданному пользователем масштабу моделирования. Внешний вид элементов управления, предоставляющих данный функционал, также соответствует таковому в RAO-Studio.

Здесь представлены кнопки старта и остановки моделирования, переключатели режимов моделирования, кнопки изменения масштаба моделирования и ползунок скорости выполнения.

5.5. Механизм обнаружения ошибок

В файле `RDOValidator.xtend` пакета `ru.bmstu.rk9.rdo.validaton` описываются правила анализа исходного кода модели, которые могут выводить те или иные сообщения об ошибках, а также предупреждения, связанные с исходным кодом имитационной модели. Пример кода, выявляющий совпадения в именах релевантных ресурсов и именах параметров образца:

```
@Check
def checkNamesInPatterns (Pattern pat)
{
	val List<EObject> paramlist  = pat.eAllContents.filter[e |
		e instanceof PatternParameter
	].toList
	val List<EObject> relreslist = pat.eAllContents.filter[e |
		e instanceof OperationRelevantResource ||
		e instanceof RuleRelevantResource      ||
		e instanceof EventRelevantResource
	].toList

	for (e : paramlist)
		if (relreslist.map[r | r.nameGeneric].contains(e.nameGeneric))
			error("Error - parameter name shouldn't match
				relevant resource name '" + e.nameGeneric + "'.",
					e, e.getNameStructuralFeature)

	for (e : relreslist)
		if (paramlist.map[r | r.nameGeneric].contains(e.nameGeneric))
```

```
                error("Error - relevant resource name shouldn't
                      match parameter name '" + e.nameGeneric + "'.", e,
                            e.getNameStructuralFeature)
}
```

5.6. Вывод

В результате работы было сделано большинство компонентов, необходимых для комфортной и продуктивной работы как с редактором имитационных моделей, так и с самими моделями в процессе их выполнения.

Диаграмма пакетов для разработанных компонентов представлена на листе А2 дипломного проекта «Диаграмма пакетов разработанных компонентов».

Подробное изображение каждого компонента графического интерфейса и его роли в системе представлено на листе А1 дипломного проекта «Внешний вид компонентов разработанной системы».

6. Исследовательская часть

В рамках исследовательской части был произведен анализ скорости работы созданной системы. Анализ производился посредством сравнения времени, затраченного на моделирование тестовых имитационных моделей, в разработанной системе и в RAO-Studio.

Тестовые модели были следующими:

- Простейшая модель многоканальной СМО (Парикмахерская)
- Модель игры «Пятнашки»

Такие модели были выбраны в силу того, что скорость работы системы – субъективное понятие, и результат во многом зависит от параметров ПК, на котором ведется сравнение, а также от реализации среды исполнения для каждой конкретной операционной системы. Эти модели задействуют разные части систем (с алгоритмической точки зрения) – для простейшей СМО в большей степени используется событийный подход, а для модели «Пятнашек» – алгоритм поиска на графе состояний.

Замеры производились с помощью таймеров, встроенных в каждую среду (rdo-xtext и RAO-Studio), которые используют одни и те же ресурсы (системные часы операционной системы).

Для каждого значения варьируемого параметра производилось по 20 замеров с последующим взятием среднего, чтобы исключить влияние случайных факторов операционной системы (захват ресурсов ПК другими, фоновыми программами).

Полученные результаты представлены в миллисекундах.

6.1. Модель простейшей СМО

Система массового обслуживания (СМО) — система, которая производит обслуживание поступающих в неё требований.

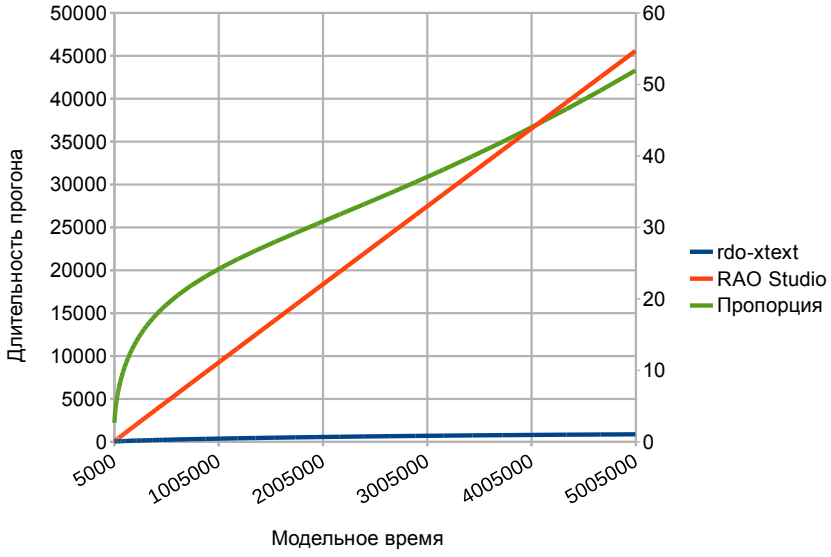
Обслуживание требований в СМО производится обслуживающими приборами. Классическая СМО содержит от одного до бесконечного числа приборов.

Данный пример является системой с ожиданием, в которой имеется накопитель бесконечной ёмкости для буферизации поступивших требований, при этом ожидающие требования образуют очередь;

В качестве варьируемого параметра выступает модельное время, по достижении которого модель будет остановлена. В качестве тестовых времен были выбраны следующие числа (единицы измерения – единицы модельного времени): 5000, 50000, 500000, 5000000. Так как на данной модели системы ведут себя стабильно, было решено остановиться именно на таком количестве параметров, так как полученный результат впоследствии хорошо интерполируется и экстраполируется.

Таблица и график с полученными результатами:

Мод. время	rdo-xtext	RAO-Studio	Пропорция
5000	21,1	56,4	2,67298578
50000	63,2	485	7,67405063
500000	242,1	4633,8	19,1400248
5000000	877,1	45561,5	51,9456162



6.2. Модель «Пятнашек»

Игра в 15 или Пятна́шки – популярная головоломка, придуманная в 1878 году Ноем Чепмэном. Представляет собой набор одинаковых квадратных костяшек с нанесёнными числами, заключённых в квадратную коробку. Длина стороны коробки в четыре раза больше длины стороны костяшек для набора из 15 элементов (и в три раза больше для набора в 8 элементов), соответственно в коробке остаётся незаполненным одно квадратное поле. Цель игры — перемещая костяшки по коробке, добиться упорядочивания их по номерам, желательно сделав как можно меньше перемещений.

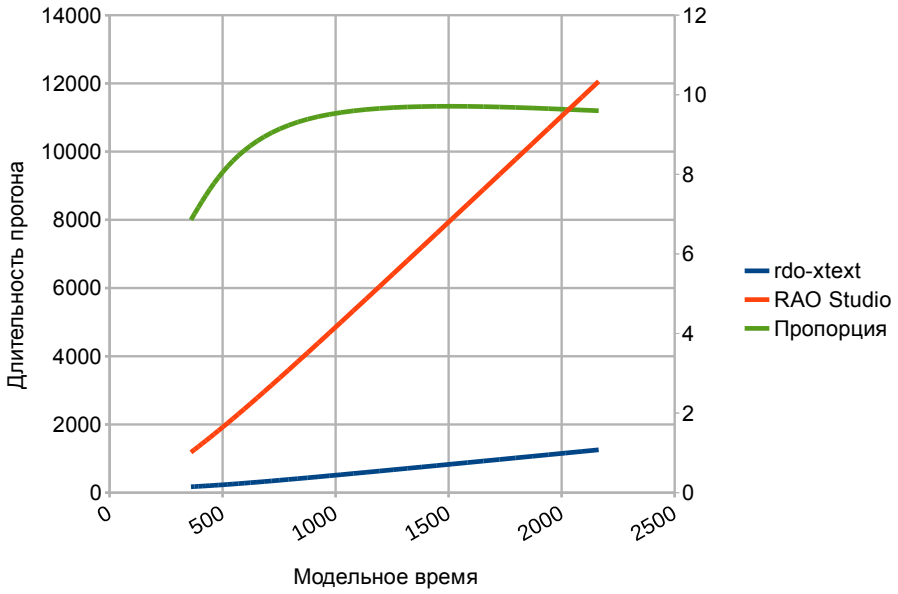
Данная задача идеально подходит для демонстрации возможностей алгоритма A*, используемого в языке РДО. И хотя полноценное поле 4x4 дает слишком большую алгоритмическую сложность, в задачах с меньшим пространством поиска (3x2, 3x3) алгоритм показывает себя очень хорошо, особенно при выборе удачной эвристики.

Для тестирования скорости работы была выбрана «конфигурация» поля размером 3x3, а варьируемым параметром – количество порожденных при поиске в ширину вершин, необходимое для решения данной задачи.

В силу того, что большое число комбинаций расстановки фишек на поле дает сравнительное небольшое количество глубин раскрытия графа (а как следствие, и порожденных вершин), было подобрано три различные комбинации фишек со следующими размерами графа: 360 вершин, 876 вершин и 2164 вершины. Так как результат субъективен, важно было качественно показать различие в скорости работы двух систем.

В результате прогона моделей в двух системах были получены следующие результаты:

Вершин	rdo-xtext	RAO-Studio	Пропорция
360	172,4	1182,1	6,85672854
876	437	4101,6	9,38581236
2164	1256,4	12058,5	9,59765998



6.3. Выводы

Скорость работы двух систем отличается в десятки раз в пользу новой разработанной системы.

В случае работы с процессным подходом видно, что порядок роста производительности rdo-xtext относительно RAO-Studio является линейной величиной, и может достигать сотен и тысяч раз на длительных прогонах.

В случае с поиском по графу пропорция между временами, затраченными системами на выполнение имитационной модели, близка к константе и примерно равна 10 раз, что также является очень хорошим результатом.

Все результаты исследования доступны на листе А2 дипломного проекта «Исследовательская часть».

7. Организационно-экономическая часть

7.1. Введение

В этой главе выполнена организационно-экономическая часть дипломного проекта, выполнен расчет затрат на разработку интегрированной среды разработки языка РДО. Для этого были 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 интегрированной среды разработки языка РДО.

Расчет действителен на 2-ой квартал 2015 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.

7.2.1. Расчет 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 [15] 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.

7.2.2. Расчет 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.

7.2.3. Расчет 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:

$$t_{РЗ} = 86 \text{ ч.},$$
$$t_{РП} = 23 \text{ ч.}$$

$K_{Р}$ - 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

7.2.4. Расчет 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_P - 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.

7.2.5. Расчет 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.

7.2.6. Расчет 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:

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

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.

7.3.1. Расчет 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: $C_{зо} = \sum_i \frac{З_i}{d} \cdot \tau_i$,

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

$З_i = 32000\,руб.$;

$d = 21$;

$\tau_i = 788{,}13$

$C_{зо} = \frac{32000}{21} \cdot 788{,}13 = 1200960\,руб.$

7.3.2. Расчет 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: $C_{зд} = C_{зо} \cdot \alpha_д$,

где $\alpha_д$ - коэффициент на дополнительную заработную плату.

