Государственное образовательное учреждение высшего профессионального образования

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

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ РК

КАФЕДРА Компьютерные системы автоматизации производства РК-9

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:

Введение в программный продукт РДО понятия неопределенного значения параметра

Студент _____ _____ Романов Я.А.
(Подпись, дата) (И.О.Фамилия)

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

Москва, 2012

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

УТВЕРЖДАЮ
Заведующий кафедрой РК9
(Индекс)
_____ _____
(И.О.Фамилия)
« ___ » _____ 20 __ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине_____

_____
(Тема курсового проекта)
_____

Студент *Романов Я.А. РК9-101*
(Фамилия, инициалы, индекс группы)

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

1. Техническое задание

_____
_____
_____
_____

2. Оформление курсового проекта

2.1. Расчетно-пояснительная записка на *30* листах формата А4.
2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.)_____
лист 1— постановка задачи;
лист 2 – диаграмма компонентов, диаграмма классов:
лист 3—диаграмма состояний, синтаксическая диаграмма, алгоритм
лист 4—алгоритм
лист 5,6—результаты;
Дата выдачи задания « » _____ 2011г.

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

Студент _____ *Романов Я.А.*
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1. Введение

В рамках имитационного моделирования приходится сталкиваться с ситуацией, когда мы не можем быть уверены в достоверности значения какого-либо параметра системы (т.е. значение параметра является неопределенным).

1.2. Основания для разработки

При помощи программного продукта РДО можно моделировать системы, когда все значения параметров задаются жестко, либо по какому-либо закону, но нет возможности моделировать системы с неопределенным значением параметров. Есть необходимость в расширении возможностей РДО, а именно в возможности работы с неопределенными значениями параметров.

1.3. Назначение разработки

Ввести понятие неопределенного значения параметра в программный продукт РДО.

1.4. Требования к программе или программному изделию

а) Требования к функциональным характеристикам:

- обеспечение РДО возможности работать с неопределенными параметрами;
- интегрирование в РДО в каждом функциональном блоке, из которого доступны параметры системы;
- доступность из каждого интерфейса РДО (текстовый и графический);
- кросс-платформенный код ПО.

б) Требования к надежности:

- стабильная работа РДО при использовании неопределенного параметра из каждого интерфейса;
- автоматическое тестирование.

в) Условия эксплуатации:

- полностью соответствуют условиям эксплуатации РДО.

г) Требования к составу и параметрам технических средств:

- полностью соответствуют требованиям к РДО.

д) Требования к информационной и программной совместимости:

- полностью соответствуют требованиям к РДО.

1.5. Требования к программной документации

а) документация для пользователей (синтаксис, входные параметры, примеры использования);

б) документация для разработчиков РДО (комментарии к коду программы, обеспечение совместимости с автоматической системой генерации документации к исходному коду РДО).

1.6. Стадии и этапы разработки

а) добавление неопределенности к параметру ресурсов;

б) реализация поддержки неопределенного значения в компиляторе;

в) обработка неопределенных значений в процессе моделирования;

- в теле Choice_from;
- в собираемых показателях;
- в анимации;

г) разработка автоматических тестов для неопределенных параметров;

д) разработка документации для пользователей;

е) разработка документации для разработчиков РДО.

1.7. Порядок контроля и приемки

а) запуск и работа имитационной модели с использованием неопределенного параметра;

б) правильное срабатывание автоматических тестов;

2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

2.1. Компоненты РДО

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

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

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

Базовый функционал представленных на диаграмме компонентов:

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

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

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

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

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

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

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

rdo_runtime отвечает за непосредственное выполнение модели, управление базой данных, базой знаний, планирование и выполнение событий, и работу процессов. Модернизируется при разработке системы.

rdo_calc отвечает за формирование калков, универсальных блоков, из которых формируются арифметические, логические, логические цепочки.

Таким образом, основные изменения должны затронуть модули rdo_parser и rdo_runtime.

2.2. Работа с алгоритмической частью

Значения параметров РДО хранятся в виде класса RDOValue, расположенного в блоке rdo_runtime. В этом же блоке будет внедрена поддержка неопределенного значения параметра (признак достоверности). Он будет реализован в виде атрибута класса RDOValue.

2.3. Работа с интерфейсом РДО

После решения поставленной задачи в распоряжении пользователя РДО должен появиться инструмент для использования в модели неопределенного значения параметра. Этим инструментом станет новая инструкция, доступная в описании ресурсов РДО.

Таким образом, лексический и синтаксический анализаторы компонента rdo_parser должны начать правильно обрабатывать новые конструкции языка.

В описании ресурсов РДО предусмотрены следующие способы задания исходных значений ресурсов:

- значение по умолчанию;
- численная константа;
- имя значения в соответствии с типом параметра.

Описание объекта ресурса[5]:

```
$Resources
<описание_ресурса> { <описание_ресурса> }
$End
```

Синтаксис описания ресурса имеет вид:

```
<имя_ресурса> : <имя_типа_ресурса> [ <признак_трассировки> ]
<начальные_значения_параметров>
```

имя_ресурса

Имя ресурса - это простое имя. Имена должны быть различными для всех ресурсов и не должны совпадать с предопределенными и ранее использованными именами.

имя_типа_ресурса

Имя типа ресурса - это имя одного из типов ресурсов, описанных в объекте типов.

признак_трассировки

При описании ресурсов после имени типа ресурса можно указать признак трассировки (подробнее смотри описание трассировки).

начальные_значения_параметров

Начальные значения параметров ресурса задают в позиционном соответствии с порядком следования параметров в описании типа. Значения задают целой или вещественной численной константой либо именем значения в соответствии с типом параметра. Для тех параметров, у которых при описании типа указано значение по умолчанию, вместо начального значения можно указать символ "*". В этом случае параметр примет значение по умолчанию. Если для параметра задан диапазон возможных значений, то проверяется соответствие начального значения этому диапазону.

В последний блок и будут вноситься дополнения.

3. ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

3.1. Проектирование алгоритмической части

Как было предложено на концептуальном этапе проектирования, признак неопределенного значения параметра ресурса является атрибутом класса RDOValue. Для того, чтобы сохранить целостность структуры класса и безопасную работу с ним, этот атрибут будет храниться точно так же, как информация о значении и типе параметра, а именно в виде закрытого поля класса.

