Государственное образовательное учреждение высшего профессионального образования

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

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ __РК (Робототехника и комплексная автоматизация)_____

КАФЕДРА __РК9 (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 курсовому проекту на тему:

___Добавление наследования типов ресурсов в язык имитационного моделирования РДО_____

_____

_____

_____

_____

_____

Студент ____группа РК9-93_____ _____ ____И.С. Намчук __
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____ ____А.В. Урусов ___
(Подпись, дата) (И.О.Фамилия)

Москва, 2010

Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« ___ » _____ 20 __ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Добавление наследования типов ресурсов в язык имитационного моделирования РДО
(Тема курсового проекта)

Студент Намчук И.С. РК9-93
(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

1. Техническое задание

Добавить наследование типов ресурсов в язык имитационного моделирования РДО

2. Оформление курсового проекта

2.1. Расчетно-пояснительная записка на 44 листах формата А4.

2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.) Лист 1: А1 Постановка задачи; Лист 2: А2 IDEF0 Компиляция RAO-Studio, А2 IDEF0 Декомпозиция Компиляция RAO-Studio; Лист 3: А2 IDEF0 Декомпозиция Компиляция rdo_parser, А2 Диаграмма компонентов; Лист 4: А2 Блок-схема работы алгоритма, А2Синтаксическая диаграмма описания типов ресурсов; Лист 5: Результаты курсового проектирования

Дата выдачи задания « ____ » _____ 2009г.

Руководитель курсового проекта _____ А.В. Урусов
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

Введение

“Сложные системы”, “системность”, “бизнес-процессы”, “управление сложными системами”, “модели” – все эти термины в настоящее время широко используются практически во всех сферах деятельности человека. Причиной этого является обобщение накопленного опыта и результатов в различных сферах человеческой деятельности и естественное желание найти и использовать некоторые общесистемные принципы и методы. Именно системность решаемых задач в перспективе должна стать той базой, которая позволит исследователю работать с любой сложной системой, независимо от ее физической сущности. Именно модели и моделирование систем является тем инструментом, которое обеспечивает эту возможность.

Имитационное моделирование (ИМ) на ЭВМ находит широкое применение при исследовании и управлении сложными дискретными системами (СДС) и процессами в них. К таким системам можно отнести экономические и производственные объекты, морские порты, аэропорты, комплексы перекачки нефти и газа, программное обеспечение сложных систем управления, вычислительные сети и многие другие. Широкое использование ИМ объясняется сложностью (а иногда и невозможностью) применения строгих методов оптимизации, которая обусловлена размерностью решаемых задач и неформализуемостью сложных систем. Так выделяют, например, следующие проблемы в исследовании операций, которые не могут быть решены сейчас и в обозримом будущем без ИМ:

1. Формирование инвестиционной политики при перспективном планировании.
2. Выбор средств обслуживания (или оборудования) при текущем планировании.
3. Разработка планов с обратной информационной связью и операционных предписаний.

Эти классы задач определяются тем, что при их решении необходимо одновременно учитывать факторы неопределенности, динамическую взаимную обусловленность текущих решений и последующих событий, комплексную взаимозависимость между управляемыми переменными исследуемой системы, а часто и строго дискретную и четко определенную последовательность интервалов времени. Указанные особенности свойственны всем сложным системам.

Проведение имитационного эксперимента позволяет:

1. Сделать выводы о поведении СДС и ее особенностях:
* без ее построения, если это проектируемая система;
* без вмешательства в ее функционирование, если это действующая система, проведение экспериментов над которой или слишком дорого, или небезопасно;
* без ее разрушения, если цель эксперимента состоит в определении пределов воздействия на систему.
2. Синтезировать и исследовать стратегии управления.
3. Прогнозировать и планировать функционирование системы в будущем.
4. Обучать и тренировать управленческий персонал и т.д.

ИМ является эффективным, но и не лишенным недостатков, методом. Трудности использования ИМ, связаны с обеспечением адекватности описания системы, интерпретацией результатов, обеспечением стохастической сходимости процесса моделирования, решением проблемы размерности и т.п. К проблемам применения ИМ следует отнести также и большую трудоемкость данного метода.

Интеллектуальное ИМ, характеризующееся возможностью использования методов искусственного интеллекта и, прежде всего, знаний, при принятии решений в процессе имитации, при управлении имитационным экспериментом, при реализации интерфейса пользователя, создании информационных банков ИМ, снимает часть проблем использования ИМ.

1.Предпроектное исследование.

1.1. Основные подходы к построению ИМ.

Системы имитационного моделирования СДС в зависимости от способов представления процессов, происходящих в моделируемом объекте, могут быть дискретными и непрерывными, пошаговыми и событийными, детерминированными и статистическими, стационарными и нестационарными.

Рассмотрим основные моменты этапа создания ИМ. Чтобы описать функционирование СДС надо описать интересующие нас события и действия, после чего создать алфавит, то есть дать каждому из них уникальное имя. Этот алфавит определяется как природой рассматриваемой СДС, так и целями ее анализа. Следовательно, выбор алфавита событий СДС приводит к ее упрощению – не рассматриваются многие ее свойства и действия, не представляющие интерес для исследователя.

Событие СДС происходит мгновенно, то есть это некоторое действие с нулевой длительностью. Действие, требующее для своей реализации определенного времени, имеет собственное имя и связано с двумя событиями – начала и окончания. Длительность действия зависит от многих причин, среди которых время его начала, используемые ресурсы СДС, характеристики управления, влияние случайных факторов и т.д. В течение времени протекания действия в СДС могут возникнуть события, приводящие к преждевременному завершению действия. Последовательность действий образует процесс в СДС (Рис. 1.).

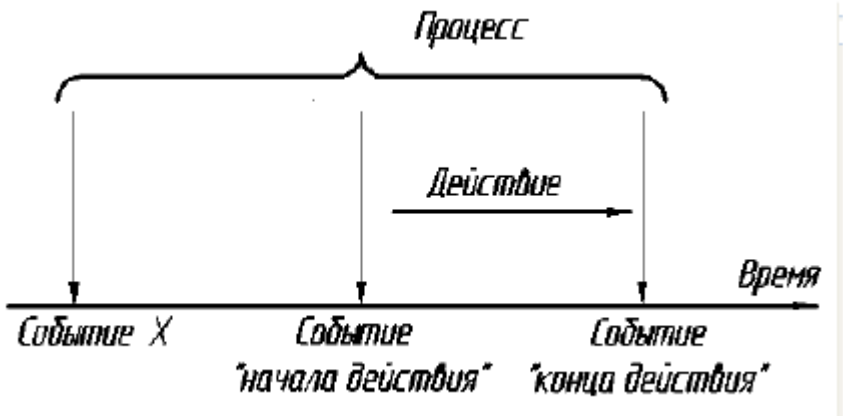

Рис. 1. Взаимосвязь между событиями, действием и процессом.

В соответствии с этим выделяют три альтернативных методологических подхода к построению ИМ: событийный, подход сканирования активностей процессно-ориентированный.

1.2. Процесс имитации в РДО.

Для имитации работы модели в РДО реализованы два подхода: событийный и сканирования активностей.

Событийный подход.

При событийном подходе исследователь описывает события, которые могут изменять состояние системы, и определяет логические взаимосвязи между ними.

Начальное состояние устанавливается путем задания значений переменным модели и параметров генераторам случайных чисел. Имитация происходит путем выбора из списка будущих событий ближайшего по времени и его выполнения.

Выполнение события приводит к изменению состояния системы и генерации будущих событий, логически связанных с выполняемым. Эти события заносятся в список будущих событий и упорядочиваются в нем по времени наступления.

Например, событие начала обработки детали на станке приводит к появлению в списке будущих событий события окончания обработки детали, которое должно наступить в момент времени равный текущему времени плюс время, требуемое на обработку детали на станке. В событийных системах модельное время фиксируется только в моменты изменения состояний.

Рис. 2. Выполнение событий в ИМ.

Подход сканирования активностей.

При использовании подхода сканирования активностей разработчик описывает все действия, в которых принимают участие элементы системы, и задает условия, определяющие начало и завершение действий. После каждого продвижения имитационного времени условия всех возможных действий проверяются и если условие выполняется, то происходит имитация соответствующего действия. Выполнение действия приводит к изменению состояния системы и возможности выполнения новых действий. Например, для начала действия обработка детали на станке необходимо наличие свободной детали и наличие свободного станка. Если хотя бы одно из этих условий не выполнено, действие не начинается.

Рис. 3. Блок-схема реализации подхода сканирования

1.3. Основные положения языка РДО.

В основе системы РДО – «Ресурсы, Действия, Операции» – лежат следующие положения:

• Все элементы сложной дискретной системы (СДС) представлены как ресурсы, описываемые некоторыми параметрами.

• Состояние ресурса определяется вектором значений всех его параметров; состояние СДС – значением всех параметров всех ресурсов.

• Процесс, протекающий в СДС, описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих определенным образом состояния ресурсов; действия ограничены во времени двумя событиями: событиями начала и конца.

