

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский университет

«Московский институт электронной техники»

Институт СПИНТех

Карасева Веслава Эдуардовна

Бакалаврская работа

по направлению 09.03.04 «Программная инженерия»

Разработка программного модуля для автоматизации процессов кредитования  
розничного банка.

ПМ АВПК

Студент

\_\_\_\_\_

Карасева В.Э.

Руководитель,

к.п.н., доцент

\_\_\_\_\_

Федотова Е.Л.

Москва 2020

## Contents

СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ .....	3
ВВЕДЕНИЕ .....	4
ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ .....	6
Анализ предметной области .....	6
1.1 Рынок розничных банковских услуг .....	6
1.2 Описание процессов рассмотрения кредитной заявки .....	8
1.4 Описание инструмента SAS Real-Time Decision Manager .....	13
1.5 Описание инструмента Credit Registry Enterprise .....	18
Выводы по разделу 1: .....	20
КОНСТРУКТОРСКИЙ РАЗДЕЛ .....	21
Автоматизация процессов кредитования на основе системы SAS RTDM ....	21
2.1 Общая схема процесса принятия решения .....	21
2.2 Ограничения типовой реализации системы принятия решений средствами SAS RTDM .....	24
2.3 Реализации СПР средствами SAS с учетом выявленных недостатков типовой схемы .....	26
Выводы по разделу 2: .....	42
ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ .....	43
Описание методов тестирования и обоснование целесообразности применения предлагаемых решений .....	43
3.1 Методы тестирования интеграционных приложений .....	43
3.2 Пример тестирования в SOAP UI .....	47
3.3 Оценка экономической эффективности инвестиций. ....	49
Выводы: .....	60
ЗАКЛЮЧЕНИЕ .....	62
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	64
Приложение 1. ....	<b>Error! Bookmark not defined.</b>
Приложение 2. ....	74
Программный код интерфейса и имплементора интеграционного веб-сервиса с сервисами БКИ .....	<b>Error! Bookmark not defined.</b>
Программный код интерфейса и имплементора интеграционного веб-сервиса фронт-офисной системы с SAS RTDM .....	<b>Error! Bookmark not defined.</b>

## СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

RTDM – автоматизированная система принятия решений SAS Real-Time Decision Manager

СПР – система принятия решений

DS2 – язык разработки компании SAS

BRMS – Business Rules Management System – информационная система управления бизнес правилами

SAS BRM – информационная система управления бизнес правилами компании SAS

CRE – Credit Registry – веб-сервис, предоставляющий данные о кредитной истории из внешних источников

ИТ – информационные технологии

БКИ – бюро кредитных историй

КИ – кредитная история

ОКБ – объединенное кредитное бюро

ВБКИ – внутреннее бюро кредитных историй

НБКИ – национальное бюро кредитных историй

## ВВЕДЕНИЕ

Одной из самых важных функций розничных банков является кредитование. Именно поэтому банки заинтересованы в усовершенствовании и модернизации этого процесса. Основными задачами для банка являются увеличение количества привлеченных клиентов, а также снижение потерь и избежание закредитованности.

Для того, чтобы решить поставленные перед банком задачи, он ищет способы автоматизаций этапов жизненного цикла заявки, с целью уменьшения времени на принятия решения по ней. В неавтоматизированном принятии решений велик риск ошибок, связанных с ручной обработкой заявок, где надо делать ставку на человеческий фактор. Неправильные действия банковских сотрудников, отвечающих за конкретную область жизненного цикла заявки, могут привести к цепочке ошибок.

В наше время почти во всех сферах жизни общества автоматизация играет огромную роль. Банковское дело не является исключением. В связи с этим тема бакалаврской выпускной квалификационной работы является актуальной. Замена интеллектуального труда человека машинным, рациональное и разумное распределение функций между человеком и компьютерной системой являются причинами повышения эффективности и скорости работы, а также уменьшения ошибок, вызванных человеческим фактором

Целью данной работы является повышение эффективности процессов кредитования физических лиц за счет разработки программного модуля.

### **Задачи:**

- исследование предметной области;
- выбор платформы для реализации системы принятия решений;
- выбор языка и среды программирования;
- разработка технической схемы реализации системы принятия решений по кредитным заявкам;

- разработка экранных форм пользовательского интерфейса;
- реализация программного модуля;
- тестирование и отладка предлагаемых программных решений;
- разработка руководства оператора.

Объектом исследования являются технологии основных процессов кредитования физических лиц крупного банка.

Предметом исследования являются технологии оптимизации и автоматизации процессов кредитования, которые основываются на технологиях компании SAS Institute.

В результате проведенной работы были решены задачи:

- определены ограничения типовой схемы реализации системы принятия решений;
- разработана техническая схема реализации системы принятия решений по кредитным заявкам, основанная на применении программных решений SAS Institute.
- разработан новый интеграционный слой, который позволил ускорить процесс обработки заявки.

Полученные результаты имеют высокую практическую значимость. Разработанный программный модуль может быть использован не только в конкретном розничном банке, а в целом для нужд департаментов различных розничных кредитных рисков и клиентской аналитики.

## ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

### Анализ предметной области.

#### 1.1 Рынок розничных банковских услуг.

Для того, чтобы автоматизировать процессы кредитования, необходимо понимать модели и процессы взаимодействия между банком и клиентом в области банковских услуг.

Федеральный закон «О банках и банковской деятельности» гласит: банк – это кредитная организация, которая имеет исключительное право на осуществление следующих операций: привлечение во вклады денежных средств физических и юридических лиц, размещение средств от своего имени и за свой счет на условиях возвратности, платности, срочности, открытие и ведение банковских счетов физических и юридических лиц [13]. Для классификации банков используют такие параметры как: тип собственности, характер деятельности, потребитель услуги, функциональное назначение, типы выполняемых операций и другое [4]. Классификация банков в зависимости от ключевых параметров представлена в таблице 1.1.

Таблица 1.1 - Классификация банков

Параметры классификации	Виды
Потребитель услуги	Физические лица
	Коммерческие предприятия
	Общественные организации
	Финансовые институты
Функциональное назначение	Венчурные банки
	Ссудо-сберегательный банк
	Ипотечные банки
	Депозитные банки

Продолжение таблицы 2.1 - Классификация банков

Функциональное назначение	Коммерческий банк
	Инвестиционный банк
Тип собственности	Государственный
	Частный
Характер деятельности	Универсальный
	Специализированный

В мировой и российской практике банковскую сферу деятельности делят на корпоративную и розничную.

Розничный банк – это специализированный банк, вид деятельности которого заключается в массовой продаже стандартизированных розничных банковских услуг физическим лицам.

Розничные банковские услуги делятся на [2]:

- кредитные карты;
- срочные вклады;
- денежные переводы;
- розничные платежи по товарам и услугам;
- вклады до востребования денежных средств;
- дебетовые карты;
- кредиты наличными;
- POS-кредитование.

В условиях кризиса банки сталкиваются с тем, что отделения становятся нерентабельными, поэтому необходимо сокращать расходы как на содержание офисов, так и на персонал и на прочее. Это приводит к повышению операционной эффективности, которая полностью зависит от уменьшения кредитного риска и от качества выполнения банковских операций. Самый распространенный риск в банковском секторе – кредитный.

Кредитный риск – это основной риск, возникающий в процессе деятельности банков[3]. Различают внутренние и внешние факторы кредитного риска (таблица 1.2).

Таблица 1.3 - Виды кредитного риска

Вид кредитного риска	Внутренние факторы кредитного риска	Внешние факторы кредитного риска
Риск индивидуального заёмщика	Ошибки персонала, связанные с нарушением должностных инструкций; Злоупотребление персонала.	Отказ заёмщика от выполнения обязательств по кредитному договору

Кредитный риск появляется, когда заемщик неспособен или не желает выполнять условия кредитного договора. Возникает риск при совершении кредитных операций [1].

## 1.2 Описание процессов рассмотрения кредитной заявки.

Объектом исследования работы являются процессы кредитования физических лиц российского розничного банка, клиентская база которого на конец 2019 года насчитывала более 5 миллионов лиц.

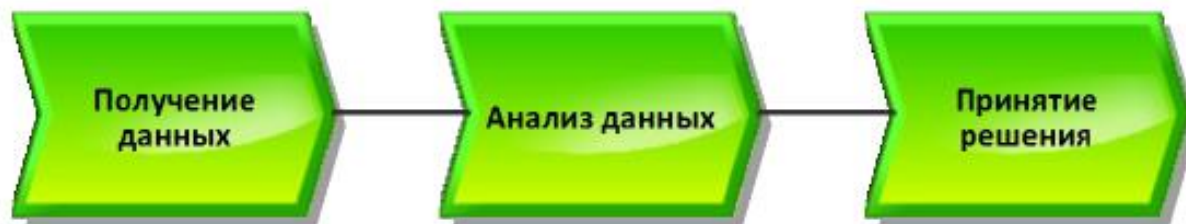
Банк занимается следующими типами услуг [5]:

- Кредитные карты;
- Кредит наличными и потребительские кредиты;
- Ипотечное кредитование;
- Автокредиты;
- Дебетовые карты;
- Вклады.

Абстрактная схема процесса кредитования изображена на рисунке 1.1.



Рисунок 1.1 - Схема кредитования



Первый шаг – это получение данных о клиенте [6]. Основные источники данных о заёмщике:

- Данные анкеты;
- Данные внутренних систем
  - фронт-офисная система;
  - black lists – черные списки клиентов;
  - прочие;
- Данные внешних систем
  - бюро кредитных историй (БКИ);
  - сервисы противодействия мошенничеству;
  - прочие;

Анкетные данные заполняются во Фронт-офисной системе вручную кредитным специалистом. Процесс получения данных из внутренних и внешних систем реализован в автоматическом режиме.

Анализ данных состоит из следующих компонентов [7]:

- Анализ данных о клиенте на предмет наличия стоп-факторов;
- Анализ кредитной истории участников кредитной заявки;
- Расчет скорингового балла;
- Расчет лимитных ограничений;
- Анализ рискованных правил;

В таблице 1.3 рассмотрены этапы, представлены характеристики этапов кредитной заявки.

Таблица 1.4 - Характеристики этапов рассмотрения заявки

Шаг процесса	Этап	Режим обработки	Система	Временные затраты	Вероятность отклонения от инструкции
Получение данных о клиенте	Заведение анкетных данных в систему	В ручном режиме	Фронт-офис	Средние	Средняя
	Получение данных из внутренних источников	В автоматическом режиме	Фронт-офис	Низкие	Низкая
	Получение данных из внешних источников	В автоматическом режиме	Фронт-офис	Низкие	Низкая
Анализ данных	Проверка на наличие стоп-факторов	В ручном режиме	-	Средние	Средняя

Продолжение таблицы 1.5 - Характеристики этапов рассмотрения заявки

Анализ данных	Расчет скорингового балла	В ручном режиме	Excel	Средние	Средняя
	Расчет лимитных ограничений	В ручном режиме	Excel	Средние	Средняя
	Анализ кредитной истории	В ручном режиме	-	Высокие	Высокая
	Анализ рисков правил	В ручном режиме	-	Высокие	Высокая
Принятие решения	Маршрутизация на отказ	В ручном режиме	-	Средние	Низкая
	Маршрутизация на дополнительную проверку	В ручном режиме	-	Средние	Низкая
	Маршрутизация на одобрение	В ручном режиме	-	Средние	Низкая

В банке проводился опрос, в результате которого выяснилось, что время рассмотрения заявки может достигать 4 рабочих дня, а анализ кредитной истории участников кредитной заявки и анализ риск-правил сильнее всего подвержены кредитному риску, который следует из ошибки сотрудников банка.

Помимо оптимизационных задач банку требовался инструмент, позволяющий быстро изменять процессы анализа кредитной заявки, что позволило бы оперативно реагировать на изменения требований бизнес задач.

Для решения данных проблем, выявленных на этапе исследования текущей реализации кредитного процесса, было принято решение о внедрении автоматизированной системы принятия решений.

Наиболее распространенные платформы для реализации системы принятия решений на отечественном рынке следующие:

- Deductor;
- SAS Real-Time Decision Manager;
- Teradata Real-Time Interaction Manager;
- SAP RTOM.

Выбор платформы для реализации системы принятия решений производился на основе метода экспертных оценок.

Независимыми экспертами были оценены вышеперечисленные программные продукты. Наиболее значимые критерии были выбраны Банком. Критерии для оценки следующие:

- стоимость лицензии, консалтинговых и услуг технической поддержки;
- быстродействие;
- наличие навыков работы с платформой у сотрудников Банка.

Оценки независимых экспертов приведены в таблице 1.4. Значения оценок находятся в диапазоне от нуля до единицы.

Таблица 1.6 - Оценки независимых экспертов

№	Критерий	Deductor	SAS RTDM	Teradata RTIM	SAP RTOM
K1	стоимость лицензии, консалтинговых и услуг технической поддержки	0.5	0.4	0.6	0.3

Продолжение таблицы 1.7 - Оценки независимых экспертов

К2	время обработки кредитной заявки	0.6	0.9	0.5	0.5
К3	наличие навыков работы с платформой у сотрудников Банка	0.6	0.9	0.3	0.7

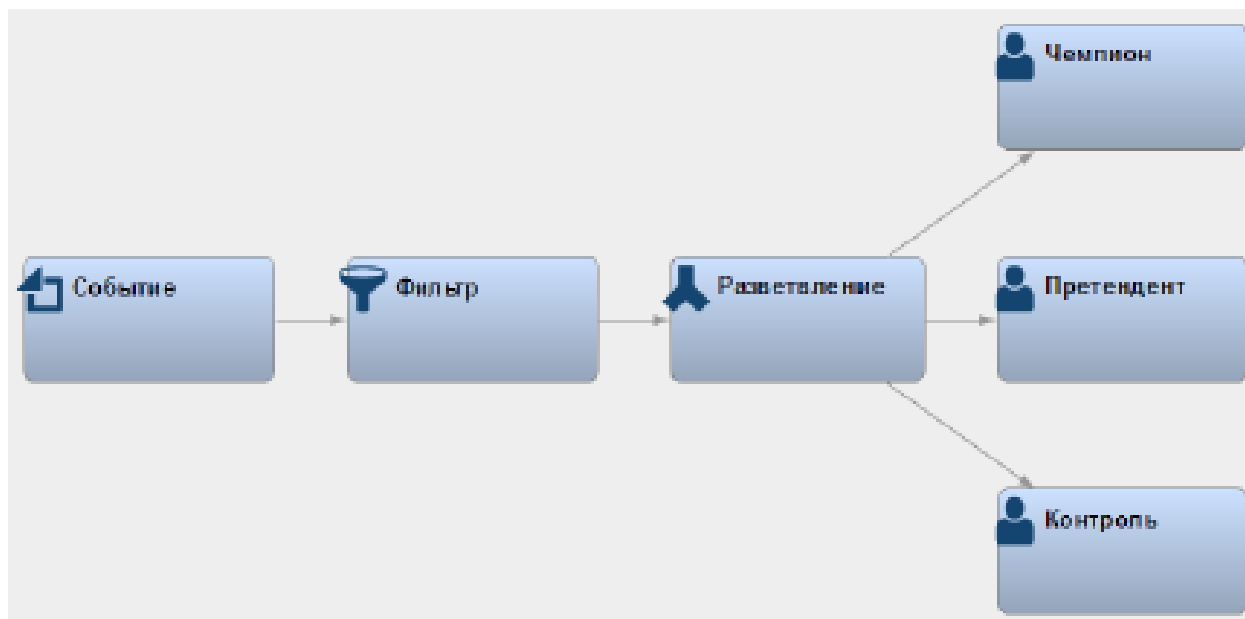
Продукт SAS Real-Time Decision Manager уступает остальным продуктам только в стоимости лицензии, по остальным критериям он превзошел конкурентов. На основании данных оценок, в банке было принято решение о внедрении системы принятия для решения задач по оптимизации и автоматизации процессов рассмотрения кредитной заявки на базе программного продукта SAS RTDM.

#### 1.4 Описание инструмента SAS Real-Time Decision Manager.

Система принятия решений SAS Real-Time Decision Manager позволяет реализовать процесс принятия решений по кредитной заявке, преобразовать процесс в исполняемый код и опубликовать на сервере приложений в виде веб-сервиса.

За разработку бизнес-логики процесса принятия решения отвечает Риск-технолог. Разработка выполняется с помощью диаграммы, которая строится способом drag-and-drop. Бизнес-пользователю доступен широкий ассортимент инструментов, которые реализованы в виде узлов диаграммы. Пример простейшей диаграммы изображен на рисунке 1.3.




Рисунок 1.3 - Простая диаграмма в SAS RTDM



Пользователи SAS RTDM могут создавать процессы принятия решений разного уровня сложности. Например, веб-процессы, которые будут вызывать определенные методы веб-процессов и получать результат от него.

Список основных узлов, используемых для создания и управления кампаниями в решении SAS Real-Time Decision Manager, приведен в таблице 1.5.

Таблица 1.5 - Список доступных узлов построения процесса принятия решений

Наименование узла	Описание функционала узла
 Ветвь	Узел позволяет разделить клиентов на макросегменты по процентному соотношению или по переменной из витрины данных
 Бизнес-правила	Узел задает специальные бизнес-правила на входные данные
 Ячейка	Узел позволяет создать временный список клиентов, который в дальнейшем можно использовать для ответа

Продолжение таблицы 1.5 - Список доступных узлов построения процесса принятия решений

 Перекрестная таблица	Узел позволяет построить таблицу пересечения одной/двух переменных для получения нового значения
 Фильтр	Узел фильтрует входные данные по одной или нескольким переменным
 Процесс	Узел позволяет написать код на языке SAS и выполнить его при вызове кампании
 ответ	Узел, определяющий передаваемую информацию клиентскому приложению
 Балл	Узел позволяет добавить модель расчета скоринговых баллов для кампании
 Начало	Является обязательным и единственным узлом диаграммы, который располагается первым.
 Поддиаграмма	Позволяет ссылаться на внешнюю кампанию. Таким образом осуществляется связь между диаграммами

Разработка процессов принятия решений ведется в компоненте SAS RTDM - Customer Intelligence Studio. CI Studio представляет собой тонкий клиент. Для пользователя наиболее значимыми функциями CI Studio являются создание диаграмм процесса принятия решения (вкладка «Designer»), разработка скриптов вспомогательных расчетов (вкладка «Definition»), управление версиями процессов принятия решений (вкладка «Administration»). На рисунке 1.4 изображено главное окно среды.

**SAS® Customer Intelligence Studio**

File Help **Designer** Definitions Setup Administration

Decision Campaigns (32 of 50) Search: Contains: SASA

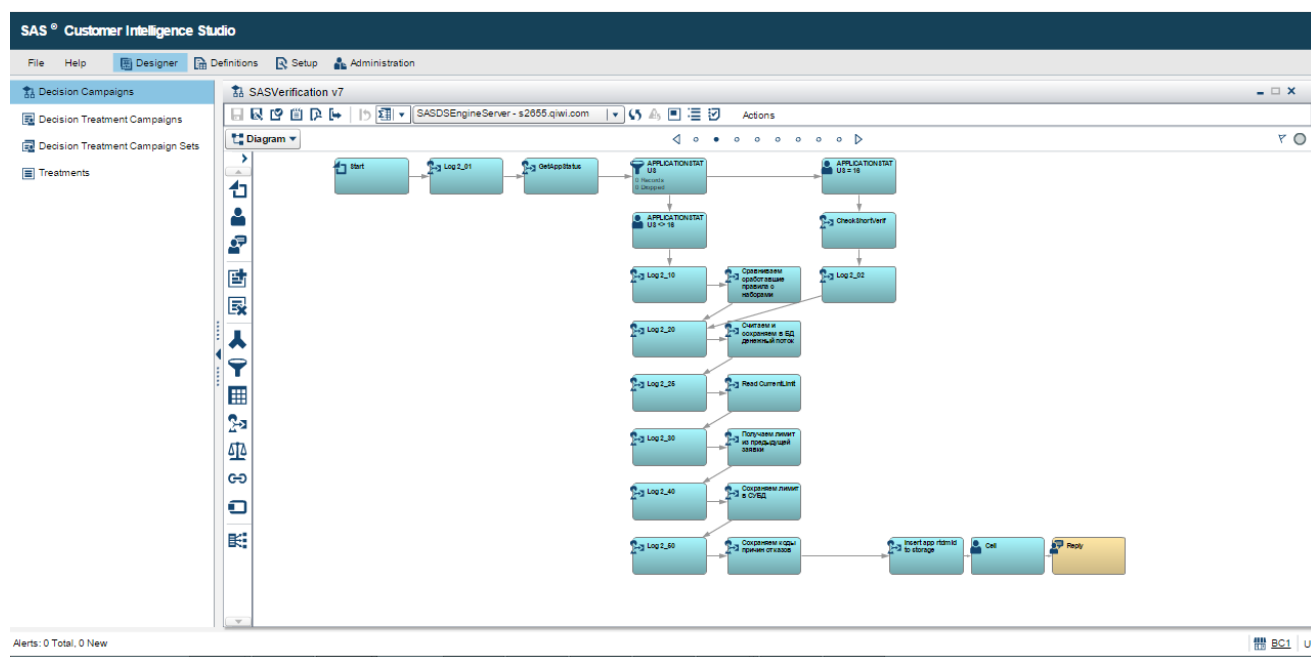
Open Actions

Name	Code	Status	Date Modified	Modified By
▼ Campaigns				
SASApproval	CAMP102	Marked for deployment	Oct 19, 2016 01:56 PM	SAS Demo User
SASApproval int_test	CAMP128	Marked for deployment	Oct 13, 2016 06:30 PM	SAS Demo User
SASApproval int_test v2	CAMP129	Designing	Oct 13, 2016 07:53 PM	SAS Demo User
SASApproval TEST	CAMP172	Designing	Dec 21, 2016 06:35 PM	Мухачева Анастасия Владимировна
SASApproval test_int_antifraud	CAMP137	Designing	Oct 21, 2016 03:27 PM	SAS Demo User
SASApproval v1		Marked for deployment	Oct 25, 2016 09:35 PM	SAS Demo User
SASApproval v1		Marked for deployment	Oct 26, 2016 04:17 PM	SAS Demo User
SASApproval v1		Marked for deployment	Oct 26, 2016 04:31 PM	SAS Demo User
SASApproval v13	CAMP161	Marked for deployment	Oct 26, 2016 06:06 PM	SAS Demo User
SASApproval v14	CAMP162	Marked for deployment	Oct 28, 2016 05:28 PM	SAS Demo User
SASApproval v15	CAMP164	Marked for deployment	Jun 8, 2017 05:13 PM	Бессонов Михаил Владимирович
SASApproval v15 4 SAS	CAMP193	Designing	Feb 3, 2017 07:03 PM	SAS Demo User
SASApproval v15 Antifraud PL SQL	CAMP168	Marked for deployment	Dec 9, 2016 07:15 PM	SAS Demo User
SASApproval v15 test for Pankov	CAMP192	Designing	Feb 20, 2017 10:12 PM	SAS Demo User
SASApproval v15_tst_Bess	CAMP173	Designing	Apr 28, 2017 03:01 PM	Бессонов Михаил Владимирович