3.2. Проектирование интерфейсной части

3.2.1. Новые синтаксические конструкции

Тело функций РДО

На концептуальном этапе проектирования был сделан вывод о необходимости реализации новой инструкции языка РДО, функция которой будет заключаться в инициализации исходного значения ресурса как неопределенного.

В синтаксис этой инструкции должно быть включено ключевое слово, определяющее, что данный ресурс имеет неопределенное значение:

#

В остальном синтаксическая конструкция формирования объекта ресурса остается неизменной.

3.3. Проектирование связей между алгоритмической и интерфейсной частями

Связь между алгоритмической и интерфейсной частью обеспечивается с помощью калков. Это универсальные блоки передачи данных в рамках РДО. С их помощью инкапсулируются знаки арифметических операций, имена переменных, функций, ресурсов, их типы и т.д.

Графический интерфейс передаст состояние значения ресурса и его параметры в калки, и блок rdo_calc передаст их в runtime. Текстовый интерфейс сначала разберет синтаксис, после этого передаст в калки и потом в runtime.

3.4. Документация

Документация для пользователя реализована в виде справки, вызываемой из диалогового окна РДО в меню «Помощь», строка «Содержание», или нажатием клавиши F1.

В документации для пользователя представлены инструкции к использованию текстового интерфейса для инициализации неопределенного значения ресурса. Описаны параметры вызова, а также приведен пример использования неопределенного значения.

4. РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

4.1. Проектирование алгоритмической части

Признак неопределенного значения хранится в следующем поле класса RDOValue:

m_undefined

Для доступа к данному полю были реализованы 2 метода, при помощи которых можно получить информацию о текущем состоянии признака достоверности параметра ресурса, а так же изменять это состояние:

getUndefined() – получает информацию о состоянии признака достоверности;
setUndefined(rbool undefined) – устанавливает значение признака достоверности в заданное разработчиком.

4.2. Реализация поддержки неопределенного значения в компиляторе

Для внедрения новой синтаксической конструкции, которая будет инициализировать значение параметра ресурса как неопределенное, необходимо ввести в систему ключевой символ #, и обеспечить правильную обработку новой инструкции в rdo_rss.y:

```
rss_value
      : '*'                {PARSER->getLastRSSResource()->addParam(rdo::Factory<RDOValue>::create(RDOParserSrcInfo(@1, _T("*"))));}
      | '#'                {PARSER->getLastRSSResource()->addParam(rdo::Factory<RDOValue>::create(RDOParserSrcInfo(@1, _T("#"))));}
      | RDO_INT_CONST      {PARSER->getLastRSSResource()->addParam(PARSER->stack().pop<RDOValue>($1));}
      | RDO_REAL_CONST     {PARSER->getLastRSSResource()->addParam(PARSER->stack().pop<RDOValue>($1));}
      | RDO_BOOL_CONST     {PARSER->getLastRSSResource()->addParam(PARSER->stack().pop<RDOValue>($1));}
      | RDO_STRING_CONST   {PARSER->getLastRSSResource()->addParam(PARSER->stack().pop<RDOValue>($1));}
      | RDO_IDENTIF        {PARSER->getLastRSSResource()->addParam(PARSER->stack().pop<RDOValue>($1));}
      | param_array_value  {PARSER->getLastRSSResource()->addParam(PARSER->stack().pop<RDOValue>($1));}
      | error
      {
            PARSER->error().error(@1, rdo::format(_T("Неправильное значение параметра: %s"), LEXER->YYText()));
      }
      ;
```

В этом фрагменте кода производится проверка значения, введенного пользователем в качестве начального. Если введенное значение не соответствует ни одному из возможных значений для инициализации, пользователю выводится сообщение об ошибке. В любом другом случае создается объект rdoRuntime::RDOValue.

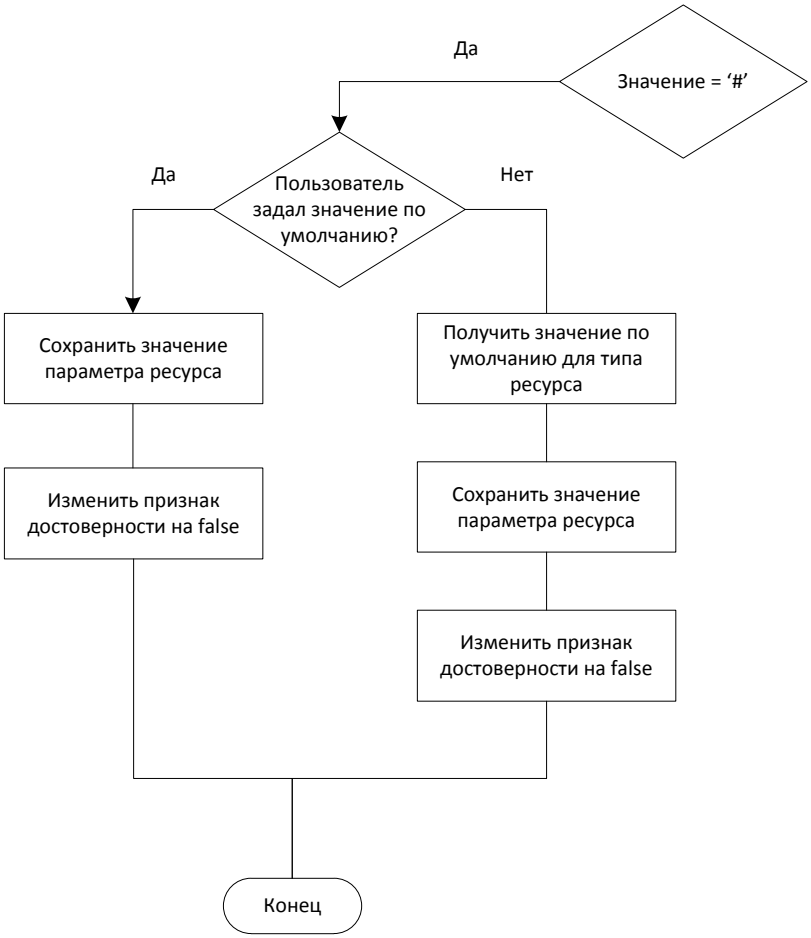
В случае, если значение было установлено пользователем, как неопределенное, необходимо не только создать объект RDOValue, но и изменить его признак неопределености. Это реализуется в файле rdo_rss.cpp в методе addParam():

```
else if (pParam->value().getAsString() == _T("#"))
	{
		LPRDOValue pValue = (*m_currParam)->getDefault()->defined()
				? (*m_currParam)->getDefault()
				:
		rdo::Factory<rdo::compiler::parser::RDOValue>::create(
			(*m_currParam)->getTypeInfo()->type()-
		>get_default(),
				(*m_currParam)->getTypeInfo()-
		>src_info(RDOParserSrcInfo()),
			(*m_currParam)->getTypeInfo()
				);
		ASSERT(pValue);

		Param param(pValue);
		param.param()->value().setUndefined(false);
		m_paramList.push_back(param);
		m_currParam++;
	}
```