• Нерегулярные события описывают изменение состояния СДС, непредсказуемые в рамках продукционной модели системы (влияние внешних по отношению к СДС факторов либо факторов, внутренних по отношению к ресурсам СДС). Моменты наступления нерегулярных событий случайны.

• Действия описываются операциями, которые представляют собой модифицированные продукционные правила, учитывающие временные связи. Операция описывает предусловия, которым должно удовлетворять состояние участвующих в операции ресурсов, и правила изменения ресурсов в начале и конце соответствующего действия.

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

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

Прогон - это единая неделимая точка имитационного эксперимента. Он характеризуется совокупностью объектов, представляющих собой исходные

данные и результаты, полученные при запуске имитатора с этими исходными данными.

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

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

Объектами исходных данных являются:

- типы ресурсов (с расширением .rtp);
- ресурсы (с расширением .rss);
- образцы операций (с расширением .pat);
- операции (с расширением .opr);
- точки принятия решений (с расширением .dpt);
- константы, функции и последовательности (с расширением .fun);
- кадры анимации (с расширением .frm);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr).

Объекты, создаваемые РДО-имитатором при выполнении прогона:

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

1.4. Постановка задачи.

Основная идея курсового проект – добавления в язык имитационного моделирования РДО наследования типов ресурсов.

На данный момент в RAO Studio отсутствует наследование типов ресурсов, это лишает язык РДО необходимой гибкости.

Рассмотрим данный недостаток на примере модели гибкой производственной системы. ГПС состоит из трех станков, обслуживаемых тремя роботом, двух тележек и двух накопителей. Детали находятся в накопителе 1, их начальное количество равно 10. При помощи робота 1, детали помещаются на тележку 1. Тележка может транспортировать одновременно только одну деталь. Тележка транспортирует деталь к первой группе станков (Станок 1 и Станок 2), которые производят одну и туже операцию. Установка и съем детали производиться при помощи робота 2. По окончании обработки, деталь помещается на тележку 2, которая транспортирует деталь к станку 3. Установка на станок 3 производиться так же с помощью робота3. По окончании обработки деталь сбрасывается в накопитель 2

Модель данной гибкой производственной системы на языке РДО представлена в Приложении 1.

Рассмотрим описание типов ресурсов на вкладке RTP.

```
$Resource_type Накопители : permanent
$Parameters
    положение                    : (станок_1, станок_2, станок_3, накопитель_1,
                                   накопитель_2, тележка_1_н, тележка_1_к,
                                   тележка_2_н, тележка_2_к, нигде)
    номер                        : integer
    максимальное_количество  : integer = 25
    текущее_количество           : integer = 0
$End

$Resource_type Тележки : permanent
$Parameters
    номер                        : integer
    положение                    : such_as Накопители.положение
    состояние                    : (свободен, занят, загружен, перемещается, прибыл,
                                   ожидает) = свободен
$End
```

```
$Resource_type Роботы : permanent
$Parameters
    номер                          : integer
    положение                      : such_as Накопители.положение
    состояние                      : (свободен, занят) = свободен
$End

$Resource_type Станки : permanent
$Parameters
    номер          : integer
    положение      : such_as Накопители.положение
    состояние      : (свободен, загружается, готов_к_обработке, работает,разгружается,
                     закончил_обработку) = свободен
    время_работы : real
$End

$Resource_type Детали : permanent
$Parameters
    номер : integer
    положение      : such_as Накопители.положение = накопитель_1
    состояние      : (хранится, транспортируется, обрабатывается, обработка_закончна) =
                     хранится
$End
```

Видно, что в приведенной модели у всех типов ресурсов есть такие параметры как “номер” и “положение”, и для каждого типа ресурса они описываются отдельно. Это уменьшает гибкость модели и увеличивает трудоемкость написания модели.

Решением данной проблемы является использование наследования, с его помощью параметры родителя будут автоматически передаваться потомку.

2. **Концептуальный этап проектирования**.

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

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

2.1.Диаграмма компонентов.

Для отображения зависимости между компонентами системы РДО и выделения среди них модернизируемых служит соответствующая диаграмма в нотации UML.

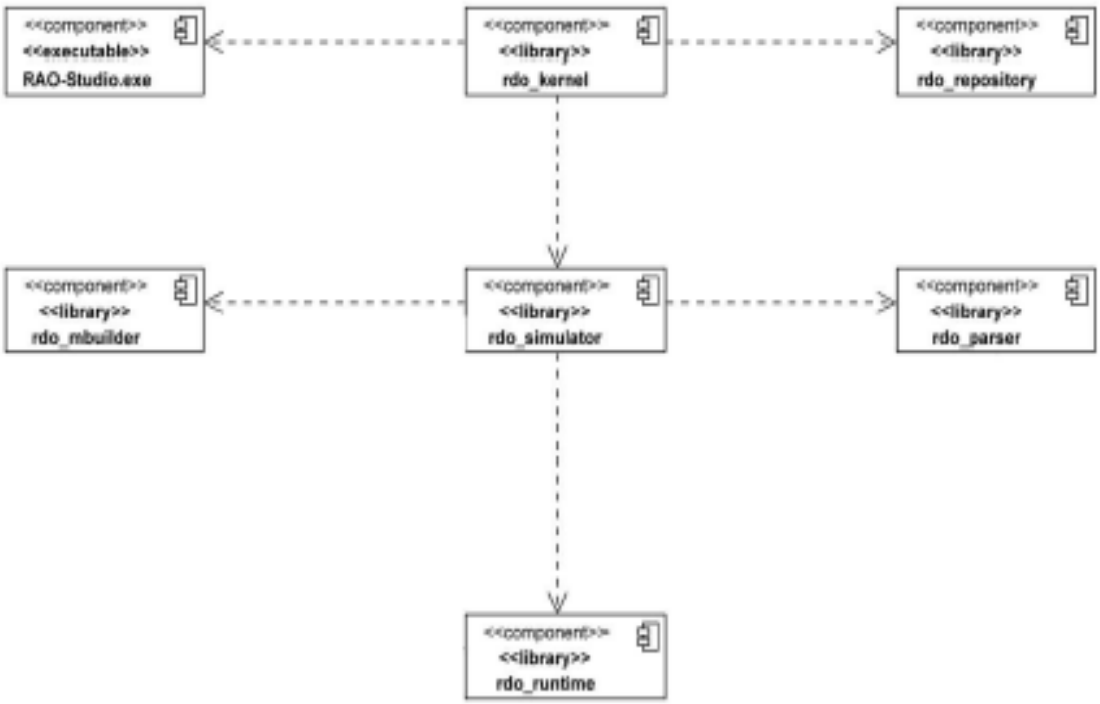

Рис. 4. Упрощенная диаграмма компонентов.

Базовый функционал представленных на диаграмме компонентов:

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

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

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

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

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

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

rdo_runtime отвечает за непосредственное выполнение модели, управление базой данных и базой знаний. Не изменяется при разработке системы.

2.2.Структура логического вывода РДО.

Логический вывод системы РДО представляет собой алгоритм, который определяет, какое событие в моделируемой системе должно произойти следующим в процессе имитации работы системы.

Во время имитации работы модели в системе существует одна МЕТА-логика. Она является контейнером для хранения разных логик. Сами логики являются контейнерами, в которых хранятся различные атомарные операции (например, нерегулярные события и правила). Таким образом, статическое представление БЗ модели на РДО представляет собой трехуровневое дерево, корнем которого является МЕТА-логика, а листьями - атомарные операции.

Интересно отметить, что реализация описанной структуры с помощью наследования – одного из основных механизмов объектно-ориентированного программирования – делает возможным на уровне логики работы РДО рекурсивное вложение логик внутрь логик. То есть архитектура имитатора РДО (***rdo_runtime***) не запрещает наличие точек принятия решений внутри точек принятия решений с любой глубиной вложенности.

Поиск активности, которая должна быть запущена следующей, начинается с обращения класса ***RDOSimulator*** к своему атрибуту ***m_logics***, в котором хранится описанная выше МЕТА-логика. Далее от корня дерева к листьям распространяется волна вызовов метода *onCheckCondition()*. Т.е. *onCheckCondition()* вызывается у МЕТА-логики, затем циклически у ее логик, и, наконец, циклически проверяются все атомарные операции каждой логики. Как только найдена активность, которая может быть выполнена, происходит ее кэширование (запоминание) внутри логики и кэширование самой логики внутри МЕТА-логики. После этого управление снова передается в ***RDOSimulator*** и найденная активность выполняется.

Для управления поиском очередной активности с помощью приоритетов точек принятия решений необходимо отсортировать список логик внутри МЕТА-логики по убыванию приоритета и в дальнейшем производить поиск в отсортированном списке.

2.3.Техническое задание.

2.3.1.Общие сведения.

В системе РДО разрабатывается наследование типов ресурсов. Основной разработчик РДО – кафедра РК-9, МГТУ им. Н.Э. Баумана.

2.3.2.Назначение и цели развития системы.