Alerts: 0 Total, 0 New

BC1 User: SAS Demo User

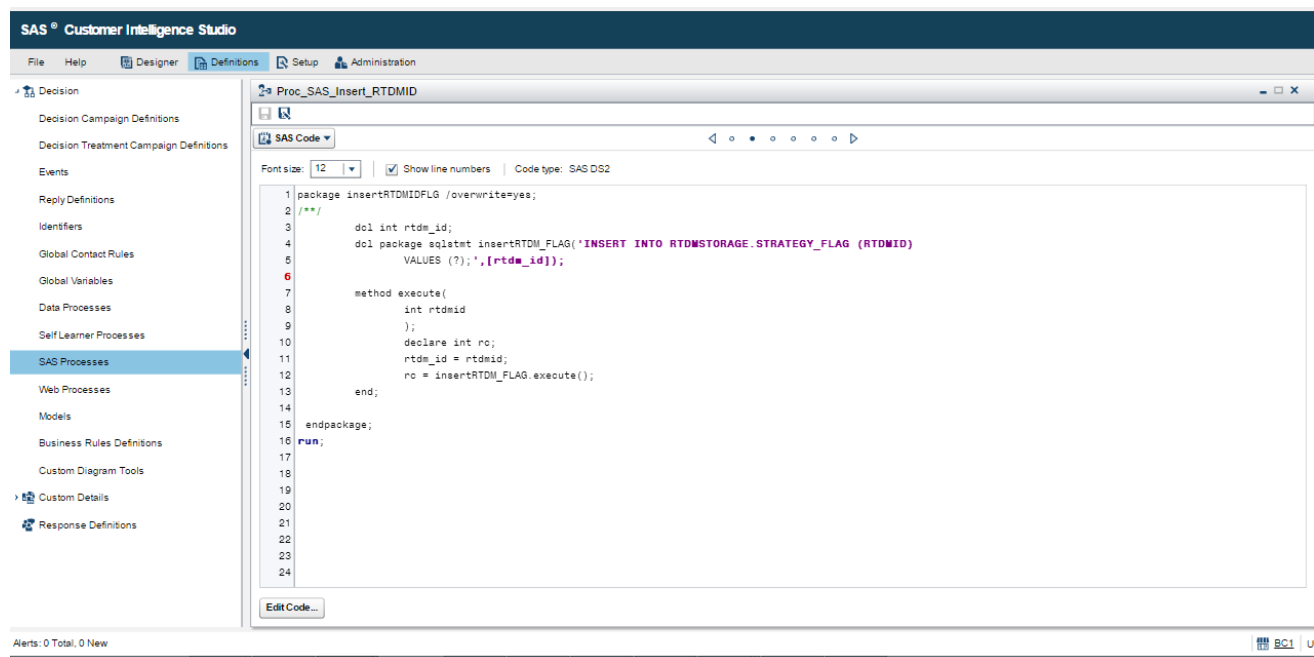
Рисунок 1.5 - Диаграмма процесса принятия решений



Пример реализованного скрипта вспомогательных расчетов изображен на рисунке 1.6.

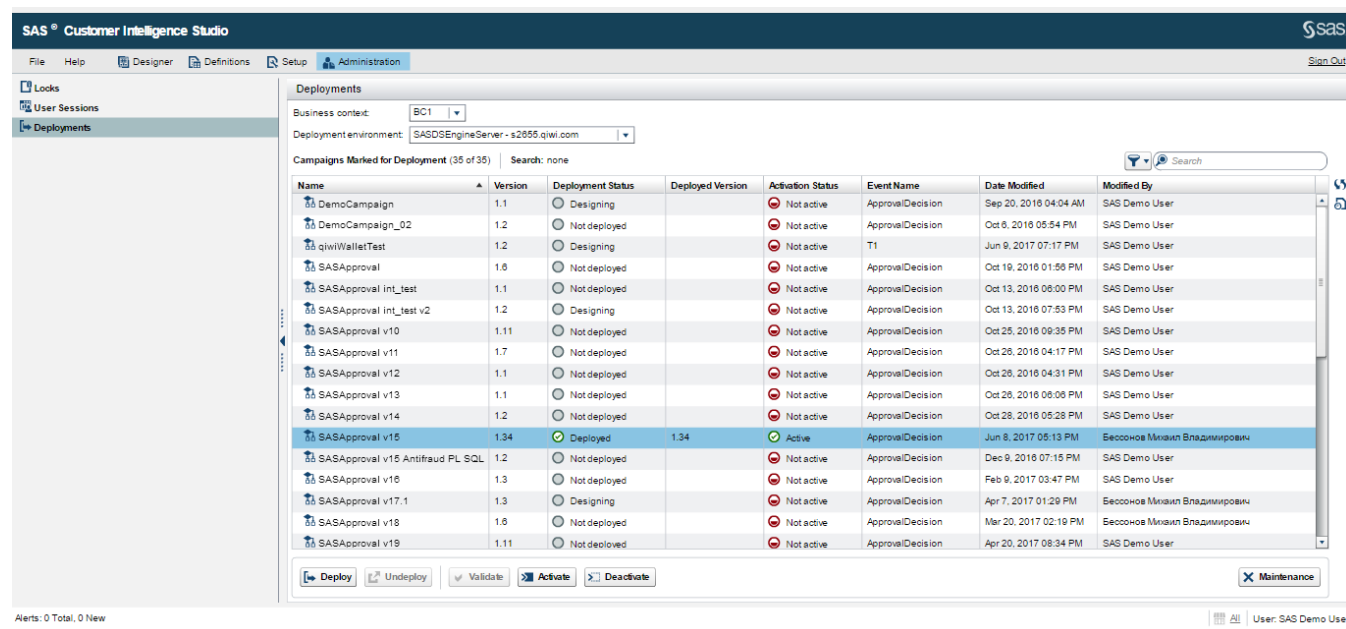


Рисунок 1.6 - Скрипт вспомогательных расчетов в CI Studio



С помощью функций администрирования пользователю CI Studio предоставляется возможность управления версий процессов принятия решений на имеющихся средах (среда разработки, среда тестирования, среда сертификации, промышленная среда).

Рисунок 1.7 - Окно администрирования процессов принятия решений



## 1.5 Описание инструмента Credit Registry Enterprise.

Каждое БКИ предоставляет отчёт о кредитной истории субъекта (КИ) в своём собственном формате. Так как обработка КИ может идти в автоматическом или автоматизированном режиме в какой-либо ИТ-системе банка, возникает необходимость реализации и поддержки нескольких различных форматов в этой системе.

Для того, чтобы избежать проблему с различными форматами, CRE предоставляет возможность получения КИ в унифицированной форме. То есть вне зависимости от БКИ, предоставляющего эту кредитную историю. Данная форма называется Единым форматом. Единый формат представляет собой документ XML, соответствующий схеме, описывающий КИ субъекта.

Так как данные по КИ для анализа по одному субъекту могут быть получены более, чем из одного БКИ, между ними могут возникать противоречия, а также данные дублируются. CRE предоставляет функциональность, позволяющую производить слияния отчётов Единого формата. Слияние включает в себя следующие шаги:

1) дедупликация в ходе которой определяются уникальные счета по всем отчётам всех БКИ,

2) сведение в ходе которой по всем счетам, информация о которых дублируется в отчётах

БКИ, производится нормализация данных.

Определение дублирования сведений об одном и том же кредите в отчёте БКИ или отчётах различных БКИ определяется на основании следующих признаков:

- совпадение даты открытия кредита;
- совпадение суммы кредита с точностью до единицы валюты (т.е. «копейки» отбрасываются) или совпадение номера договора;
- совпадение валюты кредита;

- совпадение типа кредита (после приведения к справочнику). Учёт данного критерия отключается настроечным параметром «Сравнить типы кредитования при сведении» раздела «Редактирование параметров»;
- совпадение отношения клиента (основной заёмщик/поручитель и т.д.)

При сведении, если обнаруживается противоречие в данных о кредите, CRE функционирует по следующим правилам:

- статус кредита: если противоречивые данные получены из отчёта одного БКИ, приоритет имеет статус той записи, которая имеет наиболее позднее время обновления. Если данные получены из отчётов различных БКИ, принимается статус по наибольшему приоритету согласно таблице 1.11:

Приоритет	Статус
1	Активный
2	Просрочен
3	Счет закрыт
4	Передан на обслуживание в другой банк
5	Оплачен за счет обеспечения
6	Спор
7	Проблемы с возвратом

- платёжная дисциплина: для пересекающихся диапазонов дат выбирается пессимистический вариант. Если имеются непересекающиеся диапазоны дат, в итоговом отчёте они так же будут присутствовать;
- остальные поля: используются наиболее пессимистичные значения.

Все функции Единого формата доступны через интерфейс веб-сервиса CRE. Сервис Единого формата предоставляет следующие методы:

- processRequest() – запрос КИ по субъекту в БКИ и выдача результата в виде документа Единого формата;
- groupRequest() – запрос КИ одновременно в нескольких БКИ и сведение данных о КИ субъекта, полученных из нескольких бюро, в сводный отчёт Единого формата;

- `joinUidResponses()` – сведение нескольких существующих (ранее полученных) отчётов Единого формата по одному субъекту в один отчёт Единого формата. Метод получает на вход перечень сводимых отчётов;
- `joinApplicationResponses()` – назначение метода аналогично `joinUidResponses()`, но в качестве входных данных используется идентификатор кредитной заявки.

## Выводы по разделу 1:

В разделе приведено краткое описание рынка розничных банковских услуг. Описан процесс рассмотрения кредитной заявки в исследуемом банке, которые состоит из трех основных стадий: получение данных о клиенте, анализ данных, принятие решения.

Выявлены следующие проблемы, возникающие в процессе кредитования клиентов:

ручная обработка входящих заявок, которая приводит возникновению ошибок, в связи с человеческим фактором;

невозможность СПР напрямую связываться и обмениваться данными с сервисами Бюро кредитных историй;

различный формат выходных данных CRE и входных данных SAS RTDM.

Также обоснован выбор платформы реализации системы принятия решений SAS Real-Time Decision Manager при помощи метода экспертных оценок, описаны его возможности и функции.

Описаны возможности и функции инструмента Credit Registry Enterprise.

## КОНСТРУКТОРСКИЙ РАЗДЕЛ

### Автоматизация процессов кредитования на основе системы SAS RTDM.

Достичь поставленную цель возможно, решив следующие задачи:

- проанализировать существующие кредитные системы принятия решений;
- выявить недостатки в системе принятия решений по кредитной заявке на примере конкретного Банка;
- устранить выявленные недостатки;
- оценить экономическую эффективность разработанной системы.

#### 2.1 Общая схема процесса принятия решения.

Процесс принятия решения о выдаче представляет собой проверку информации о потенциальном заёмщике по различным внутренним правилам, при выполнении которых определяется платежеспособность клиента и его благонадёжность.

Каждый процесс реализует определенный набор проверочных и расчетных блоков.

Каждый блок в качестве входных атрибутов принимает необходимые значения для расчетов. В результате обработки блока, в СПР возвращается определенный состав рассчитанных бизнес полей.

Из системы принятия решений может быть осуществлен вызов внешней системы для обогащения данных по заявке и участникам сделки.

Процесса принятия решений состоит из этапов, которые перечислены на рисунке 2.1.

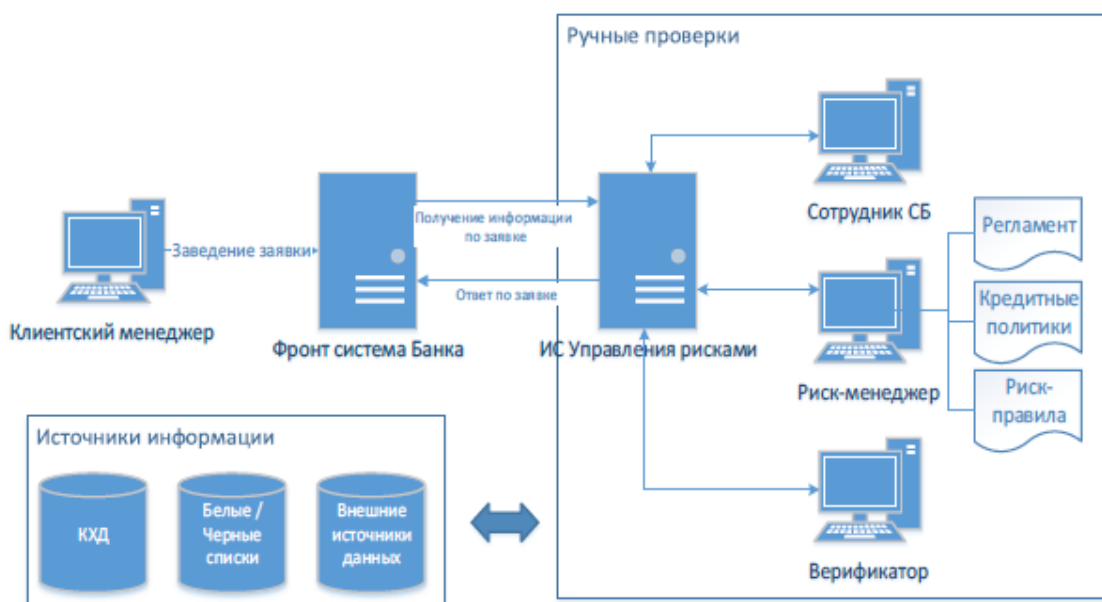
Рисунок 2.1 - Этапы процесса принятия решений



Когда процесс принятия решения осуществляется без автоматизации, анализ приложения в банке осуществляется в полуавтоматическом режиме, то есть после того, как приложение открывается менеджером клиента в системе фронт-офиса, приложение При этом определенный набор рассчитанных параметров передается менеджеру по управлению рисками, который затем, в соответствии с требованиями, выполняет проверки в ручном режиме и определяет необходимость маршрутизации для андеррайтера (верификатора) или сотрудника службы безопасности.

Схематично процесс отражен на рисунке 2.2.

Рисунок 2.2 - Процесс ручного рассмотрения заявки



Процесс принятия решений представляет собой вызов стратегии SAS RTDM с входящими параметрами.

Система принятия решений SAS RTDM обеспечивает автоматическое принятие решений по кредитным заявкам Банка в соответствии с требованиями Банка, настроенными с использованием инструментов SAS RTDM. SAS RTDM предоставляет инструменты для настройки процесса кредитования, создания, удаления и изменения блоков.

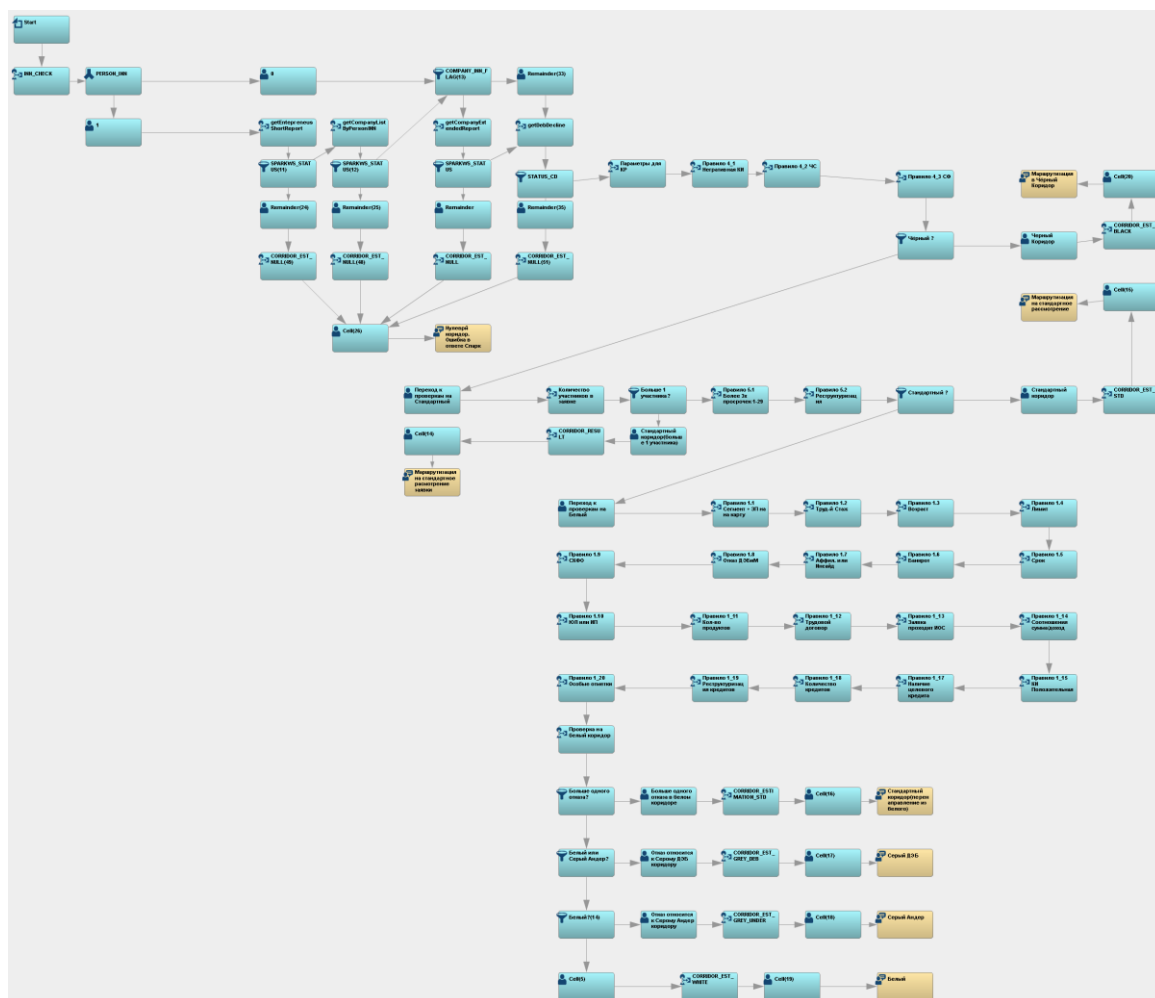
Основные функции системы принятия решений SAS RTDM:

- Написание процессов с использованием Web-интерфейса;

- Удобный перенос изменений с одного контура на другой;
- Принятие решения по кредитной заявке согласно требованиям, настроенным в SAS RTDM;
- Интеграция с внешними источниками данных по протоколу SOAP;
- Интеграция с внутренними источниками данных (БД);
- Тестирование стратегии принятия решений, так же есть возможность batch-тестирования;
- Запись данных в БД;

Алгоритм рассмотрения кредитной заявки и принятия решения по ней оформляется в SAS RTDM в виде схемы, состоящей из узлов. Пример такого алгоритма можно увидеть на рисунке 2.3.

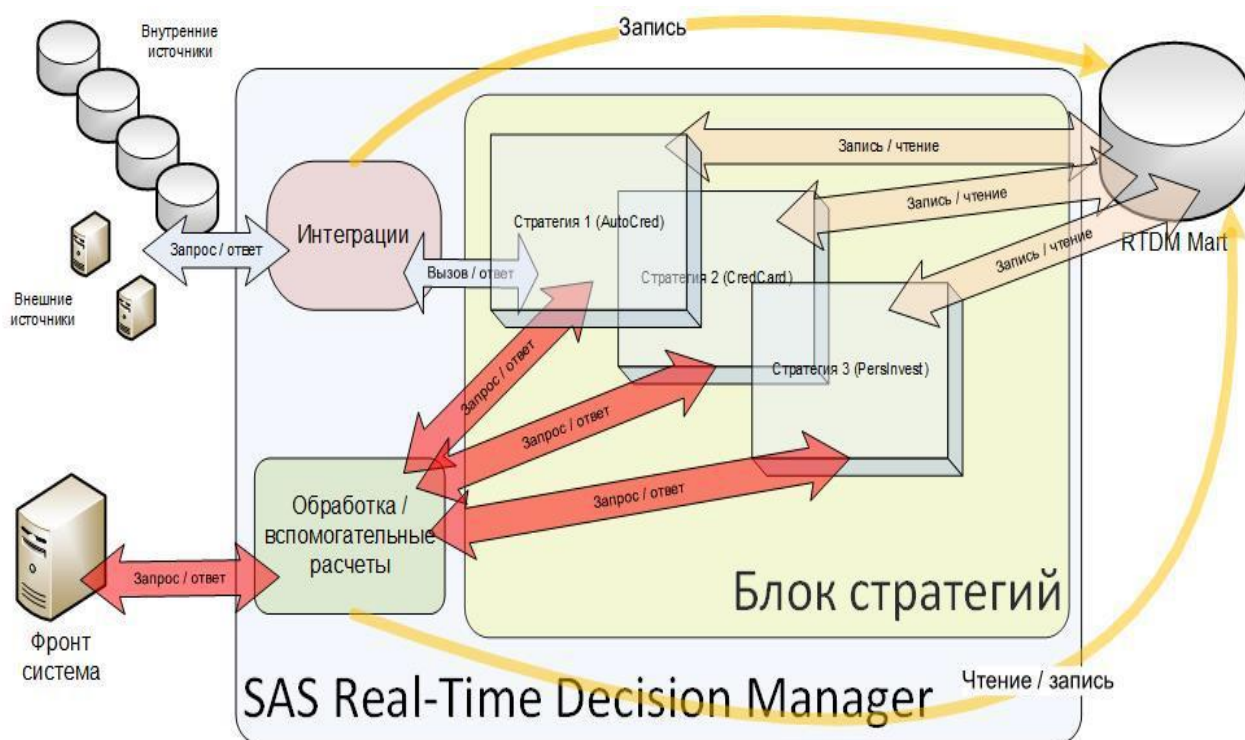
Рисунок 2.3 – Алгоритм процесса принятия решения по кредитной заявке



## 2.2 Ограничения типовой реализации системы принятия решений средствами SAS RTDM.

Рассмотрим типовую реализацию системы принятия решений на базе SAS RTDM, которая отражена на рисунке 2.4.