В данном фрагменте кода создается объект RDOValue и задается признак неопределенности. Если пользователем задано значение по умолчанию для данного параметра ресурса, берется это значение, если нет, берется значение по умолчанию для данного типа параметра ресурса.

Фрагмент алгоритма, показывающий инициализацию неопределенного значения:

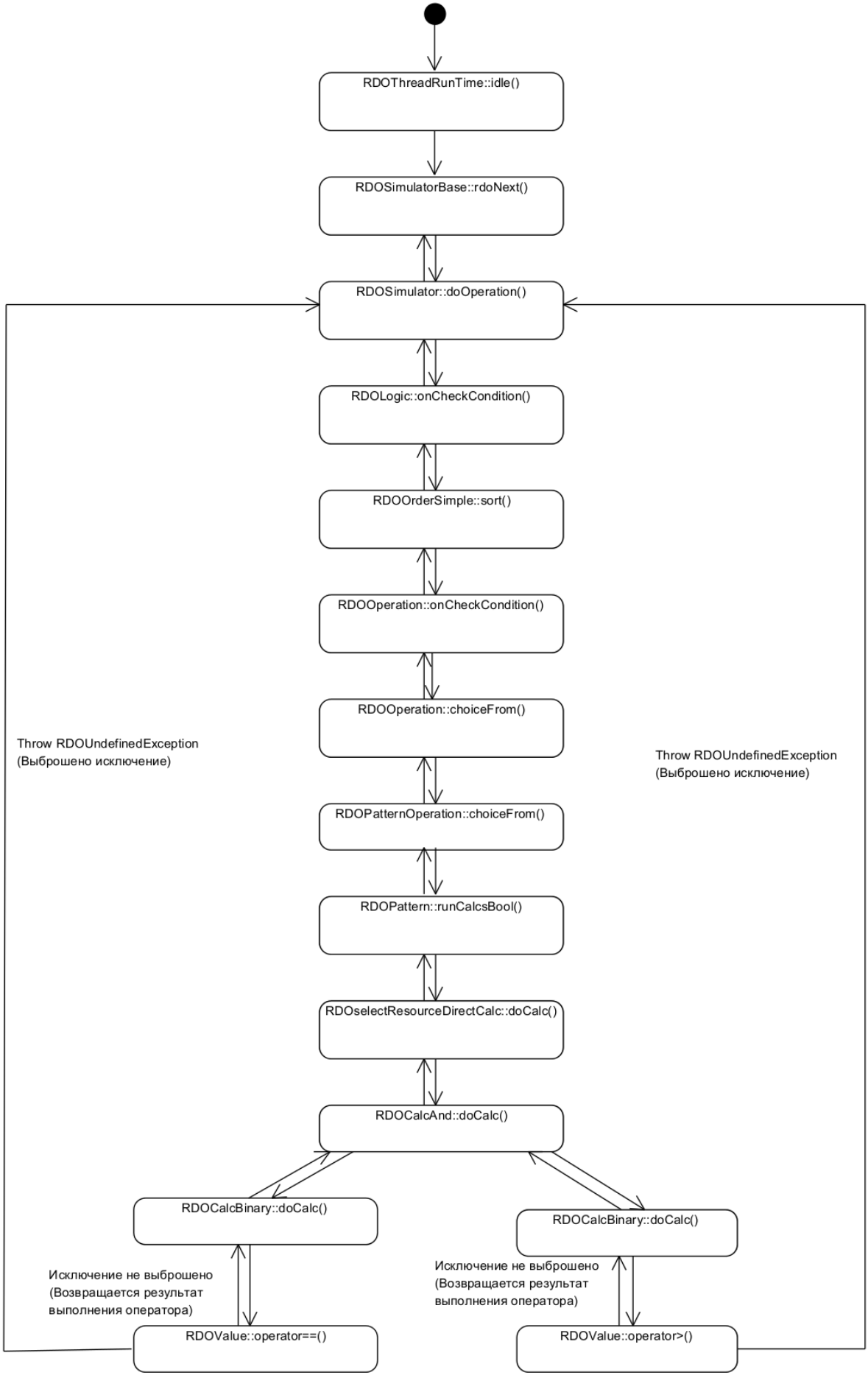


4.3. Реализация обработки неопределенного значения параметра в процессе моделирования

Для того, чтобы процесс моделирования был адекватным, необходимо правильно обрабатывать неопределенные параметры ресурса. Существует несколько мест, где они могут быть использованы:

- choice_from тела паттерна;
- собираемые показатели;
- анимация;
- тело паттерна;
- трассировка.