Основная цель данного курсового проекта – реализовать синтаксис описания наследования типов ресурсов в языке РДО, а так же организовать передачу параметров типов ресурсов от родителя потомкам.

2.3.3.Характеристики объекта автоматизации.

РДО – язык имитационного моделирования, включающий все три основные подхода описания дискретных систем: процессный, событийный и сканирования активностей.

2.3.4.Требования к системе.

При описании модели гибкой производственной системы в языке РДО, описание параметров “номер” и ”положение” производится однажды для новой сущности “Элементы_участка”. Это обеспечивает более адекватное описание модели системы и сокращение кода.

Таким образом с учетом возможности наследования типов ресурсов вкладка RTP примет вид:

```
$Resource_type Система : permanent
$Parameters
        положение : (станок_1, станок_2, станок_3, накопитель_1, накопитель_2,
                тележка_1_н, тележка_1_к, тележка_2_н, тележка_2_к, нигде)
$End

$Resource_type Элементы_участка : permanent
$Parameters
    номер : integer = 1
    положение  : such_as Система.положение
$End

$Resource_type Накопители : Элементы_участка : permanent
$Parameters
```

```
    максимальное_количество : integer = 25
    текущее_количество     : integer = 0
$End

$Resource_type Тележки : Элементы_участка : permanent
$Parameters
    состояние      : (свободен, занят, загружен, перемещается, прибыл, ожидает) = свободен
$End

$Resource_type Роботы : Элементы_участка : permanent
$Parameters
    состояние  : (свободен, занят) = свободен
$End

$Resource_type Станки : Элементы_участка : permanent
$Parameters
    состояние          :(свободен, загружается, готов_к_обработке,  работает,разгружается,
                        закончил_обработку) = свободен
    время_работы : real
$End

$Resource_type Детали : permanent
$Parameters
    номер : integer
    положение     : such_as Система.положение = накопитель_1
    состояние      :(хранится, транспортируется, обрабатывается, обработка_закончна) =
                    хранится
$End
```

3.1.Разработка синтаксиса описания типа ресурса.

Типы ресурсов определяют структуру глобальной базы данных программы (модели) и их описывают в отдельном объекте (имеет расширение **.rtp**).

Описание каждого типа ресурса имеет один из следующих форматов:

$Resourse_type<имя_типа>:<вид_ресурсов>
$Parameters
<описание_параметра>{< описание_параметра >}
$End

либо

$Resourse_type<имя_типа>:<имя_типа_родителя>:<вид_ресурсов>
$Parameters
<описание_параметра>{< описание_параметра >}
$End

имя_типа
Имя типа представляет собой простое имя. Имена типов должны быть различными для всех типов и не должны совпадать с предопределенными и ранее использованными именами.

имя_типа_родителя
Имя типа родителя представляет собой простое имя. Имя типа родителя должно быть определено ранее.

вид_ресурсов
Вид ресурсов данного типа может быть одним из следующих:

permanent: Постоянные ресурсы; ресурсы этого вида всегда присутствуют в модели, они не могут быть уничтожены или созданы во время прогона

temporary: Временные ресурсы; ресурсы этого вида могут во время прогона создаваться и уничтожаться при выполнении операций, правил и совершении нерегулярных событий

описание_параметра
Описание параметра ресурса имеет формат:

< имя_параметра>:< тип_параметра >[=< значение_по_умолчанию>]

имя_параметра

Имя параметра - это простое имя. Имена параметров должны быть различными для всех параметров данного типа и не должны совпадать предопределеннымии ранее использованными именами. Имя параметра может совпадать с именем параметра другого типа ресурсов.

тип_параметра

Тип параметра - это один из возможных типов данных языка. Ссылки возможны на параметры ранее описанных типов ресурсов и на ранее описанные параметры данного типа ресурсов.

значение_по_умолчанию

Для параметра любого типа может быть задано значение по умолчанию. Это значение указывают после знака равенства целой или вещественной численной константой, либо именем значения для перечислимого параметра. При указании типа ссылкой также возможно задание значения по умолчанию. При этом задаваемое значение может отличаться от значения по умолчанию того параметра, на тип которого проводится ссылка.

Тип данных языка РДО

- целый тип - ***integer***;
- вещественный тип - ***real***;
- строковый тип – ***string*;**
- логический тип – ***bool*** ;
- перечислимый тип;
- ссылка на один из выше определенных типов - ***such_as***.

3.2.Разработка архитектуры компонента rdo_parser.

Для возможности обработки новой конструкции в коде модели требуют изменений лексический и синтаксический анализаторы РДО.

4.Рабочий этап проектирования.

4.1.Синтаксический анализ типов данных.

Для реализации в среде имитационного моделирования нового инструмента разработанного на концептуальном и техническом этапах проектирования необходимо добавить в генератор синтаксического анализатора (bison) описание наследования в rtp_header:

```
RDO_Resource_type RDO_IDENTIF_COLON RDO_IDENTIF_COLON rtp_vid_res
{
      LEXER->m_enum_param_cnt = 0;
      RDOValue*     type_name =   reinterpret_cast<RDOValue*>($2);
      std::string   name      = type_name->value().getIdentificator();
      const RDORTPResType* _rtp      = PARSER->findRTPResType( name );
      if ( _rtp ) {
            PARSER->error_push_only( type_name->src_info(), rdo::format("Тип
            ресурса уже существует: %s", name.c_str()) );
            PARSER->error_push_only( _rtp->src_info(), "См. первое определение" );
            PARSER->error_push_done();
      }
      RDOValue*      prnt_type_name = reinterpret_cast<RDOValue*>($3);
      std::string    prnt_name      = prnt_type_name->value().getIdentificator();
      const RDORTPResType* _rtp_prnt      = PARSER->findRTPResType( prnt_name );
      if ( _rtp_prnt ) {
            RDORTPResType* rtp = new RDORTPResType( PARSER, type_name->src_info(),
            $4 != 0 );
            rsint t_ind=0, col_par=_rtp_prnt->getParams().size();
            while (t_ind<col_par)
            {
                  rtp->addParam(_rtp_prnt->getParams()[t_ind]);
                  PARSER->warning( rdo::format("Параметр %s передан от родителя %s
                  потомку %s", _rtp_prnt->getParams()[t_ind]-
                  >src_info().src_text().c_str(), prnt_name.c_str(),name.c_str())
                  );
                  t_ind=t_ind+1;
            }
            $$ = (int)rtp;
            PARSER->warning(   rdo::format("Тип ресурса %s является потомком типа
            ресурса %s", name.c_str(), prnt_name.c_str()) );
            }
      else {
            PARSER->error_push_only( rdo::format("Родительский тип ресурса не
            существует: %s", prnt_name.c_str()) );
            PARSER->error_push_done();
            }
      }
```

Из этого кода можно сделать вывод, что параметры типа ресурса – родителя будут скопированы потомку при его создании.

Заключение

В рамках данного курсового проекта были получены следующие результаты:

1) Проведено предпроектное исследование системы имитационного моделирования РДО и сформулированы предпосылки создания в системе наследования типов данных.

2) На этапе концептуального проектирования системы с помощью диаграммы компонентов нотации UML укрупнено, показано внутреннее устройство РДО и выделены те компоненты, которые потребуют внесения изменений в ходе этой работы. Разработаны функциональные диаграммы (в нотации IDEF0) процесса компиляции RAO-Studio.

3) На этапе технического проектирования разработан новый синтаксис для описания наследования типов ресурсов, который представлен на синтаксической диаграмме. Разработана блок – схема создаваемого алгоритма.

4) На этапе рабочего проектирования написан программный код для реализации изменеий в ***rdo_parser*** системы РДО. Проведены отладка и тестирование новой системы, в ходе которых исправлялись найденные ошибки.

5) Измененная модель ГПС при моделировании дает теже результаты, что и исходная, т.е. добавление наследования не изменило логику работы модели.

Поставленная цель курсового проекта достигнута.

Список использованных источников

1. RAO-Studio – Руководство пользователя, 2007 [**http://rdo.rk9.bmstu.ru/forum/viewtopic.php?t=900**].
2. Справка по языку РДО (в составе программы) [**http://rdo.rk9.bmstu.ru/forum/viewforum.php?f=15**].
3. Емельянов В.В., Ясиновский С.И. Имитационное моделирование систем: Учеб. пособие. – М.: Изд-во МГТУ им.Н.Э. Баумана, 2009. – 584с.: ил. (Информатика в техническом университете).
4. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78.
5. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701-90. Условные обозначения и правила выполнения.
6. Бьерн Страуструп. Язык моделирования C++. Специальное издание. Пер. с англ. – М.: ООО «Бином-пресс», 2007 г. – 1104 с.: ил.