Рисунок 2.4 - Концептуальная типовая модель реализации системы принятия решений на базе SAS RTDM



Рассмотрим отдельно узел «Обработка/вспомогательные расчеты». Данный элемент используется для выполнения усложненных расчетов, требующих написания отдельных скриптов. В общей практике данный элемент реализуется с помощью языка SAS Base.

На SAS Base реализуются алгоритмы обработки информации, далее скомпилированный скрипт «публикуется» на сервере приложений как веб-сервис, после чего к нему возможно обратиться SOAP запросом. Схематично процесс отражен на рисунке 2.5.



Рисунок 2.5 - Формирование скрипта вспомогательных расчетов



Этот подход эффективен, если в процессе принятия решения не требуется большого количества сценариев, в противном случае общее время обработки заявки может значительно увеличиться, что напрямую влияет на качество всего процесса кредитования. Эффективность реализации этой кампании вспомогательных расчетов оценивается дополнительно.

#### Структура запроса WEB сервисов.

В системе принятия решений на базе SAS RTDM поддерживается только простая входная структура для запроса и ответа в веб-сервисы с помощью встроенного элемента “Web-Process”. Большинство систем, с которыми необходимо интегрироваться, принимают на вход сложную структуру.

Эта проблема решается путем разработки службы интеграции между системой принятия решений и веб-службой. Конкретный метод уровня интеграции с простой структурой вызывается из системы принятия решений. Кроме того, служба интеграции собирает данные из базы данных, из которой она формирует структуру сообщения запроса для веб-службы, из которой DSS должен получать данные. После этого запрос отправляется в виде сообщения SOAP, в ответ сервис интеграции получает ответ и инициализирует классы, которые были созданы с помощью XSD стороннего веб-сервиса. Ответ от веб-службы записывается в соответствующие таблицы, и в ответ от службы интеграции это простая структура, в которой передается ключ, с помощью которого DSS может получать данные из базы данных.

Далее представлено описание реализации интеграции системы принятия решения на базе SAS RTMD и системы доступа к Бюро кредитных историй Credit Registry Enterprise.

Таким образом, выявлено несколько недостатков текущей реализации СПР, которые необходимо решить посредством разработки интеграционных приложений.

До разработки интеграционных приложений	После разработки и внедрения интеграционных приложений
Данные, полученные из CRE и фронт-офисной системы не принимаются на вход SAS RTDM из-за сложной структуры	Интеграционные приложения приводят данные, полученные на выходе из CRE и фронт-офисной системы, к тому виду данных, который ожидается на входе в SAS RTDM
Нет возможности получить данные из БКИ непосредственно из SAS RTDM	Данные из БКИ доступны в едином интерфейсе SAS RTDM
Нет возможности получать анкетные данные из фронт-офисной системы	Данные из фронт-офисной системы доступны из SAS RTDM

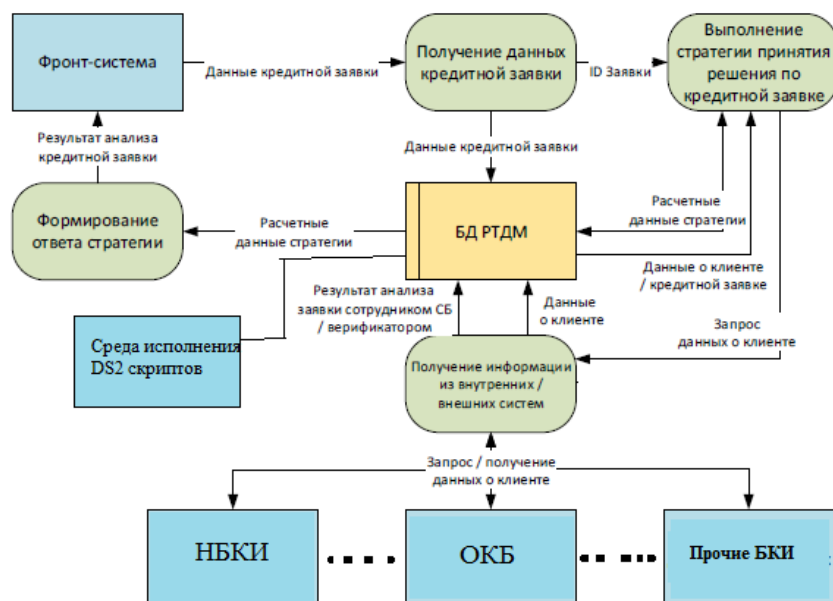
### 2.3 Реализации СПР средствами SAS с учетом выявленных недостатков типовой схемы.

Для устранения выявленных недостатков типовой реализации разработана новая схема реализации СПР. Схематично модель отображена на рисунке 2.6. Ниже перечислены ключевые пункты, устраняющие отмеченные узкие места.

1. Скрипты «вспомогательных расчетов» разрабатываются на языке Groovy. Вызов скриптов реализован при помощи встроенного бесшовного интеграционного механизма;
2. Получение ответа от Бюро кредитных историй осуществляется с помощью интеграционного веб-сервиса.

Схематично модель изображена на рисунке 2.6.

Рисунок 2.6 - Новая модель системы принятия решений



## Реализация блоков вспомогательных расчетов на основе Groovy

Groovy — объектно-ориентированный язык программирования, разработанный для платформы Java как дополнение к языку Java с возможностями Python, Ruby и Smalltalk.

Использует Java-подобный синтаксис с динамической компиляцией в JVM байт-код и напрямую работает с другим Java-кодом и библиотеками. Язык может использоваться в любом Java-проекте или как скриптовый язык.

Уменьшение времени обработки кредитной заявки системой принятия решений при использовании языка Groovy относительно веб-сервисов, реализованных при помощи языка SAS Base, экспериментально доказано ниже.

При тестировании производительности обоих подходов были реализованы скрипты на языке Groovy и на языке SAS Base, который в последствие доступен для системы принятия решений в виде веб-сервиса. Скрипты содержат все необходимые конструкции, которые используются при разработке вспомогательных расчетов в системе принятия решений: математические операторы, работа со строками, DML запросы к базе данных.

В ходе испытаний использовался один тестовый кейс. В процессе выполнения скрипта на SAS Base и Groovy на основе входных данных тестового

кейсы выполняются аналогичные шаги. В связи с этим, сложность алгоритма, разработанного для двух подходов, считается равной. В ходе эксперимента получены результаты, которые отображена в таблице 2.6.

Таблица 2.1 - Результаты эксперимента

Подход реализа ции расчетов	Количес тво испытан ий	Оценка математическ ого ожидания, сек.	Несмещенн ая оценка дисперсии, сек.в кв.	Стандарт ное отклонен ие	Доверительный интервал при уровне значимости 0.1
Groovy	1000	3.017 сек.	35.750	5.979	(2.706;3.328)
SAS Base	1000	9.081 сек.	164.609	12.830	(8.413;9.749)

Скрипты, написанные на языке Groovy, исполняются в среднем в 3 раза быстрее, чем скрипт, реализованный на языке SAS Base.

Опираясь на результаты эксперимента, было решено использовать в качестве инструмента для реализации вспомогательных расчетов в системе принятия решений язык Groovy.

Пример программы на языке Groovy:

```
class LoopingCallsGroovy implements Runnable {

    private static final Logger log = Logger.getLogger(LoopingCallsGroovy.class);
    private final Stopwatch watch = new Stopwatch();

    Long rtdmId = null;
    String EndpointURL = null;
    Long breakIfErrorFlag = null;
    RTDMTable inputVar = null;
    RTDMTable outputVarTempl = null;
    private Long resultCode = null;
    private String resultDescription = null;
    private RTDMTable outputVar = null;

    public void run() {
        SASDSResponse response = null;
        SASDSRequestFactory DSfactory = null;
        SASDSRequest DSrequest = null;

        if (log.isTraceEnabled()) {
            watch.start();
            log.trace("Start Looping Calls. RTDM_ID = " + rtdmId + ";
BREAK_IF_ERROR_FLAG = " + breakIfErrorFlag + ";");
        }

        try {
            //log.trace(EndpointURL + "/RTDM/Custom");
            DSfactory = SASDSRequestFactory.getInstance(EndpointURL +
"/RTDM/Custom", new Properties());
            //log.trace("After DSfactory = SASDSRequestFactory.getInstance");
            String CORRELATION_ID =
Long.toHexString(System.currentTimeMillis());
```

```

        outputVar = outputVarTemp1;
        for (Row row : inputVar.iterator()) {
            String eventName = null;
            for (RTDTable.Column c : inputVar.getColumns()) {
                if (c.getType().toString().equals("STRING")) {
                    if (c.getName().equals("EVENT_NAME")) {
                        //log.trace("Var Name = " + c.getName() +
"; Var value = " + row.columnDataGet(c.getName()));
                        eventName =
row.columnDataGet(c.getName());
                        //log.trace("eventName = " + eventName);
                    }
                }
            }
            DSrequest = DSfactory.create(eventName, CORRELATION_ID,
"GMT");
            for (RTDTable.Column c : inputVar.getColumns()) {
                if (c.getType().toString().equals("STRING")) {
                    if (row.columnDataGet(c.getName()) != null
                        &&
!c.getName().equals("EVENT_NAME")) {
                        DSrequest.setString(c.getName(),
row.columnDataGet(c.getName()));
                        //log.trace("Var Name = " + c.getName() +
"; Var value = " + row.columnDataGet(c.getName()));
                    }
                    else
                        DSrequest.setString(c.getName(), null);
                }
            }
        }
    }
}

```

Интеграционное приложение, обеспечивающее взаимодействие фронт-офисной системы с SAS RTDM.

Для начала необходимо выбрать язык программирования для реализации необходимых интеграционных приложений.

Критерий	C++	Python	Java	Scala
Объектно-ориентированный	+	+	+	+
Нативная поддержка связи с БД	-	+	+	+
Автоматический сборщик мусора	-	+	+	+
Наличие веб-фреймворков	-	+	+	+
Опыт использования	+	-	+	-

Исходя их совокупности необходимых для выбора языка факторов, разрабатываться интеграционные приложения будут на Java.

Следующим шагом будет выбор среды разработки.

Критерий	NetBeans	Eclipse	IntelliJ IDEA
Кроссплатформенность	+	+	+
Встроенный отладчик	+	+	+
Интеллектуальное автодополнение	+	-	+
Наличие установленного ПО на предприятии	-	+	+

Учитывая необходимые критерии, выбираем IntelliJ IDEA.

SAS RTDM принимает только плоскую структуру запроса и ответа. В следствии этого перед консультантами SAS возникает задача интеграции фронт-офисной ситемы с SAS RTDM.

Интеграционное приложение будет являться клиентом для веб-сервиса SAS RTDM с одной стороны, а с другой фронт-офисная система будет являться клиентом интеграционного приложения.

Интеграционное приложение реализовано при помощи платформы Java Enterprise Edition с использованием открытых библиотек:

- Библиотека MyBatis – предназначена для автоматизации работы между Java-классами и СУБД;
- Библиотека CXF – предназначена для разработки веб-сервисов.
- Log4j – инструмент логирования действий.

Интеграционное приложение разворачивается в виде SOAP веб-сервиса на сервер приложений SAS Web Application Server (изначально разработан на базе Apache WebAppServer).

У приложения должен быть один метод, который будет использоваться для принятия запроса от фронт-офисной системы и возврата ответа обратно.

Так же для реализации интеграционного приложения нам понадобятся:

- XSD запроса
- XSD ответа
- WSDL события в SAS RTDM

Запрос к интеграционному приложению от фронт-офисной системы происходит в синхронном режиме. На вход подается сложная структура, описанная в xsd-файле запроса. На выход подается так же сложная структура, которая описана в xsd-файле ответа.

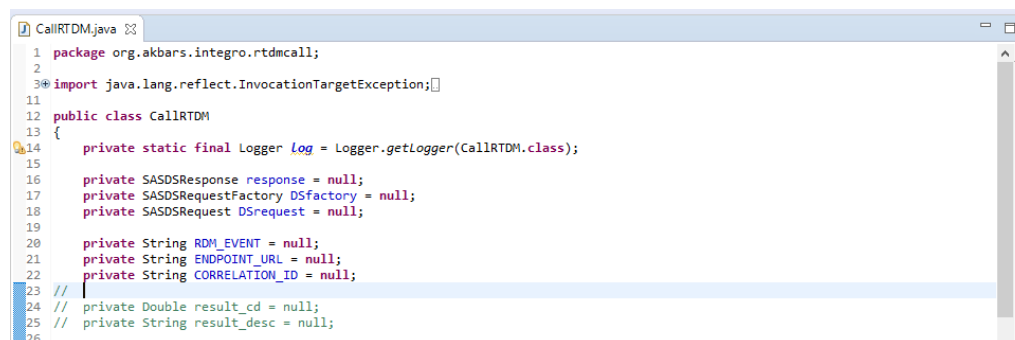
Ниже приведен общий алгоритм выполнения запроса:

1. Генерация/получение идентификатора запроса для SAS RTDM;
2. Парсинг данных, полученных из заявки в операционную базу SAS RTDM;
3. Вызов события стратегии в SAS RTDM с помощью динамического заполнения атрибутов этого события;
4. Получения ответа от SAS RTDM;
5. Обработка ответа от SAS RTDM;
6. Сбор ответной xml в соответствии с классами, сгенерированными по xsd ответа;
7. Отправка ответа в фронт-офисную систему;
8. Логирование действий;

Опыт внедрения интеграционных приложений в различных банках сформировал определенный набор методов для их написания. В рамках данной работы была поставлена задача придумать новые методы разработки, которые позволили бы усовершенствовать процесс, а так же увеличить контроль за его ходом. В следствии этого были реализованы два интеграционных приложения.

Первое интеграционное приложение было разработано с помощью JavaApi, которое позволяло вызывать SAS RTDM с помощью методов классов из библиотек.

Рисунок 2.6 - Объявление классов для вызова SAS RTDM в первом интеграционном приложении



```
1 package org.akbars.integro.rtdmcall;
2
3 import java.lang.reflect.InvocationTargetException;
4
5
6
7
8
9
10
11 public class CallRTDM
12 {
13     private static final Logger Log = Logger.getLogger(CallRTDM.class);
14
15     private SASDSResponse response = null;
16     private SASDSRequestFactory DSfactory = null;
17     private SASDSRequest DSrequest = null;
18
19     private String RDM_EVENT = null;
20     private String ENDPOINT_URL = null;
21     private String CORRELATION_ID = null;
22
23     //
24     // private Double result_cd = null;
25     // private String result_desc = null;
26 }
```

Рисунок 2.7 - Вызов SAS RTDM в первом интеграционном приложении

```
35
36 CORRELATION_ID = Long.toHexString(System.currentTimeMillis());
37
38 System.out.println("calling RTDM");
39 System.out.println("phaseid == " + phaseid);
40 System.out.println("strategyid == " + strategyid);
41 List<RTDMParameters> rtdmPrmtrs = abRepMapper.getStrategyAndEvent(phaseid, strategyid);
42 if (rtdmPrmtrs != null)
43 {
44     for (RTDMParameters rtdmPrmtr : rtdmPrmtrs)
45     {
46         ENDPOINT_URL = rtdmPrmtr.getUri();
47         RDM_EVENT = rtdmPrmtr.getEventName();
48     }
49     System.out.println("ENDPOINT_URL == " + ENDPOINT_URL);
50     System.out.println("RDM_EVENT == " + RDM_EVENT);
51     try
52     {
53         DSfactory = SASDSRequestFactory.getInstance(ENDPOINT_URL, props);
54         DSrequest = DSfactory.create(RDM_EVENT, CORRELATION_ID, "GMT");
55         DSrequest.setString("APPLICID", applicid);
56         DSrequest.setLong("RTDMID", rtdmid);
57         DSrequest.setString("PHASEID", phaseid);
58         DSrequest.setString("STAGEID", "1");
59
60         response = DSrequest.execute();
61
62         System.out.println("ErrorCode == " + response.getString("ErrorCode") + " ;"
63             + " ErrorType == " + response.getString("ErrorType") + " ;"
64             + " ErrorDescription == " + response.getString("ErrorDescription"));
65     } catch (ClassNotFoundException | SecurityException | IllegalArgumentException | NoSuchMethodException
66         | InstantiationException | IllegalAccessException | InvocationTargetException e)
67     {
68         System.out.println("Errors in rtdm execute");
69     }
70 }
```

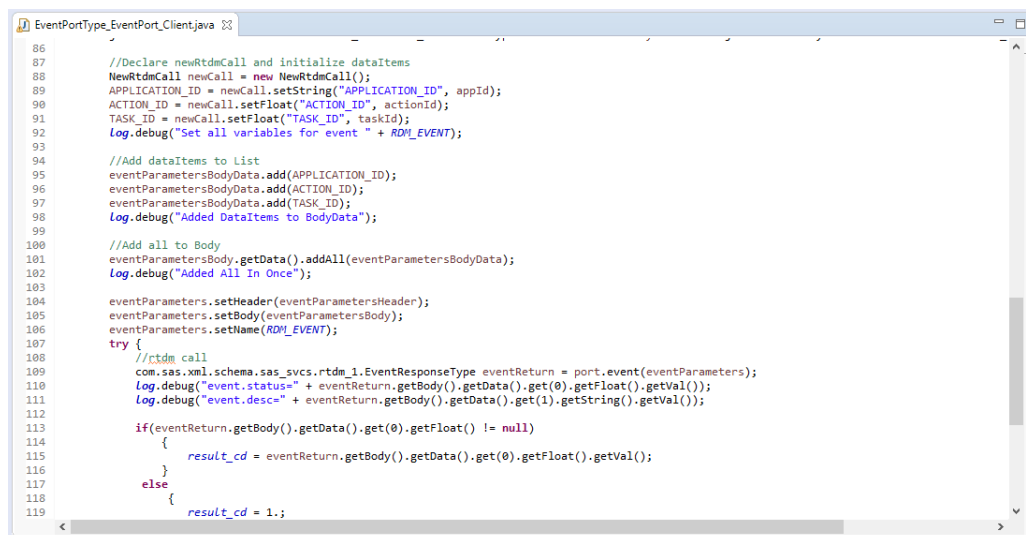
Второе интеграционное приложение вызывало SAS RTDM, как клиент веб-сервиса, динамически заполняя входящие параметры для события стратегии, формируя классы и отправляя их на адрес развернутого приложения SAS RTDM с помощью методов, которые были сгенерированы для клиента веб-сервиса, по WSDL события SAS RTDM. Данный подход позволил включить логирование, что помогло качественнее отлавливать ошибки и обеспечило прозрачность процесса, контроль за его ходом.

Рисунок 2.8 - Класс, в котором динамически заполняются параметры для события

```
1 package com.sas.newrtm.call;
2
3 public class NewRtdmCall
4 {
5
6     public NewRtdmCall()
7     {
8     }
9
10
11     public com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType setString(String valueName, String value)
12     {
13         com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType eventParametersBodyDataVall = new com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType();
14         com.sas.xml.schema.sas_svcs.rtdm_1.String stringEventValue = new com.sas.xml.schema.sas_svcs.rtdm_1.String();
15         stringEventValue.setVal(value);
16         eventParametersBodyDataVall.setName(valueName);
17         eventParametersBodyDataVall.setString(stringEventValue);
18         return eventParametersBodyDataVall;
19     }
20
21     public com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType setFloat(String valueName, Double value)
22     {
23         com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType eventParametersBodyDataVall = new com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType();
24         com.sas.xml.schema.sas_svcs.rtdm_1.Float floatEventValue = new com.sas.xml.schema.sas_svcs.rtdm_1.Float();
25         floatEventValue.setVal(value);
26         eventParametersBodyDataVall.setName(valueName);
27         eventParametersBodyDataVall.setFloat(floatEventValue);
28         return eventParametersBodyDataVall;
29     }
30 }
31
```



## Рисунок 2.8 - Пример заполнения классов и вызов SAS RTDM



```
86
87 //Declare newRtdmCall and initialize dataItems
88 NewRtdmCall newCall = new NewRtdmCall();
89 APPLICATION_ID = newCall.setString("APPLICATION_ID", appId);
90 ACTION_ID = newCall.setFloat("ACTION_ID", actionId);
91 TASK_ID = newCall.setFloat("TASK_ID", taskId);
92 Log.debug("Set all variables for event " + RDM_EVENT);
93
94 //Add dataItems to List
95 eventParametersBodyData.add(APPLICATION_ID);
96 eventParametersBodyData.add(ACTION_ID);
97 eventParametersBodyData.add(TASK_ID);
98 Log.debug("Added DataItems to BodyData");
99
100 //Add all to Body
101 eventParametersBody.getData().addAll(eventParametersBodyData);
102 Log.debug("Added All In Once");
103
104 eventParameters.setHeader(eventParametersHeader);
105 eventParameters.setBody(eventParametersBody);
106 eventParameters.setName(RDM_EVENT);
107 try {
108 //rtdm call
109 com.sas.xml.schema.sas_svcs.rtdm_1.EventResponseType eventReturn = port.event(eventParameters);
110 Log.debug("event.status=" + eventReturn.getBody().getData().get(0).getFloat().getVal());
111 Log.debug("event.desc=" + eventReturn.getBody().getData().get(1).getString().getVal());
112
113 if(eventReturn.getBody().getData().get(0).getFloat() != null)
114 {
115     result_cd = eventReturn.getBody().getData().get(0).getFloat().getVal();
116 }
117 else
118 {
119     result_cd = 1.;
120 }
```

В дальнейшем проводилось нагрузочное тестирование обоих приложений. На вход поступал большой поток заявок (около 9000 заявок в час).