$\alpha_д = 0{,}2$

$C_{зд} = 1200960 \cdot 0{,}2 = 240192\,руб.$

7.3.3. Отчисления на социальное страхование

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

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

$$\alpha_{д} = 0,30 + 0,002 = 0,302$$

$$C_{зд} = (1200960 + 240192) \cdot 0,302 = 435228 \; руб.$$

7.3.4. Накладные 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.

$$C_{н} = C_{зо} \cdot \alpha_{н},$$

$$\alpha_{н} = 1,8$$

$$C_{зд} = 1200960 \cdot 1,8 = 2161728 \; руб.$$

7.3.5. Расходы на оборудование (амортизация)

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

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

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

α_{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	35000	35000

Затраты на амортизацию оборудования (ПЭВМ):

C_{oo} = 35 000 · 20 · 449,44 · 8 / (100 · 2080) = 12101 руб.

7.3.6. Результаты расчетов затрат на разработку программного продукта

№ пп	Наименование статьи	Сметная стоимость, руб.
1	основная заработная плата	1200960
2	Дополнительная заработная плата	240192
3	Отчисления на социальное страхование	435228
4	Накладные расходы	2161728
5	Расходы на использование оборудования	12101
	Итого	3810017

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 3810017 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 разработке и проектировании интегрированной среды разработки языка РДО. 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

8.2.1. Опасные 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.

8.2.2. Вредные факторы:

8.2.2.1. Физические

- повышенные уровни электромагнитного излучения;
- повышенный уровень статического электричества;
- повышенные уровни запыленности воздуха рабочей зоны;
- повышенный уровень шума;
- повышенный уровень вибрации;

- повышенный или пониженный уровень освещенности;

8.2.2.2. Психофизиологические

- напряжение зрения;
- напряжение внимания;
- интеллектуальные нагрузки;
- эмоциональные нагрузки;
- длительные статические нагрузки;

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 м^2, в помещениях культурно-развлекательных учреждений и с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 м^2. 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 м^2 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. Повышенное значение в электрической цепи, замыкание которой может произойти через тело человека

Причины возникновения:

- В помещении установлены устройства, требующие электропитания от сети переменного тока с номинальным напряжением 220В, к ним проложена необходимая проводка. Максимальное опасное напряжение в цепи, замыкание которой может пройти через тело человека, составляет 220В.
- Устройства имеют металлические и пластиковые корпуса, которые являются потенциально опасными местами, в результате соприкосновения с которыми человек может попасть в электрическую цепь.

Влияние на человека:

Электрический ток, протекающий через организм человека, воздействует на него термически, электролитически и биологически.

- Термическое действие характеризуется нагревом тканей, вплоть до ожогов;
- Электролитическое — разложением органических жидкостей, в том числе и крови;
- Биологическое действие электрического тока проявляется в нарушении биоэлектрических процессов и сопровождается раздражением и возбуждением живых тканей и сокращением мышц, в том числе – сердечной и мышц, ответственных за дыхание.

Предлагаемые меры по обеспечению безопасности:

- Инструктаж сотрудников по организации безопасной работы с электроприборами;
- Создание регламента по обслуживанию электроприборов с целью своевременного определения ненадежных соединений и возможных мест замыкания на корпус;
- Оборудование защитного заземления/зануления;
- Размещение предупреждающих наклеек;
- Установка устройств защитного отключения.

8.3.2. Повышенный уровень электромагнитного излучения

Причина возникновения:

- Электропроводка и большое количество электроприборов на малой площади (например, несколько рабочих мест ПЭВМ)

Влияние на человека:

Механизм воздействия заключается в том, что в электрическом поле атомы и молекулы, из которых состоит человеческое тело, поляризуются,

а полярные молекулы (например, воды), кроме того, ориентируются по направлению распространения электромагнитного поля. В электролитах, которыми являются жидкие составляющие тканей, крови, межклеточной жидкости и т. п., после приложения внешнего поля появляются ионные токи. Переменное электрическое поле вызывает нагрев тканей тела человека как за счет переменной поляризации диэлектриков, так и за счет появления токов проводимости.

Тепловой эффект является следствием поглощения энергии электромагнитного поля. Кроме того, имеет место отражение электромагнитных волн от поверхности человеческого тела из-за изменения на этой границе волнового сопротивления среды. Поглощение энергии и возникновение ионных токов сопровождается специфическим воздействием на биологические ткани, поскольку нарушается тонкая структура электрических потенциалов и циркуляции жидкости в клетках и внутренних органах.

Переменное магнитное поле приводит к изменению ориентации магнитных моментов атомов и молекул. Этот эффект слабее вызываемого внешним электрическим полем, но он также небезразличен для организма.

Чем больше напряженность поля и чем больше время воздействия, тем указанные эффекты проявляются сильнее.

Регламентирующие документы:

- СанПиН 2.2.4.1191-03 "Электромагнитные поля в производственных условиях".

Табл. 4 - ПДУ воздействия периодического магнитного поля частотой 50Гц

Время пребывания, час	Допустимые уровни МП, Н [А/м] / В [мкТл] при воздействии	
	Общем	локальном
<= 1	1600 / 2000	6400/8000
2	800 / 1000	3200 / 4000
4	400 / 500	1600 / 2000
8	80 / 100	800 / 1000

8.3.3. Повышенный уровень статического электричества

Статическое электричество — это совокупность явлений, связанных с возникновением, сохранением и релаксацией свободного электрического заряда на поверхности и в объеме диэлектрических и полупроводниковых веществ, материалов изделий или на изолированных проводниках. Широкое использование в промышленности диэлектрических и полупроводниковых материалов значительно расширило область проявления статического электричества.

Влияние на человека:

Воздействие статического электричества на человека может проявляться в виде слабого длительно протекающего тока или в форме кратковременного разряда через его тело. Такой разряд вызывает у человека рефлекторное движение, что в ряде случаев может привести к попаданию работающего в опасную зону оборудования и закончиться несчастным случаем. Кроме того, электростатическое поле повышенной напряженности отрицательно влияет на организм человека, вызывая функциональные изменения со стороны центральной нервной, сердечно-сосудистой и других систем организма. Для ограничения вредного воздействия электростатического поля проводится его нормирование в

соответствии с СанПиН 2.2.4.1191-03 «Электромагнитные поля в производственных условиях».

Предельно допустимый уровень напряженности электростатического поля ($E_{ПДУ}$) при воздействии <= 1 час за смену устанавливается равным 60 кВ/м. При воздействии ЭСП более 1 часа за смену $E_{ПДУ}$ определяются по формуле $E_{ПДУ} = \frac{60}{\sqrt{t}}$, где t - время воздействия (час).

8.3.4. Повышенная запыленность воздуха рабочей зоны

Пыль может оказывать на организм различное действие: фиброгенное, токсическое, раздражающее и т. д, а также может вызывать *профессиональные заболевания* легких — пневмокониозы, пылевые бронхиты, а также др. хронические заболевания органов дыхания.

Предлагаемые меры по обеспечению безопасности:

- Установлена вентиляция;
- Проводится регулярный медицинский осмотр.

Регламентирующие документы:

ПДК вредных веществ по ГН 2.2.5.686-98 «Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны. Гигиенические нормативы».

8.3.5. Повышенный уровень шума

Согласно санитарным нормам СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки» уровень звука для работ, не требующих постоянной концентрации не должен превышать 80 дБА.

Причина возникновения:

- Работа вентиляторов ПЭВМ, системы кондиционирования.

Влияние на человека:

Снижение умственной и физической активности.

Предлагаемые меры по обеспечению безопасности:

- Периодический мониторинг шумовой нагрузки;
- Своевременная замена неисправных узлов, создающих повышенные звуковые нагрузки;
- Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

8.3.6. Повышенный уровень вибрации

Воздействие вибраций не только ухудшает самочувствие работающего и снижает производительность труда, но часто приводит к тяжелому профессиональному заболеванию – виброболезни.

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 8.1).

Таблица 8.1. 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 ×	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	10	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.

8.3.7. Повышенный или пониженный уровень освещенности

Причина возникновения:

- Рабочее место располагается в помещении с искусственным освещением

Регламентирующий документ:

- СанПиН 2.2.2/2.4.1340-03 "Гигиенические требования к персональным электронно-вычислительным машинам и организации работы"

Требования к освещению на рабочих местах, оборудованных ПЭВМ:

Освещенность на рабочем столе:	300-500 лк
Освещенность на экране ПЭВМ:	не выше 300лк
Блики на экране:	не выше 40 $кд/м^2$
Прямая блесткость источника света:	200 $кд/м^2$
Показатель ослепленности:	не более 20

Показатель дискомфорта:	не более 15
Отношение яркости	
- между рабочими поверхностями:	3:1-5:1
- между поверхностями стен и оборудования:	10:1
Коэффициент пульсации:	не более 5%.

Влияние на человека:

- Неправильные параметры освещения влияют на функционирование зрительного аппарата, то есть определяет зрительную работоспособность, на психику человека, его эмоциональное состояние, вызывает усталость центральной нервной системы, возникающей в результате прилагаемых усилий для опознания четких или сомнительных сигналов.

Предлагаемые меры по обеспечению безопасности:

- Установка дополнительного освещения.

8.3.8. Психофизические нагрузки

Причина появления:

Разработка длится в течение длительного времени, и ведется полностью на ПЭВМ, что приводит к напряжению зрения, внимания, длительным статическим и интеллектуальным нагрузкам.

Влияние на человека:

Совокупность этих вредных факторов влечёт за собой психические расстройства, проявляемые в виде депрессий, нарушении сна, нервных срывах.

Меры по обеспечению безопасности:

- Введение регламентированных перерывов в работе;
- Регулярный медицинский осмотр.

8.3.9. 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: $f_0 = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{q_1 \cdot N}{m}}$, где

q_1 - вертикальная жесткость виброизолятора, Н/см

N - число виброизоляторов, штук

m - масса виброизолятора, кг

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

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

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.)	Предель-ная (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 $f_0 = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{q_1 \cdot N}{m}} = \frac{1}{2 \cdot 3.14} \cdot \sqrt{600} = 3.89 Гц$

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.

8.4.2. 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»

8.4.2.1. 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).