Приложение 1. Модель гибкой производственной системы на языке РДО

Primer.pat (Образцы):

```
$Pattern Погрузка_детали :  operation trace
$Relevant_resources
    накопитель_  : Накопители Keep NoChange
    деталь_      : Детали Keep Keep
    робот_       : Роботы Keep Keep
    тележка_     : Тележки Keep Keep
$Time = Экспоненциальный( Время_погрузки )
$Body
    накопитель_
        Choice from накопитель_.номер = 1 and накопитель_.текущее_количество > 0
        first
        Convert_begin
            текущее_количество set накопитель_.текущее_количество - 1

    деталь_
        Choice from деталь_.положение = накопитель_1
        first
        Convert_begin
            положение set нигде
            состояние set транспортируется
        Convert_end
            положение set тележка_1_н

    робот_
        Choice from робот_.номер = 1 and робот_.положение = накопитель_1 and
        робот_.состояние = свободен
        first
        Convert_begin
            положение set нигде
            состояние set занят
        Convert_end
            положение set тележка_1_н
            состояние set свободен

    тележка_
        Choice from тележка_.номер = 1 and тележка_.положение = тележка_1_н and
        тележка_.состояние = свободен
        first
        Convert_begin
            состояние set занят
        Convert_end
            состояние set загружен
$End
```

```
$Pattern Доставка_детали :  operation trace
$Parameters
     номер_тележки                          : integer
     начальное_положение_устройства : such_as Накопители.положение
     конечное_положение_устройства  : such_as Накопители.положение
$Relevant_resources
     тележка_       : Тележки Keep Keep
     деталь_        : Детали NoChange Keep
$Time = Экспоненциальный( Время_доставки )
$Body
     тележка_
          Choice from тележка_.положение = начальное_положение_устройства
                         and тележка_.состояние = загружен
          first
          Convert_begin
               состояние set перемещается
          Convert_end
               положение set конечное_положение_устройства
               состояние set прибыл

     деталь_
          Choice from деталь_.положение = начальное_положение_устройства
          first
          Convert_end
               положение set конечное_положение_устройства
$End

$Pattern Возврат_робота :  operation trace
$Relevant_resources
     робот_  : Роботы Keep Keep
$Time = Время_возврата
$Body
     робот_
          Choice from робот_.положение <> место_возврата(робот_.положение)
                         and робот_.состояние = свободен
          first
          Convert_begin
               состояние set занят
          Convert_end
               состояние set свободен
               положение set место_возврата(робот_.положение)
$End

$Pattern Возврат_тележки :  operation trace
$Parameters
     номер_тележки                          : integer
     начальное_положение_устройства : such_as Накопители.положение
     конечное_положение_устройства  : such_as Накопители.положение
$Relevant_resources
     тележка_  : Тележки Keep Keep
$Time = Экспоненциальный( Время_доставки )
$Body
```

```
    тележка_
        Choice from тележка_.положение = начальное_положение_устройства
                    and тележка_.состояние = свободен
        first
        Convert_begin
            состояние set занят
        Convert_end
            состояние set свободен
            положение set конечное_положение_устройства
$End

$Pattern Установка_на_станке :  operation trace
$Parameters
    номер_станка            : such_as Накопители.положение
    положение_устройства : such_as Накопители.положение
$Relevant_resources
    станок_      : Станки Keep Keep
    тележка_     : Тележки Keep Keep
    робот_       : Роботы Keep Keep
    деталь_      : Детали NoChange Keep
$Time = Экспоненциальный( Время_установки_на_станок )
$Body
    станок_
        Choice from станок_.положение = номер_станка and станок_.состояние = свободен
        first
        Convert_begin
            состояние set загружается
        Convert_end
            состояние set готов_к_обработке

    тележка_
        Choice from тележка_.положение = положение_устройства
                    and тележка_.состояние = прибыл
        first
        Convert_begin
            состояние set ожидает
        Convert_end
            состояние set свободен

    робот_
        Choice from робот_.положение = положение_устройства
                    and робот_.состояние = свободен
        first
        Convert_begin
            состояние set занят
        Convert_end
            состояние set свободен
            положение set станок_.положение

    деталь_
        Choice from деталь_.положение = положение_устройства
        first
```

```
            Convert_end
                положение set номер_станка
$End

$Pattern Обработка_на_станке :  operation trace
$Parameters
    номер_станка : such_as Накопители.положение
$Relevant_resources
    станок_          : Станки Keep Keep
    деталь_          : Детали Keep Keep
$Time = станок_.время_работы
$Body
    станок_
            Choice from станок_.положение = номер_станка and станок_.состояние =
            готов_к_обработке
            first
            Convert_begin
                состояние set работает
            Convert_end
                состояние set закончил_обработку

    деталь_
            Choice from деталь_.положение = номер_станка
            first
            Convert_begin
                состояние set обрабатывается
            Convert_end
                состояние set обработка_закончна
$End

$Pattern Разгрузка_станков :  operation trace
$Parameters
    номер_станка : such_as Накопители.положение
$Relevant_resources
    станок_      : Станки Keep Keep
    робот_       : Роботы Keep Keep
    тележка_     : Тележки Keep Keep
    деталь_      : Детали Keep Keep
$Time = Экспоненциальный( Время_разгрузки_станка )
$Body
    станок_
            Choice from станок_.положение = номер_станка and станок_.состояние =
            закончил_обработку
            first
            Convert_begin
                состояние set разгружается
            Convert_end
                состояние set свободен

    робот_
            Choice from робот_.состояние = свободен and робот_станок(робот_.номер, с
            танок_.номер) = 1
```

```
            first
            Convert_begin
                состояние set занят
            Convert_end
                состояние set свободен
                положение set тележка_2_н

    тележка_
            Choice from тележка_.номер = 2 and тележка_.состояние = свободен
            first
            Convert_begin
                состояние set занят
            Convert_end
                состояние set загружен

    деталь_
            Choice from деталь_.положение = номер_станка
            first
            Convert_begin
                состояние set транспортируется
            Convert_end
                положение set тележка_2_н
$End

$Pattern Окончание_обработки :  operation trace
$Parameters
    номер_станка : such_as Накопители.положение
$Relevant_resources
    станок_       : Станки Keep Keep
    деталь_       : Детали Keep Keep
    накопитель_  : Накопитель_2 NoChange Keep
$Time = Экспоненциальный( Время_разгрузки_станка )
$Body
    станок_
            Choice from станок_.положение = номер_станка and станок_.состояние =
            закончил_обработку
            first
            Convert_begin
                состояние set разгружается
            Convert_end
                состояние set свободен

    деталь_
            Choice from деталь_.положение = номер_станка
            first
            Convert_begin
                состояние set транспортируется
            Convert_end
                положение set накопитель_2

    накопитель_
            Choice NoCheck
```

```
            first
            Convert_end
                текущее_количество set накопитель_.текущее_количество + 1
$End

$Pattern Работа_таймера : irregular_event trace
$Relevant_resources
    накопитель     :     Накопитель_1 Keep
$Time = 0.5
$Body
    накопитель
            Convert_event
                положение set накопитель.положение
$End
```

Primer.rtp (Типы ресурсов):

```
$Resource_type Система : permanent
$Parameters
    положение : (станок_1, станок_2, станок_3, накопитель_1, накопитель_2, тележка_1_н,
    тележка_1_к, тележка_2_н, тележка_2_к, нигде)
$End
$Resource_type Элементы_участка : permanent
$Parameters
    номер          : integer = 1
    положение   : such_as Система.положение
$End
$Resource_type Накопители : Элементы_участка : permanent
$Parameters
    максимальное_количество : integer = 25
    текущее_количество                : integer = 0
$End
$Resource_type Тележки : Элементы_участка : permanent
$Parameters
    состояние                          : (свободен, занят, загружен, перемещается, прибыл,
    ожидает) = свободен
$End
$Resource_type Роботы : Элементы_участка : permanent
$Parameters
    состояние                          : (свободен, занят) = свободен
$End
$Resource_type Станки : Элементы_участка : permanent
$Parameters
    состояние     : (свободен, загружается, готов_к_обработке, работает,разгружается,
    закончил_обработку) = свободен
    время_работы : real
$End
$Resource_type Детали : permanent
$Parameters
    номер          : integer
    положение    : such_as Система.положение = накопитель_1
```

```
    состояние      : (хранится, транспортируется, обрабатывается, обработка_закончна) =
    хранится
$End
```

Primer.rss (Ресурсы):

```
$Resources
    Система_1 :  Система trace накопитель_1

    Накопитель_1        : Накопители trace 1 накопитель_1 * 10
    Накопитель_2        : Накопители trace 2 накопитель_2 * *

    Робот_1             : Роботы trace 1 накопитель_1      *
    Робот_2             : Роботы trace 2 тележка_1_к       *
    Робот_3             : Роботы trace 3 тележка_2_к       *

    Тележка_1           : Тележки trace 1 тележка_1_н      *
    Тележка_2           : Тележки trace 2 тележка_2_н      *

    Станок_1            : Станки trace 1 станок_1 * 28
    Станок_2            : Станки trace 2 станок_2 * 28
    Станок_3            : Станки trace 3 станок_3 * 15

    Деталь_1            : Детали       1 * *
    Деталь_2            : Детали       2 * *
    Деталь_3            : Детали       3 * *
    Деталь_4            : Детали       4 * *
    Деталь_5            : Детали       5 * *
    Деталь_6            : Детали       6 * *
    Деталь_7            : Детали       7 * *
    Деталь_8            : Детали       8 * *
    Деталь_9            : Детали       9 * *
    Деталь_10           : Детали       10 * *

$End
```