Таблица 2.2 - Результаты эксперимента

Подход реализации	Количество испытаний	Выдача ответа на фронт-офисную систему по одной заявке, сек.
JavaApi	9000	130.081 сек.
Клиент веб-сервиса	9000	92.582 сек.

Приложение, работающее с SAS RTDM, как клиент веб-сервиса в 1,5 раза быстрее обрабатывает заявку, через приложение, работающее с SAS RTDM через JavaApi.

Опираясь на результаты эксперимента, было решено в дальнейшем при промышленной разработке использовать данный подход к разработке интеграционных приложений.

К интеграционному сервису можно обратиться напрямую из системы принятия решений, реализованной на базе SAS Real-Time Decision Manager.

Интеграционное приложение, обеспечивающее взаимодействие SAS RTDM с сервисами БКИ

Стандартные средства SAS RTDM не позволяют напрямую обратиться к сервисам Бюро кредитных историй в силу плоской структуры запроса и ответа.

Credit Registry предоставляет функции направления запросов и обработки получаемых ответов от БКИ, функции накопления полученных ответов для возможности повторного их использования без запроса в БКИ (кэширование), функции сведения и дедупликации ответов от нескольких БКИ по одному субъекту в единый унифицированный отчет (УСО). Наличие у CRE доступа к различным БКИ, и различным сервисам в рамках БКИ определяется настройкой кодов БКИ и заключением договора между конкретным БКИ и Заказчиком.

Интеграционное приложение реализовано при помощи платформы Java Enterprise Edition с использованием открытых библиотек:

- Библиотека MyBatis – предназначена для автоматизации работы между Java-классами и БД;
- Библиотеки CXF – предназначены для автоматизированной передачи и получения ответа от веб-сервиса.
- Log4j – логгер, позволяющий контролировать процесс выполнения программы, с возможностью записи в разные источники.
- Apache Tomcat – позволяет опубликовать свое приложение на веб-сервере.

Интеграционное приложение разворачивается в виде SOAP веб-сервиса на сервер Apache Tomcat.

Разработанное интеграционное может обращаться к следующим сервисам (таблица 2.7):

Таблица 2.3 - Описание сервисов БКИ и коннекторов Credit Registry

БКИ	Коннекторы	CRE connector code	SubrequestCode
CRE	Не используется	Вызов метода сведения ранее полученных отчетов	

Продолжение таблицы 2.3 - Описание сервисов БКИ и коннекторов Credit Registry

NBKI	credit report	302	6
	scoring FICO 3	302	3
	Scoring Antifraud	302	4
	Social Connections	302	5
	AFS (NBKI-AFS)	27	
СМЭВ	FMS (NBKI-FMS)	7	нет
Equifax	credit report	33	28006
	Application PD Score (Equifax- Scoring)	33	28016
	FPS (Equifax-FPS)	20	
OKB	credit report	6	6
	Scoring (OKB- Scoring)	6	1
	NH (OKB-National Hunter)	21	нет
Spark	ConnectorService	22	1
	ConnectorService	22	31
	ConnectorService	22	5
	ConnectorService	22	8
KBRS	credit report	19	1
	FSSP (KBRS- FSSP)	19	16

К интеграционному сервису можно обратиться непосредственно из системы принятия решений, базе SAS Real-Time Decision Manager.

Запрос в Credit Registry происходит в синхронном режиме. На вход подается плоская структура, что является важным условием для элемента web-process системы SAS RTDM. Структура запроса представлена в таблице 2.7. Структура ответа – в таблице 2.8.

Таблица 2.4 - Структура запроса интеграционного сервиса

Атрибут	Описание	Тип
RTDMID	Идентификатор вызова Системы принятия решения	Long
CUSTOMERID	Идентификатор участника кредитной заявки	Character

Таблица 2.5 - Структура ответа интеграционного сервиса

Атрибут	Описание	Тип
REPLYCD	Код возврата интеграционного сервиса SAS RTDM	Character
CREID	Идентификатор отчета кредитной истории	Long

Ниже приведен общий алгоритм выполнения запроса:

1. Получение данных клиента из БД для заполнения в параметров запроса в CRE;
2. Получение аутентификационных данных для доступа к CRE (логин и пароль);
3. Получение идентификатора запроса и идентификатора кредитной заявки. При этом, идентификатора запроса генерируется для каждого запроса в БКИ создается уникальный, а идентификатора кредитной заявки – уникальный на одну обрабатываемую заявку;
4. Сохранение данных запроса, идентификатора запроса, идентификатора кредитной заявки, RTDMID в базу данных;

5. Отправка запроса в CRE;
6. Получение ответа CRE;
7. Генерация ответа CRE (CREID), демаршалинг и запись кредитного отчета в базу данных;
8. Генерация кода возврата сервиса (REPLYCD);
9. Логирование действий;

REPLYCD формируется на основе наличия технических ошибок в процессе выполнения запроса и атрибута “Response Status” в ответе Credit Registry. Статусная модель REPLYCD отображена в таблице 2.10.

Таблица 2.10 - Статусная модель статуса возврата ИРС

Значение REPLYCD	Описание REPLYCD	Значение Response Status	Описание Response Status	Ошибки в процессе интеграции
1	Критическая ошибка	-	-	Ошибки, связанные с выполнением кода веб-сервиса.
0	Успешное выполнение запроса	0	кредитная история найдена	-
0	Успешное выполнение запроса	1	данные заемщика не найдены	-
1	Критическая ошибка	2	ошибка в ответе внешнего источника	-
1	Критическая ошибка	4	техническая ошибка	-

Продолжение таблицы 2.10 - Статусная модель статуса возврата ИРС

2	Событие предупреди тельного характера	3	ошибка таймаута ответа	-
2	Событие предупреди тельного характера	5	ошибка проверки данных	-
2	Событие предупреди тельного характера	6	прочие ошибки	-

В интеграции были реализованы следующие методы (представлены в таблице 11), которые отвечают за обращение в конкретные сервисы Credit Registry по определенному коннектору (перечень сервтсов и соответствующих им коннекторов перечислен в таблице 2.11).

Таблица 2.11 - Название методов интеграционного сервиса

Название сервиса	Соответствующий метод интеграционного сервиса	Значение глубины поиска в кэше	Значение таймаута запроса
Запрос компании по ИНН	GetCompanyExtendedReport	5 дней	20 секунд
Проверка по ИНН ФЛ принадлежности к участнику бизнеса	GetCompanyListByPersonINN	5 дней	20 секунд

Продолжение таблицы 2.11 - Название методов интеграционного сервиса

Проверка по ФИО ФЛ принадлежности к участнику бизнеса	GetCompanyListByFIO	5 дней	20 секунд
Проверка по ИНН ФЛ принадлежности к ИП	GetEntrepreneurShortReport	5 дней	20 секунд
Получение из НБКИ кредитного отчета	getNBCHcredreport	5 дней	20 секунд
Обращение в CRE для получения унифицированного сводного	getCREUSO	-	10 секунд
Получение из КБРС кредитного отчета	getKBRScredreport	5 дней	60 секунд
Получение из НБКИ скоринга FICO 3	getNBCHFICO3	5 дней	20 секунд
Получение из НБКИ Анти-фрод скоринга	getNBCHAntiFraudScore	1 дней	20 секунд
Получение из НБКИ информации о социальных связях	getNBCHSocialConnects	10 дней	30 секунд

Продолжение таблицы 2.11 - Название методов интеграционного сервиса

Получение из СМЭВ информации ФМС	getSMEVFMS	10 дней	20 секунд
Получение из ОКБ информации National Hunter	getOKBNatHunter	1 день	20 секунд
Получение из НБКИ информации Anti Fraud Scoring	getNBCHAFS	1 дней	30 секунд
Получение из Эквифакс кредитного отчета	getEQUcredreport	5 дней	20 секунд
Получение из Эквифакс Аппликационного PD скоинга	getEQUApplicScore	5 дней	20 секунд
Получение из Эквифакс информации Fraud Prevention Service	getEQUFPS	1 дней	30 секунд
Получение из ОКБ кредитного отчета	getOKBcredreport	5 дней	20 секунд
Получение из КБРС информации ФССП	getKBRSFSSP	10 дней	20 секунд



Ответ от сервиса приходит в одинаковом виде и имеет структуру, описанную в таблице 2.12. Каждой сущности из ответа соответствует аналогичная таблица в модели данных. Подробную детализацию отчета по основным БКИ (ОКБ, НБКИ, ВБКИ, Equifax) можно найти в документации «CRE.v28. Сведение и унификация. Описание использования», а для других сервисов есть отдельные документации.

Таблица 2.12 - Используемые сущности кредитного отчета и маппинг на модель данных

Название сущности	Описание сущности	Соответствующая таблица операционной БД RTDM
MAIN	Идентификационные данные	CRE_MAIN
CONNECTOR_DATA_DETAIL	Расширенная информация об источнике КИ	CRE_CONNECTOR_DATA_DETAIL
SECONDARY	Статусные данные	CRE_SECONDARY
GENERAL	Информация о содержимом отчёта	CRE_GENERAL
DUCOMENT	Информация о документах	CRE_DOCUMENT
ADDRESS	Информация об адресе	CRE_ADDRESS
PHONE	Данные о телефонном номере	CRE_PHONE
EMPLOYMENT	Сведения о работодателе	CRE_EMPLOYMENT
LOANS_OVERVIEW	Сводные данные о кредитах субъекта	CRE_LOANS_OVERVIEW

Продолжение таблицы 2.12 - Используемые сущности кредитного отчета и маппинг на модель данных

LOAN	Описание кредитного договора	CRE_LOAN
SCORE	Скоринговый отчёт	CRE_SCORE
INQUIRY	Информация о запросах в БКИ	CRE_INQUIRY
FRAUD	Одиночный блок с информацией для предотвращения мошенничества	CRE_FRAUD

Выводы по разделу 2:

За счет использования в системе принятия решений механизмов Groovy удалось реализовать блоки вспомогательных расчетов в SAS RTDM и уменьшить время выполнения вспомогательных расчетов в 3 раза.

Разработанное интеграционное решение позволило реализовать взаимодействие между системой принятия решений и сервисами, предоставляющими кредитную историю, а так же взаимодействие системы принятия решений с фронт-офисной системой.

## ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

Описание методов тестирования и обоснование целесообразности применения предлагаемых решений.

Будем рассматривать тестирование как процесс отладки (проверки) какой-либо программы, который заключается в выполнении последовательности различных наборов контрольных тестов, для которых заранее известен ожидаемый результат. Т.е. при тестировании предполагается, что будет выполнена программа и получены конкретные результаты выполнения написанных тестов.

При подборе необходимых тестов, надо охватить как можно больше ситуаций алгоритма проверяемой программы. Менее жесткое требование - выполнение хотя бы один раз каждой ветви программы.

Первым видом тестирования была отладка.

Отладка - это проверка описания программного объекта на языке программирования с целью обнаружения в нем синтаксических и логических ошибок и последующее их устранение. Ошибки обнаруживаются компиляторами при их синтаксическом контроле. После этого проводится верификация по проверке правильности кода и валидация по проверке соответствия программного продукта заданным требованиям.

Целью тестирования является проверка работы реализованных функций в соответствии с их описанием и спецификацией. На основе внешних спецификаций функций и проектной информации на процессах ЖЦ создаются функциональные тесты, с помощью которых проводится тестирование с учетом требований, сформулированных на этапе анализа предметной области.

### 3.1 Методы тестирования интеграционных приложений.

Тот факт, что SOAP является протоколом, имеет огромное значение для тестирования: нужно изучить не только сам протокол, но и «первичные» стандарты и протоколы, на которых он базируется.

SOAP определяет формат сообщений, которыми обмениваются клиент и сервер; он также описывает детали того, как приложения обрабатывают определенные фрагменты сообщений. Например, некоторые элементы в заголовке позволяют создавать приложения, в которых сообщения сначала проходят через несколько промежуточных «станций», а только затем достигают конечного получателя.

Тестирование SOAP практически всегда подразумевает использование SoapUI.

SoapUI - это инструмент, который можно использовать для функционального и нефункционального тестирования. Он не ограничивается веб-службами, хотя это инструмент де-факто, используемый при тестировании веб-служб.

SoapUI богат следующими пятью аспектами:

- функциональное тестирование;
- тестирование безопасности;
- нагрузочное тестирование;
- протоколы и технологии;
- интеграция с другими инструментами.

Функциональное тестирование.

- SoapUI позволяет тестировщикам писать функциональные API-тесты в SoapUI.
- SoapUI поддерживает функцию Drag-Drop, которая ускоряет разработку скрипта.
- SoapUI поддерживает отладку тестов и позволяет тестировщикам разрабатывать управляемые данными тесты.
- SoapUI поддерживает несколько сред, облегчая переключение между средами QA, Dev и Prod.
- SoapUI позволяет выполнять расширенные сценарии (тестировщик может разрабатывать свой собственный код в зависимости от сценариев).

Тестирование безопасности.

- SoapUI выполняет полный набор сканирования уязвимостей.

- SoapUI предотвращает SQL-инъекцию для защиты баз данных.
- SoapUI сканирует переполнения стека, вызванные огромными по размеру документами.
- SoapUI сканирует межсайтовый скриптинг, который происходит, когда параметры сообщений отображаются в сообщениях.
- SoapUI выполняет фазинговое сканирование и сканирование границ, чтобы избежать ошибочного поведения сервисов.

#### Нагрузочное тестирование.

- SoapUI распределяет нагрузочные тесты по n числу агентов LoadUI.
- SoapUI с легкостью имитирует нагрузочное тестирование в больших объемах и в реальных условиях.
- SoapUI позволяет расширенные пользовательские отчеты для сбора параметров производительности.
- SoapUI позволяет осуществлять сквозной мониторинг производительности системы.

SoapUI поддерживает широкий спектр протоколов:

- SOAP – простой протокол доступа к объектам;
- WSDL – язык определения веб-сервисов;
- REST – представительский государственный трансферт;
- HTTP – протокол передачи гипертекста;
- HTTPS – протокол передачи гипертекста;
- AMF – формат сообщения о действии;
- JDBC – соединение с базой данных Java;
- JMS – служба сообщений Java.

Начинать тестирование надо на стадии написания документации, для этого необходимо использовать специализированные редакторы проверки схем. Самые известные – Oxygen и Altova, которые используются аналитиками при

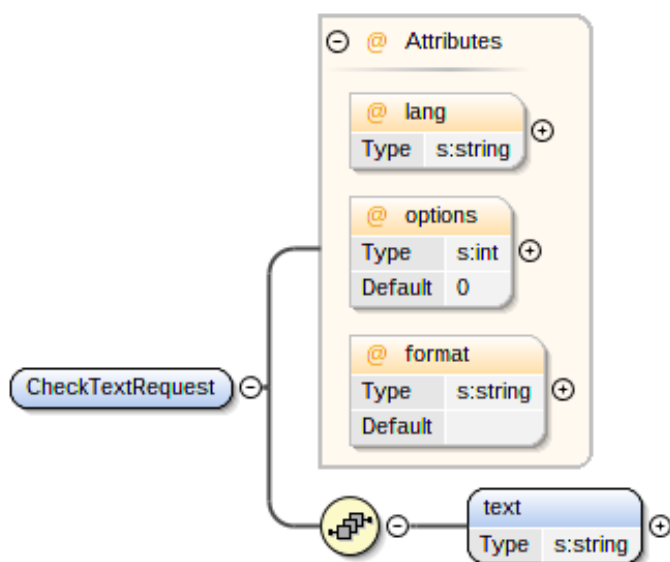
описании сервисов. Самые полезные функции редакторов, используемые при тестировании: визуализация XSD, генерация XML на основе XSD и валидация XML по XSD. Разберем, зачем они нужны.

## Визуализация XSD

Необходимая функция для наглядного представления схемы, которое дает возможность быстро извлечь обязательные элементы и атрибуты, а также существующие ограничения.

Например, для запроса `CheckTextRequest` обязательным является элемент `text`, а необязательными – все три атрибута (при этом у атрибута `options` установлено значение по умолчанию – ноль). (Рисунок 3.1)

Рисунок 3.1 – Пример визуализации XSD запроса



Использовать визуализацию очень удобно и даже необходимо при наличии большого количества типов и ограничений в схеме.

## Генерация XML на основе XSD

Данную функцию надо использовать для того, чтобы увидеть пример корректного сообщения. На практике часто используется для проверки нюансов работы ограничений.

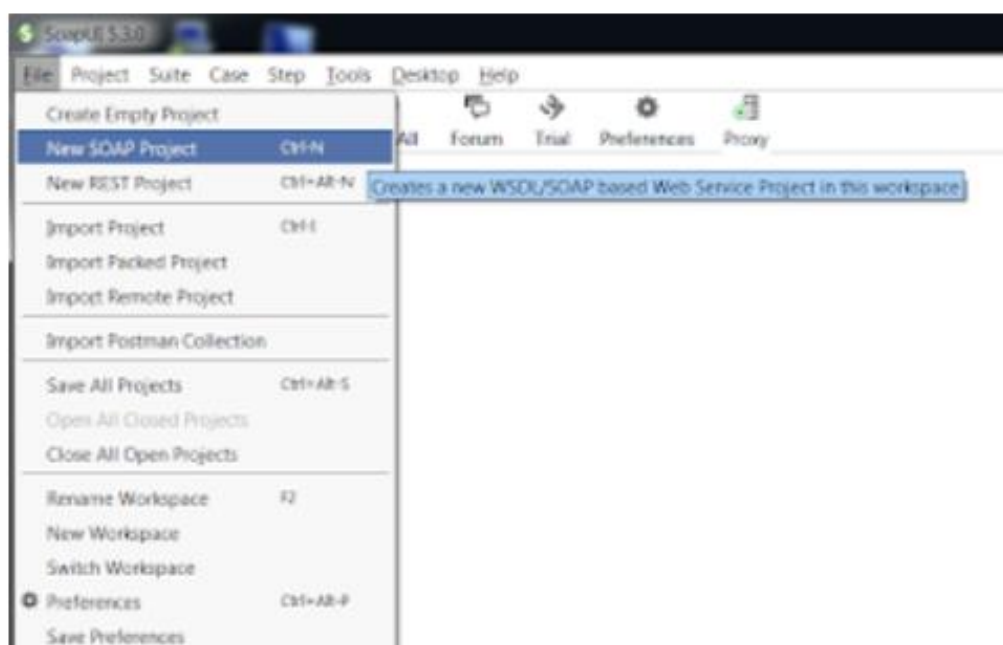
## Валидация XML по XSD

Использовать этот функционал полезно после генерации XML на основе XSD, то есть проверить сообщение на корректность. Эти две функции дают возможность отлавливать неочевидные дефекты в XSD еще на стадии разработки самого сервиса.

### 3.2 Пример тестирования в SOAP UI.

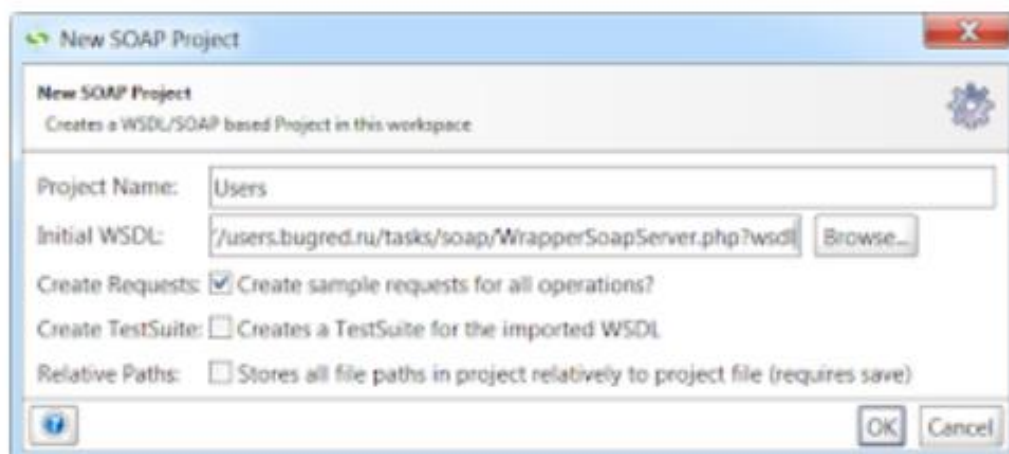
Запускаем SOAP UI и создаем новый файл.