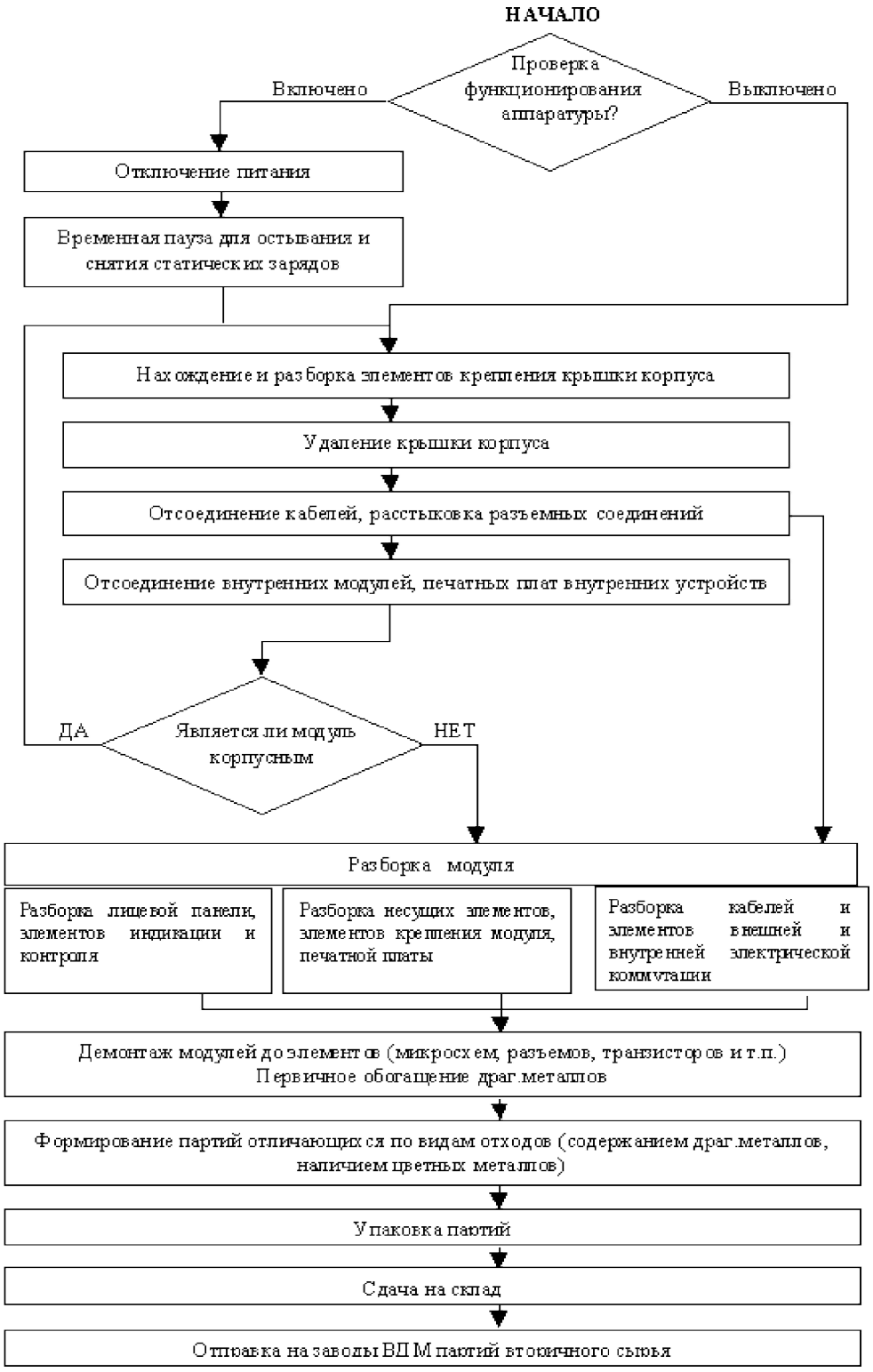


Рисунок 8.1 Технологическая схема разборки ПЭВМ

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.

8.4.2.2. 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.

8.4.2.3. 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.

8.4.2.4. 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.

8.4.3. 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.

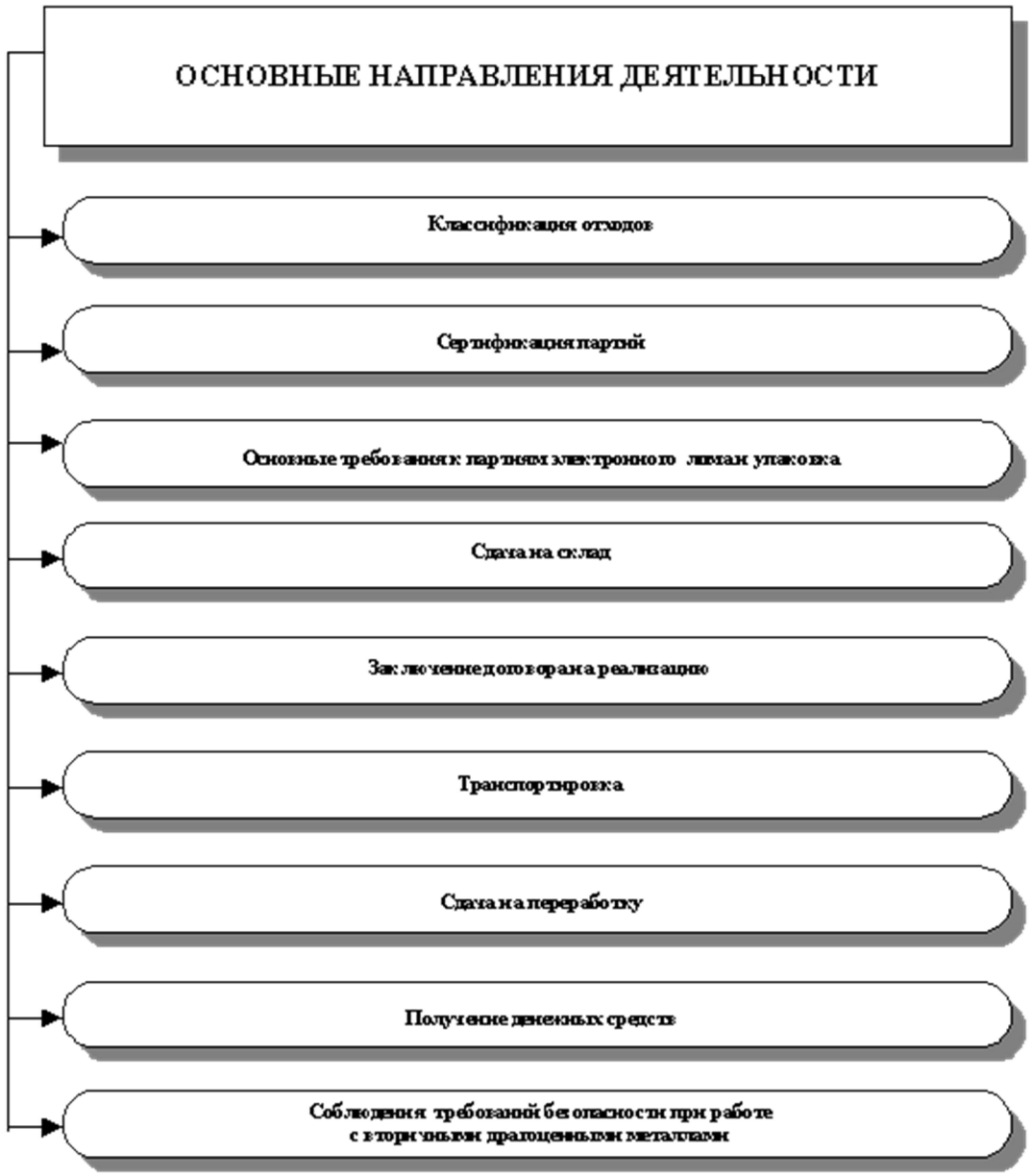


Рисунок 8.2 Основные направления деятельности на этапе «Реализация партий»

8.4.3.1. 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.

8.4.3.2. 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
11.	драгметаллами
12.	Элементы питания, аккумуляторы, ампульные батареи
	Сыпучие материалы (шихта катализаторов, зола фотоматериалов, шлам фиксажный и т.п.)

8.4.3.3. 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.

8.4.3.4. 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.

8.4.3.5. 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.

8.4.3.6. 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.

8.4.3.7. 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.

8.4.3.8. 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 проник-новения	Класс опас-ности	ПДК вредных веществ в воздухе рабочей зоны, мг/м^3
Серебро и его соединения	Отходы могут оказывать раздражающее действие на слизистую оболочку носа и дыхательных путей.	Органы дыхания	11	0,5-1
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 создания новой интегрированной среды разработки языка РДО.

На этапе концептуального проектирования системы выбрана платформа для разработки, описаны ее преимущества и очерчен круг задач, выполнение которых необходимо для достижения поставленной цели.

На этапе технического проектирования разработана грамматика языка и структура библиотеки симулятора языка РДО, используемой имитационными моделями. Сформированы начальные версии компонентов, предоставляемых библиотекой Xtext для реализации функционала интегрированной среды разработки.

На этапе рабочего проектирования разработаны алгоритмы и написан программный код для их реализации. Созданы элементы графического интерфейса пользователя, как присущие прошлым версиям сред разработки языка РДО, так и новые, типичные для систем, основанных на платформе Eclipse. Проведены отладка и тестирование системы, в ходе которых исправлялись найденные ошибки. Работоспособность системы была проверена в операционных системах Windows, Linux, MacOS на написанных ранее моделях.

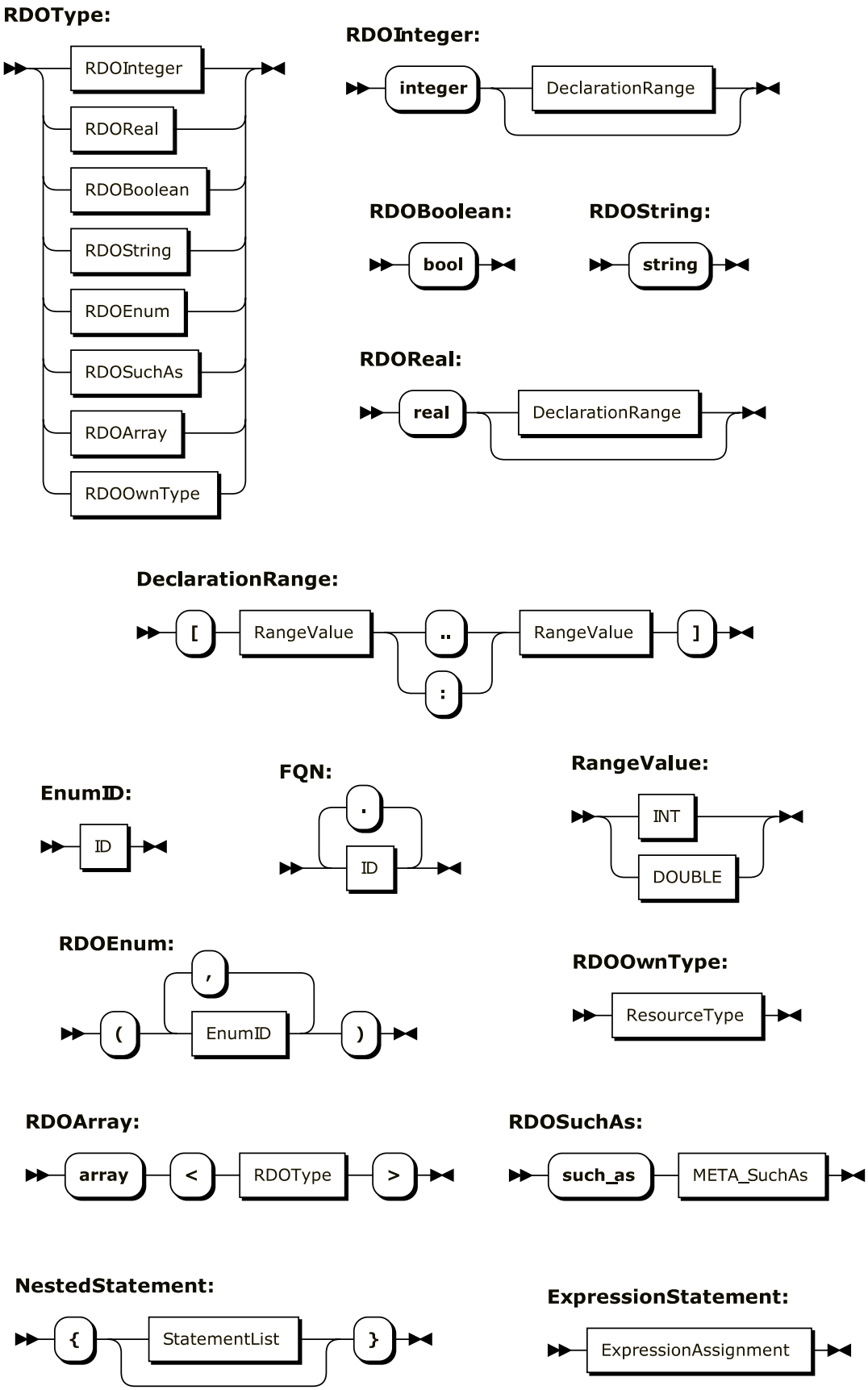
В исследовательской части произведен анализ скорости работы системы относительно RAO-Studio, с положительными результатами.