Primer.opr (Операции):

```
$Operations
    Таймер_                 : Работа_таймера
    Работа_на_погрузке_1    : Погрузка_детали

    Доставка_детали_1  : Доставка_детали 1 тележка_1_н тележка_1_к
    Доставка_детали_2  : Доставка_детали 2 тележка_2_н тележка_2_к

    Установка_на_станке_1   : Установка_на_станке станок_1 тележка_1_к
    Установка_на_станке_2   : Установка_на_станке станок_2 тележка_1_к
    Установка_на_станке_3   : Установка_на_станке станок_3 тележка_2_к

    Возврат_робота_         : Возврат_робота
```

```
    Возврат_тележки_1  : Возврат_тележки 1 тележка_1_к тележка_1_н
    Возврат_тележки_2  : Возврат_тележки 2 тележка_2_к тележка_2_н

    Обработка_на_станке_1     : Обработка_на_станке станок_1
    Обработка_на_станке_2     : Обработка_на_станке станок_2
    Обработка_на_станке_3     : Обработка_на_станке станок_3

    Разгрузка_станков_1       : Разгрузка_станков станок_1
    Разгрузка_станков_2       : Разгрузка_станков станок_2

    Окончание_обработки_      : Окончание_обработки станок_3
$End
```

Primer.frm (Анимация):

```
$Frame fram_1
$Back_picture = <127 127 127> 800 800
Show

text   [10, 5,  50, 25, <127 127 127>, <100 255 0>, 'Время:' ]
text   [60, 5, 150, 25, <127 127 127>, <100 255 0>, Time_now ]


text   [10,70 ,350 ,25 , <127 127 127>, <0 0 0>, 'Станок 1 в состоянии:' ]
text   [350,70 ,350 ,25 , <127 127 127>, <0 0 0>, Станок_1.состояние]
text   [500,70 ,350 ,25 , <127 127 127>, <0 0 0>, Станок_1.положение]
text   [10,85 ,350 ,25 , <127 127 127>, <0 0 0>, 'Станок 2 в состоянии:' ]
text   [350,85 ,350 ,25 , <127 127 127>, <0 0 0>, Станок_2.состояние]
text   [500,85 ,350 ,25 , <127 127 127>, <0 0 0>, Станок_2.положение]
text   [10,100 ,350 ,25 , <127 127 127>, <0 0 0>, 'Станок 3 в состоянии:' ]
text   [350,100 ,350 ,25 , <127 127 127>, <0 0 0>, Станок_3.состояние]
text   [500,100 ,350 ,25 , <127 127 127>, <0 0 0>, Станок_3.положение]

text   [10,120 ,350 ,25 , <127 127 127>, <0 0 0>, 'Тележка_1 в состоянии:' ]
text   [350,120 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка_1.состояние]
text   [500,120 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка_1.положение]

text   [10,135 ,350 ,25 , <127 127 127>, <0 0 0>, 'Тележка_2 в состоянии:' ]
text   [350,135 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка_2.состояние]
text   [500,135 ,350 ,25 , <127 127 127>, <0 0 0>, Тележка_2.положение]

text   [10,150 ,350 ,25 , <127 127 127>, <0 0 0>, 'Робот_1 в состоянии:' ]
text   [350,150 ,350 ,25 , <127 127 127>, <0 0 0>, Робот_1.состояние]
text   [500,150 ,350 ,25 , <127 127 127>, <0 0 0>, Робот_1.положение]

text   [10,165 ,350 ,25 , <127 127 127>, <0 0 0>, 'Робот_2 в состоянии:' ]
text   [350,165 ,350 ,25 , <127 127 127>, <0 0 0>, Робот_2.состояние]
text   [500,165 ,350 ,25 , <127 127 127>, <0 0 0>, Робот_2.положение]

text   [10,180 ,350 ,25 , <127 127 127>, <0 0 0>, 'Робот_3 в состоянии:' ]
```

```
text   [350,180 ,350 ,25 , <127 127 127>, <0 0 0>, Робот_3.состояние]
text   [500,180 ,350 ,25 , <127 127 127>, <0 0 0>, Робот_3.положение]

text   [10,200 ,350 ,25 , <127 127 127>, <0 0 0>, 'Кол-во детале в 1-м накопителе:' ]
text   [350,200 ,350 ,25 , <127 127 127>, <0 0 0>, Накопитель_1.текущее_количество]
text   [10,215 ,350 ,25 , <127 127 127>, <0 0 0>, 'Кол-во детале в 2-м накопителе:' ]
text   [350,215 ,350 ,25 , <127 127 127>, <0 0 0>, Накопитель_2.текущее_количество]

text   [10,300 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 1:' ]
text   [250,300 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_1.положение]
text   [350,300 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_1.состояние]

text   [10,315 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 2:' ]
text   [250,315 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_2.положение]
text   [350,315 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_2.состояние]

text   [10,330 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 3:' ]
text   [250,330 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_3.положение]
text   [350,330 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_3.состояние]

text   [10,345 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 4:' ]
text   [250,345 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_4.положение]
text   [350,345 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_4.состояние]

text   [10,360 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 5:' ]
text   [250,360 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_5.положение]
text   [350,360 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_5.состояние]

text   [10,375 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 6:' ]
text   [250,375 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_6.положение]
text   [350,375 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_6.состояние]

text   [10,390 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 7:' ]
text   [250,390 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_7.положение]
text   [350,390 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_7.состояние]

text   [10,405 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 8:' ]
text   [250,405 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_8.положение]
text   [350,405 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_8.состояние]

text   [10,420 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 9:' ]
text   [250,420 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_9.положение]
text   [350,420 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_9.состояние]

text   [10,435 ,350 ,25 , <127 127 127>, <0 0 0>, 'Деталь 10:' ]
text   [250,435 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_10.положение]
text   [350,435 ,350 ,25 , <127 127 127>, <0 0 0>, Деталь_10.состояние]

$End
```

Primer.fun (Константы, последовательности, функции):

```
$Constant
    Время_погрузки                  : real = 1.5
    Время_установки_на_станок : real = 1.0
    Время_разгрузки_станка    : real = 0.5
    Время_возврата                : real = 0.2
    Время_доставки                  : real = 2.0
$End

$Sequence Экспоненциальный : real
$Type = exponential 123456789
$End

$Function место_возврата : such_as Накопители.положение
$Type = algorithmic
$Parameters
    текущее_место : such_as Накопители.положение
$Body
    Calculate_if текущее_место = станок_1          место_возврата = тележка_1_к
    Calculate_if текущее_место = станок_2          место_возврата = тележка_1_к
    Calculate_if текущее_место = тележка_2_н       место_возврата = тележка_1_к
    Calculate_if текущее_место = тележка_1_н       место_возврата = накопитель_1
    Calculate_if текущее_место = станок_3          место_возврата = тележка_2_к
    Calculate_if          0 = 0                    место_возврата = текущее_место
$End

$Function робот_станок : integer[0..2]
$Type = table
$Parameters
    Номер_робота_ : integer [1..3]
    Номер_станка_ : integer [1..3]
$Body
    {Номер_робота}
    {                          1       2       3       }
    {Но-   1}            0       1       0
    {мер   2}            0       1       0
    {станка 3}           0       0       1
$End
```

Primer.smr (Прогон):

```
Model_name = Primer
Resource_file = Primer
OprIev_file = Primer
Statistic_file = Primer
Results_file = Primer
Trace_file = Primer
Frame_file = Primer
Frame_number = 1
Show_mode = Animation
Show_rate = 5000.0
Terminate_if Накопитель_выходной.Количество_деталей = 10
```