Для решения данной задачи было решено использовать исключения. Наглядно обработку исключений можно увидеть на примере choice_from, при любой попытке обращения к неопределенному параметру ресурса, выбрасывается исключение, которое перехватывается в симуляторе РДО и переводит флаг выполнения следующей операции в false. Наглядно это можно увидеть на диаграмме состояний:


RDOThreadRunTime::idle()
RDOSimulatorBase::rdoNext()
RDOSimulator::doOperation()
RDOLogic::onCheckCondition()
RDOOrderSimple::sort()
RDOOperation::onCheckCondition()
RDOOperation::choiceFrom()
RDOPatternOperation::choiceFrom()
RDOPattern::runCalcsBool()
RDOselectResourceDirectCalc::doCalc()
RDOCalcAnd::doCalc()
RDOCalcBinary::doCalc()
RDOCalcBinary::doCalc()
RDOValue::operator==()
RDOValue::operator>()
Throw RDOUndefinedException
(Выброшено исключение)
Throw RDOUndefinedException
(Выброшено исключение)
Исключение не выброшено
(Возвращается результат
выполнения оператора)
Исключение не выброшено
(Возвращается результат
выполнения оператора)

Как показано на диаграмме, в случае выброса исключения, обратная цепочка состояний прерывается, и информация сразу передается в RDOSimulator. При обычном ходе моделирования результаты выполнения операторов возвращаются последовательно вверх по цепочке.

4.4. Документация

Документация для разработчиков РДО генерируется с помощью автоматизированного средства получения документации к программному коду Doxygen. Эта система формирует описание классов, функций, их параметров с помощью определенным образом форматированных комментариев к коду. Представляет описание в виде структурированной системы, содержащей подробное описание кода программы, диаграммы зависимостей классов, а так же сам код программы.

5. АВТОМАТИЧЕСКИЕ ТЕСТЫ

5.1. Введение

Автоматические тесты необходимы для того, чтобы удостовериться в том, что изменения в коде не затронули логику работы методов класса RDOValue.

Кроме того, необходимо убедиться в том, что до внесения изменений в код класса, все методы работали адекватно, и дальнейшие изменения не повлияли на их работу, т.е. качественно ничего не изменилось.

5.2. Тест методов класса RDOValue

В классе RDOValue может храниться информация разных типов. Для всех этих типов определены общие методы, тем не менее, над некоторыми типами невозможно выполнить определенные операции. Более того, часть типов несовместима друг с другом.

Принимая во внимание эти факторы, были реализованы три типа тестов:

1) тесты, проверяющие работу методов в рамках параметров одного типа;
2) тесты, проверяющие работу методов в рамках совместимых типов;
3) тесты, проверяющие работу методов в рамках несовместимых типов.

В последнем случае критерием адекватности работы методов является сообщение об ошибке при попытке работы с несовместимыми типами параметров ресурсов.

Всего было произведено 26 различных тестов, в которых выполнено 280 проверок.

Процент покрытия кода автоматическими тестами составляет 87%.

ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта были получены следующие результаты:

Проведено предпроектное исследование системы ресурсов РДО и сформулированы предпосылки создания в системе инструмента для работы с неопределенными значениями параметров ресурсов.

На этапе концептуального проектирования системы выделены компоненты РДО, которые потребуют внесения изменений в ходе этой работы. Так же были обзорно рассмотрены эти компоненты.

На этапе технического проектирования разработан новый синтаксис инициализации параметров ресурсов, разработана документация для пользователя.

На этапе рабочего проектирования разработана структура автоматических тестов. Написан программный код для реализации спроектированных ранее алгоритмов работы и архитектуры компонентов rdo_parser и rdo_runtime системы РДО. Программный код является кросс-платформенным. Разработана документация для разработчиков РДО. Проведены отладка и тестирование новой системы, в ходе которых исправлялись найденные ошибки. Работоспособность системы была проверена на разработанной ранее тестовой модели-прототипе, использующей неопределенное значение параметра ресурса.

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

СПИСОК ЛИТЕРАТУРЫ

1) Кельтон В., Лоу А. Имитационное моделирование. Классика CS. 3-е изд. – Спб.: Питер: Киев: Издательская группа BHV, 2004. – 847 с.: ил.

2) Справка РДО. [В Интернете] http://rdo.rk9.bmstu.ru/forum/download/file.php?id=4

3) Справка Doxygen. [В Интернете] http://www.stack.nl/~dimitri/doxygen/commands.html

4) Емельянов В.В., Ясинский С.И. Введение в интеллектуальное моделирование сложных дискретных систем и процессов. Язык РДО. – М.: «АНВИК», 1998. – 427 с., ил. 136.

СПИСОК ПРИМЕНЯЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1. Microsoft Office 2010
2. КОМПАС – 3D V10
3. Mathcad 14
4. Microsoft Paint 6.1
5. Visual Paradigm 8.0
6. Microsoft Visual Studio 2010

Приложение 1. Программный код интерфейсной части

Часть файла rdo_rss.y

```
rss_value
      : '*'                {PARSER->getLastRSSResource()-
>addParam(rdo::Factory<RDOValue>::create(RDOParserSrcInfo(@1, _T("*"))));}
      | '#'                {PARSER->getLastRSSResource()-
>addParam(rdo::Factory<RDOValue>::create(RDOParserSrcInfo(@1, _T("#"))));}
      | RDO_INT_CONST      {PARSER->getLastRSSResource()->addParam(PARSER-
>stack().pop<RDOValue>($1));}
      | RDO_REAL_CONST     {PARSER->getLastRSSResource()->addParam(PARSER-
>stack().pop<RDOValue>($1));}
      | RDO_BOOL_CONST     {PARSER->getLastRSSResource()->addParam(PARSER-
>stack().pop<RDOValue>($1));}
      | RDO_STRING_CONST   {PARSER->getLastRSSResource()->addParam(PARSER-
>stack().pop<RDOValue>($1));}
      | RDO_IDENTIF        {PARSER->getLastRSSResource()->addParam(PARSER-
>stack().pop<RDOValue>($1));}
      | param_array_value {PARSER->getLastRSSResource()->addParam(PARSER-
>stack().pop<RDOValue>($1));}
      | error
      {
            PARSER->error().error(@1, rdo::format(_T("Неправильное значение
параметра: %s"), LEXER->YYText()));
      }
      ;
```