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

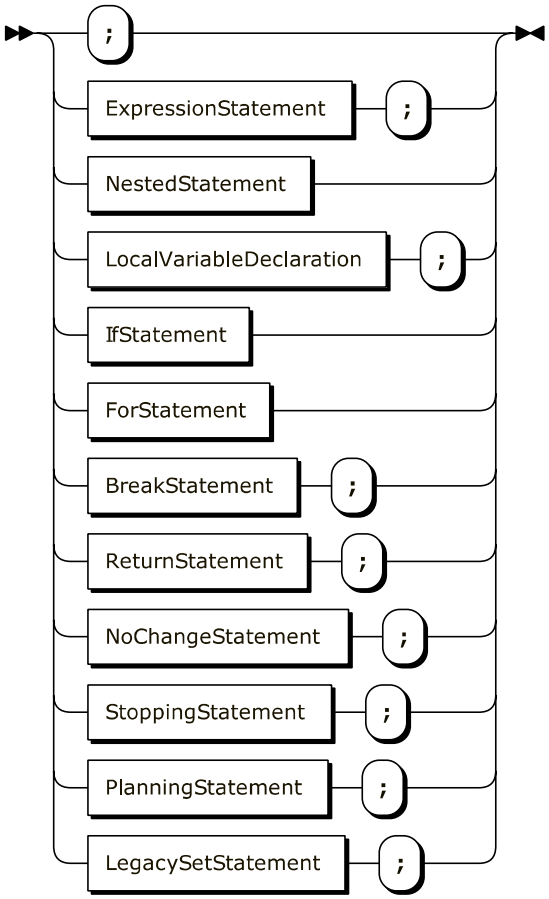
Список литературы

1. Справка по языку РДО [http://rdostudio.com/help/];
2. Справка по RAO-studio [http://rdostudio.com/help/];
3. Емельянов В. В., Ясиновский С. И. Имитационное моделирование систем: Учеб. Пособие – М.: Издательство МГТУ им. Н. Э. Баумана, 2009. – 584 с.: ил. (Информатика в техническом университете);
4. [http://ru.wikipedia.org/wiki/Интегрированная_среда_разработки]
5. [http://ru.wikipedia.org/wiki/Синтаксический_анализ]
6. [http://ru.wikipedia.org/wiki/Eclipse_(среда_разработки)]
7. Martin Fowler – Domain-Specific Languages: Addison-Wesley, 2010. 640 c.
8. Lorenzo Bettini – Implementing Domain-Specific Languages with Xtext and Xtend: Packt Publishing, 2013. – 343 c.
9. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78;
10. Леоненков. Самоучитель по UML
11. [https://ru.wikipedia.org/wiki/Алгоритм_поиска_A*];
12. *Единая система программной документации. Техническое задание.*
13. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701-90. Условные обозначения и правила выполнения.
14. 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.
15. *Сборник типовых расчетов по курсу «Охрана труда» для студентов.* МВТУ: б.н., 1984 г.
16. СНиП 21.01-97. *Нормы пожарной безопасности «Определение категорий»*

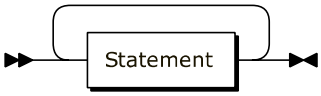
Приложение 1



Statement:

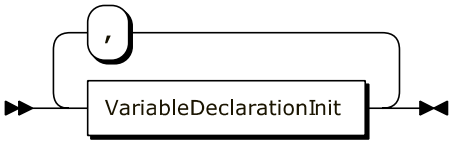


StatementList:

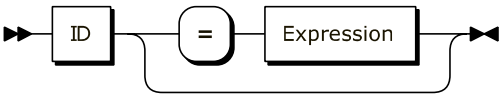


LocalVariableDeclaration:

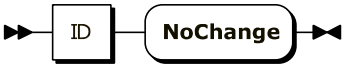
VariableDeclarationList:



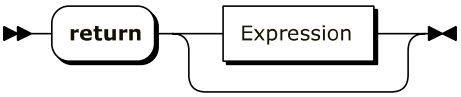
VariableDeclarationInit:



NoChangeStatement:



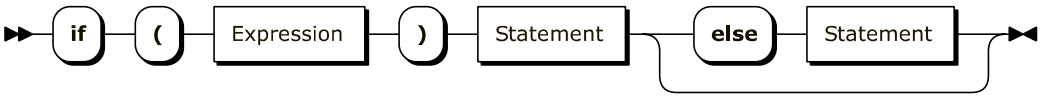
ReturnStatement:



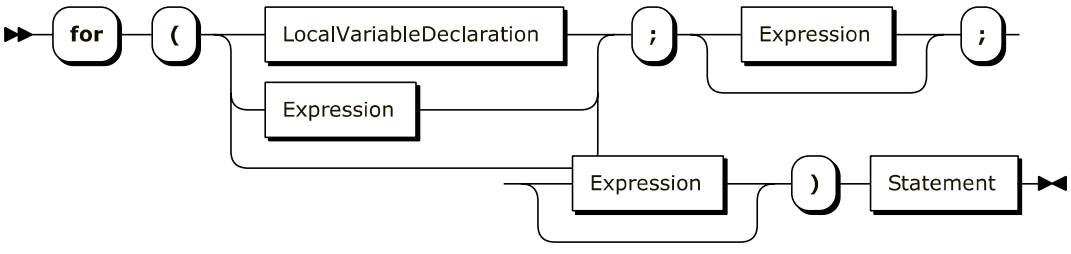
BreakStatement:

StoppingStatement:

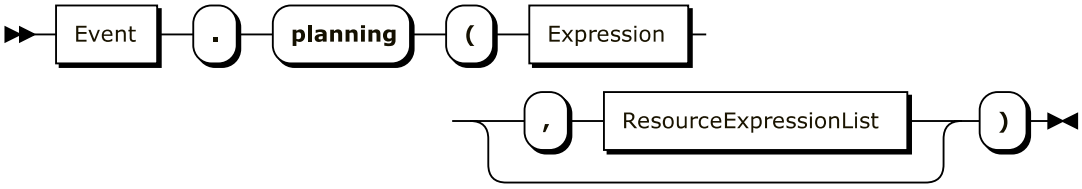
IfStatement:



ForStatement:

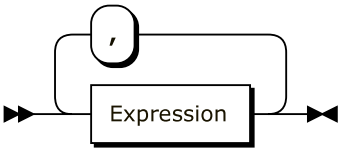


PlanningStatement:



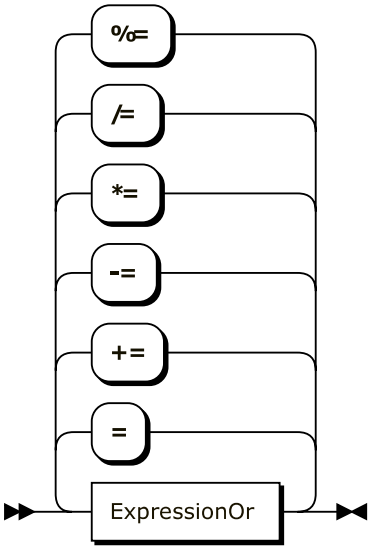
LegacySetStatement:

ExpressionList:

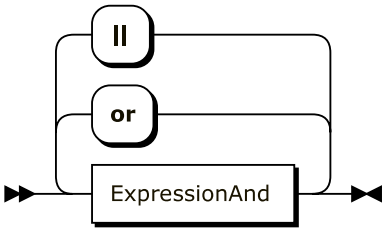


Expression:

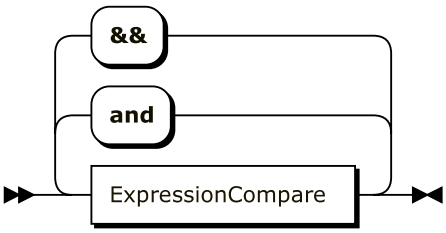
ExpressionAssignment:



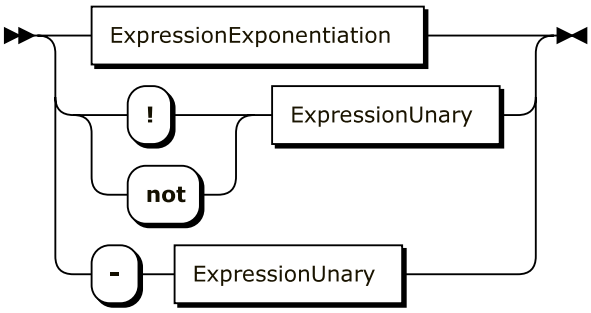
ExpressionOr:



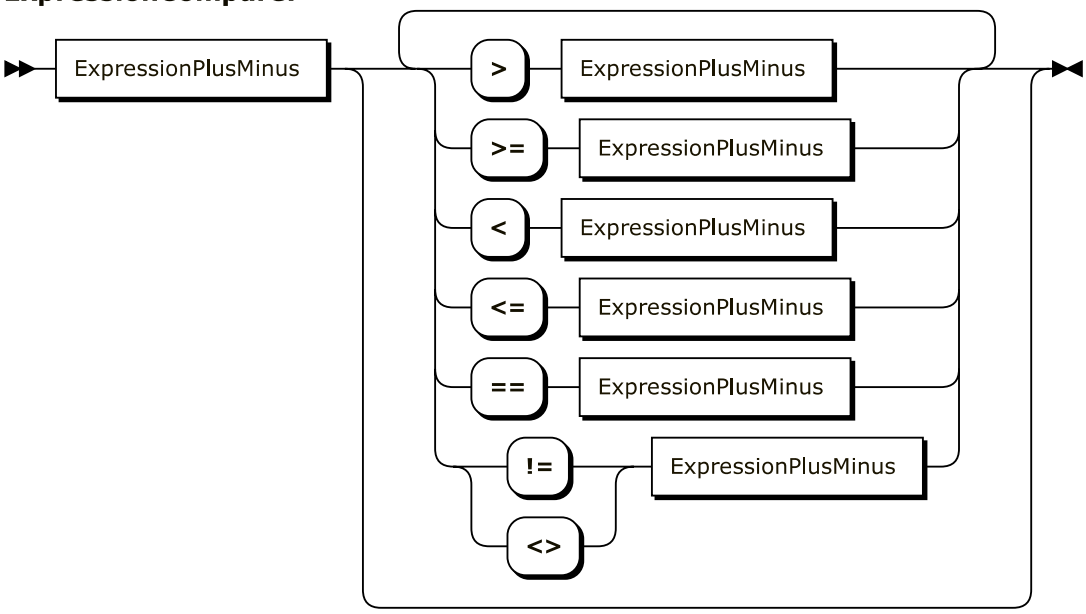
ExpressionAnd:



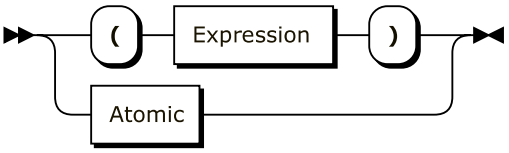
ExpressionUnary:



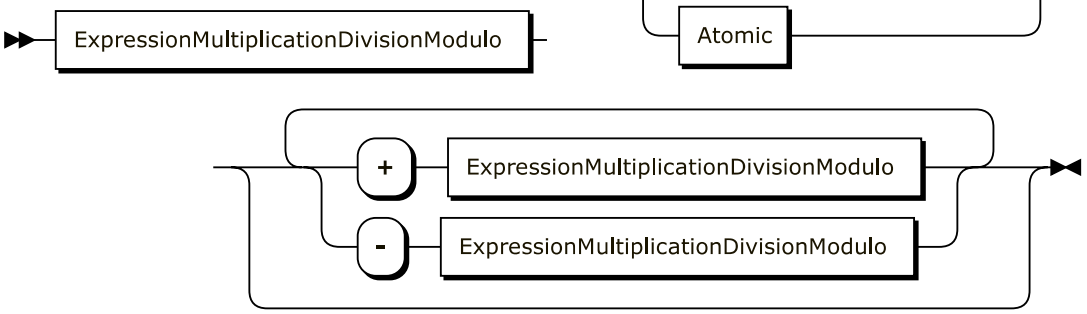
ExpressionCompare:



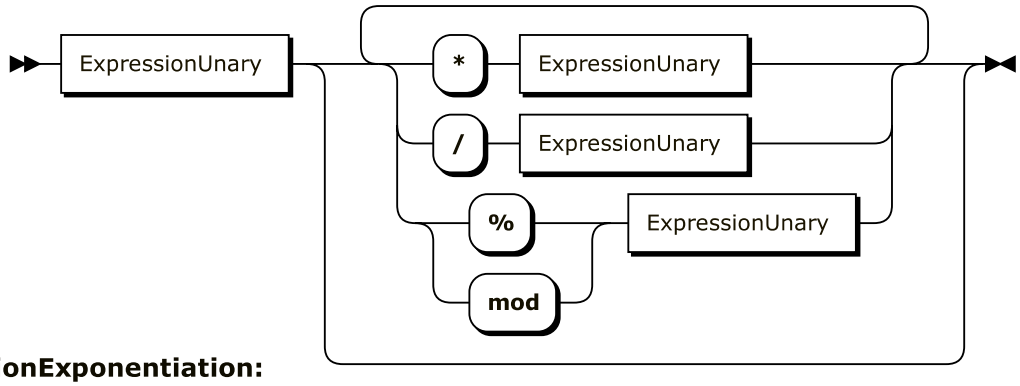
Primary:



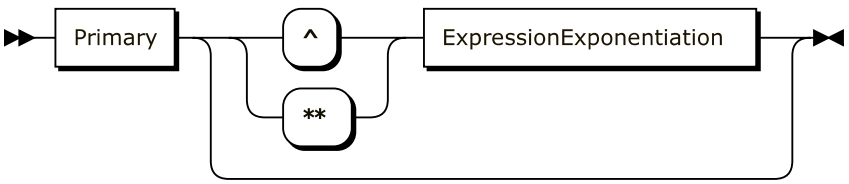
ExpressionPlusMinus:



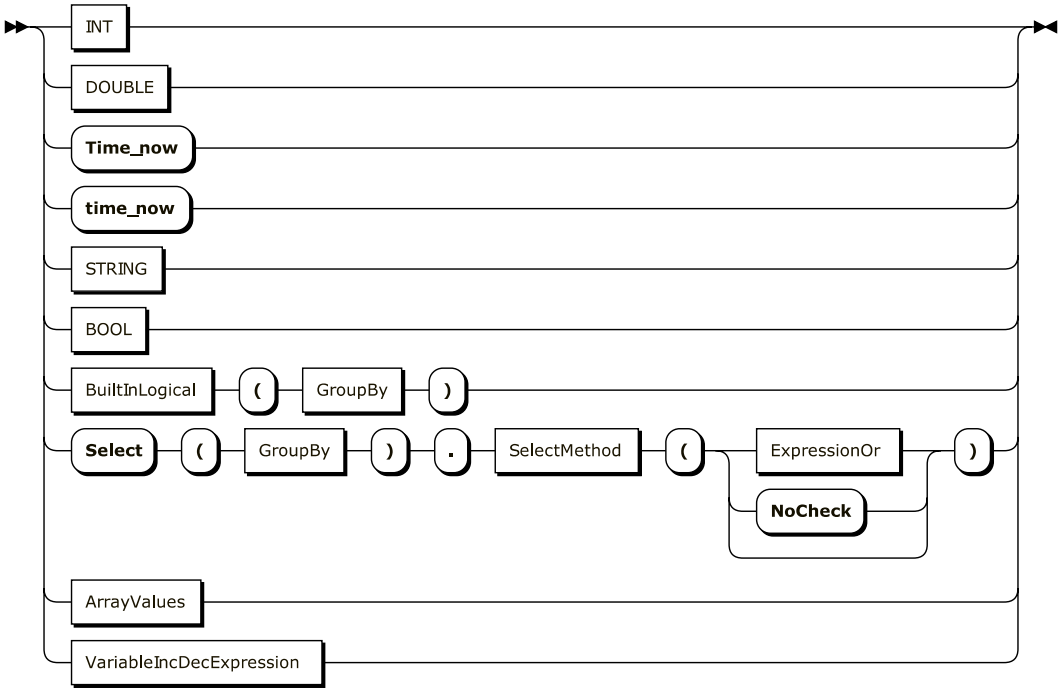
ExpressionMultiplicationDivisionModulo:



ExpressionExponentiation:

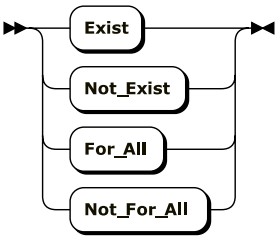


Atomic:

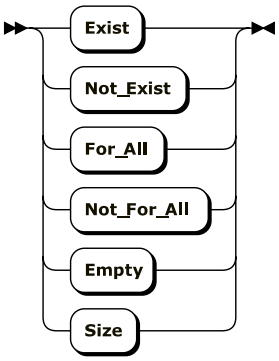


ArrayValues:

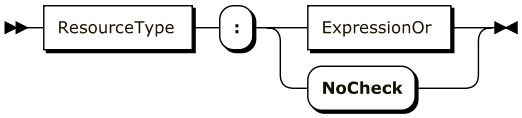
BuiltInLogical:



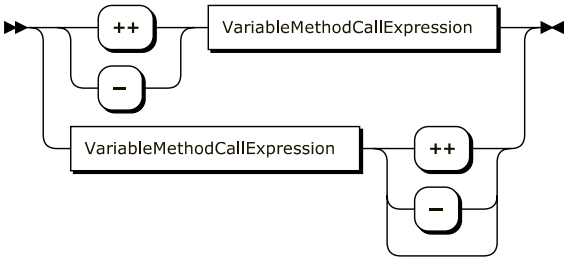
SelectMethod:



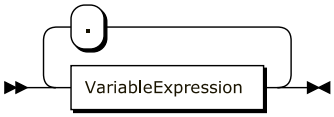
GroupBy:



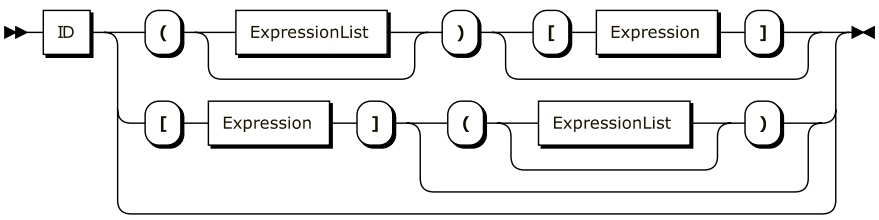
VariableIncDecExpression:



VariableMethodCallExpression:



VariableExpression:



```
// =========================================================
//                      Statement language
// =========================================================
StatementList:
        (statements += Statement)+
;

Statement
        : empty ?= ';'
        | ExpressionStatement ';'
        | NestedStatement
        | LocalVariableDeclaration ';'
        | IfStatement
        | ForStatement
        | BreakStatement ';'
        | ReturnStatement ';'
//      | process_input_statement
        | NoChangeStatement ';'
        | StoppingStatement ';'
        | PlanningStatement ';'
        | LegacySetStatement ';'
//      | watch_start
//      | watch_stop
;

ExpressionStatement:
        expr = ExpressionAssignment
;

NestedStatement:
        invoke = '{' statements = StatementList?  '}'
;

LocalVariableDeclaration:
        type = RDOType list = VariableDeclarationList
;

VariableDeclarationList:
        declarations += VariableDeclarationInit (',' declarations +=
VariableDeclarationInit)*
;

VariableDeclarationInit:
        name = ID ('=' value = Expression)?
;

IfStatement:
        'if' '(' condition = Expression ')'
                then = Statement
        ( => 'else'
                else = Statement
        )?
;
```

```
ForStatement:
        'for' '('
        ( declaration = LocalVariableDeclaration | init = Expression )? ';'
          condition = Expression? ';'
          update = Expression? ')'
        body = Statement
;

BreakStatement:
        break ?= 'break'
;

ReturnStatement:
        invoke = 'return' return = Expression?
;

NoChangeStatement:
        name = ID 'NoChange'
;

StoppingStatement:
        event = [Event|FQN] '.' 'stopping' '(' ')'
;

PlanningStatement:
        event = [Event|FQN] '.' 'planning' '(' value = Expression ')'
        ('(' parameters = ResourceExpressionList ')')?
;

LegacySetStatement:
        call = FQN 'set' value = Expression
;

// =============================================================
//                       Expression language
// =============================================================
ExpressionList:
        values += Expression  (',' values += Expression)*
;

Expression
        : ExpressionAssignment
;

// Assignment: left associative, higher priority (-1)
ExpressionAssignment returns Expression:
        ExpressionOr ( => op = ('='|'+='|'-='|'*='|'/='|'%=')
                {ExpressionAssignment.left = current} next = ExpressionOr
        )*
;
```

```
// Or: left associative, priority 0
ExpressionOr returns Expression:
        ExpressionAnd (
                ('or' | '||') {ExpressionOr.left = current}
right = ExpressionAnd
        )*
;
// And: left associative, priority 1
ExpressionAnd returns Expression:
        ExpressionCompare (
                ('and' | '&&') {ExpressionAnd.left = current}
                        right = ExpressionCompare
        )*
;

// Comparisons: left associative, priority 2
ExpressionCompare returns Expression:
        ExpressionPlusMinus (
                ('>'  {ExpressionLarger.left=current}
                        right=ExpressionPlusMinus) |
                ('>=' {ExpressionLarger_Equal.left=current}
                        right=ExpressionPlusMinus) |
                ('<'  {ExpressionSmaller.left=current}
                        right=ExpressionPlusMinus) |
                ('<=' {ExpressionSmaller_Equal.left=current}
                        right=ExpressionPlusMinus) |
                ('==' {ExpressionEqual.left=current}
                        right=ExpressionPlusMinus) |
                (('!=' | '<>') {ExpressionNot_Equal.left=current}
                        right=ExpressionPlusMinus)
        )*
;

// addition/subtraction: left associative, priority 3
ExpressionPlusMinus returns Expression:
        ExpressionMultiplicationDivisionModulo (
                ('+' {ExpressionPlus.left=current}
                        right=ExpressionMultiplicationDivisionModulo) |
                ('-' {ExpressionMinus.left=current}
                        right=ExpressionMultiplicationDivisionModulo)
        )*
;
// multiplication/division, left associative, priority 4
ExpressionMultiplicationDivisionModulo returns Expression:
        ExpressionUnary (
                ('*'  {ExpressionMultiplication.left=current}
                        right=ExpressionUnary) |
                ('/'  {ExpressionDivision.left=current}
                        right=ExpressionUnary) |
                (('%' | 'mod') {ExpressionModulo.left=current}
                        right=ExpressionUnary)
        )*
;
```

```
// Unary operators: right associative, priority 5
ExpressionUnary returns Expression
        : ExpressionExponentiation
        | (('!' | 'not') {ExpressionNegate} exp=ExpressionUnary)
        | ('-'   {ExpressionInvert} exp=ExpressionUnary)
;

// exponentiation: right associative, priority 6
ExpressionExponentiation returns Expression:
        Primary
        ( ('^' | '**') {ExpressionExponentiation.left=current}
                right=ExpressionExponentiation )?
;

Primary returns Expression
        : {Primary}'(' exp = Expression ')'
        | Atomic
;

Atomic returns Expression
        : {IntConstant}    value = INT
        | {DoubleConstant} value = DOUBLE
        | {TimeNow} invoke = ('Time_now' | 'time_now')
        | {StringConstant} value = STRING
        | {BoolConstant}   value = BOOL
        | {GroupExpression} type = BuiltInLogical '(' arg = GroupBy ')'
        | {SelectExpression} 'Select' '(' arg = GroupBy ')' '.'
                method = SelectMethod '(' (arg2 = ExpressionOr | 'NoCheck'?) ')'
        | ArrayValues
        | VariableIncDecExpression
;

ArrayValues:
        '[' {ArrayValues} values = ExpressionList? ']'
;

enum BuiltInLogical
        : EXIST = 'Exist'
        | NOTEXIST = 'Not_Exist'
        | FORALL = 'For_All'
        | NOTFORALL = 'Not_For_All'
;

enum SelectMethod
        : EXIST = 'Exist'
        | NOTEXIST = 'Not_Exist'
        | FORALL = 'For_All'
        | NOTFORALL = 'Not_For_All'
        | EMPTY = 'Empty'
        | SIZE = 'Size'
;
```

```
GroupBy:
        type = [ResourceType|FQN] ':'
                (condition = ExpressionOr | 'NoCheck')
;

VariableIncDecExpression
        : (pre  = '++' | pre  = '--' ) => var = VariableMethodCallExpression
        | var = VariableMethodCallExpression ( => post = '++' | => post = '--'
        )?
;

VariableMethodCallExpression
        : calls += VariableExpression ('.' calls += VariableExpression)*
;

VariableExpression
        : call = ID
        ( => functionfirst ?= '(' (args = ExpressionList)? ')'
                ( => arraylast ?= '[' iterator = Expression ']' )?
        | => arrayfirst ?= '[' iterator = Expression ']'
                ( => functionlast ?= '(' (args = ExpressionList)? ')' )?
        )?
;
```