Приложение 2. Полный синтаксический анализ описания типа данных (rdortp.y).

```
namespace rdoParse
{
%}

%start rtp_list

%%

rtp_list:               /* empty */
                        | rtp_list rtp_res_type
                        | error {
                              PARSER->error( "Ожидается ключевое слово
$Resource_type" );
                        };

rtp_res_type:           rtp_header RDO_Parameters rtp_body RDO_End
                        {
                              RDORTPResType* res_type =
reinterpret_cast<RDORTPResType*>($1);
                              if ( res_type->getParams().empty() )
                              {
                                    PARSER->warning( @2, rdo::format( "Тип
ресурса '%s' не содежит параметров", res_type->name().c_str() ) );
                              }
                        }
                        | rtp_header RDO_Parameters rtp_body {
                              PARSER->error( @2, "Не найдено ключевое слово
$End" );
                        }
                        | rtp_header error {
                              PARSER->error( @2, "Не найдено ключевое слово
$Parameters" );
                        };

rtp_header:             RDO_Resource_type RDO_IDENTIF_COLON rtp_vid_res
                        {
                              LEXER->m_enum_param_cnt = 0;
                              RDOValue*            type_name =
reinterpret_cast<RDOValue*>($2);
                              std::string          name      = type_name-
>value().getIdentificator();
                              const RDORTPResType* _rtp      = PARSER-
>findRTPResType( name );
                              if ( _rtp ) {
                                    PARSER->error_push_only( type_name-
>src_info(), rdo::format("Тип ресурса уже существует: %s", name.c_str()) );
                                    PARSER->error_push_only( _rtp-
>src_info(), "См. первое определение" );
                                    PARSER->error_push_done();
                              }
                              RDORTPResType* rtp = new RDORTPResType( PARSER,
type_name->src_info(), $3 != 0 );
                              $$ = (int)rtp;
                        }
                        | RDO_Resource_type RDO_IDENTIF_COLON
RDO_IDENTIF_COLON rtp_vid_res
```

```
                              {
                                    LEXER->m_enum_param_cnt = 0;
                                    RDOValue*            type_name =
reinterpret_cast<RDOValue*>($2);
                                    std::string          name      = type_name-
>value().getIdentificator();
                                    const RDORTPResType* _rtp      = PARSER-
>findRTPResType( name );
                                    if ( _rtp ) {
                                          PARSER->error_push_only( type_name-
>src_info(), rdo::format("Тип ресурса уже существует: %s", name.c_str()) );
                                          PARSER->error_push_only( _rtp-
>src_info(), "См. первое определение" );
                                          PARSER->error_push_done();
                                    }
                                    RDOValue*            prnt_type_name =
reinterpret_cast<RDOValue*>($3);
                                    std::string          prnt_name      =
prnt_type_name->value().getIdentificator();
                                    const RDORTPResType* _rtp_prnt      = PARSER-
>findRTPResType( prnt_name );
                                    if ( _rtp_prnt ) {
                                          RDORTPResType* rtp = new RDORTPResType(
PARSER, type_name->src_info(), $4 != 0 );
                                          rsint t_ind=0, col_par=_rtp_prnt-
>getParams().size();
                                          while (t_ind<col_par)
                                                {
                                                rtp->addParam(_rtp_prnt-
>getParams()[t_ind]);
                                                PARSER->warning(
rdo::format("Параметр %s передан от родителя %s потомку %s", _rtp_prnt-
>getParams()[t_ind]->src_info().src_text().c_str(),
prnt_name.c_str(),name.c_str()) );
                                                t_ind=t_ind+1;
                                                }
                                          $$ = (int)rtp;

                                          PARSER->warning(   rdo::format("Тип
ресурса %s является потомком типа ресурса %s", name.c_str(), prnt_name.c_str()) );
                                          }
                                    else {
                                          PARSER->error_push_only(
rdo::format("Родительский тип ресурса не существует: %s", prnt_name.c_str()) );
                                          PARSER->error_push_done();
                                          }
                              }
                              | RDO_Resource_type RDO_IDENTIF_COLON error {
                                    PARSER->error( @2, "Не указан вид ресурса" );
                              }
                              | RDO_Resource_type RDO_IDENTIF_COLON
RDO_IDENTIF_COLON error {
                                    PARSER->error( @3, "Не указан вид ресурса" );
                              }
                              | RDO_Resource_type error {
                                    std::string str( LEXER->YYText() );
                                    PARSER->error( @2, rdo::format("Ошибка в
описании имени типа ресурса: %s", str.c_str()) );
                              };

rtp_vid_res:            RDO_permanent     { $$ = 1; }
                              | RDO_temporary   { $$ = 0; };
```

```
rtp_body:               /* empty */ {
                        }
                        | rtp_body rtp_param {
                              RDORTPParam* param =
reinterpret_cast<RDORTPParam*>($2);
                              PARSER->getLastRTPResType()->addParam( param );
                        };

rtp_param:              RDO_IDENTIF_COLON param_type fuzzy_terms_list
                        {
                              RDOValue*            param_name =
reinterpret_cast<RDOValue*>($1);
                              RDORTPParamType*     param_type =
reinterpret_cast<RDORTPParamType*>($2);
                              RDORTPFuzzyTermsSet* terms_set  =
reinterpret_cast<RDORTPFuzzyTermsSet*>($3);
                              if ( terms_set->empty() )
                              {
                                    RDORTPParam* param = new RDORTPParam(
PARSER->getLastRTPResType(), param_name->src_info(), param_type );
                                    param_type->reparent( param );
                                    if ( param_type->typeID() ==
rdoRuntime::RDOType::t_enum ) {

      static_cast<RDORTPEnumParamType*>(param_type)->enum_name = rdo::format(
"%s.%s", PARSER->getLastRTPResType()->name().c_str(), param_name-
>src_info().src_text().c_str() );
                                    }
                                    $$ = (int)param;
                              }
                              else
                              {
                                    RDORTPFuzzyParam* param = new
RDORTPFuzzyParam( PARSER, param_name->src_info(), terms_set );
                                    param_type->reparent( param );
                                    $$ = (int)param;
                              }
                        }
                        | RDO_IDENTIF_COLON error {
                              if ( PARSER->lexer_loc_line() == @1.last_line )
{
                                    std::string str( LEXER->YYText() );
                                    PARSER->error( @2, rdo::format( "Неверный
тип параметра: %s", str.c_str() ) );
                              } else {
                                    PARSER->error( @1, "Ожидается тип
параметра" );
                              }
                        }
                        | error {
                              PARSER->error( @1, "Неправильное описание
параметра" );
                        };

fuzzy_terms_list: /* empty */ {
                              RDORTPFuzzyTermsSet* terms_set = new
RDORTPFuzzyTermsSet( PARSER );
                              $$ = (int)terms_set;
                        }
                        | fuzzy_terms_list fuzzy_term {
                              RDORTPFuzzyTermsSet* terms_set =
reinterpret_cast<RDORTPFuzzyTermsSet*>($1);
```

```
                                    RDORTPFuzzyTerm*     term      =
reinterpret_cast<RDORTPFuzzyTerm*>($2);
                                    terms_set->add( term );
                                    $$ = $1;
                                };

fuzzy_term:             RDO_Fuzzy_Term RDO_IDENTIF {
                                    RDOValue* param_name =
reinterpret_cast<RDOValue*>($2);
//                                  RDORTPFuzzyMembershiftFun*
fuzzy_membershift_fun = reinterpret_cast<RDORTPFuzzyMembershiftFun*>($3);
//                                  RDORTPFuzzyTerm* fuzzy_term = new
RDORTPFuzzyTerm( PARSER, param_name->src_info(), fuzzy_membershift_fun );
//                                  fuzzy_membershift_fun->reparent( fuzzy_term );
//                                  $$ = (int)fuzzy_term;
                                };

fuzzy_membershift_fun: /* empty */ {
                                    RDORTPFuzzyMembershiftFun* fun = new
RDORTPFuzzyMembershiftFun( PARSER );
                                    $$ = (int)fun;
                                }
                                | fuzzy_membershift_fun membership_point {

                                    RDORTPFuzzyMembershiftFun*   fun   =
reinterpret_cast<RDORTPFuzzyMembershiftFun*>($1);
                                    RDORTPFuzzyMembershiftPoint* point =
reinterpret_cast<RDORTPFuzzyMembershiftPoint*>($2);
                                    fun->add( point );
                                    $$ = $1;
                                    //Задание функции принадлежности точками -
вершинами ломанных кривых
                                };

membership_point:      '(' RDO_REAL_CONST ',' RDO_REAL_CONST ')' {

                                    double x_value =
reinterpret_cast<RDOValue*>($2)->value().getDouble();
                                    double y_value =
reinterpret_cast<RDOValue*>($4)->value().getDouble();
                                    RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                    $$ = (int)fuzzy_membershift_point;
                                }
                                | '(' RDO_REAL_CONST ',' RDO_REAL_CONST ')' ',' {

                                    double x_value =
reinterpret_cast<RDOValue*>($2)->value().getDouble();
                                    double y_value =
reinterpret_cast<RDOValue*>($4)->value().getDouble();
                                    RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                    $$ = (int)fuzzy_membershift_point;
                                }
                                | '(' RDO_REAL_CONST ',' RDO_INT_CONST ')' {

                                    double x_value =
reinterpret_cast<RDOValue*>($2)->value().getDouble();
                                    double y_value =