Рисунок 3.1 – Создание нового проекта в SOAP UI



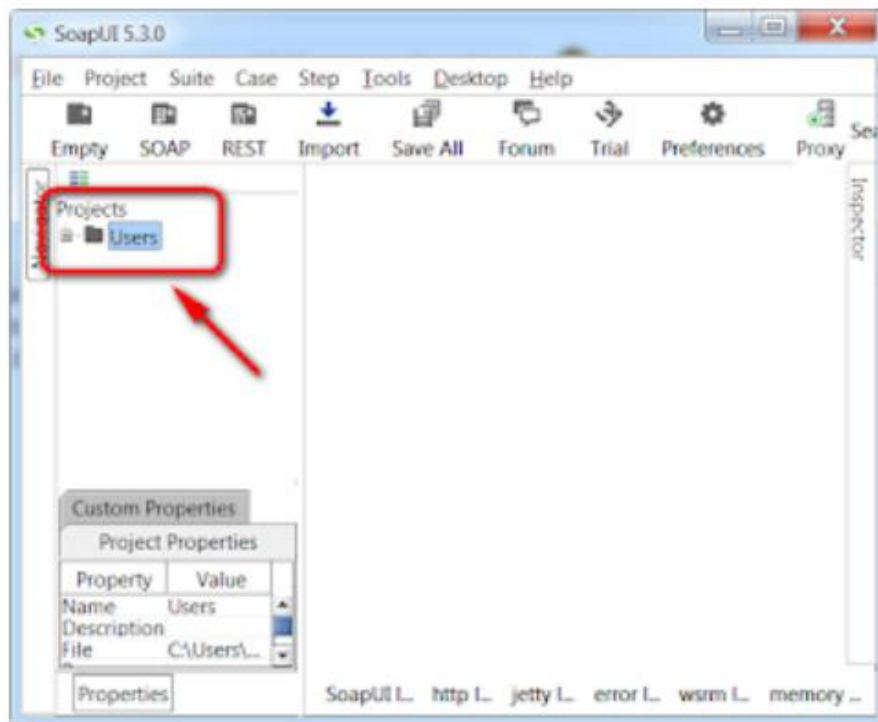
Указываем имя и WSDL

Рисунок 3.2 – Указание имени проекта и WSDL



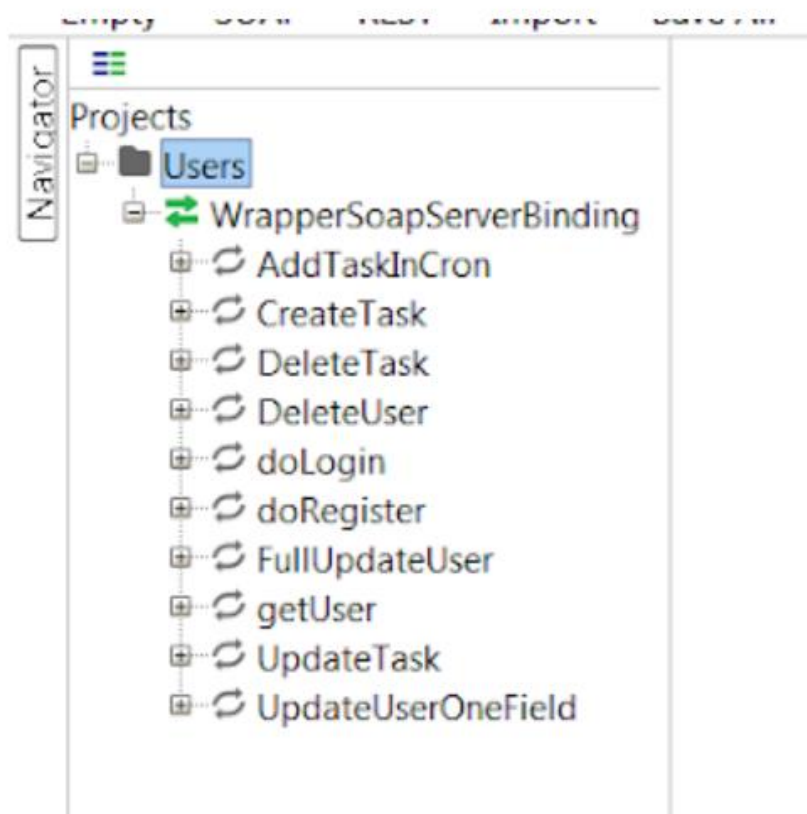
Создался проект

Рисунок 3.3 – Окно с проектом



Soap Ui по WSDL автоматически сгенерит проект и все доступные методы

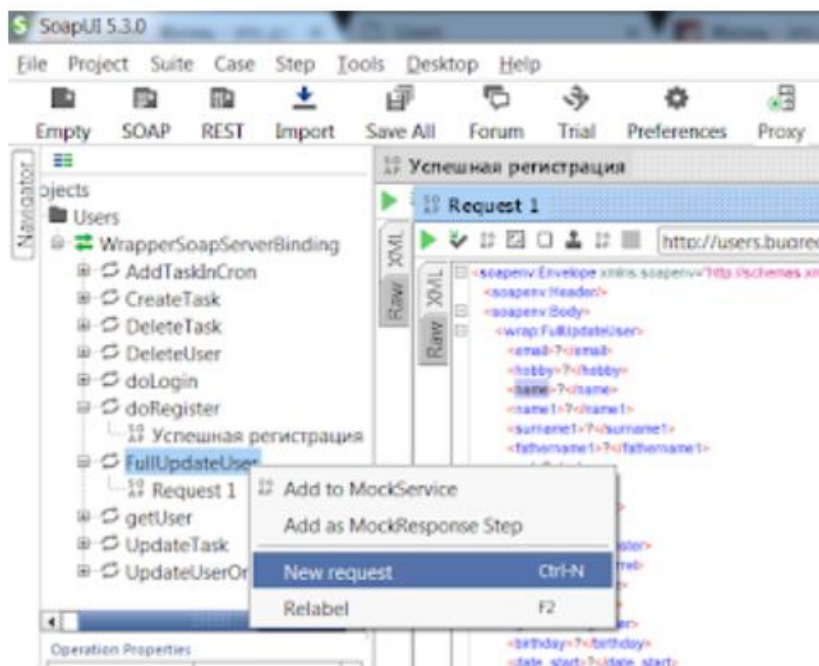
Рисунок 3.4 – Доступные для проекта методы





Чтобы создать новый запрос щелкаем правой кнопкой на название метода  
→ new request

Рисунок 3.1 – Создание запроса в SOAP UI



Это будет запрос, сформированный на основании XSD.

### 3.3 Оценка экономической эффективности инвестиций.

Проектируя любую информационную систему, необходимо обосновать экономическую эффективность ее внедрения. Без данного этапа невозможно ответить на вопрос об экономической целесообразности затрат на разработку системы.

Эффективность экономической информационной системы определяют сопоставлением результатов от функционирования информационной системы и затрат всех видов ресурсов, необходимых для ее создания и развития.

В тех случаях, когда результаты и затраты представлены в денежном выражении, говорят об экономической эффективности информационной системы. Причем оценка результатов и затрат проводится в пределах конкретного расчетного периода. В качестве начала расчетного периода, в пределах которого учитывают затраты, принимают год начала разработки информационной системы. Конец расчетного периода определяют в

соответствии со сроком морального старения технических средств и проектных решений системы.

#### Расчет экономических выгод

На первом шаге рассмотрим процесс увеличения объема обрабатываемых заявок. За счет автоматизации бизнес-правил и использования системы RTDM, как было отмечено в предыдущем пункте, Банк увеличивает объем выдаваемых кредитов на 25%. До использования системы RTDM Банк имел портфель розничных кредитов в 16 350 млн рублей, растущих на 13% в год.

Для определения размера увеличения объема выдаваемых кредитов за счет фиксированного темпа роста воспользуемся формулой 3.5.

$$V_i = V_0 \cdot (1 + g)^{i-1} \quad (3.5)$$

где:  $V_i$  - портфель розничных кредитов на  $i$ -ый год, руб;

$V_0$  - портфель розничных кредитов на начало рассматриваемого периода, руб;

$g$  - фиксированный рыночный темп роста.

Увеличение кредитного портфеля за счет использования SAS RTDM рассчитывается по формуле 3.6.

$$\Delta V_{i, rtdm} = V_i \cdot g_{rtdm} \quad (3.6)$$

где:  $\Delta V_{i, rtdm}$  - увеличение кредитного портфеля за счет использования SAS RTDM на  $i$ -ый год, руб;

$V_i$  - портфель розничных кредитов на  $i$ -ый год, руб;

$g_{rtdm}$  - процент увеличения объема выдаваемых кредитов за счет внедрения SAS RTDM.

Для определения прибыли за счет увеличения кредитного портфеля воспользуемся формулой 3.7.

$$\Pi_i = p \cdot \Delta V_{i, rtdm} \quad (3.7)$$

где:  $\Pi_i$  - прибыль за счет увеличения кредитного портфеля на  $i$ -ый год, руб;

$p$  - уровень прибыльности кредитного портфеля, %;

$\Delta V_{i, rtdm}$  - увеличение кредитного портфеля за счет использования SAS RTDM, руб.

Воспользовавшись формулами 3.5-3.7, получаем, что фиксированный рыночный тринадцатипроцентный ( $g = 13\%$ ) темп роста розничного кредитования плюс рост, обеспечиваемый за счет использования системы RTDM, позволит в течении трех лет увеличить кредитный портфель банка с 16350 млн рублей ( $V_0$ ) До более чем 26000 млн рублей. При уровне прибыльности в  $p = 2,5\%$ , рост доходов обеспечивает увеличение прибыли более чем на 345 млн рублей. Результаты расчетов приведены в таблице 3.4.

Таблица 3.4 – Расчет прибыли за счет увеличения количества выдаваемых кредитов

Показатель	Параметр	Первый год	Второй год	Третий год
Кредитный портфель	$V_i$	16 350 000 000 р.	18 475 500 000 р.	20 877 315 000 р.
Увеличение кредитного портфеля за счет использования SAS RTDM	$\Delta V_i^{rtdm}$	4 087 500 000 р.	4 618 875 00 р.	5 219 328 750 р.
Прибыль за счет увеличения кредитного портфеля	$\Pi_i$	102 187 500 р.	115 471 875 р.	130 483 246 р.

В результате использования RTDM, банк упростил свои предыдущие методы автоматизации бизнес-правил, для которых было необходимо привлечение ИТ-персонала. Бизнес подразделение, отвечающее за блок розничного кредитования, уменьшает свои расходы за счет того, что теперь при

внесении изменений в процесс принятия решений не требуется оплачивать услуги одного привлеченного ИТ специалиста. Рассчитаем сокращение расходов на ИТ персонал по формуле 3.8.

$$R_{IT} = 12 \cdot Wage \quad (3.8)$$

где:  $R_{IT}$  - сокращение расходов на ИТ персонал в год, руб;

$Wage$  - заработная плата, руб.

Затраты на привлечение ИТ-персонала сокращаются в размере 1 080 000 р в год (формула 10), т.е. сокращение затрат на 3 240 00 р за 3 года. Расчет производится из полной занятости сотрудника и заработной платой в 90 000 р в месяц.

В результате, общие выгоды за 3 года использования SAS RTDM превышают 350 млн рублей и выгоды, приведенные на момент рассмотрения, при ставке дисконтирования в 12,5 %, составляют более 276 млн рублей. Для подсчета приведенной выгоды использовалась формула 3.9:

$$PV_{profit} = \sum_{t=1}^N \frac{\Pi_i + R_{IT}}{(1+i)^t} \quad (3.9)$$

где:  $PV_{profit}$  – дисконтированные выгоды, руб;

$R_{IT}$  – сокращение расходов на ИТ персонал в год, руб

$\Pi_i$  – прибыль за счет увеличения кредитного портфеля на t-ый год, руб;

$i$  – ставка дисконтирования;

$t$  – период.

В таблице 3.5 представлtnы результаты расчета общих выгод.

Таблица 3.5 – Общие выгоды

Показатель	Параметр	Исходное значение	Первый год, руб	Второй год, руб	Третий год, руб	Итого, руб	PVprofit, руб

Продолжение таблицы 3.5 – Общие выгоды

Прибыль за счет увеличения кредитного портфеля	$\Pi_i$	0 р	102 187 500	115 471 875 р	130 483 246 р	348 142 621 р	273 712 924 р
Сокращение затрат на ИТ персонал	$R_{IT}$	0 р	1 080 000	1 080 00	1 080 00 0	3 240 00 0	2 571 85 2
Общие выгоды	$PV_{profit}$	0 р	103 267 50 0	116 551 875	131 563 264	351 382 621	276 284 776

Банк ежегодно оплачивает лицензионные сборы и обслуживание. В оплату лицензионных сборов последующих периодов включен процент, связанный с ежегодным повышением цен. В среднем годовой темп роста цен составляет 15%. Формула 3.10 используется для выполнения расчета оплаты лицензионных сборов:

$$L_i = L_0 \cdot (1 + infl)^{i-1} \quad (3.10)$$

где:  $L_i$  – затраты на лицензионные сборы за  $i$ -ый год, руб;

$L_0$  – затраты на лицензионные сборы за первый год, руб;

$infl$  – среднегодовой темп роста цен.

Размер оплаты технической поддержки составляет 18% от стоимости лицензии (ставка вендора) и рассчитывается по формуле 3.11.

$$TechSupport_i = L_i \cdot 0,18 \quad (3.11)$$

где:  $TechSupport_i$  – затраты на обслуживание за  $i$ -ый год, руб;

$L_i$  – затраты на лицензионные сборы за  $i$ -ый год, руб.

Дополнительно Банк понес капитальные расходы на покупку серверов в размере 8 175 000 р и консультационные услуги по установке, конфигурации и интеграции системы в размере 15 805 000 р.

Общие затраты на серверы, лицензии и консультационные услуги на по установке, конфигурации и интеграции рассчитываются по формуле 3.12.

$$PlatformCosts = TechSupport_i + L_i + Servers + Installs \quad (3.12)$$

где: *PlatformCosts* – затраты на серверы, лицензии и консультационные услуги на по установке, конфигурации и интеграции, руб;

*TechSupport<sub>i</sub>* – затраты на обслуживание за i-ый год, руб;

*Servers* – расходы на покупку серверов, руб;

*Installs* – консультационные услуги по установке, конфигурации и интеграции системы, руб;

*L<sub>i</sub>* – затраты на лицензионные сборы за i-ый год, руб.

Таблица 3.6 - Серверы, Лицензии и Консультационные услуги на по установке, конфигурации и интеграции

Показатель	Параметр	Разовый платеж, руб	Первый год, руб	Второй год, руб	Третий год, руб
Лицензионные сборы	<i>L<sub>i</sub></i>	0	5 995 000	6 894 250	7 928 388
Обслуживание (тех. поддержка)	<i>TechSupport<sub>i</sub></i>	0	1 079 100	1 240 965	1 427 110
Серверы для размещения RTDM	<i>Servers</i>	8 175 000	0	0	0
Услуги по установке, конфигурации и интеграции	<i>Installs</i>	15 850 000	0	0	0

Продолжение таблицы 3.6 - Серверы, Лицензии и Консультационные услуги на по установке, конфигурации и интеграции

Суммарные платформенные затраты	<i>PlatformCosts</i>	24 025 000	7 074 100	8 135 215	9 355 498
---------------------------------	----------------------	------------	-----------	-----------	-----------

Далее рассмотрим затраты на консультационные услуги по внедрению системы. Для выполнения работ по внедрению была привлечена команда из 4 человек, состоящая из двух консультантов, архитектора и менеджера проекта. Длительность внедрения 9 месяцев (185 рабочих дней). Проектная занятость архитектора 50%, руководителя проекта – 25%, консультанта – 100%. Дополнительно предполагается, что после внедрения в целях поддержки внедренной СПР привлекается 1 сотрудник сроком на 3 года. Требуемые для расчета параметры проектной команды приведены в таблице 3.7.

Таблица 3.7 - Ставки участников проектной команды

Роль	Ставка	Процент времени занятости	Число участников на консультации	Число участников на поддержке
Руководитель проекта	37300 р/день	50%	1	0
Архитектор	41500 р/день	25%	1	0
Консультант	29500 р/день	100%	2	1

Расчет итоговых значений выполняется по формуле 3.13.

$$StaffCosts = Num + Salary + T + Period \quad (3.13)$$

где: *StaffCosts* – стоимость услуг участников проекта, руб;

*Num* – число участников;

*Salary* – ставка, рублей/день;

*T* – процент времени занятости;

*Period* – длительность внедрения.

Общие затраты на консультационные услуги по внедрению системы отражены в таблице 15 и рассчитываются по формуле 3.14.

$$ConsultingCosts = \sum StaffCosts \quad (3.14)$$

где: *ConsultingCosts* – затраты на консультационные услуги по внедрению системы, руб;

*StaffCosts* – стоимость услуг участников проекта, руб.

Таблица 3.8 - Консультационные услуги на внедрение

Показатель	Параметр	Разовый плате, руб	Первый год, руб	Второй год, руб	Третий год
Стоимость услуг руководителя проекта	<i>StaffCosts</i>	3 450 250	0	0	0
Стоимость услуг архитектора	<i>StaffCosts</i>	1 919 375	0	0	0
Стоимость услуг консультанта	<i>StaffCosts</i>	10 915 000	7 268 500	7 268 500	7 268 500
Суммарные затраты на консалтинг	<i>ConsultingCo.</i>	16 284 625	7 268 500	7 268 500	7 268 500

В результате, общие затраты на внедрение и сопровождение системы в течение последующих трех лет составляет менее 87 млн рублей и затраты, приведенные на момент рассмотрения, при ставке дисконтирования в 12,5%, составляют менее 77 млн. рублей. Результаты отражены в таблице 16. Для подсчета приведенных затрат использовалась формула 3.15:



$$PV_{costs} = \sum_{t=1}^N \frac{ConsultingCosts_t + PlatformCosts_t}{(1+i)^t} \quad (3.15)$$

где:  $PV_{costs}$  – дисконтированные затраты, руб;

$ConsultingCosts_t$  – затраты на консультационные услуги по внедрению системы, руб;

$PlatformCosts_t$  – затраты на серверы, лицензии и консультационные услуги на по установке, конфигурации и интеграции, руб;

$i$  – ставка дисконтирования;

$t$  – период.

Таблица 3.9 - Консультационные услуги на внедрение

Показатель	Параметр	Исходное значение	Первый год	Второй год	Третий год	Итого	$PV_{costs}$
Серверы, Лицензии и Консультационные услуги на по установке, конфигурации и интеграции	<i>Platform Costs</i>	24 025 000р.	7 074 100р.	8 135 215р.	9 355 498р.	48 589 813 р.	43 311 578р.
Консультационные услуги на внедрение	<i>Consulting Costs</i>	16 284 625р.	7 268 500 р	7 268 500 р	7 268 500 р	38 090 125 р.	33 593 426р.
Общие затраты	<i>Total Costs</i>	40 309 625р.	14 342 600р.	15 403 715р.	16 623 998р.	86 679 938 р.	76 905 004р

## Расчет итоговых экономических показателей

Для анализа экономической эффективности от внедрения необходимо сравнить полученные результаты общих выгод с общими затратами. Итоговые значения приведены в таблице 3.10. Рассчитаем денежный поток, воспользовавшись формулой 3.16

$$CF_i = Profit_i - TotalCosts_i \quad (3.16)$$

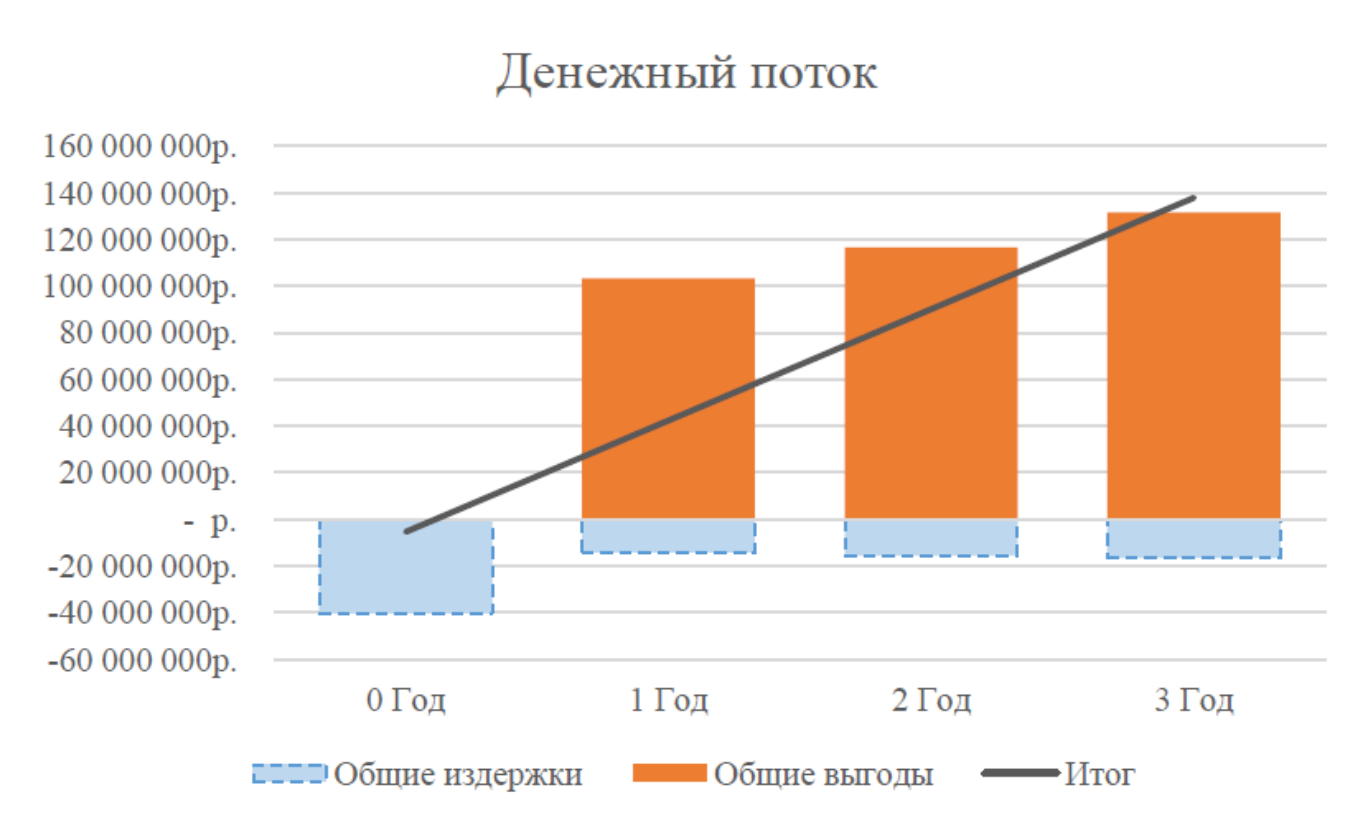
где:  $CF_t$  – денежный поток  $i$ -го периода, руб;;

$Profit_i$  – общие выгоды  $i$ -го периода, руб;;

$TotalCosts_i$  – общие затраты  $i$ -го периода, руб.