Приложение 2

Компилятор типов ресурсов

```
class RDOResourceTypeCompiler
{
	private static var chunkstart = 0;
	private static var chunknumber = 0;

	def public static compileResourceType(ResourceType rtp, String filename,
Iterable<ResourceDeclaration> instances)
	{
		'''
		package «filename»;

		import java.nio.ByteBuffer;

		import java.util.Collection;
		import java.util.LinkedList;
		import java.util.ArrayList;
		import java.util.HashMap;

		import ru.bmstu.rk9.rdo.lib.json.*;

		import ru.bmstu.rk9.rdo.lib.*;
		@SuppressWarnings("all")

		public class «rtp.name» implements Resource,
ResourceComparison<«rtp.name»>
		{
			private static ResourceManager<«rtp.name»> managerCurrent;

			private String name = null;

			@Override
			public String getName()
			{
				return name;
			}

			@Override
			public String getTypeName()
			{
				return "«rtp.fullyQualifiedName»";
			}

			private Integer number = null;

			@Override
			public Integer getNumber()
			{
				return number;
			}

			public void register(String name)
			{
				this.name = name;
				this.number = managerCurrent.getNextNumber();
				managerCurrent.addResource(this);
```

```
            lastCreated = this;
      }

      public void register()
      {
            this.number = managerCurrent.getNextNumber();
            managerCurrent.addResource(this);
            lastCreated = this;
      }

      private static «rtp.name» lastCreated;

      public static «rtp.name» getLastCreated()
      {
            return lastCreated;
      }

      public static «rtp.name» getResource(String name)
      {
            return managerCurrent.getResource(name);
      }

      public static «rtp.name» getResource(int number)
      {
            return managerCurrent.getResource(number);
      }

      public static java.util.Collection<«rtp.name»> getAll()
      {
            return managerCurrent.getAll();
      }

      public static Collection<«rtp.name»> getTemporary()
      {
            return managerCurrent.getTemporary();
      }

      public static void eraseResource(«rtp.name» res)
      {
            managerCurrent.eraseResource(res);
            lastDeleted = res;
            notificationManager.notifySubscribers("ResourceDeleted");
      }

      private static «rtp.name» lastDeleted;

      public static «rtp.name» getLastDeleted()
      {
            return lastDeleted;
      }

      private static NotificationManager notificationManager =
            new NotificationManager
            (
                  new String[]
                  {
                        "ResourceDeleted"
                  }
            );

      public static Notifier getNotifier()
      {
```

```
                return notificationManager;
        }

        private ResourceManager<«rtp.name»> managerOwner = managerCurrent;

        public static void setCurrentManager(ResourceManager<«rtp.name»>
manager)
        {
                managerCurrent = manager;
        }

        «IF rtp.eAllContents.filter(typeof(RDOEnum)).toList.size > 0»//
ENUMS«ENDIF»
        «FOR e : rtp.eAllContents.toIterable.filter(typeof(RDOEnum))»
                public enum «e.getEnumParentName(false)»_enum
                {
                        «e.makeEnumBody»
                }

        «ENDFOR»
        «FOR parameter : rtp.parameters»
                private volatile «parameter.type.compileType»
«parameter.name»«parameter.type.getDefault»;

                public «parameter.type.compileType» get_«parameter.name»()
                {
                        return «parameter.name»;
                }

                public «parameter.type.compileType»
set_«parameter.name»(«parameter.type.compileType» «parameter.name»)
                {
                        if(managerOwner == managerCurrent)
                                this.«parameter.name» = «parameter.name»;
                        else
                                this.copyForNewOwner().«parameter.name» =
«parameter.name»;

                        return «parameter.name»;
                }

                public «parameter.type.compileType»
set_«parameter.name»_after(«parameter.type.compileType» «parameter.name»)
                {
                        «parameter.type.compileTypePrimitive» copy =
this.«parameter.name»;

                        set_«parameter.name»(«parameter.name»);

                        return copy;
                }

        «ENDFOR»
        private «rtp.name» copyForNewOwner()
        {
                «rtp.name» copy = new
«rtp.name»(«rtp.parameters.compileResourceTypeParametersCopyCall»);

                copy.name = name;
                copy.number = number;
                managerCurrent.addResource(copy);
                return copy;
```

```
		}

		public «rtp.name»(«rtp.parameters.compileResourceTypeParameters»)
		{
			«FOR parameter : rtp.parameters»
				if(«parameter.name» != null)
					this.«parameter.name» = «parameter.name»;
			«ENDFOR»
		}

		@Override
		public boolean checkEqual(«rtp.name» other)
		{
			«FOR parameter : rtp.parameters»
	if(!this.«parameter.name».equals(other.«parameter.name»))
					return false;
			«ENDFOR»

			return true;
		}

		public final static JSONObject structure = new JSONObject()
			«rtp.compileStructure»;

		@Override
		public ByteBuffer serialize()
		{
			int size = «chunkstart + chunknumber * basicSizes.INT»;
			«rtp.parameters.filter
			[ p |
				val type = p.type.compileType
				if(type == "String" ||
type.startsWith("java.util.ArrayList"))
					return true
				else
					return false
			].compileBufferCalculation»

			ByteBuffer entry = ByteBuffer.allocateDirect(size);

			«rtp.parameters.compileSerialization»

			return entry;
		}
	}
	'''
	}

	def public static
compileResourceTypeParametersCopyCall(List<ResourceTypeParameter> parameters)
	{
		'''«IF parameters.size > 0»«
			parameters.get(0).name»«
			FOR parameter : parameters.subList(1, parameters.size)», «
				parameter.name»«
			ENDFOR»«
		ENDIF»'''
	}

	private static class basicSizes
	{
```

```
		static val INT = 4
		static val DOUBLE = 8
		static val BOOL = 1
		static val ENUM = 2
	}

	def private static String compileStructure(ResourceType rtp)
	{
		val parameters = rtp.parameters
		var offset = 0
		var chunkindex = 1;

		var cparams = ""
		for(p : parameters)
		{
			val type = p.compileType
			var ctype = ""

			val coffset = ".put(\"offset\", " + offset + ")"
			val cchunk = ".put(\"index\", " + chunkindex + ")"
			var depth = 0
			var ischunk = false
			var isenum = false
			var enums = ""

			if(type == "Integer")
			{
				ctype = "integer"
				offset = offset + basicSizes.INT
			}
			if(type == "Double")
			{
				ctype = "real"
				offset = offset + basicSizes.DOUBLE
			}
			if(type == "Boolean")
			{
				ctype = "boolean"
				offset = offset + basicSizes.BOOL
			}
			if(type.endsWith("_enum"))
			{
				ctype = "enum"
				offset = offset + basicSizes.ENUM
				isenum = true
				if(type.substring(0, type.length - 5) ==
p.fullyQualifiedName)
				{
					enums =
						'''
.put
(
	"enums", new JSONArray()
'''
					for(e : (p.resolveAllSuchAs as RDOEnum).enums)
						enums = enums + "\t\t.put(\"" + e.name + "\")\n"
					enums = enums +
						'''
)
'''
				}
				enums = enums +
```

```
                                        '''
                                        .put("enum_origin", "«type.substring(0, type.length -
5)»")
                                        '''
                        }
                        if(type.startsWith("java.util.ArrayList"))
                        {
                                ischunk = true
                                ctype = p.arrayType
                                if(ctype.endsWith("_enum"))
                                {
                                        isenum = true
                                        if(ctype.substring(0, ctype.length - 5) ==
p.fullyQualifiedName)
                                        {
                                                enums =
                                                        '''
                                                        .put
                                                        (
                                                                "enums", new JSONArray()
                                                        '''
                                                for(e : (p.resolveAllArrays as RDOEnum).enums)
                                                        enums = enums + "\t\t.put(\"" + e.name +
"\")\n"
                                                enums = enums +
                                                        '''
                                                        )
                                                        '''
                                        }
                                        enums = enums +
                                                '''
                                                .put("enum_origin", "«ctype.substring(0,
ctype.length - 5)»")
                                                '''
                                        ctype = "enum"
                                }
                                depth = p.arrayDepth
                                chunkindex = chunkindex + 1
                                ctype = "array\")\n.put(\"array_type\", \"" + ctype
                        }
                        if(type == "String")
                        {
                                ischunk = true
                                ctype = "string"
                                chunkindex = chunkindex + 1
                        }

                        cparams = cparams + '''
                                .put
                                (
                                        new JSONObject()
                                                .put("name", "«p.name»")
                                                .put("type", "«ctype»")
                                                «IF isenum»
                                                        «enums»
                                                «ENDIF»
                                                «IF ischunk»
                                                        «cchunk»
                                                        «IF depth > 0».put("depth",
«depth»)«ENDIF»
                                                «ELSE»
                                                        «coffset»
```

```
                                        «ENDIF»
                                )
                                '''
                }

                chunkstart = offset
                chunknumber = chunkindex - 1

                return
                        '''
                        .put
                        (
                                "parameters", new JSONArray()
                                        «cparams»
                        )
                        .put("last_offset", «offset»)'''

        }

        def private static String
compileBufferCalculation(Iterable<ResourceTypeParameter> parameters)
        {
                var ret = ""

                for(p : parameters)
                {
                        var typename = p.compileType
                        val depth = p.arrayDepth
                        ret = ret + "\n"
                        for(i : 0 ..< depth - 1)
                        {
                                typename = typename.substring("java.util.ArrayList<".length,
typename.length - 1)
                                ret = ret + '''
                                        «i.TABS»size += «basicSizes.INT» * («IF i ==
0»«p.name»«ELSE»inner«i - 1»«ENDIF».size() + 1);
                                        «i.TABS»for(«typename» inner«i» : «IF i ==
0»«p.name»«ELSE»inner«i - 1»«ENDIF»)
                                        «i.TABS»{
                                        '''
                        }

                        if(depth > 0)
                        {
                                typename = typename.substring("java.util.ArrayList<".length,
typename.length - 1)
                                ret = ret + '''
                                «(depth - 1).TABS»size += «basicSizes.INT» + «IF typename ==
"Integer"
                                        »«basicSizes.INT» * «IF depth > 1»inner«depth -
2»«ELSE»«p.name»«ENDIF».size();«
                                ENDIF»«
                                IF typename == "Double"
                                        »«basicSizes.DOUBLE» * «IF depth > 1»inner«depth -
2»«ELSE»«p.name»«ENDIF».size();«
                                ENDIF»«
                                IF typename == "Boolean"
                                        »«basicSizes.BOOL» * «IF depth > 1»inner«depth -
2»«ELSE»«p.name»«ENDIF».size();«
                                ENDIF»«
                                IF typename.endsWith("_enum")
```

```
                                    »«basicSizes.ENUM» * «IF depth > 1»inner«depth -
2»«ELSE»«p.name»«ENDIF».size();«
                              ENDIF»«
                              IF typename == "String"
                                    »«basicSizes.INT» * «IF depth > 1»inner«depth -
2»«ELSE»«p.name»«ENDIF».size();
                              «(depth - 1).TABS»for(String inner : «IF depth >
1»inner«depth - 2»«ELSE»«p.name»«ENDIF»)
                              «(depth - 1).TABS»  size += «basicSizes.INT» +
inner.getBytes().length;«
                              ENDIF»
                              '''
                        }
                        else
                        {
                              ret = ret + '''
                              «IF typename == "String"»byte[] bytes_of_«p.name» =
«p.name».getBytes();
                              size += «basicSizes.INT» + bytes_of_«p.name».length;
                              «ENDIF»'''
                        }

                        for(i : 0 ..< depth - 1)
                        {
                              for(j : 0 ..< depth - i - 2)
                                    ret = ret + "\t"
                              ret = ret + "}\n"
                        }
                  }

                  return ret
            }

            def private static String compileSerialization(Iterable<ResourceTypeParameter>
parameters)
            {
                  var ret = ""
                  val constsize = parameters.filter
                        [ p |
                              val type = p.type.compileType
                              if(type == "String" ||
type.startsWith("java.util.ArrayList"))
                                    return false
                              else
                                    return true
                        ]
                  val chunks =  parameters.filter
                        [ p |
                              val type = p.type.compileType
                              if(type == "String" ||
type.startsWith("java.util.ArrayList"))
                                    return true
                              else
                                    return false
                        ]

                  for(p : constsize)
                  {
                        val type = p.compileType

                        if(type == "Integer")
                              ret = ret + '''
```

```
                                    entry.putInt(«p.name»);
                                    '''
                        if(type == "Double")
                                ret = ret + '''
                                    entry.putDouble(«p.name»);
                                    '''
                        if(type == "Boolean")
                                ret = ret + '''
                                    entry.put(«p.name» ? (byte)1 : (byte)0);
                                    '''
                        if(type.endsWith("_enum"))
                                ret = ret + '''
                                    entry.putShort((short)«p.name».ordinal());
                                    '''
                }

                if(chunknumber > 0)
                        ret = ret + '''
                                int chunkstart = entry.position(); // «chunkstart»
                                int cposition = chunkstart;
                                int cnumber = 0;

                                LinkedList<Integer> stack = new LinkedList<Integer>();
                                stack.add(entry.position());

                                entry.position(chunkstart + «basicSizes.INT * chunknumber»);
                                '''

                var pnum = 0
                for(p : chunks)
                {
                        var typename = p.compileType
                        val depth = p.arrayDepth
                        ret = ret + '''

                                entry.putInt(stack.peekLast() + «basicSizes.INT * pnum»,
entry.position());
                                {
                                '''
                        for(i : 0 ..< depth - 1)
                        {
                                typename = typename.substring("java.util.ArrayList<".length,
typename.length - 1)
                                ret = ret + '''
                                    «IF i > 0»
                                    «(i+1).TABS»int size«i - 1» = inner«i - 1».size();
                                    «(i+1).TABS»entry.putInt(size«i - 1»);
                                    «(i+1).TABS»stack.add(entry.position());
                                    «(i+1).TABS»entry.position(entry.position() + size«i -
1» * «basicSizes.INT»);
                                    «ENDIF»
                                    «(i+1).TABS»int counter«i» = 0;
                                    «(i+1).TABS»for(«typename» inner«i» : «IF i ==
0»«p.name»«ELSE»inner«i - 1»«ENDIF»)
                                    «(i+1).TABS»{
                                    «(i+1).TABS» entry.putInt(stack.peekLast() +
(counter«i»++) * «basicSizes.INT», entry.position());
                                    '''
                        }
                        pnum = pnum + 1

                        if(depth > 0)
```

```
		{
			typename = typename.substring("java.util.ArrayList<".length,
typename.length - 1)
			ret = ret + '''
				«depth.TABS»int size«depth - 1» = «IF depth >
1»inner«depth - 2»«ELSE»«p.name»«ENDIF».size();
				«depth.TABS»entry.putInt(size«depth - 1»);
				«IF typename == "Integer"
					»«depth.TABS»for(Integer inner«depth - 1» : «IF
depth > 1»inner«depth - 2»«ELSE»«p.name»«ENDIF»)
				«depth.TABS» entry.putInt(inner«depth - 1»);«
				ENDIF»«
				IF typename == "Double"
					»«depth.TABS»for(Double inner«depth - 1» : «IF
depth > 1»inner«depth - 2»«ELSE»«p.name»«ENDIF»)
				«depth.TABS» entry.putDouble(inner«depth - 1»);«
				ENDIF»«
				IF typename == "Boolean"
					»«depth.TABS»for(Boolean inner«depth - 1» : «IF
depth > 1»inner«depth - 2»«ELSE»«p.name»«ENDIF»)
				«depth.TABS» entry.put(inner«depth - 1» ? (byte)1 :
(byte)0);«
				ENDIF»«
				IF typename.endsWith("_enum")
					»«depth.TABS»for(«typename» inner«depth - 1» :
«IF depth > 1»inner«depth - 2»«ELSE»«p.name»«ENDIF»)
				«depth.TABS» entry.putShort((short)inner«depth -
1».ordinal());«
				ENDIF»«
				IF typename == "String"
					»«depth.TABS»stack.add(entry.position());
				«depth.TABS»entry.position(entry.position() +
size«depth - 1» * «basicSizes.INT»);
				«depth.TABS»int counter = 0;
				«depth.TABS»for(String inner : «IF depth >
1»inner«depth - 2»«ELSE»«p.name»«ENDIF»)
				«depth.TABS»{
				«depth.TABS» entry.putInt(stack.peekLast() +
(counter++) * «basicSizes.INT», entry.position());
				«depth.TABS» byte[] bytes_of_inner = inner.getBytes();
				«depth.TABS» int size«depth» = bytes_of_inner.length;
				«depth.TABS» entry.putInt(size«depth»);
				«depth.TABS» entry.put(bytes_of_inner);
				«depth.TABS»}
				«depth.TABS»stack.removeLast();«
				ENDIF»
				'''
		}
		else
		{
			ret = ret + '''
				«IF typename == "String"»
	entry.putInt(bytes_of_«p.name».length);
					entry.put(bytes_of_«p.name»);
				«ENDIF»'''
		}

		for(i : 0 ..< depth - 1)
		{
			ret = ret + (depth - i - 1).TABS + "}\n" +
				if(i < depth - 2) (depth - i - 1).TABS +
"stack.removeLast();\n" else ""
```

```
			}

			ret = ret + "}\n"
		}

		return ret
}

def static String TABS(int number)
{
		return '''«FOR i : 0 ..< number»«ENDFOR»'''
}

def static int getArrayDepth(ResourceTypeParameter parameter)
{
		var EObject type = parameter.type
		var depth = 0;

		while(type instanceof RDORTPParameterArray || type instanceof RDOArray)
		{
			if(type instanceof RDORTPParameterArray)
				type = (type as RDORTPParameterArray).type.arraytype
			else
				type = (type as RDOArray).arraytype
			depth = depth + 1
		}

		return depth
}

def static String getArrayType(ResourceTypeParameter parameter)
{
		var EObject type = parameter.type

		while(type instanceof RDORTPParameterArray || type instanceof RDOArray)
			if(type instanceof RDORTPParameterArray)
				type = (type as RDORTPParameterArray).type.arraytype
			else
				type = (type as RDOArray).arraytype

		type.getTypename
}

def static String getTypename(EObject type)
{
		switch(type)
		{
			RDOInteger: return "integer"
			RDOReal   : return "real"
			RDOBoolean: return "boolean"
			RDOEnum   : return type.compileType
			RDOString : return "string"
			RDOSuchAs : return type.resolveAllSuchAs.getTypename
			default: return null
		}
}

def static String getDefault(RDORTPParameterType parameter)
{
		switch parameter
		{
			RDORTPParameterBasic:
```

```
                        return if(parameter.^default != null) " = " +
parameter.^default.compileExpression.value else ""

                  RDORTPParameterEnum:
                        return if(parameter.^default != null) " = " +
parameter.type.compileType + "." + parameter.^default.name else ""

                  RDORTPParameterSuchAs:
                        if(parameter.type.compileType.endsWith("_enum"))
                              return if(parameter.^default != null) " = " +
parameter.^default.compileExpressionContext((new LocalContext).
                                    populateWithEnums(parameter.type.resolveAllSuchAs
as RDOEnum)).value else ""
                        else
                              return if(parameter.^default != null) " = " +
parameter.^default.compileExpression.value else ""

                  RDORTPParameterString:
                        return if(parameter.^default != null) ' = "' +
parameter.^default + '"' else ""

                  default:
                        return ""
            }
      }

      def public static compileResourceTypeParameters(List<ResourceTypeParameter>
parameters)
      {
            '''«IF parameters.size > 0»«parameters.get(0).type.compileType» «
                  parameters.get(0).name»«
                  FOR parameter : parameters.subList(1, parameters.size)», «
                        parameter.type.compileType» «
                        parameter.name»«
                  ENDFOR»«
            ENDIF»'''
      }
}
```