reinterpret_cast<RDOValue*>($4)->value().getDouble();
```

```
                                    RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                    $$ = (int)fuzzy_membershift_point;
                               }
                               | '(' RDO_REAL_CONST ',' RDO_INT_CONST ')' ',' {

                                    double x_value =
reinterpret_cast<RDOValue*>($2)->value().getDouble();
                                    double y_value =
reinterpret_cast<RDOValue*>($4)->value().getDouble();
                                    RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                    $$ = (int)fuzzy_membershift_point;
                               };

// ----------------------------------------------------------------------------
// ---------- Описание типа параметра
// ----------------------------------------------------------------------------
param_type:       RDO_integer param_int_diap param_int_default_val
                         {
                              RDORTPIntDiap*    diap =
reinterpret_cast<RDORTPIntDiap*>($2);
                              RDORTPDefVal*       dv =
reinterpret_cast<RDORTPDefVal*>($3);
                              RDORTPIntParamType* rp = new RDORTPIntParamType(
PARSER->getLastParsingObject(), diap, dv, RDOParserSrcInfo( @1, @3 ) );
                              $$ = (int)rp;
                         }
                         | RDO_real param_real_diap param_real_default_val
                         {
                              RDORTPRealDiap*    diap =
reinterpret_cast<RDORTPRealDiap*>($2);
                              RDORTPDefVal*        dv =
reinterpret_cast<RDORTPDefVal*>($3);
                              RDORTPRealParamType* rp = new RDORTPRealParamType(
PARSER->getLastParsingObject(), diap, dv, RDOParserSrcInfo( @1, @3 ) );
                              $$ = (int)rp;
                         }
                         | RDO_string param_string_default_val
                         {
                              RDORTPDefVal*          dv =
reinterpret_cast<RDORTPDefVal*>($2);
                              RDORTPStringParamType* rp = new
RDORTPStringParamType( PARSER->getLastParsingObject(), dv, RDOParserSrcInfo( @1,
@2 ) );
                              $$ = (int)rp;
                         }
                         | RDO_bool param_bool_default_val
                         {
                              RDORTPDefVal*        dv =
reinterpret_cast<RDORTPDefVal*>($2);
                              RDORTPBoolParamType* rp = new RDORTPBoolParamType(
PARSER->getLastParsingObject(), dv, RDOParserSrcInfo( @1, @2 ) );
                              $$ = (int)rp;
                         }
                         | param_enum param_enum_default_val
                         {
                              LEXER->m_enum_param_cnt = 0;
                              RDORTPEnum*  enu = reinterpret_cast<RDORTPEnum*>($1);
                              RDORTPDefVal* dv =
reinterpret_cast<RDORTPDefVal*>($2);
```

```
                              if ( dv->isExist() )
                              {
                                    enu->findEnumValueWithThrow( dv-
>value().src_pos(), dv->value().value().getAsString() ); // Если не найдено, то
будет сообщение об ошибке, т.е. throw
                              }
                              RDORTPEnumParamType* rp = new RDORTPEnumParamType(
PARSER->getLastParsingObject(), enu, dv, RDOParserSrcInfo( @1, @2 ) );
                              $$ = (int)rp;
                        }
                        | param_such_as
                        {
                              const RDORTPParam* param =
reinterpret_cast<RDORTPParam*>($1);
                              RDOParserSrcInfo src_info( @1 );
                              src_info.setSrcText( "such_as " + (param-
>getResType() ? param->getResType()->name() + "." : "") + param->name() );
                              $$ = (int)param->getType()->constructorSuchAs(
src_info );
                        }
                        | param_such_as '=' RDO_INT_CONST
                        {
                              const RDORTPParam* param =
reinterpret_cast<RDORTPParam*>($1);
                              RDOParserSrcInfo src_info( @1, @3 );
                              src_info.setSrcText( "such_as " + (param-
>getResType() ? param->getResType()->name() + "." : "") + param->name() );
                              $$ = (int)param->getType()->constructorSuchAs(
src_info, *reinterpret_cast<RDOValue*>($3) );
                        }
                        | param_such_as '=' RDO_REAL_CONST
                        {
                              const RDORTPParam* param =
reinterpret_cast<RDORTPParam*>($1);
                              RDOParserSrcInfo src_info( @1, @3 );
                              src_info.setSrcText( "such_as " + (param-
>getResType() ? param->getResType()->name() + "." : "") + param->name() );
                              $$ = (int)param->getType()->constructorSuchAs(
src_info, *reinterpret_cast<RDOValue*>($3) );
                        }
                        | param_such_as '=' RDO_IDENTIF
                        {
                              const RDORTPParam* param =
reinterpret_cast<RDORTPParam*>($1);
                              RDOParserSrcInfo src_info( @1, @3 );
                              src_info.setSrcText( "such_as " + (param-
>getResType() ? param->getResType()->name() + "." : "") + param->name() );
                              $$ = (int)param->getType()->constructorSuchAs(
src_info, *reinterpret_cast<RDOValue*>($3) );
                        }
                        | param_such_as '=' error
                        {
                              PARSER->error( "Ожидается зачение по-умолчанию" );
                        };
/*
                        | RDO_integer error {
                              PARSER->error( @2, "Ошибка после ключевого слова
integer. Возможно, не хватает значения по-умолчанию." );
                        }
                        | RDO_real error {
                              PARSER->error( @2, "Ошибка после ключевого слова
real. Возможно, не хватает значения по-умолчанию." );
                        }
```

```
                        | param_enum error {
                              PARSER->error( @2, "Ошибка после перечислимого типа.
Возможно, не хватает значения по-умолчанию." );
                        };
*/
param_int_diap:   /* empty */ {
                              YYLTYPE pos = @0;
                              pos.first_line   = pos.last_line;
                              pos.first_column = pos.last_column;
                              RDORTPIntDiap* diap = new RDORTPIntDiap( PARSER, pos
);
                              $$ = (int)diap;
                        }
                        | '[' RDO_INT_CONST RDO_dblpoint RDO_INT_CONST ']' {
                              RDORTPIntDiap* diap = new RDORTPIntDiap( PARSER,
reinterpret_cast<RDOValue*>($2)->value().getInt(),
reinterpret_cast<RDOValue*>($4)->value().getInt(), RDOParserSrcInfo( @1, @5 ), @4
);
                              $$ = (int)diap;
                        }
                        | '[' RDO_REAL_CONST RDO_dblpoint RDO_REAL_CONST {
                              PARSER->error( @2, "Требуется целочисленный диапазон,
указан вещественный" );
                        }
                        | '[' RDO_REAL_CONST RDO_dblpoint RDO_INT_CONST {
                              PARSER->error( @2, "Требуется целочисленный диапазон,
указан вещественный" );
                        }
                        | '[' RDO_INT_CONST RDO_dblpoint RDO_REAL_CONST {
                              PARSER->error( @4, "Требуется целочисленный диапазон,
указан вещественный" );
                        }
                        | '[' RDO_INT_CONST RDO_dblpoint RDO_INT_CONST error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                        }
                        | '[' RDO_INT_CONST RDO_dblpoint error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                        }
                        | '[' error {
                              PARSER->error( @2, "Диапазон задан неверно" );
                        };

param_real_diap:  /* empty */ {
                              YYLTYPE pos = @0;
                              pos.first_line   = pos.last_line;
                              pos.first_column = pos.last_column;
                              RDORTPRealDiap* diap = new RDORTPRealDiap( PARSER,
pos );
                              $$ = (int)diap;
                        }
                        | '[' RDO_REAL_CONST RDO_dblpoint RDO_REAL_CONST ']' {
                              double min = reinterpret_cast<RDOValue*>($2)-
>value().getDouble();
                              double max = reinterpret_cast<RDOValue*>($4)-
>value().getDouble();
                              RDORTPRealDiap* diap = new RDORTPRealDiap( PARSER,
min, max, RDOParserSrcInfo( @1, @5 ), @4 );
                              $$ = (int)diap;
                        }
                        | '[' RDO_REAL_CONST RDO_dblpoint RDO_INT_CONST ']' {
                              double min = reinterpret_cast<RDOValue*>($2)-
>value().getDouble();
```

```
                              double max = reinterpret_cast<RDOValue*>($4)-
>value().getDouble();
                              RDORTPRealDiap* diap = new RDORTPRealDiap( PARSER,
min, max, RDOParserSrcInfo( @1, @5 ), @4 );
                              $$ = (int)diap;
                         }
                         | '[' RDO_INT_CONST RDO_dblpoint RDO_REAL_CONST ']' {
                              double min = reinterpret_cast<RDOValue*>($2)-
>value().getDouble();
                              double max = reinterpret_cast<RDOValue*>($4)-
>value().getDouble();
                              RDORTPRealDiap* diap = new RDORTPRealDiap( PARSER,
min, max, RDOParserSrcInfo( @1, @5 ), @4 );
                              $$ = (int)diap;
                         }
                         | '[' RDO_INT_CONST RDO_dblpoint RDO_INT_CONST ']' {
                              double min = reinterpret_cast<RDOValue*>($2)-
>value().getDouble();
                              double max = reinterpret_cast<RDOValue*>($4)-
>value().getDouble();
                              RDORTPRealDiap* diap = new RDORTPRealDiap( PARSER,
min, max, RDOParserSrcInfo( @1, @5 ), @4 );
                              $$ = (int)diap;
                         }
                         | '[' RDO_REAL_CONST RDO_dblpoint RDO_REAL_CONST error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                         }
                         | '[' RDO_REAL_CONST RDO_dblpoint RDO_INT_CONST error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                         }
                         | '[' RDO_INT_CONST RDO_dblpoint RDO_REAL_CONST error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                         }
                         | '[' RDO_INT_CONST RDO_dblpoint RDO_INT_CONST error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                         }
                         | '[' RDO_REAL_CONST RDO_dblpoint error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                         }
                         | '[' RDO_INT_CONST RDO_dblpoint error {