Часть файла rdo_rss.cpp

```
if (pParam->value().getAsString() == _T("*"))
            {
                  if (!(*m_currParam)->getDefault()->defined())
                  {
                        RDOParser::s_parser()->error().push_only(pParam->src_info(),
_T("Невозможно использовать '*', к.т. отсутствует значение по-умолчанию"));
                        /// @todo src_info() без параметра RDOParserSrcInfo()
                        RDOParser::s_parser()->error().push_only((*m_currParam)-
>getTypeInfo()->src_info(RDOParserSrcInfo()), _T("См. описание параметра"));
                        RDOParser::s_parser()->error().push_done();
                  }
                  m_paramList.push_back(Param((*m_currParam)->getDefault()));
                  m_currParam++;
            }
            else if (pParam->value().getAsString() == _T("#"))
            {
                  LPRDOValue pValue = (*m_currParam)->getDefault()->defined()
                        ? (*m_currParam)->getDefault()
                        : rdo::Factory<rdo::compiler::parser::RDOValue>::create(
                              (*m_currParam)->getTypeInfo()->type()->get_default(),
                              (*m_currParam)->getTypeInfo()-
>src_info(RDOParserSrcInfo()),
                              (*m_currParam)->getTypeInfo()
                        );
                  ASSERT(pValue);

                  Param param(pValue);
                  param.param()->value().setUndefined(false);
                  m_paramList.push_back(param);
                  m_currParam++;
            }
            else
```

```
            {
                  m_paramList.push_back(Param((*m_currParam)->getTypeInfo()-
>value_cast(pParam)));
                  m_currParam++;
            }
```

Приложение 2. Программный код алгоритмической части

Часть файла rdo_value.inl

```
inline void RDOValue::setUndefined(CREF(rbool) undefined)
{
	m_undefined = undefined;
}

inline CREF(rbool) RDOValue::getUndefined() const
{
	return m_undefined;
}
```

Часть файла rdo_exception.inl

```
// --------------------------------------------------------------------------------
// -------------------- RDOUndefinedException
// --------------------------------------------------------------------------------
inline RDOUndefinedException::RDOUndefinedException()
{}

inline tstring RDOUndefinedException::getType() const
{
	return "RDOValue Undefined Error";
}
```

Часть файла rdo_simulator

```
if (!foundPlaned)
		{
			// Не нашли запланированное событие
			// Проверить все возможные события и действия, вызвать первое,
которое может быть вызвано
			LPIBaseOperation pMetaLogic =
m_pMetaLogic.query_cast<IBaseOperation>();
			try
			{
				res = pMetaLogic->onCheckCondition(pRuntime);
			}
			catch(CREF(RDOUndefinedException))
			{
				res = false;
			}
			if (res)
			{
				res = pMetaLogic->onDoOperation(pRuntime) !=
IBaseOperation::BOR_cant_run;
			}
			if (!res)
			{
				m_checkOperation = false;
			}
		}
```