Компилятор точек принятия решений

```
class RDODecisionPointCompiler
{
	def public static compileDecisionPoint(DecisionPoint dpt, String filename)
	{
		val activities = switch dpt
		{
			DecisionPointSome : dpt.activities
			DecisionPointPrior: dpt.activities.map[a | a.activity]
		}

		val priorities = if(dpt instanceof DecisionPointPrior)
			(dpt as DecisionPointPrior).activities.map[a | a.priority] else null

		val parameters = activities.map[a |
			if(a.pattern instanceof Operation)
				(a.pattern as Operation).parameters
			else
				(a.pattern as Rule).parameters
		]

		val priority = if(dpt instanceof DecisionPointPrior)
			(dpt as DecisionPointPrior).priority else null

		return
			'''
			package «filename»;

			import ru.bmstu.rk9.rdo.lib.json.*;

			import ru.bmstu.rk9.rdo.lib.*;
			@SuppressWarnings("all")

			public class «dpt.name»
			{
				«FOR i : 0 ..< activities.size»
				«IF activities.get(i).parameters.size ==
parameters.get(i).size»
				private static
«activities.get(i).pattern.fullyQualifiedName».Parameters «activities.get(i).name» =
					new
«activities.get(i).pattern.fullyQualifiedName».Parameters(«activities.get(i).compileExp
ression.value»);
				«ENDIF»
				«ENDFOR»

				private static DecisionPoint«IF dpt instanceof
DecisionPointPrior»Prior«ENDIF» dpt =
					new DecisionPoint«IF dpt instanceof
DecisionPointPrior»Prior«ENDIF»
					(
						"«dpt.fullyQualifiedName»",
						«IF priority != null»new DecisionPoint.Priority()
						{
							@Override
							public void calculate()
							{
								priority =
«priority.compileExpression.value»;
							}
```

```
                    }«ELSE»null«ENDIF»,
                    «IF dpt.condition != null
                    »new DecisionPoint.Condition()
                    {
                          @Override
                          public boolean check()
                          {
                                return
«dpt.condition.compileExpression.value»;
                          }
                    }«ELSE»null«ENDIF»
              );

        public static void init()
        {
              «FOR i : 0 ..< activities.size»
                    dpt.addActivity(
                          new DecisionPoint«IF dpt instanceof
DecisionPointPrior»Prior«
                                ENDIF».Activity«IF dpt instanceof
DecisionPointPrior»
                          (
      «ELSE»(«ENDIF»"«activities.get(i).name»"«IF dpt instanceof DecisionPointPrior»,
                                «IF priorities.get(i) != null»new
DecisionPoint.Priority()
                                {
                                      @Override
                                      public void calculate()
                                      {
                                            priority =
«priorities.get(i).compileExpression.value»;
                                      }
                                }«ELSE»null«ENDIF»
                          «ENDIF»)
                          {
                                @Override
                                public boolean checkActivity()
                                {
                                      return
«activities.get(i).pattern.fullyQualifiedName
      ».findResources(«activities.get(i).name»);
                                }

                                @Override
                                public Rule executeActivity()
                                {
      «activities.get(i).pattern.fullyQualifiedName» executed =
      «activities.get(i).pattern.fullyQualifiedName
      ».executeRule(«activities.get(i).name»);

      Simulator.getDatabase().addDecisionEntry
                                      (
                                            dpt, this,
Database.PatternType.«
                                                  IF
activities.get(i).pattern instanceof Rule»RULE«ELSE»OPERATION_BEGIN«ENDIF»,
```

```
                                                executed
                                        );

executed.addResourceEntriesToDatabase(Pattern.ExecutedFrom.«
                                                IF dpt instanceof
DecisionPointPrior»PRIOR«ELSE»SOME«ENDIF»);

                                        return executed;
                                    }
                                }
                            );

                        «ENDFOR»
                        «IF dpt.parent == null»
                            Simulator.addDecisionPoint(dpt);
                        «ELSE»
«dpt.parent.fullyQualifiedName».getDPT().addChild(dpt);
                        «ENDIF»
                    }

                    public static DecisionPoint getDPT()
                    {
                        return dpt;
                    }

                    public static final JSONObject structure = new JSONObject()
                        .put("name", "«dpt.fullyQualifiedName»")
                        .put("type", "«IF dpt instanceof
DecisionPointSome»some«ELSE»prior«ENDIF»")
                        .put("parent", «IF dpt.parent !=
null»"«dpt.parent.fullyQualifiedName»"«ELSE»(String)null«ENDIF»)
                        .put
                        (
                            "activities", new JSONArray()
                                «FOR a : activities»
                                .put
                                (
                                    new JSONObject()
                                        .put("name", "«a.name»")
                                        .put("pattern",
"«a.pattern.fullyQualifiedName»")
                                )
                                «ENDFOR»
                        );
                    }
                    '''
    }

    def public static compileDecisionPointSearch(DecisionPointSearch dpt, String
filename)
    {
        val activities = dpt.activities
        val parameters = activities.map[a | a.pattern.parameters]

        return
        '''
            package «filename»;

            import ru.bmstu.rk9.rdo.lib.json.*;
```

```
import ru.bmstu.rk9.rdo.lib.*;
@SuppressWarnings("all")

public class «dpt.name»
{
	«FOR i : 0 ..< activities.size»
	«IF (activities.get(i).parameters != null &&
activities.get(i).parameters.size == parameters.get(i).size) ||
		(activities.get(i).parameters == null &&
activities.get(i).pattern.parameters == null)»
	private static
«activities.get(i).pattern.fullyQualifiedName».Parameters «activities.get(i).name» =
		new
«activities.get(i).pattern.fullyQualifiedName».Parameters(«activities.get(i).compileExp
ression.value»);
	«ENDIF»
	«ENDFOR»

	private static
DecisionPointSearch<rdo_model.«dpt.eResource.URI.projectName»Model> dpt =
		new
DecisionPointSearch<rdo_model.«dpt.eResource.URI.projectName»Model>
		(
			"«dpt.fullyQualifiedName»",
			«IF dpt.condition != null
			»new DecisionPoint.Condition()
			{
				@Override
				public boolean check()
				{
					return
«dpt.condition.compileExpression.value»;
				}
			}«ELSE»null«ENDIF»,
			new DecisionPoint.Condition()
			{
				@Override
				public boolean check()
				{
					return
«dpt.termination.compileExpression.value»;
				}
			},
			new DecisionPointSearch.EvaluateBy()
			{
				@Override
				public double get()
				{
					return
«dpt.evaluateby.compileExpression.value»;
				}
			},
			«IF dpt.comparetops»true«ELSE»false«ENDIF»,
			new
DecisionPointSearch.DatabaseRetriever<rdo_model.«dpt.eResource.URI.projectName»Model>()
			{
				@Override
				public
rdo_model.«dpt.eResource.URI.projectName»Model get()
				{
					return
rdo_model.«dpt.eResource.URI.projectName»Model.getCurrent();
```

```
                        }
                    }
                );

                public static void init()
                {
                    «FOR a : activities»
                        dpt.addActivity(
                            new
DecisionPointSearch.Activity("«filename».«a.name»",«
                                IF a.valueafter !=
null»DecisionPointSearch.Activity.ApplyMoment.after«
        ELSE»DecisionPointSearch.Activity.ApplyMoment.before«ENDIF»)
                            {
                                @Override
                                public boolean checkActivity()
                                {
                                    return
«a.pattern.fullyQualifiedName».findResources(«a.name»);
                                }

                                @Override
                                public Rule executeActivity()
                                {
        «a.pattern.fullyQualifiedName» executed =
«a.pattern.fullyQualifiedName».executeRule(«a.name»);
                                    return executed;
                                }

                                @Override
                                public double calculateValue()
                                {
                                    return «IF a.valueafter !=
null»«a.valueafter.compileExpression.value
        »«ELSE»«a.valuebefore.compileExpression.value»«ENDIF»;
                                }
                            }
                        );
                    «ENDFOR»

                    «IF dpt.parent == null»
                        Simulator.addDecisionPoint(dpt);
                    «ELSE»
        «dpt.parent.fullyQualifiedName».getDPT().addChild(dpt);
                    «ENDIF»
                }

                public static DecisionPoint getDPT()
                {
                    return dpt;
                }

                public static final JSONObject structure = new JSONObject()
                        .put("name", "«dpt.fullyQualifiedName»")
                        .put("type", "search")
                        .put("parent", «IF dpt.parent !=
null»"«dpt.parent.fullyQualifiedName»"«ELSE»(String)null«ENDIF»)
```

```
                        .put("compare_tops", "«IF
dpt.comparetops»YES«ELSE»NO«ENDIF»")
                        .put
                        (
                              "activities", new JSONArray()
                                    «FOR a : activities»
                                    .put
                                    (
                                          new JSONObject()
                                                .put("name", "«a.name»")
                                                .put("pattern",
"«a.pattern.fullyQualifiedName»")
                                    )
                                    «ENDFOR»
                        );
                  }
            '''
      }
}
```