                              PARSER->error( @4, "Диапазон задан неверно" );
                         }
                         | '[' error {
                              PARSER->error( @2, "Диапазон задан неверно" );
                         };

param_int_default_val:  /* empty */ {
                                   $$ = (int)new RDORTPDefVal(PARSER);
                              }
                              | '=' RDO_INT_CONST {
                                   $$ = (int)new RDORTPDefVal(PARSER,
*reinterpret_cast<RDOValue*>($2) );
                              }
                              | '=' RDO_REAL_CONST {
                                   PARSER->error( @2, rdo::format("Целое число
инициализируется вещественным: %f", reinterpret_cast<RDOValue*>($2)-
>value().getDouble()) );
                              }
                              | '=' error {
                                   RDOParserSrcInfo _src_info(@1, @2, true);
                                   if ( _src_info.src_pos().point() )
                                   {
```

```
                                        PARSER->error( _src_info, "Не указано
значение по-умолчанию для целого типа" );
                                  }
                                  else
                                  {
                                        PARSER->error( _src_info, "Неверное
значение по-умолчанию для целого типа" );
                                  }
                            };

param_real_default_val: /* empty */ {
                                  $$ = (int)new RDORTPDefVal(PARSER);
                            }
                            | '=' RDO_REAL_CONST {
                                  $$ = (int)new RDORTPDefVal(PARSER,
*reinterpret_cast<RDOValue*>($2));
                            }
                            | '=' RDO_INT_CONST {
                                  $$ = (int)new RDORTPDefVal(PARSER,
*reinterpret_cast<RDOValue*>($2));
                            }
                            | '=' error {
                                  RDOParserSrcInfo _src_info(@1, @2, true);
                                  if ( _src_info.src_pos().point() )
                                  {
                                        PARSER->error( _src_info, "Не указано
значение по-умолчанию для вещественного типа" );
                                  }
                                  else
                                  {
                                        PARSER->error( _src_info, "Неверное
значение по-умолчанию для вещественного типа" );
                                  }
                            };

param_string_default_val:     /* empty */
                            {
                                  $$ = (int)new RDORTPDefVal(PARSER);
                            }
                            | '=' RDO_STRING_CONST
                            {
                                  $$ = (int)new RDORTPDefVal(PARSER,
*reinterpret_cast<RDOValue*>($2));
                            }
                            | '=' error
                            {
                                  RDOParserSrcInfo _src_info(@1, @2, true);
                                  if ( _src_info.src_pos().point() )
                                  {
                                        PARSER->error( _src_info, "Не указано
значение по-умолчанию для строчного типа" );
                                  }
                                  else
                                  {
                                        PARSER->error( _src_info, "Неверное
значение по-умолчанию для строчного типа" );
                                  }
                            };

param_bool_default_val: /* empty */
                            {
                                  $$ = (int)new RDORTPDefVal(PARSER);
                            }
```

```
                              | '=' RDO_BOOL_CONST
                              {
                                    $$ = (int)new RDORTPDefVal(PARSER,
*reinterpret_cast<RDOValue*>($2));
                              }
                              | '=' error
                              {
                                    RDOParserSrcInfo _src_info(@1, @2, true);
                                    if ( _src_info.src_pos().point() )
                                    {
                                          PARSER->error( _src_info, "Не указано
значение по-умолчанию для булевского типа" );
                                    }
                                    else
                                    {
                                          PARSER->error( _src_info, "Неверное
значение по-умолчанию для булевского типа" );
                                    }
                              };

param_enum: '(' param_enum_list ')' {
                        RDORTPEnum* enu = reinterpret_cast<RDORTPEnum*>($2);
                        enu->setSrcPos( @1, @3 );
                        enu->setSrcText( enu->getEnums().asString() );
                        $$ = $2;
                  }
                  | '(' param_enum_list error {
                        PARSER->error( @2, "Перечисление должно заканчиваться
скобкой" );
                  };

param_enum_list: RDO_IDENTIF {
                              RDORTPEnum* enu = new RDORTPEnum( PARSER-
>getLastParsingObject(), *reinterpret_cast<RDOValue*>($1) );
                              enu->setSrcInfo( reinterpret_cast<RDOValue*>($1)-
>src_info() );
                              LEXER->m_enum_param_cnt = 1;
                              $$ = (int)enu;
                        }
                        | param_enum_list ',' RDO_IDENTIF {
                              if ( LEXER->m_enum_param_cnt >= 1 ) {
                                    RDORTPEnum* enu  =
reinterpret_cast<RDORTPEnum*>($1);
                                    enu->add( *reinterpret_cast<RDOValue*>($3) );
                                    $$ = (int)enu;
                              } else {
                                    PARSER->error( @3, "Ошибка в описании значений
перечислимого типа" );
                              }
                        }
                        | param_enum_list RDO_IDENTIF {
                              if ( LEXER->m_enum_param_cnt >= 1 ) {
                                    RDORTPEnum* enu  =
reinterpret_cast<RDORTPEnum*>($1);
                                    enu->add( *reinterpret_cast<RDOValue*>($2) );
                                    $$ = (int)enu;
                                    PARSER->warning( @1, rdo::format("Пропущена
запятая перед: %s", reinterpret_cast<RDOValue*>($2)-
>value().getIdentificator().c_str()) );
                              } else {
                                    PARSER->error( @2, "Ошибка в описании значений
перечислимого типа" );
                              }
```

```
                        }
                        | param_enum_list ',' RDO_INT_CONST {
                              PARSER->error( @3, "Значение перечислимого типа не
может быть цифрой" );
                        }
                        | param_enum_list ',' RDO_REAL_CONST {
                              PARSER->error( @3, "Значение перечислимого типа не
может быть цифрой" );
                        }
                        | param_enum_list RDO_INT_CONST {
                              PARSER->error( @2, "Значение перечислимого типа не
может быть цифрой" );
                        }
                        | param_enum_list RDO_REAL_CONST {
                              PARSER->error( @2, "Значение перечислимого типа не
может быть цифрой" );
                        }
                        | RDO_INT_CONST {
                              PARSER->error( @1, "Значение перечислимого типа не
может начинаться с цифры" );
                        }
                        | RDO_REAL_CONST {
                              PARSER->error( @1, "Значение перечислимого типа не
может начинаться с цифры" );
                        };

param_enum_default_val: /* empty */ {
                              $$ = (int)new RDORTPDefVal(PARSER);
                        }
                        | '=' RDO_IDENTIF {
                              $$ = (int)new RDORTPDefVal(PARSER,
*reinterpret_cast<RDOValue*>($2));
                        }
                        | '=' error {
                              RDOParserSrcInfo _src_info(@1, @2, true);
                              if ( _src_info.src_pos().point() )
                              {
                                    PARSER->error( _src_info, "Не указано
значение по-умолчанию для перечислимого типа" );
                              }
                              else
                              {
                                    PARSER->error( _src_info, "Неверное
значение по-умолчанию для перечислимого типа" );
                              }
                        };

param_such_as:    RDO_such_as RDO_IDENTIF '.' RDO_IDENTIF {
                        std::string type  = reinterpret_cast<RDOValue*>($2)-
>value().getIdentificator();
                        std::string param = reinterpret_cast<RDOValue*>($4)-
>value().getIdentificator();
                        const RDORTPResType* const rt = PARSER-
>findRTPResType( type );
                        if ( !rt ) {
                              PARSER->error( @2, rdo::format("Ссылка на
неизвестный тип ресурса: %s", type.c_str()) );
                        }
                        const RDORTPParam* const rp = rt->findRTPParam( param
);
                        if ( !rp ) {
                              PARSER->error( @4, rdo::format("Ссылка на
неизвестный параметр ресурса: %s.%s", type.c_str(), param.c_str()) );
```

```
                              }
                              $$ = (int)rp;
                        }
                        | RDO_such_as RDO_IDENTIF {
                              std::string constName =
reinterpret_cast<RDOValue*>($2)->value().getIdentificator();
                              const RDOFUNConstant* const cons = PARSER-
>findFUNConstant( constName );
                              if ( !cons ) {
                                    PARSER->error( @2, rdo::format("Ссылка на
несуществующую константу: %s", constName.c_str()) );
                              }
                              $$ = (int)cons->getDescr();
                        }
                        | RDO_such_as RDO_IDENTIF '.' error {
                              std::string type = reinterpret_cast<RDOValue*>($2)-
>value().getIdentificator();
                              const RDORTPResType* const rt = PARSER-
>findRTPResType( type );
                              if ( !rt ) {
                                    PARSER->error( @2, rdo::format("Ссылка на
неизвестный тип ресурса: %s", type.c_str()) );
                              } else {
                                    PARSER->error( @4, "Ошибка при указании
параметра" );
                              }
                        }
                        | RDO_such_as error {
                              PARSER->error( @2, "После ключевого слова such_as
необходимо указать тип и параметер ресурса для ссылки" );
                        };
   // --------------------------------------------------------------------------
```