Приложение 3. Программный код автоматического теста

Часть файла main.cpp

```
/*!
  \copyright (c) RDO-Team, 2009-2012
  \file      main.cpp
  \authors   Урусов Андрей (rdo@rk9.bmstu.ru)
  \authors   Пройдаков Евгений (lord.tiran@gmail.com)
  \authors   Романов Ярослав (robot.xet@gmail.com)
  \date      13.07.2009
  \brief     Тест для RDOValue
  \indent    4T
*/

// ---------------------------------------------------------------------------- PCH
// ---------------------------------------------------------------------- INCLUDES
#include <iostream>
#include <boost/bind.hpp>
#define BOOST_TEST_MODULE RDOValue_Test
#include <boost/test/included/unit_test.hpp>
// ---------------------------------------------------------------------- SYNOPSIS
#include "simulator/runtime/rdo_runtime.h"
#include "simulator/runtime/rdo_res_type.h"
// --------------------------------------------------------------------------------

OPEN_RDO_RUNTIME_NAMESPACE

BOOST_AUTO_TEST_SUITE(RDOValue_Test)

template <class F>
void testException(F binder)
{
	rbool flag = false;
	try
	{
		binder();
	}
	catch(CREF(RDOValueException))
	{
		flag = true;
	}
	BOOST_CHECK(flag);
}

void testing(RDOValue value1, RDOValue value2)
{
	testException(boost::bind(&RDOValue::operator+=, &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator-=, &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator*=, &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator/=, &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator>,  &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator<,  &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator>=, &value1,
boost::cref(value2)));
	testException(boost::bind(&RDOValue::operator<=, &value1,
boost::cref(value2)));
```

```
	testException(boost::bind(&RDOValue::operator==, &value1,
boost::cref(value2)));
}

void compare(RDOValue value1, RDOValue value2)
{
	BOOST_CHECK(value1 != value2);
	BOOST_CHECK(value1 <  value2);
	BOOST_CHECK(value2 >  value1);
	BOOST_CHECK(value1 <= value2);
	BOOST_CHECK(value2 >= value1);
}

template <class T1>
void compareOne(CREF(T1) param1, CREF(T1) param2)
{
	const T1 val1 = param1;
	const T1 val2 = param2;
	RDOValue value1(val1);
	RDOValue value2(val2);

	BOOST_CHECK(value1 != value2);
	BOOST_CHECK(value1 >  value2);
	BOOST_CHECK(value2 <  value1);
	BOOST_CHECK(value1 >= value2);
	BOOST_CHECK(value2 <= value1);
	value2 = val1;
	BOOST_CHECK(value1 >= value2);
	BOOST_CHECK(value2 <= value1);
}

template <class T1, class T2>
void compareValue(CREF(T1) param1, CREF(T2) param2, CREF(RDOType::TypeID) type1,
CREF(RDOType::TypeID) type2)
{
	T1 val1 = param1;
	T2 val2 = param2;
	RDOValue value1(val1);
	RDOValue value2(val2);

	compare(value1, value2);

	value1 = value2;
	BOOST_CHECK(value1 == value2);

	value1 = val1;
	value2 = val2;
	value1 += value2;
	BOOST_CHECK(value1          == T1(param1 + param2));
	BOOST_CHECK(value1.typeID() == type1);
	value1 -= value2;
	BOOST_CHECK(value1 == param1);
	value1 *= value2;
	BOOST_CHECK(value1 == param1 * param2);
	value1 /= value2;
	BOOST_CHECK(value1 == param1);

	BOOST_CHECK(value1.typeID() == type2);
}

template <class T1, class T2>
void compareStr(CREF(T1) param1, CREF(T2) param2)
{
	T1 val1 = param1;
	T2 str1 = param2;
```

```
	RDOValue value1(val1);
	RDOValue value2(str1);

	testing(value1, value2);
}

template <class T1, class T2>
void compareChr(CREF(T1) param1, CREF(T2) param2)
{
	T1  val1 = param1;
	T2  ch1  = param2;
	RDOValue value1(val1);
	RDOValue value2(ch1 );

	value1 += value2;
	BOOST_CHECK(value1 == val1 + ch1);
	value1 -= value2;
	BOOST_CHECK(value1 == val1      );
	value1 *= value2;
	BOOST_CHECK(value1 == val1 * ch1);
	value1 /= value2;
	BOOST_CHECK(value1 == val1      );

	compare(value1, value2);
}

BOOST_AUTO_TEST_CASE(RDOValue_String)
{
	const tstring str1 = _T("qqq");
	RDOValue value1(str1);
	BOOST_CHECK(value1.getString  () == str1);
	BOOST_CHECK(value1.getAsString() == str1);

	RDOValue value2 = value1;
	BOOST_CHECK(value2.getString  () == str1);
	BOOST_CHECK(value2.getAsString() == str1);
	BOOST_CHECK(value2 == value1);

	const tstring str2 = _T("aaa");
	value2 += str2;
	BOOST_CHECK(value2.getString  () == str1 + str2);
	BOOST_CHECK(value2.getAsString() == str1 + str2);
}

BOOST_AUTO_TEST_CASE(RDOValue_Rsint_Arifmethic)
{
	const rsint val1 = 30;
	RDOValue value1(val1);
	BOOST_CHECK(value1.getInt      () == val1);
	BOOST_CHECK(value1.getUInt     () == val1);
	BOOST_CHECK(value1.getDouble   () == val1);
	BOOST_CHECK(value1.getAsString () == "30");
	BOOST_CHECK(value1.getEnumAsInt() == 30  );

	RDOValue value2 = value1;
	BOOST_CHECK(value2 == val1  );
	BOOST_CHECK(value2 == value1);

	const rsint val2 = 20;
	value1 += val2;
	BOOST_CHECK(value1 == val1 + val2);
	value1 = val1;
	value1 = value1 + val2;
	BOOST_CHECK(value1 == val1 + val2);
```

```
	const rsint val3 = 10;
	value1 -= val3;
	BOOST_CHECK(value1 == val1 + val2 - val3);
	value1 += val3;
	value1 = value1 - val3;
	BOOST_CHECK(value1 == val1 + val2 - val3);

	const rsint val4 = 2;
	value2 = value1;
	value2 *= val4;
	BOOST_CHECK(value2 == (val1 + val2 - val3) * val4);
	value2 /= val4;
	value2 = value2 * val4;
	BOOST_CHECK(value2 == (val1 + val2 - val3) * val4);

	const rsint val5 = 5;
	value2 = value1;
	value2 /= val5;
	BOOST_CHECK(value2 == ((val1 + val2 - val3) / val5));
	value2 *= val5;
	value2 = value2 / val5;
	BOOST_CHECK(value2 == ((val1 + val2 - val3) / val5));

	value2 = value1;
	value2 --;
	BOOST_CHECK(value2 == value1 - 1);
	value2 ++;
	BOOST_CHECK(value2 == value1);
}

BOOST_AUTO_TEST_CASE(RDOValue_Rsint_Compare)
{
	compareOne<rsint>(30, 20);
}


BOOST_AUTO_TEST_CASE(RDOValue_Ruint_Arifmethic)
{
	const ruint val1 = 30;
	RDOValue value1(val1);
	BOOST_CHECK(value1 == val1);
	BOOST_CHECK(value1.getInt      () == val1);
	BOOST_CHECK(value1.getUInt     () == val1);
	BOOST_CHECK(value1.getDouble   () == val1);
	BOOST_CHECK(value1.getAsString () == "30");
	BOOST_CHECK(value1.getEnumAsInt() == 30  );

	RDOValue value2 = value1;
	BOOST_CHECK(value2 == val1  );
	BOOST_CHECK(value2 == value1);

	const ruint val2 = 20;
	value1 += val2;
	BOOST_CHECK(value1 == val1 + val2);
	value1 = val1;
	value1 = value1 + val2;
	BOOST_CHECK(value1 == val1 + val2);

	const ruint val3 = 10;
	value1 -= val3;
	BOOST_CHECK(value1 == val1 + val2 - val3);
	value1 += val3;
	value1 = value1 - val3;
	BOOST_CHECK(value1 == val1 + val2 - val3);
```

```
	const ruint val4 = 2;
	value2 = value1;
	value2 *= val4;
	BOOST_CHECK(value2 == (val1 + val2 - val3) * val4);
	value2 /= val4;
	value2 = value2 * val4;
	BOOST_CHECK(value2 == (val1 + val2 - val3) * val4);

	const ruint val5 = 5;
	value2 = value1;
	value2 /= val5;
	BOOST_CHECK(value2 == ((val1 + val2 - val3) / val5));
	value2 *= val5;
	value2 = value2 / val5;
	BOOST_CHECK(value2 == ((val1 + val2 - val3) / val5));

	value2 = value1;
	value2 --;
	BOOST_CHECK(value2 == value1 - 1);
	value2 ++;
	BOOST_CHECK(value2 == value1);
}

BOOST_AUTO_TEST_CASE(RDOValue_Ruint_Compare)
{
	compareOne<ruint>(30, 20);
}

BOOST_AUTO_TEST_CASE(RDOValue_Double_Arifmethic)
{
	const double doub1 = 30.2;
	RDOValue value1(doub1);
	BOOST_CHECK(value1 == doub1);
	BOOST_CHECK(value1.getInt      () == 30    );
	BOOST_CHECK(value1.getUInt     () == 30    );
	BOOST_CHECK(value1.getDouble   () == doub1 );
	BOOST_CHECK(value1.getAsString () == "30.2");