В результате положительный денежный поток получаем уже после первого года использования системы. Динамика роста денежного потока отражена на рисунке 3.1. Чистый денежный поток при ставке дисконтирования в 12,5% за 3 года достигает чуть менее 200 млн рублей, таким образом чистый денежный поток больше нуля, соответственно, при данной ставке дисконтирования внедрение системы является экономически эффективным.

Рисунок 3.1 - Денежный поток



Показатель окупаемости инвестиций (ROI) характеризует уровень доходности от владения активом. При показателе  $ROI > 100\%$  можно судить о том, что инвестиционные затраты эффективны. Математический расчет ROI можно представить формулой 3.17.

$$ROI = \frac{\sum_{i=1}^N CF_t - TotalCosts_0}{N \cdot TotalCosts_0} \cdot 100\% \quad (3.17)$$

где: ROI – показатель окупаемости инвестиций;

$CF_t$  – денежный поток  $i$ -го периода, руб;

$N$  – число периодов;

$TotalCosts_0$  – общие начальные затраты, руб.

Согласно расчетам по формуле 19 показатель  $ROI = 165\%$ , таким образом,  $ROI > 100\%$ , следовательно, инвестиционные затраты на внедрение системы SAS RTDM эффективны.

Также немаловажным показателем является период окупаемости инвестиций, рассчитываемой по формуле 3.18.

$$T_{co} = \frac{t \cdot TotalCosts_0}{(\sum_{t=1}^N CP_t) > 0} \cdot 12 \quad (3.18)$$

где:  $T_{co}$  – период окупаемости инвестиций, мес.;

$(\sum_{t=1}^N CP_t) > 0$  – денежный поток  $t$ -го периода, превышающий ноль

$N$  – число периодов

$t$  – число лет.

Общий денежный поток положителен уже после первого года использования системы, таким образом  $t=1$ , воспользовавшись формулой 20, получаем, что период окупаемости инвестиций составляет  $T_{co} = 5,4$  месяца.

Срок окупаемости — период времени, необходимый для того, чтобы доходы, генерируемые инвестициями, покрыли затраты на инвестиции.

Таблица 3.10 - Расчет итоговых экономических показателей

Показатель	Параметр	Исходное значение	Первый год	Второй год	Третий год	Итог	PV
Издержки	<i>Total Costs</i>	40 309 625р.	14 342 600р.	15 403 715р.	16 623 998р.	86 679 938р.	76 905 004р
Выгоды	<i>Profit</i>	0 р	103 267 500 р	116 551 875 р	131 563 264 р	351 382 621р.	276 284 776р.
Итоговый денежный поток	<i>CF</i>	-40 309 625,00р.	88 924 900,00 р.	101 148 160,00 р.	114 939 266,00 р.	264 702 683,00 р.	199 379 772,00 р.

### Выводы по разделу 3:

В данном разделе были рассмотрены методы тестирования разработанных интеграционных приложений. В ходе работы были произведены тестирование и отладка интеграционных приложений, в результате чего они являются полностью исправными и выполняют свои функции.

Также в третьей главе приведены результаты анализа оценки эффективности предложенной модели. В качестве оцениваемого показателя рассматривался параметр «пропускная способность процесса рассмотрения кредитной заявки». В результате использования системы пропускная способность увеличилась на 48%, что в свою очередь увеличивает суммарный объем выдаваемых кредитов на 25%.

Проведен экономический анализ и выполнен расчет экономической эффективности инвестиций в проект по внедрению системы, которые показали целесообразность внедрения на основании следующих результатов:

- чистый денежный поток положителен и составляет порядка 200 млн. рублей;
- показатель окупаемости инвестиций превышает 100% и составляет 168%;
- срок окупаемости составляет 5,4 месяца.

## ЗАКЛЮЧЕНИЕ

В бакалаврской выпускной квалификационной работе была поставлена цель — автоматизация и повышение эффективности бизнес-процессов кредитования физических лиц розничного банка за счет разработки программного модуля.

Результаты работы:

- исследована предметная область;
- выбрана платформа для реализации системы принятия решений;
- выбран язык программирования;
- разработана схема данных ПМ;
- разработана схема алгоритма ПМ;
- разработаны экранные формы пользовательского интерфейса;
- реализован программный модуль;
- проведено тестирование предлагаемых программных решений.

Также в ходе проведенной работы были получены следующие результаты:

- проведен анализ процесса кредитования клиентов и ИС, поддерживающих эти процессы в исследуемом Банке;
- обоснован выбор платформы для реализации системы принятия решений SAS RTDM при помощи метода экспертных оценок;
- проведен анализ существующего подходе к реализации системы принятия решений на основе платформы SAS RTDM;
- Предложен новый подход к реализации вспомогательных расчетов в процессе принятия решений. Время выполнения скриптов при таком подходе в 3 раза меньше относительно существующего подхода;
- Реализовано интеграционное приложение, позволяющее установить взаимодействие между фронт-офисной системой и SAS RTDM.

- Реализовано интеграционное приложение, позволяющее установить взаимодействие между системой принятия решений и сервисами Бюро кредитных историй.

Проведенные оценки экономической эффективности могут быть использованы практической деятельности в процессе продажи решений на базе SAS RTDM в качестве обоснования для заказчика при принятии решения о внедрении СПР.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Л.Г. Гагарина, С.Ю. Голова, Р.А. Касимов, В.М. Трояновский, Е.Л. Федотова Методические указания по выполнению выпускной квалификационной работы по направлению подготовки бакалавров 231000.62 "Программная инженерия" / Под ред. Л.Г. Гагариной. -М.: МИЭТ, 2014.
2. Балакина Р.Т. Кредитная политика коммерческого банка: Учебно-практическое пособие – Омск: Изд-во Ом. гос. ун-та, 2009г.
3. Борисов А.Б. Большой экономический словарь. — М.: Книжный мир, 2003г.;
4. Пухов А. «Продажи и управление бизнесом в розничном банке» / Москва: ЦИПСИР, КноРус, 2012г.
5. Горелая Н.В. «Организация кредитования в коммерческом банке»: учебное пособие. – М. : ИД «ФОРУМ» : ИНФРА-М, 2012г.
6. Ермаков С.Л., Юденков Ю.Н. Основы Организации деятельности коммерческого банка: Учебник.- М.: КНОРУС, 2009г.
7. Банковское дело: Учебник/под ред. проф. Колесникова В.И– М.: Финансы и статистика, 2010г.
8. Банковское дело: современная система кредитования : учебное пособие / О.И. Лаврушин, О.Н. Афанасьева, С.Л. Корниенко ; под ред. засл. деят. науки РФ, д-ра экон. наук, проф. О.И. Лаврушина. — 3-е изд., доп. - М. : КНОРУС, 2007г.
9. Вахрин П.И. Финансы и кредит: учеб. / П.И. Вахрин, А.С. Нешиной. - Москва: Дашков и К, 2009г.
10. Все кредиты России. Большая кредитная энциклопедия. М.: Русинвест, 2006г.
11. Закон РФ от 2 декабря 1990 г. N 395-1 (ред. от 07.05.2013) "О банках и банковской деятельности". Правовая система Консультант Плюс.
12. Лецкий Э.К. Информационные технологии на железнодорожном транспорте, УМК МПС России, 200г.
13. Краюхин, Г. А. Техничко-экономическое обоснование проектов: учебное пособие. - Санкт-Петербург: СПбГИЭУ, 2011.
14. Генри Бекет Java SOAP. – Москва: ЛОРИ 2004г.



15. Ньюкомер Э. Веб-сервисы. XML, WSDL, SOAP и UDDI. Для профессионалов.  
- Издательский дом "Питер"
16. Хабибуллин И. Самоучитель XML. – Москва. 2019г.
17. Малкольм Г. Программирование для Microsoft SQL Server 2000 с использованием XML. – Москва 2017г.
18. Интернет-ресурс с описанием тестирования протокола SOAP -  
<https://okiseleva.blogspot.com/2017/07/soap-soap-ui.html>
19. Курс SAS Real Time Decision Manager: создание и управление маркетинговыми кампаниями
20. Фастовский Э.Г. Сервис-ориентированные технологии интеграции информации.- 2011 г.
21. Интернет-ресурс с примерами запросов в формате SOAP  
<https://yandex.ru/dev/direct/doc/dg-v4/concepts/SOAP-docpage/>
22. Интернет-ресурс с примерами тестирования в SOAP UI  
<http://www.proghouse.ru/programming/20-soapui#>
23. Интернет-ресурс с кратким руководством по WSDL  
<https://coderlessons.com/tutorials/xml-tekhnologii/uznaite-wsdl/wsdl-kratkoe-rukovodstvo>
24. Бобби Вульф, Грегор Хоп Шаблоны интеграции корпоративных приложений
25. Гагарина Л.Г., Киселев Д.В., Федотова Е.Л. Разработка и эксплуатация автоматизированных информационных систем. Учебное пособие.- М: ИД «ФОРУМ» - ИНФРА-М, 2009
26. Черников Б.В. Методические рекомендации по подготовке квалификационной работы по направлению подготовки бакалавров 09.03.04 «Программная инженерия». / Б.В. Черников — М.: МИЭТ, 2016
- 27.ГОСТ Р 7.0.5-2008. «Справки по оформлению списка литературы».
- 28.ГОСТ 7.32-2001 «Отчёт о научно-исследовательской работе. Структура и правила оформления».
- 29.ГОСТ 19.701-90 (ISO 5807-85). «ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

- 30.ГОСТ 19.505-79. «ЕСПД. Руководство оператора. Требования к содержанию и оформлению».
- 31.ГОСТ 19.102-77. «ЕСПД. Стадии разработки».
- 32.Введение в тестирование [Электронный ресурс] - Режим доступа: <http://delta-course.org/docs/Delta2014S-T2-L5.pdf>
- 33.Интеграционное тестирование [электронный ресурс] - <http://www.protesting.ru/testing/levels/integration.html>
- 34.Гагарина Л.Г., Колдаев В.Д. Алгоритмы и структуры данных. -М.: Финансы и статистика; ИНФРА-М, 2009.
- 35.ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению»;
- 36.ГОСТ 19.781-90. Программное обеспечение систем обработки информации. Термины и определения;
- 37.Обзор языка программирования C# [Электронный ресурс] <http://www.microsoft.com/>;
- 38.Обзор языка программирования Java. [Электронный ресурс] <https://java.com>;
- 39.Баула В.Г., Основы программирования и алгоритмические языки: Учебное пособие / В.Г. Баула, Н.Д. Васюкова, В.В. Тюляева, П.М. Уманец. - М.: Энергоатомиздат, 2011.-312 с.;
- 40.Обзор среды разработки Eclipse . [Электронный ресурс] <https://eclipse.org/>;
- 41.Обзор среды разработки IntelliJIDEA [Электронный ресурс] <https://www.jetbrains.com/>;
- 42.Java [Электронный ресурс]. М., 2015. <http://www.oracle.com/technetwork/java/index.html>
43. Java [Электронный ресурс]. М., 2015. <https://ru.wikipedia.org/wiki/Java>
44. Сравнение C#, C++ и Java от создателя C#. [Электронный ресурс]. М., 2005-2015.: <http://it.icmp.ru/post/view/2792>
45. Бадд Т. Объектно-ориентированное программирование [Электронный ресурс]. М., 2014. [http://kit.znu.edu.ua/iLec/9sem/OOP/lit/Badd\\_T\\_-\\_obektno\\_orienntirovannoe\\_programmirovanie.pdf](http://kit.znu.edu.ua/iLec/9sem/OOP/lit/Badd_T_-_obektno_orienntirovannoe_programmirovanie.pdf)

- 46.ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.
- 47.Орлик С. Введение в программную инженерию и управление жизненным циклом программного обеспечения [Электронный ресурс] / Пер. с англ. SWEBOK URL: <http://sorlik.blogspot.ru/2005/07/swebok.html>
- 48.Гагарина Л.Г. Введение в архитектуру проектирования программного обеспечения: учебное пособие /Л.Г.Гагарина, А.Р.Федоров, П.А.Федоров. -М.: ИД«ФОРУМ»: ИНФРА-М, 2016.
- 49.Гордеев А. В. Операционные системы: Учебник для вузов. — 2-е изд. — СПб.: Питер, 2007.
- 50.Обзор IDE средств для программирования. [Электронный ресурс]. М., 2016.  
Режим доступа: <http://www.javaportal.ru/projects/taidej/results.html>
- 51.Тестирование программного обеспечения. Основные понятия и определения. [Электронный ресурс]. - Режим доступа: <http://www.protesting.ru/testing/>
- 52.Тестирование программного обеспечения. [Электронный ресурс]. М., 2016.  
Режим доступа: [https://ru.wikipedia.org/wiki/Тестирование\\_программного\\_обеспечения/](https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения/)

## ПРИЛОЖЕНИЕ 1

### РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ КРЕДИТОВАНИЯ РОЗНИЧНОГО БАНКА. (ПМ АВПК)

Техническое задание

## 1. Введение

Розничные банки постоянно находятся в поиске модернизации процессов кредитования. Основными задачами для банка являются увеличение количества привлеченных клиентов, а также снижение потерь и избежание закредитованности.

Для данных задач банк ищет способы автоматизаций этапов жизненного цикла заявки, с целью уменьшения времени на принятия решения по ней. Так же всегда важно само решение, которое при ручной обработке заявки не может исключать человеческого фактора и последующих ошибок, связанных с неправильными действиями банковских сотрудников, отвечающих за конкретную область жизненного цикла заявки. Тема работы является актуальной ввиду того, что автоматизации технологических процессов является на текущий момент одним из ключевых звеньев в банковском деле.

Объектом исследования являются технологии основных процессов кредитования физических лиц крупного банка.

Предметом исследования являются технологии оптимизации и автоматизации процессов кредитования, которые основываются на технологиях компании SAS Institute.

## 2. Основания для разработки

### 2.1. Основание для разработки

задание на выпускную работу;  
решение зав. Института СПИНТех.

### 2.2. Наименование разработки:

«Разработка программного модуля для автоматизации процессов кредитования на основе системы SAS RTDM».

### 2.3. Исполнитель:

Исполнителем является студент группы МП-41 НИУ «МИЭТ» Карасева Веслава Эдуардовна.

## 3. Назначение разработки

Данный ПМ создается с целью является повышение эффективности процессов кредитования физических лиц.

#### **4. Технические требования**

##### **4.1. Требования к функциональным характеристикам**

###### **4.1.1. Состав выполняемых функций**

Создаваемый ПМ должен обеспечивать выполнение следующих функций:

Получение данных клиента из БД для заполнения в параметров запроса в CRE;

Получение аутентификационных данных для доступа к CRE (логин и пароль);

Получение идентификатора запроса и идентификатора кредитной заявки.

Сохранение данных запроса, идентификатора запроса, идентификатора кредитной заявки в базу данных;

Отправка запроса в CRE;

Получение ответа CRE;

Генерация ответа CRE, демаршалинг и запись кредитного отчета в базу данных;

Генерация кода возврата сервиса;

Логирование действий;

###### **4.1.2. Организация входных и выходных данных**

*Входные данные*

Тип вызова, идентификатор обращения к SAS RTDM в рамках обработки одной заявки, тип используемой стратегии, структура с параметрами заявки.

*Выходные данные*

Идентификатор обращения к SAS RTDM в рамках обработки одной заявки, структура с параметрами ответа

##### **4.2. Требования к надежности**

Работа ПМ не должна приводить к фатальным сбоям операционной системы.

ПМ должен работать с входными данными, предусмотренными техническими требованиями в соответствии с алгоритмом функционирования, выдавать сообщения об ошибках при неверно заданных исходных данных и прочих

нештатных ситуациях, поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей.

#### 4.3. Условия эксплуатации

Персонал, использующий ПМ, должен обладать навыками работы с компьютером, навыками работы с системой SAS RTDM.

#### 4.4. Требования к составу и параметрам технических средств

В состав технических средств должен входить компьютер на базе ОС Windows XP и выше, включающий в себя:

процессор с частотой не менее 1500 МГц;

оперативную память DDR не менее 1Гб;

дисковое пространство не менее 500 Мб.

сетевую карту

**Дополнительное техническое оснащение.** Для проведения демонстрации необходимо наличие следующих технических средств:

Компьютера, включающего в себя:

процессор с частотой не менее 1500 МГц;

оперативную память DDR не менее 1 Гб;

жесткий диск 1 Гб;

сетевую карту;

графический адаптер

Монитора с разрешением не менее 1280x1024 (для обеспечения комфортности восприятия),

Установленной программы SAS RTDM

Установленной программы SAS CI Studio

#### 4.5. Требования к информационной и программной совместимости

Базовые языки программирования: Java, Groovy, среда разработки Eclipse. ПМ должен работать под системой Windows XP и выше.

#### 4.6. Специальные требования

Специальных требований к характеристикам программы не предъявляется.

#### 5. Требования к программной документации

### 5.1. Требования к составу программной документации

В комплект документации должны входить: руководство оператора.

### 5.2. Требования к оформлению документации

Программная документация должна быть разработана и оформлена в соответствии с ЕСПД.

### 6. Порядок контроля и приёмки

Контроль и приёмка разработки осуществляются на устройстве заказчика.

Проверяется выполнение всех ранее заявленных функций ПМ.

### 7. Стадии и этапы разработки

В течение периода с февраля 2019 года по июнь 2019 года должны быть проведены следующие работы:

Наименование работ	Сроки исполнения
Изучение предметной области, обзор литературы и существующих аналогов, разработка обобщенных структур данных, основных алгоритмов	11.02.2019 – 15.02.2019
Предварительная разработка структуры входных и выходных данных	16.02.2019 – 01.03.2019
Уточнение структуры входных и выходных данных, определение формы представления отчетов, разработка структуры ПМ (в рамках технического проекта)	02.03.2019 – 20.03.2019
Программирование и отладка ПМ	21.03.2019 – 21.04.2019
Доработка ПМ, согласование и утверждение методики испытаний, проведение предварительных испытаний, корректировка ПМ с учетом испытаний	22.04.2019 – 18.05.2019
Составление пояснительной записки	19.05.2019-21.05.2019



Подготовка слайдов	22.05.2019-23.05.2019
Внедрение, подготовка и передача ПМ заказчику	24.05.2019 – 31.05.2019

## ПРИЛОЖЕНИЕ 2

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ АВТОМАТИЗАЦИИ  
ПРОЦЕССОВ КРЕДИТОВАНИЯ РОЗНИЧНОГО БАНКА.  
(ПМ АВПК)

РУКОВОДСТВО ПРОГРАММИСТА

## АННОТАЦИЯ

В данном программном документе приведено руководство программиста по использованию ПМ АВПК.

В разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые программным модулем, условия, необходимые для выполнения.

В разделе «Характеристики программы» приведено описание основных характеристик и особенностей ПМ.

В разделе «Обращение к программе» приведено описание обращения к ПМ.

В разделе «Входные и выходные данные» приведено описание организации используемой входной и выходной информации.

В разделе «Сообщения» указаны тексты сообщений, выдаваемых программисту в ходе выполнения программы, описание их содержания и действия, которые необходимо предпринять по этим сообщениям.

Оформление программного документа «Руководство программиста» произведено по требованиям ЕСПД (ГОСТ 19.101-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.504-79).

## СОДЕРЖАНИЕ

1.	НАЗНАЧЕНИЕ И УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ ....	77
1.1.	Функциональное назначение .....	77
1.2.	Эксплуатационное назначение .....	77
1.3.	Требования к аппаратным средствам .....	77
1.4.	Требования к программному обеспечению.....	78
1.5.	Требования к программисту .....	78
2.	ХАРАКТЕРИСТИКИ ПРОГРАММЫ.....	79
3.	ОБРАЩЕНИЕ К ПРОГРАММЕ .....	80
4.	ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ .....	82
5.	СООБЩЕНИЯ .....	83
	ПЕРЕЧЕНЬ СОКРАЩЕНИЙ .....	<b>Error! Bookmark not defined.</b>

## 1. НАЗНАЧЕНИЕ И УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

### 1.1. Функциональное назначение

ПМ АВПК выполняет следующие функции:

- Получение данных клиента из БД для заполнения в параметров запроса в CRE;
- Получение аутентификационных данных для доступа к CRE (логин и пароль);
- Получение идентификатора запроса и идентификатора кредитной заявки. При этом, идентификатора запроса генерируется для каждого запроса в БКИ создается уникальный, а идентификатора кредитной заявки – уникальный на одну обрабатываемую заявку;
- Сохранение данных запроса, идентификатора запроса, идентификатора кредитной заявки, RTDMID в базу данных;
- Отправка запроса в CRE;
- Получение ответа CRE;
- Генерация ответа CRE (CREID), демаршалинг и запись кредитного отчета в базу данных;
- Генерация кода возврата сервиса (REPLYCD);
- Логирование действий;

### 1.2. Эксплуатационное назначение

Основным назначением ПМ АВПК является автоматизация процесса принятия решения по кредитным заявкам.

### 1.3. Требования к аппаратным средствам

Минимальный состав технических средств и их технические характеристики:

- процессор 1.6 ГГц или выше;
- ОЗУ 1 ГБ;
- жесткий диск HDD, 5400 об/мин;
- объем доступного пространства на жестком диске не менее 10 ГБ;
- видеоадаптер с поддержкой DirectX 9 и разрешения экрана 1024x768;
- устройства ввода/вывода: мышь, клавиатура, монитор.