	BOOST_CHECK(value1.getEnumAsInt() == 30    );

	RDOValue value2 = value1;
	BOOST_CHECK(value2 == doub1);
	BOOST_CHECK(value2 == value1);

	const double doub2 = 20.5;
	value1 += doub2;
	BOOST_CHECK(value1 == doub1 + doub2);
	value1 = doub1;
	value1 = value1 + doub2;
	BOOST_CHECK(value1 == doub1 + doub2);

	const double doub3 = 10.3;
	value1 -= doub3;
	BOOST_CHECK(value1 == doub1 + doub2 - doub3);
	value1 += doub3;
	value1 = value1 - doub3;
	BOOST_CHECK(value1 == doub1 + doub2 - doub3);

	const double doub4 = 2.5;
	value2 = value1;
	value2 *= doub4;
	BOOST_CHECK(value2 == (doub1 + doub2 - doub3) * doub4);
	value2 /= doub4;
	value2 = value2 * doub4;
	BOOST_CHECK(value2 == (doub1 + doub2 - doub3) * doub4);
```

```
	const double doub5 = 5;
	value2 = value1;
	value2 /= doub5;
	BOOST_CHECK(value2 == ((doub1 + doub2 - doub3) / doub5));
	value2 *= doub5;
	value2 = value2 / doub5;
	BOOST_CHECK(value2 == ((doub1 + doub2 - doub3) / doub5));

	value2 = value1;
	value2 --;
	BOOST_CHECK(value2 == value1 - 1);
	value2 ++;
	BOOST_CHECK(value2 == value1);
}

BOOST_AUTO_TEST_CASE(RDOValue_Double_Compare)
{
	compareOne<double>(30.2, 20.5);
}

BOOST_AUTO_TEST_CASE(RDOValue_Bool)
{
	rbool bool1 = true;
	rbool bool2 = false;
	rbool bool3 = true;
	rbool bool4 = false;

	RDOValue value1(bool1);
	RDOValue value2(bool2);
	RDOValue value3(bool3);
	RDOValue value4(bool4);

	BOOST_CHECK(value1.getBool     ()     );
	BOOST_CHECK(value1.getInt      () == 1);
	BOOST_CHECK(value1.getUInt     () == 1);
	BOOST_CHECK(value1.getEnumAsInt() == 1);
	BOOST_CHECK(value1.getDouble   () == 1);
	BOOST_CHECK(!value2);
	BOOST_CHECK(  value2 == value4 );
	BOOST_CHECK(  value1 != value2 );
	BOOST_CHECK(!(value1 == value2));
	BOOST_CHECK(  value1 && value3 );
	BOOST_CHECK(!(value2 && value3));
	BOOST_CHECK(  value1 || value2 );
	BOOST_CHECK(!(value2 || value4));
}

BOOST_AUTO_TEST_CASE(RDOValue_Char)
{
	tchar ch1 = 'a';
	tchar ch2 = 'b';
	RDOValue value1 = ch1;
	RDOValue value2 = ch2;
	BOOST_CHECK(value1 == ch1   );
	BOOST_CHECK(value1 <  value2);
	BOOST_CHECK(value2 >  value1);
	BOOST_CHECK(value1 != value2);
	value1 = value2;
	BOOST_CHECK(value1 == value2);
}

BOOST_AUTO_TEST_CASE(RDOValue_Enum)
{
	LPRDOEnumType pEnum = rdo::Factory<RDOEnumType>::create();
	BOOST_CHECK(pEnum);
```

```
	BOOST_CHECK(pEnum->empty());
	pEnum->add(_T("test0"));
	pEnum->add(_T("test1"));
	pEnum->add(_T("test2"));
	pEnum->add(_T("test3"));
	BOOST_CHECK(pEnum->findEnum(_T("test1")) == 1);
	BOOST_CHECK(pEnum->exist   (_T("test3"))     );

	RDOValue value(pEnum);
	BOOST_CHECK(value.typeID      () == RDOType::t_enum);
	BOOST_CHECK(value.getEnum     () == pEnum          );
	BOOST_CHECK(value.getInt      () == 0              );
	BOOST_CHECK(value.getDouble   () == 0              );
	BOOST_CHECK(value.getEnumAsInt() == 0              );
	BOOST_CHECK(!value.getAsBool  ()                   );
	BOOST_CHECK(value.getAsString () == "test0"        );

	RDOValue value1(pEnum, "test2");
	BOOST_CHECK(value1.typeID      () == RDOType::t_enum);
	BOOST_CHECK(value1.getEnum     () == pEnum          );
	BOOST_CHECK(value1.getInt      () == 2              );
	BOOST_CHECK(value1.getDouble   () == 2              );
	BOOST_CHECK(value1.getEnumAsInt() == 2              );
	BOOST_CHECK(value1.getAsBool   ()                   );
	BOOST_CHECK(value1.getAsString () == "test2"        );

	RDOValue value2(pEnum, 2);
	BOOST_CHECK(value2.typeID      () == RDOType::t_enum);
	BOOST_CHECK(value2.getEnum     () == pEnum          );
	BOOST_CHECK(value2.getInt      () == 2              );
	BOOST_CHECK(value2.getDouble   () == 2              );
	BOOST_CHECK(value2.getEnumAsInt() == 2              );
	BOOST_CHECK(value2.getAsBool   ()                   );
	BOOST_CHECK(value2.getAsString () == "test2"        );

}

BOOST_AUTO_TEST_CASE(RDOValue_Rsint_Ruint)
{
	compareValue<rsint, ruint>(10, 15, RDOType::t_int, RDOType::t_real);
}

BOOST_AUTO_TEST_CASE(RDOValue_Ruint_Rsint)
{
	compareValue<ruint, rsint>(10, 15, RDOType::t_int, RDOType::t_real);
}

BOOST_AUTO_TEST_CASE(RDOValue_Rsint_Double)
{
	compareValue<rsint, double>(10, 15.2, RDOType::t_int, RDOType::t_real);
}

BOOST_AUTO_TEST_CASE(RDOValue_Double_rsint)
{
	compareValue<double, rsint>(10.2, 15, RDOType::t_real, RDOType::t_real);
}

BOOST_AUTO_TEST_CASE(RDOValue_Ruint_Double)
{
	compareValue<ruint, double>(10, 15.2, RDOType::t_int, RDOType::t_real);
}

BOOST_AUTO_TEST_CASE(RDOValue_Double_ruint)
{
	compareValue<double, ruint>(10.2, 15, RDOType::t_real, RDOType::t_real);
```

```
}

BOOST_AUTO_TEST_CASE(RDOValue_Rsint_String)
{
	compareStr<rsint, tstring>(10, _T("abc"));
}

BOOST_AUTO_TEST_CASE(RDOValue_Ruint_String)
{
	compareStr<ruint, tstring>(10, _T("abc"));
}

BOOST_AUTO_TEST_CASE(RDOValue_Double_String)
{
	compareStr<double, tstring>(10, _T("abc"));
}

BOOST_AUTO_TEST_CASE(RDOValue_Char_Rsint)
{
	compareChr<rsint, tchar>(10, _T('a'));
}

BOOST_AUTO_TEST_CASE(RDOValue_Char_Ruint)
{
	compareChr<ruint, tchar>(10, _T('a'));
}

BOOST_AUTO_TEST_CASE(RDOValue_Char_Double)
{
	compareChr<double, tchar>(10, _T('a'));
}

BOOST_AUTO_TEST_CASE(RDOValue_String_Char)
{
	compareStr<tchar, tstring>(_T('a'), _T("abc"));
}

BOOST_AUTO_TEST_CASE(RDOValue_Identificator)
{
	tstring str = _T("abc");
	RDOValue value1(str, g_identificator);
	BOOST_CHECK(value1.typeID() == RDOType::t_identificator);
	tstring iden = value1.getIdentificator();
	BOOST_CHECK(iden == str);
	iden = value1.getAsString();
	BOOST_CHECK(iden == str);

	tstring str2 =_T("dba");
	RDOValue value2(str2, g_identificator);
	BOOST_CHECK(value1 != value2);

	RDOValue value3(str, g_identificator);
	BOOST_CHECK(value1 == value3);
}

template <class T1>
void testUndef(CREF(T1) param1)
{
	T1 val = param1;
	RDOValue value(val);
	BOOST_CHECK(value.getUndefined() == true);
}

BOOST_AUTO_TEST_CASE(RDOValue_Undefined)
{
```

```
	testUndef<rsint>  (10       );
	testUndef<ruint>  (10       );
	testUndef<double> (10.5     );
	testUndef<tstring>(_T("abc"));
	testUndef<tchar>  (_T('a')  );
	testUndef<rbool>  (true     );

	rsint val1 = 10;
	RDOValue value1(val1);
	BOOST_CHECK(value1);

	BOOST_CHECK(value1.getUndefined() == true);
	value1.setUndefined(false);
	BOOST_CHECK(value1.getUndefined() == false);

	RDOValue value2(value1);
	BOOST_CHECK(value2.getUndefined() == false);
}

BOOST_AUTO_TEST_CASE(RDOValue_Resource)
{
	LPRDORuntime pRuntime = rdo::Factory<RDORuntime>::create();
	BOOST_CHECK(pRuntime);

	LPRDOResourceType pResourceType = rdo::Factory<RDOResourceType>::create(0);
	BOOST_CHECK(pResourceType);
	LPIResourceType pResourceFactory =
pResourceType.interface_cast<IResourceType>();
	BOOST_CHECK(pResourceFactory);

	std::vector<RDOValue> paramList;
	paramList.push_back(RDOValue(1      ));
	paramList.push_back(RDOValue(2.2    ));
	paramList.push_back(RDOValue(_T("3")));

	LPRDOResource pResource = pResourceFactory->createRes(pRuntime, pRuntime-
>getResourceId(), paramList, true, true);
	BOOST_CHECK(pResource);

	RDOValue value1(pResourceType, pResource);

	LPRDOResourceType pType = value1.type().object_dynamic_cast<RDOResourceType>();
	BOOST_CHECK(pType);

	LPRDOResource pResource1 = value1.getPointerSafety<RDOResourceType>();

	pRuntime   = NULL;
	value1     = RDOValue();
	pResource1 = NULL;

	BOOST_CHECK(pResource.owner());

	pResource = NULL;
}

BOOST_AUTO_TEST_SUITE_END() // RDOValue_Test

CLOSE_RDO_RUNTIME_NAMESPACE
```