#### 1.4. Требования к программному обеспечению

ПМ АВПК работает в системах под управлением ОС Windows, начиная с версии 7. Необходимо наличие Java EE.

#### 1.5. Требования к программисту

Программист, использующий ПМ АВПК, должен владеть языками программирования Java, Groovy. А так же владение SAS RTDM.

## 2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

Интеграционное приложение будет являться клиентом для веб-сервиса SAS RTDM с одной стороны, а с другой фронт-офисная система будет являться клиентом интеграционного приложения.

Интеграционное приложение реализовано при помощи платформы Java Enterprise Edition с использованием открытых библиотек:

- Библиотека MyBatis – предназначена для автоматизации работы между Java-классами и СУБД;
- Библиотека CXF – предназначена для разработки веб-сервисов.
- Log4j – инструмент логирования действий.

Интеграционное приложение разворачивается в виде SOAP веб-сервиса на сервер приложений SAS Web Application Server (изначально разработан на базе Apache WebAppServer).

У приложения должен быть один метод, который будет использоваться для принятия запроса от фронт-офисной системы и возврата ответа обратно.

Так же для реализации интеграционного приложения нам понадобятся:

- XSD запроса
- XSD ответа
- WSDL события в SAS RTDM

Интеграционное приложение реализовано при помощи платформы Java Enterprise Edition с использованием открытых библиотек:

- Библиотека MyBatis – предназначена для автоматизации работы между Java-классами и БД;
- Библиотеки CXF – предназначены для автоматизированной передачи и получения ответа от веб-сервиса.
- Log4j – логгер, позволяющий контролировать процесс выполнения программы, с возможностью записи в разные источники.
- Apache Tomcat – позволяет опубликовать свое приложение на веб-сервере.

Интеграционное приложение разворачивается в виде SOAP веб-сервиса на сервер Apache Tomcat.

### 3. ОБРАЩЕНИЕ К ПРОГРАММЕ

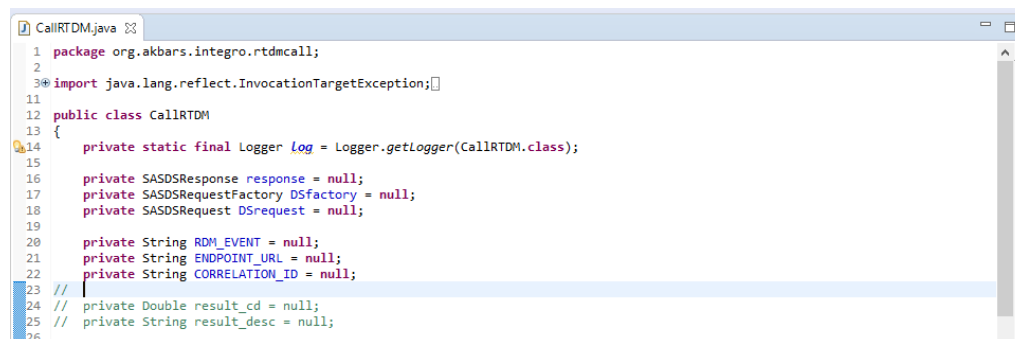
Интеграционные веб-сервисы доступны по адресам:

[https://server\\_hostname:8996/SASIntegro/services](https://server_hostname:8996/SASIntegro/services)

[https://server\\_hostname:8996/SASCRE/services](https://server_hostname:8996/SASCRE/services)

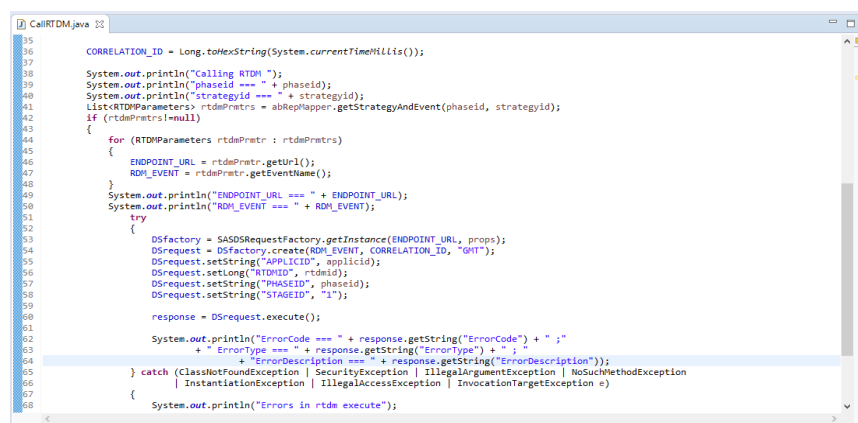
Первое интеграционное приложение было разработано с помощью JavaApi, которое позволяло вызывать SAS RTDM с помощью методов классов из библиотек.

Рисунок 2.6 - Объявление классов для вызова SAS RTDM в первом интеграционном приложении



```
1 package org.akbars.integro.rtdmcall;
2
3 import java.lang.reflect.InvocationTargetException;
4
11
12 public class CallRTDM
13 {
14     private static final Logger Log = Logger.getLogger(CallRTDM.class);
15
16     private SASDSResponse response = null;
17     private SASDSRequestFactory DSfactory = null;
18     private SASDSRequest DSrequest = null;
19
20     private String RDM_EVENT = null;
21     private String ENDPOINT_URL = null;
22     private String CORRELATION_ID = null;
23
24     // private Double result_cd = null;
25     // private String result_desc = null;
26 }
```

Рисунок 2.7 - Вызов SAS RTDM в первом интеграционном приложении



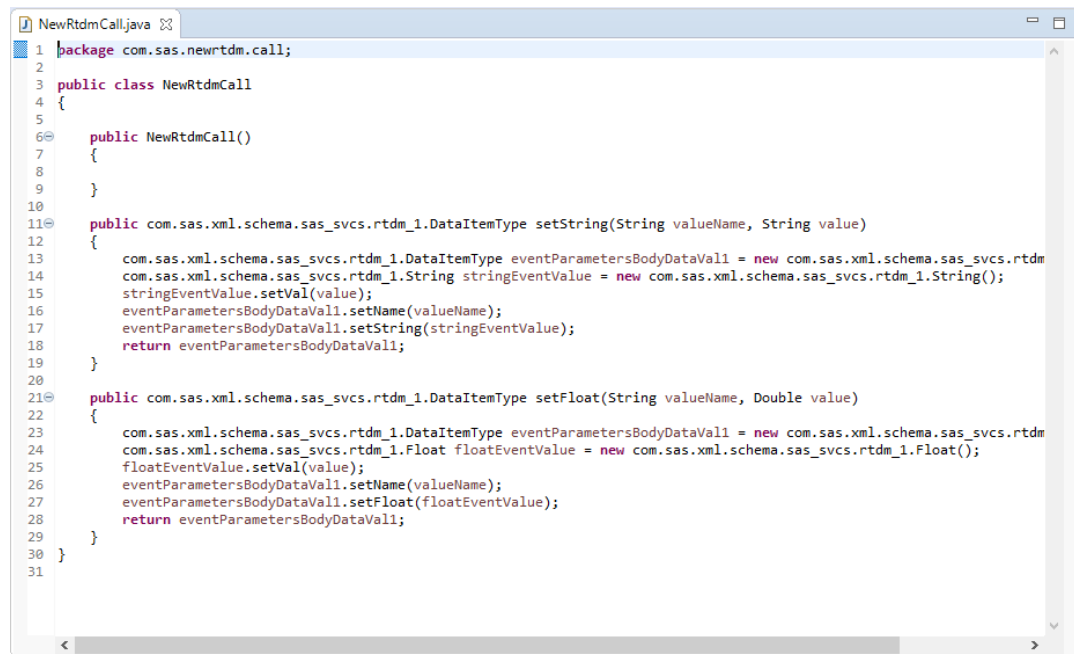
```
35
36 CORRELATION_ID = Long.toHexString(System.currentTimeMillis());
37
38 System.out.println("calling RTDM");
39 System.out.println("phaseid == " + phaseid);
40 System.out.println("strategyid == " + strategyid);
41 List<RTDMParameters> rtdmPrmtrs = abRepMapper.getStrategyAndEvent(phaseid, strategyid);
42 if (rtdmPrmtrs != null)
43 {
44     for (RTDMParameters rtdmPrmtr : rtdmPrmtrs)
45     {
46         ENDPOINT_URL = rtdmPrmtr.getUri();
47         RDM_EVENT = rtdmPrmtr.getEventName();
48     }
49     System.out.println("ENDPOINT_URL == " + ENDPOINT_URL);
50     System.out.println("RDM_EVENT == " + RDM_EVENT);
51     try
52     {
53         DSfactory = SASDSRequestFactory.getInstance(ENDPOINT_URL, props);
54         DSrequest = DSfactory.create(RDM_EVENT, CORRELATION_ID, "GMT");
55         DSrequest.setString("APPLICID", applicid);
56         DSrequest.setLong("RTDMID", rtdmid);
57         DSrequest.setString("PHASEID", phaseid);
58         DSrequest.setString("STAGEID", "1");
59         response = DSrequest.execute();
60         System.out.println("ErrorCode == " + response.getString("ErrorCode") + " ;"
61             + " ErrorType == " + response.getString("ErrorType") + " ;"
62             + " ErrorDescription == " + response.getString("ErrorDescription"));
63     } catch (ClassNotFoundException | SecurityException | IllegalArgumentException | NoSuchMethodException
64         | InstantiationException | IllegalAccessException | InvocationTargetException e)
65     {
66         System.out.println("Errors in rtdm execute");
67     }
68 }
```

Второе интеграционное приложение вызывало SAS RTDM, как клиент веб-сервиса, динамически заполняя входящие параметры для события стратегии, формируя классы и отправляя их на адрес развернутого приложения SAS RTDM с помощью методов, которые были сгенерированы для клиента веб-сервиса, по WSDL события SAS RTDM. Данный подход позволил включить



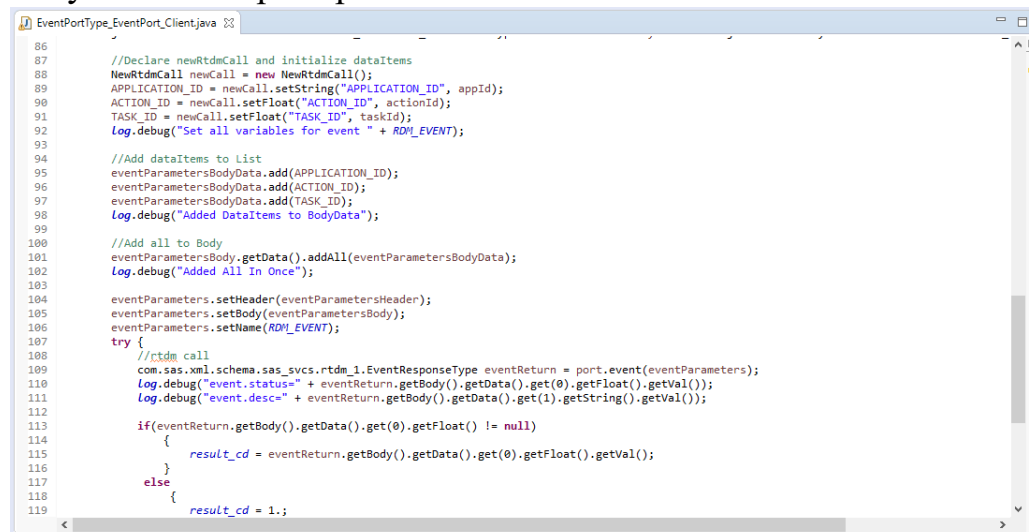
логирование, что помогло качественнее отлавливать ошибки и обеспечило прозрачность процесса, контроль за его ходом.

Рисунок 2.8 - Класс, в котором динамически заполняются параметры для события



```
1 package com.sas.newrtdm.call;
2
3 public class NewRtdmCall
4 {
5
6     public NewRtdmCall()
7     {
8
9     }
10
11     public com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType setString(String valueName, String value)
12     {
13         com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType eventParametersBodyDataVal1 = new com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType();
14         com.sas.xml.schema.sas_svcs.rtdm_1.String stringEventValue = new com.sas.xml.schema.sas_svcs.rtdm_1.String();
15         stringEventValue.setVal(value);
16         eventParametersBodyDataVal1.setName(valueName);
17         eventParametersBodyDataVal1.setString(stringEventValue);
18         return eventParametersBodyDataVal1;
19     }
20
21     public com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType setFloat(String valueName, Double value)
22     {
23         com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType eventParametersBodyDataVal1 = new com.sas.xml.schema.sas_svcs.rtdm_1.DataItemType();
24         com.sas.xml.schema.sas_svcs.rtdm_1.Float floatEventValue = new com.sas.xml.schema.sas_svcs.rtdm_1.Float();
25         floatEventValue.setVal(value);
26         eventParametersBodyDataVal1.setName(valueName);
27         eventParametersBodyDataVal1.setFloat(floatEventValue);
28         return eventParametersBodyDataVal1;
29     }
30 }
31
```

Рисунок 2.8 - Пример заполнения классов и вызов SAS RTDM



```
86
87 //Declare newRtdmCall and initialize dataItems
88 NewRtdmCall newCall = new NewRtdmCall();
89 APPLICATION_ID = newCall.setString("APPLICATION_ID", appId);
90 ACTION_ID = newCall.setFloat("ACTION_ID", actionId);
91 TASK_ID = newCall.setFloat("TASK_ID", taskId);
92 Log.debug("Set all variables for event " + RDM_EVENT);
93
94 //Add dataItems to List
95 eventParametersBodyData.add(APPLICATION_ID);
96 eventParametersBodyData.add(ACTION_ID);
97 eventParametersBodyData.add(TASK_ID);
98 Log.debug("Added DataItems to BodyData");
99
100 //Add all to Body
101 eventParametersBody.getData().addAll(eventParametersBodyData);
102 Log.debug("Added All In Once");
103
104 eventParameters.setHeader(eventParametersHeader);
105 eventParameters.setBody(eventParametersBody);
106 eventParameters.setName(RDM_EVENT);
107 try {
108     //rtdm call
109     com.sas.xml.schema.sas_svcs.rtdm_1.EventResponseType eventReturn = port.event(eventParameters);
110     Log.debug("event.status=" + eventReturn.getBody().getData().get(0).getFloat().getVal());
111     Log.debug("event.desc=" + eventReturn.getBody().getData().get(1).getString().getVal());
112
113     if(eventReturn.getBody().getData().get(0).getFloat() != null)
114     {
115         result_cd = eventReturn.getBody().getData().get(0).getFloat().getVal();
116     }
117     else
118     {
119         result_cd = 1.;
120     }
121 }
122
```

#### 4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Запрос к интеграционному приложению от фронт-офисной системы происходит в синхронном режиме. На вход подается сложная структура, описанная в xsd-файле запроса. На выход подается так же сложная структура, которая описана в xsd-файле ответа.

Ниже приведен общий алгоритм выполнения запроса:

- Генерация/получение идентификатора запроса для SAS RTDM;
- Парсинг данных, полученных из заявки в операционную базу SAS RTDM;
- Вызов события стратегии в SAS RTDM с помощью динамического заполнения атрибутов этого события;
- Получения ответа от SAS RTDM;
- Обработка ответа от SAS RTDM;
- Сбор ответной xml в соответствии с классами, сгенерированными по xsd ответа;
- Отправка ответа в фронт-офисную систему;
- Логирование действий;

## 5. СООБЩЕНИЯ

В процессе работы модуля программисту/оператору могут выдаваться сообщения об ошибках.

- 1) Ошибка «Ошибка инициализации подключения к БД перед сохранением запроса!. Код ошибки: <код\_ошибки>. Описание ошибки: <описание\_ошибки>».

Решение: проверить подключение к БД.

- 2) Ошибка «Ошибка при записи параметров запроса в БД! Код ошибки: <код\_ошибки>. Описание ошибки: <описание\_ошибки>».

Решение: проверить корректность запроса.

- 3) Ошибка «Ошибка при получении значений параметров AppId, ActionId, TaskId из входного XML сообщения! Код ошибки: <код\_ошибки>. Описание ошибки: <описание\_ошибки>».

Решение: убедиться в корректности содержания XML файла.

- 4) Ошибка «Ошибка при вызове РТДМ. При обработке заявки возникли ошибки RTDM! Код ошибки: <код\_ошибки>. Описание: <описание\_ошибки>».

Решение: убедиться в корректности инициализации API вызова RTDM.

## ПРИЛОЖЕНИЕ 3

### ПРОГРАММНЫЙ МОДУЛЬ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССОВ КРЕДИТОВАНИЯ РОЗНИЧНОГО БАНКА

Текст программы

ПМ АВПК

## Содержание

1. Программный код интеграции сервисами БКИ.....	86
1.Запросы отчетов ВБКИ, НБКИ, ОКР .....	87
2.Проверки принадлежности ФЛ к бизнесам, ИП по ФИО и ИНН.....	90
2. Программный код интеграции фронт-офисной системы с SAS RTDM .....	93
1.Обработка анкетных данных клиента.....	95
2.Подключение к БД .....	97
3.Вызов SAS RTDM.....	100
4.Запись в БД .....	100

## 1. Программный код интеграции сервисами БКИ

```
/**
 *
 * @author SAS1
 */

@WebService(targetNamespace = "http://my.org/ns/")
public interface CRE {

    @WebMethod(operationName = "getIBCHResponse", action =
"urn:GetIBCHResponse")

    IntegrationResponse getIBCHResponse(@WebParam(name = "request_id") int
rtdmid, @WebParam(name = "applicant_id") String customerid);

    @WebMethod(operationName = "getNBCHResponse", action =
"urn:GetNBCHResponse")

    IntegrationResponse getNBCHResponse(@WebParam(name = "request_id") int
rtdmid, @WebParam(name = "applicant_id") String customerid);

    @WebMethod(operationName = "getUCBResponse", action = "urn:GetUCBResponse")

    IntegrationResponse getUCBResponse(@WebParam(name = "request_id") int
rtdmid, @WebParam(name = "applicant_id") String customerid);

    @WebMethod(operationName = "joinResponseByCreId", action =
"urn:JoinResponseByCreId")

    IntegrationResponse joinResponseByCreId(@WebParam(name = "cre_id")
ArrayList<Integer> creidList, @WebParam(name = "request_id") int rtdmid,

        @WebParam(name = "applicant_id") String customerid);

    @WebMethod(operationName = "getGroupRequestNBCHandUCB", action =
"urn:GetGroupRequestNBCHandUCB")

    IntegrationResponse getGroupRequestNBCHandUCB(@WebParam(name =
"request_id") int rtdmid,

        @WebParam(name = "applicant_id") String customerid);

    @WebMethod(operationName = "getCompanyListByPersonINN", action =
"urn:getCompanyListByPersonINN")

    IntegrationResponse getCompanyListByPersonINN(@WebParam(name =
"request_id") int rtdmid,

        @WebParam(name = "applicant_id") String customerid);
```

```

        @WebMethod(operationName = "getCompanyListByPersonFIO", action =
"urn:getCompanyListByPersonFIO")

        IntegrationResponse getCompanyListByPersonFIO(@WebParam(name =
"request_id") int rtdmid,

                @WebParam(name = "applicant_id") String customerid);

        @WebMethod(operationName = "getCompanyExtendedReport", action =
"urn:getCompanyExtendedReport")

        IntegrationResponse getCompanyExtendedReport(@WebParam(name =
"request_id") int rtdmid,

                @WebParam(name = "applicant_id") String customerid);

        @WebMethod(operationName = "getEntrepreneusShortReport", action =
"urn:getEntrepreneusShortReport")

        IntegrationResponse getEntrepreneusShortReport(@WebParam(name =
"request_id") int rtdmid,

                @WebParam(name = "applicant_id") String customerid);
    }

    /**
     *
     * @author SAS1
     */

    @WebService(endpointInterface = "com.sas.creclient.CRE",
serviceName = "CRE", targetNamespace = "http://my.org/ns/")
    public class CREImpl implements CRE {

        public static final Logger log = Logger.getLogger(CREImpl.class);

```

### 1.3 Запросы отчетов ВБКИ, НБКИ, ОКР

```

    @Override

    public IntegrationResponse getIBCHResponse(int rtdmid, String customerid) {

        log.info("Start getIBCHResponse; request_id = " + rtdmid + ";
applicant_id = " + customerid);

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("1011");

        String reportType = "Отчет ВБКИ";

```

```

        IntegrationResponse response =

            (new CreRequestSender()).execProcessRequest(rtdmid,
customerid, reportType, connCode);

        log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

        return response;
    }

    @Override

    public IntegrationResponse getNBCHResponse(int rtdmid, String customerid) {

        log.info("Start getNBCHResponse; request_id = " + rtdmid + ";
applicant_id = " + customerid);

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("0");

        connCode.getSubRequestCode().add("1");

        String reportType = "Отчет НКН";

        IntegrationResponse response =

            (new CreRequestSender()).execProcessRequest(rtdmid,
customerid,reportType, connCode);

        log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

        return response;
    }

    @Override

    public IntegrationResponse getUCBResponse(int rtdmid, String customerid) {

        log.info("Start getUCBResponse; request_id = " + rtdmid + "; applicant_id
= " + customerid);

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("6");

        connCode.getSubRequestCode().add("0");

        String reportType = "Отчет ОКБ";

        IntegrationResponse response =

            (new CreRequestSender()).execProcessRequest(rtdmid,
customerid, reportType, connCode);

        log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

        return response;
    }

    @Override