Приложение 4. Программный код автоматического теста

```
Вкладка RTP

$Resource_type Парикмахерские: permanent
$Parameters
    состояние_парикмахера : ( Свободен, Занят)
    количество_в_очереди  : integer
    количество_обслуженных: integer
$End
Вкладка RSS

$Resources
    Парикмахерская: Парикмахерские trace # 0 0
$End
Вкладка EVN

$Pattern Образец_прихода_клиента : event
$Relevant_resources
    _Парикмахерская: Парикмахерская Keep
$Body
_Парикмахерская
    Convert_event
        Образец_прихода_клиента.planning( time_now +
Интервал_прихода( 30 ) );
        количество_в_очереди++;
$End

$Pattern Образец_прихода_парикмахера : event
$Relevant_resources
    _Парикмахерская: Парикмахерская Keep
$Body
_Парикмахерская
    Convert_event
        состояние_парикмахера = Свободен;
$End

Вкладка PAT
```

```
$Pattern Образец_обслуживания_клиента : operation
$Relevant_resources
     _Парикмахерская: Парикмахерская Keep Keep
$Time = Длительность_обслуживания( 20, 40 )
$Body
_Парикмахерская
     Choice from _Парикмахерская.состояние_парикмахера ==
Свободен and _Парикмахерская.количество_в_очереди > 0
     Convert_begin
          количество_в_очереди--;
          состояние_парикмахера = Занят;
     Convert_end
          состояние_парикмахера  = Свободен;
          количество_обслуженных++;
$End

Вкладка SMR

Show_mode       = Animation
Show_rate       = 3600.0

Образец_прихода_клиента.planning( time_now +
Интервал_прихода( 30 ) )
Образец_прихода_парикмахера.planning( 100 )

Terminate_if Time_now >= 12 * 7 * 60
```