```



```

        public IntegrationResponse joinResponseByCreId(ArrayList<Integer>
creidList, int rtdmid,

                String customerid) {

            log.info("Start joinResponseByCreId; request_id = " + rtdmid + ";
applicant_id = " + customerid + "; creidList" + creidList.toString());

            IntegrationResponse response = (new
CreRequestSender()).execJoinUidResponse(creidList, rtdmid, customerid);

            log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

            return response;

        }

        @Override

        public IntegrationResponse getGroupRequestNBCHandUCB(int rtdmid, String
customerid) {

            log.info("Start getGroupRequestNBCHandUCB; request_id = " + rtdmid + ";
applicant_id = " + customerid);

            ArrayList<ConnectorCode> connCodeList = new ArrayList<ConnectorCode>();

            ConnectorCode connCodeNBCH = new ConnectorCode();

            connCodeNBCH.setConnectorCode("0");

            connCodeNBCH.getSubRequestCode().add("1");

            connCodeList.add(connCodeNBCH);

            ConnectorCode connCodeUCB = new ConnectorCode();

            connCodeUCB.setConnectorCode("6");

            connCodeUCB.getSubRequestCode().add("0");

            connCodeList.add(connCodeUCB);

            String reportType = "Сводный отчет НБКИ и ОКБ";

            IntegrationResponse response =

                (new CreRequestSender()).execGroupRequest(rtdmid, customerid,
reportType, connCodeList);

            log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

            return response;

        }

```

## 2.Проверки принадлежности ФЛ к бизнесам, ИП по ФИО и ИНН

```
@Override

    public IntegrationResponse getCompanyListByPersonINN(int rtdmid, String
customerid)

    {

        log.info("Start getCompanyListByPersonINN; request_id = " + rtdmid
+ "; applicant_id = " + customerid);

        ArrayList<ConnectorCode> connCodeList = new
ArrayList<ConnectorCode>();

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("22");

        connCode.getSubRequestCode().add("31");

        connCodeList.add(connCode);

        String reportType = "Проверка по ИНН ФЛ принадлежности к участнику
бизнеса";

        IntegrationResponse response =

            (new CreRequestSender()).execGroupRequestSparkInn(rtdmid,
customerid,reportType, connCodeList);

        log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

        return response;

    }
```

```
@Override

    public IntegrationResponse getCompanyListByPersonFIO(int rtdmid, String
customerid)

    {

        log.info("StartgetCompanyListByPersonFIO; request_id = " + rtdmid
+ "; applicant_id = " + customerid);

        ArrayList<ConnectorCode> connCodeList = new
ArrayList<ConnectorCode>();

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("22");

        connCode.getSubRequestCode().add("23");

        connCodeList.add(connCode);

        String reportType = "Проверка по ФИО ФЛ принадлежности к участнику
бизнеса";

        IntegrationResponse response =
```

```

        (new CreRequestSender()).execGroupRequestSparkFIO(rtdmid,
customerid,reportType, connCodeList);

        log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

        return response;
    }

    @Override

    public IntegrationResponse getCompanyExtendedReport(int rtdmid, String
customerid)

    {

        log.info("Start getCompanyExtendedReport; request_id = " + rtdmid +
"; applicant_id = " + customerid);

        ArrayList<ConnectorCode> connCodeList = new
ArrayList<ConnectorCode>();

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("22");

        connCode.getSubRequestCode().add("1");

        connCodeList.add(connCode);

        String reportType = "Запрос компании по ИНН";

        IntegrationResponse response =

            (new CreRequestSender()).execGroupRequestSparkEmpInn(rtdmid,
customerid,reportType, connCodeList);

        log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

        return response;
    }

    @Override

    public IntegrationResponse getEntrepreneusShortReport(int rtdmid, String
customerid)

    {

        log.info("Start getEntrepreneusShortReport; request_id = " + rtdmid
+ "; applicant_id = " + customerid );

        ArrayList<ConnectorCode> connCodeList = new
ArrayList<ConnectorCode>();

        ConnectorCode connCode = new ConnectorCode();

        connCode.setConnectorCode("22");

        connCode.getSubRequestCode().add("8");

        connCodeList.add(connCode);

```

```
String reportType = "Проверка по ИНН ФЛ принадлежности к ИП";

IntegrationResponse response =

    (new CreRequestSender()).execGroupRequestSparkInn(rtdmid,
customerid,reportType, connCodeList);

    log.info("CRE_ID = " + response.getCreId() + "; STATUS = " +
response.getStatus());

    return response;

}

}
```

## 2. Программный код интеграции фронт-офисной системы с SAS RTDM

```
@WebService(targetNamespace = "http://ru/integro/category", name =
"SASCategoryPort")

@XmlSeeAlso({ru.integro.category.request.ObjectFactory.class,
ru.integro.fault.ObjectFactory.class,
ru.integro.category.response.ObjectFactory.class})

@SOAPBinding(parameterStyle = SOAPBinding.ParameterStyle.BARE)

public interface SASCategoryPort {

    @WebResult(name = "response_category", targetNamespace =
"http://ru/integro/category/response", partName = "parameters")

    @WebMethod(action = "http://ru/integro/category/processRequest")

    public ru.integro.category.response.ResponseCategory processRequest(

        @WebParam(partName = "parameters", name = "category_request",
targetNamespace = "http://ru/integro/category/request")

        ru.integro.category.request.RequestCategory parameters

    ) throws BusinessFault;

}

@WebService(targetNamespace = "http://ru/integro/category", endpointInterface =
"ru.integro.category.SASCategoryPort", portName = "SASCategoryPort",
serviceName = "SASCategoryService", wsdlLocation = "SASCategoryService.wsdl")

@Features(features = "ru.integro.category.ExtendedLoggingFeature")
// @Features(features = "org.apache.cxf.feature.LoggingFeature")

public class SASCategoryPortImpl implements SASCategoryPort

{

    /*-----private static final Logger LOG =
Logger.getLogger(SASCategoryPortImpl.class.getName());*/

    private static final Logger log = Logger.getLogger("LoggerForDebugging");

    private static final String _STRATEGY_ID = "SASCategory";

    private static final String _STRATEGY_Auto_ID = "SASAuto";

    private static String ModusOperandi = "OFFI";

    public String iHostName = null;

    private RtdmStrategy rtdmStrategy = null;

    private RtdmStrategy rtdmStrategyAuto = null;
```

```

        ru.integro.fault.ObjectFactory fOF = new
ru.integro.fault.ObjectFactory();

        ru.integro.category.request.ObjectFactory reqOF = new
ru.integro.category.request.ObjectFactory();

        ru.integro.category.response.ObjectFactory repOF = new
ru.integro.category.response.ObjectFactory();


        PersistCategory iPersistCategory = new PersistCategory();

        PopulateRyplyForCategory iPopulateReplyForCategory = new
PopulateRyplyForCategory();

        // CallRTDM iCallRTDM = new CallRTDM();

        EventPortType_EventPort_Client rtdmClient = new
EventPortType_EventPort_Client();

        // 20190320 new class for logging

        EachXMLForEachRun rollingXmlFile = new EachXMLForEachRun(); // 2019.04.12
logs

    public SASCategoryPortImpl()
    {
        /*-----LOG.info("ZZZZZZZZZZZZZZInitializing the
Service.");*/

        try
        {

            SqlSessionFactory factory =
MyBatisUtil.getSqlSessionFactory();

            ModusOperandi =
factory.getConfiguration().getEnvironment().getId();

            // LOG.info("The service and myBatis are set for mode(env):
"+ModusOperandi);

            RtdmSomeDao rtdmStrategyDao = new RtdmSomeDao();

            rtdmStrategy =
rtdmStrategyDao.populateRtdmStrategy(_STRATEGY_ID,
ModusOperandi.toUpperCase().substring(0, 4));

            rtdmStrategyAuto =
rtdmStrategyDao.populateRtdmStrategy(_STRATEGY_Auto_ID,
ModusOperandi.toUpperCase().substring(0, 4));

            iHostName = InetAddress.getLocalHost().getHostName();

        } catch (Exception e1)
        {

```

```

        e1.printStackTrace();

        iHostName = "localhost";

    }

    // LOG.info("SASCalculator service is running on
host:"+iHostName+"; Configured

    // strategy="+rtDmStrategy+"; \n Finish Initializing the
Service.");

}

public Fault fillTheFault(String faultCode, String faultDescription)
{

    /* экземпляр fault, если случится */

    Fault iFault = fOF.createFault();

    iFault.setFaultCode(faultCode);

    iFault.setFaultMessage(faultDescription);

    return iFault;

}

```

## 1.Обработка анкетных данных клиента

```

@Override

    public ResponseCategory processRequest(RequestCategory parameters) throws
BusinessFault

    {

        //=====

        Long actionId_ = parameters.getActionId();

        String appId_ = parameters.getAppInfo().getAppId();

        rollingXmlFile.generateRequestFile(parameters, actionId_, appId_);

        //=====

        ResponseCategory _return = null;

        int maxTries = 3;

        int count;

        // Проверка на дубликаты по action_id

        List<ResponseCategoryDao.AppInfo> checkDuplicate;

```

```

SqlSession sqlSession = null;
CategoryRequestMapper reqMapper = null;
CategoryResponseMapper respMapper = null;
SingleFormatMapper sfsMapper = null;

Long actionId = null;
String appId = null;
Integer taskId = null;

String hostname = "";
String FORMATER = "yyyy-MM-dd'T'HH:mm:ss";

DateFormat format = new SimpleDateFormat(FORMATER);

// Date date = new Date();
XMLGregorianCalendar request_time = null;
String process_type = "Category Request Start";
String description = null;

Properties propForHost = new Properties();
InputStream inputForHost = null;
try
{
    inputForHost =
Resources.getResourceAsStream("config.properties");
    propForHost.load(inputForHost);
    hostname = propForHost.getProperty("hostname");

} catch (IOException e4)
{
    log.error("Error loading hosthame : ");
    log.error(e4);
    e4.printStackTrace();
}

```



```
log.info("Start sql session*****");
```

```
try
```

```
{
```

```
    try
```

```
    {
```

## 2.Подключение к БД

```
/* открываем сессию */
```

```
        sqlSession =
```

```
MyBatisUtil.getSqlSessionFactory().openSession();
```

```
/* грузим маппер для запроса и для ответа по  
Calculator-интерфейсу */
```

```
        reqMapper =
```

```
sqlSession.getMapper(CategoryRequestMapper.class);
```

```
        respMapper =
```

```
sqlSession.getMapper(CategoryResponseMapper.class);
```

```
        sfsMapper =
```

```
sqlSession.getMapper(SingleFormatMapper.class);
```

```
        log.info("Opened sql session*****");
```

```
    } catch (Exception e)
```

```
    {
```

```
        log.error("Error trying open sql session : ");
```

```
        log.error(e);
```

```
        e.printStackTrace();
```

```
        // ошибка открытия сессии к БД
```

```
        throw new BusinessFault("Error occurred with processing  
of acctionId=" + actionId,
```

```
                                fillTheFault("105", "Ошибка инициализации  
подключения к БД перед сохранением запроса!"));
```

```
    }
```

```
try
```

```
{
```

```
    actionId = parameters.getActionId();
```

```
    appId = parameters.getAppInfo().getAppId();
```

```
    taskId = parameters.getAppInfo().getTaskId();
```

```

        // Место где можно вызвать метод проверки на
        существование action_id. Через If

        // check_id = 1 ...

        // Getting date and values for logs_table insert,
        changed initialization for

        // mybatis.

        // For changes compare with other integrations

    } catch (Exception e)
    {

        log.error(e);

        throw new BusinessFault("Error occured with initialize
        actionId, appId, taskId", fillTheFault("109",
        "Ошибка при получении значений параметров
        AppId, ActionId, TaskId из входного XML сообщения!"));
    }

    try
    {

        request_time = DatatypeFactory.newInstance()

        .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

    } catch (DatatypeConfigurationException e2)
    {

        log.error("Error dates, error : ");

        log.error(e2);

        e2.printStackTrace();

    }

    process_type = "Category Start";

    description = null;

    // Starting to log process

    reqMapper.insertLogsInfo(request_time, actionId,
    process_type, description, hostname);

```

```

        checkDuplicate =
respMapper.loadAppInfoForDuplicates(actionId);

        log.info("Duplicate check === " + checkDuplicate.size());
        if (checkDuplicate.size() != 0)
        {
            process_type = "Category Duplicate Error";
            description = "Присутствуют дублирующиеся записи";
            reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

            throw new BusinessFault("Error occured with processing
of acctionId=" + actionId, fillTheFault("108",
                "Присутствуют дублирующиеся записи в
количестве : " + checkDuplicate.size() + " штук!"));
        }

        // LOG.info("Executing operation calculator for actionId=" +
actionId + "
        // appId="
        // + appId + " taskId=" + taskId);
        log.debug(
            "
");
        log.debug("Executing operation calculator for actionId=" +
actionId + " appId=" + appId + " taskId="
            + taskId);

        if (rtdmStrategy == null)
        {
            try
            {
                request_time = DatatypeFactory.newInstance()
.newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
            } catch (DatatypeConfigurationException e2)
            {
                log.error("Error dates");
                e2.printStackTrace();
            }
        }
    }
}

```

```

    }

    process_type = "Category Request Error";
    description = "В справочнике не настроена стратегия для
этой заявки!";

    // Starting to log process
    reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

    throw new BusinessFault("Error occurred with processing
of actionId=" + actionId,

        fillTheFault("101", "В справочнике не
настроена стратегия для этой заявки!"));
    }

    /* экземпляр ответа */

```

### 3.Вызов SAS RTDM

```

    // try {
    // iCallRTDM.initialize(rtdmStrategy);
    // } catch (Exception e) {
    // e.printStackTrace();
    // // ошибка инициализации вызова RTDM...
    // throw new BusinessFault("Error occurred with processing of
actionId=" +

    // actionId,
    // fillTheFault("102", "Ошибка инициализации API вызова
RTDM!"));

    // }

    int rc = 0; /* статус выполнения */

```

### 4.Запись в БД

```

    try
    {

        request_time = DatatypeFactory.newInstance()

        .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

    } catch (DatatypeConfigurationException e3)
    {

```

```

        log.debug("Error in dates, error : ");
        log.error(e3);
    }

//        process_type = "Category Request Persist Category Start";
//        description = null;
//        // Starting to log process
//        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        Long RtdmId = 0L;
        Long RtdmCallStatus = -1L;
        try
        {
            RtdmId =
this.iPersistCategory.insertRequest(parameters, reqMapper, sfsMapper,
iHostName);

            request_time = DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

//        process_type = "Category Request Persist Category End";
//        description = null;
//        // Starting to log process
//        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        log.debug("RTDMID === " + RtdmId + " for actionId === "
+ actionId);

        if (RtdmId == 0L)
        {
            log.error("RTDMID === 0 for actionId == " +
actionId);

            request_time = DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

            process_type = "Category Request Error";
            description = "Ошибка при записи параметров
запроса в БД!";

```

```

        // Starting to log process
        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        sqlSession.rollback();

        throw new BusinessFault("Error occured while
persisting the ActionId=" + actionId,

                                fillTheFault("103", "Ошибка при
записи параметров запроса в БД!"));
    }

    sqlSession.commit();

    // 2019.02.21 Изменение в вызове в стратегии
    for (count = 1; count <= maxTries; ++count)
    {
        log.debug("Trying to execute with " + count + "
attempt RTDM event with actionId === " + actionId);

        request_time = DatatypeFactory.newInstance()

.newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

        process_type = "Category RTDM Call Start";
        description = null;

        // Starting to log process
        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        request_time = DatatypeFactory.newInstance()

.newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

//
count;        process_type = "Category RTDM Call Number " +
//
        description = null;
//
        // Starting to log process
//
        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        try

```

```

        {

            Long ProductId = 0l;

            if(parameters.getCreditParams() != null)

            {

                if(parameters.getCreditParams().getValue().getRequested() != null) {

                    ProductId =
parameters.getCreditParams().getValue().getRequested().getValue().getProductId(
);

                }

            }

            System.out.println("=== Strategy = " +
ProductId);

            if (ProductId==1) {RtdmCallStatus =
rtdmClient.execute(appId, actionId, taskId, rtdmStrategy);}

            if (ProductId==2) {RtdmCallStatus =
rtdmClient.execute(appId, actionId, taskId, rtdmStrategyAuto);}

            break;

        } catch (Exception e)

        {

            log.debug("Error with " + count + " try to
execute RTDM Event for actionId = " + actionId);

            log.error(e);

            request_time =
DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

            process_type = "Category Request Looping
Error Number " + count;

            description = "Looping Error";

            // Starting to log process

            reqMapper.insertLogsInfo(request_time,
actionId, process_type, description, hostname);

            try

            {

                Thread.sleep(2000);

```

```

        } catch (InterruptedException e1)
        {
            log.error("Could not sleep for
actionId === " + actionId + " error : ");
            log.error(e1);
            e1.printStackTrace();
        }
    }

    }

    request_time = DatatypeFactory.newInstance()

    .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

    process_type = "Category RTDM Call End";
    description = null;
    // Starting to log process
    reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

    if ((RtdmCallStatus == -1L &&
rtdmStrategy.getINTEGROIGNOREFLG() != 1) || (count == maxTries))
    {
        log.error("Error in strategy call for action_id
=== " + actionId);

        request_time = DatatypeFactory.newInstance()

        .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

        process_type = "Category Request Error ";
        description = "Ошибка при вызове РТДМ. При
обработке заявки возникли ошибки RTDM";
        // Starting to log process
        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        throw new BusinessFault("Error occured while RTDM
processing the ActionId=" + actionId,

```



```

        fillTheFault("104", "Ошибка при
вызове РТДМ. При обработке заявки возникли ошибки РТДМ.");
    }

    } catch (BusinessFault e)
    {
        throw e;
    } catch (Exception e1)
    {

        try
        {
            request_time = DatatypeFactory.newInstance()

.newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
        } catch (DatatypeConfigurationException e)
        {
            log.error("Error date" + e);
            e.printStackTrace();
        }

        process_type = "Category Request Error";
        description = "Ошибка! Exception не относящийся к
Business Fault!";

        // Starting to log process
        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        log.debug("Error in throwing Business Fault Exception
for actionId === " + actionId + " error is : ");
        log.error(e1);

    }

    try
    {
        request_time = DatatypeFactory.newInstance()

```

```

        .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
    } catch (DatatypeConfigurationException e2)
    {
        log.error("Error date" + e2);
        e2.printStackTrace();
    }

//        process_type = "Category Generate Request File";
//        description = null;
//        // Starting to log process
//        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        // 20190320 generate each file for each run for request
        //rollingXmlFile.generateRequestFile(parameters, actionId,
appId);

        _return = repOF.createResponseCategory();

        try
        {
            request_time = DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
        } catch (DatatypeConfigurationException e2)
        {
            log.error("Error date" + e2);
            e2.printStackTrace();
        }

//        process_type = "Category Reply Populate Start ";
//        description = null;
//        // Starting to log process
//        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        // Changes 14.02.2019

```

```

// 2019.02.21 Изменение в сборе ответа
for (count = 1; count <= maxTries; ++count)
{
    log.debug("Trying to populate reply with " + count + "
attempt with actionId === " + actionId);

    try
    {
        request_time = DatatypeFactory.newInstance()

.newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
    } catch (DatatypeConfigurationException e2)
    {
        log.error("Error date" + e2);
        e2.printStackTrace();
    }

//        process_type = "Category Reply Populate Number " +
count;
//        description = null;
//        // Starting to log process
//        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

    try
    {
        _return =
this.iPopulateReplyForCategory.populateReply(_return, respMapper, actionId,
RtdmId,

repOF);

        break;
    } catch (Exception e)
    {
        log.debug("Error with " + count + " try to
populate reply for actionId = " + actionId);
        log.error(e);

        try
        {

```

```

        Thread.sleep(2000);
    } catch (InterruptedException e1)
    {
        log.debug("Error while trying to sleep for
two second on actionId === " + actionId);
        e1.printStackTrace();
        log.error(e1);
    }
    e.printStackTrace();
    if (count == maxTries)
    {
        try
        {
            request_time =
DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
        } catch (DatatypeConfigurationException e2)
        {
            log.error("Error date" + e2);
            e2.printStackTrace();
        }

        process_type = "Category Reply Error";
        description = "Ошибка при сборке ответа от
системы принятия решений";

        // Starting to log process
        reqMapper.insertLogsInfo(request_time,
actionId, process_type, description, hostname);

        throw new BusinessFault(e.getMessage(),
fillTheFault("110",
                "Ошибка при сборке ответа от
системы принятия решений, actionId: " + actionId));
    }
}

}

try

```

```

        {

            request_time = DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

        } catch (DatatypeConfigurationException e2)
        {

            log.error("Error date" + e2);
            e2.printStackTrace();

        }

//            process_type = "Category Reply Populate End ";
//            description = null;
//            // Starting to log process
//            reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

        try
        {

            request_time = DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));

        } catch (DatatypeConfigurationException e2)
        {

            log.error("Error date" + e2);
            e2.printStackTrace();

        }

//            process_type = "Category Generate Reply File";
//            description = null;
//            // Starting to log process
//            reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

// 20190320 generate each file for each run for reply
rollingXmlFile.generateResponseFile(_return, actionId,
appId);

        try

```

```

        {
            request_time = DatatypeFactory.newInstance()

                .newXMLGregorianCalendar(format.format(System.currentTimeMillis()));
        } catch (DatatypeConfigurationException e2)
        {
            log.error("Error date" + e2);
            e2.printStackTrace();
        }

        process_type = "Category End";
        description = null;
        // Starting to log process
        reqMapper.insertLogsInfo(request_time, actionId,
process_type, description, hostname);

    }
    catch (Exception e)
    {
        log.error(e);

        throw new BusinessFault("Error occured while processing the
ActionId=" + actionId,

            fillTheFault("777", e.toString()));
    }
    // Changes 21.02.2019 - one finally for all SqlSession.
    finally
    {
        // rc = pespMapper.saveToRTDMReply(_return, RequestID,
"SASCalculator", "OK",
        // new BigDecimal("0.00"), "OK");
        if (sqlSession != null)
        {
            sqlSession.commit();
            sqlSession.close();
            log.debug("Closed connection for actionId === " +
actionId);
        }
    }
}

```

```
        System.out.println("End of integration");  
        return _return;  
    }  
}
```