МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Информационные системы и технологии корпоративного управления
(направленность (профиль)

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему **«Исследование методов усовершенствования PLM-процессов в BSS-системах»**

Студент П.В. Еремейчук _____
(И.О. Фамилия) (личная подпись)

Научный руководитель Е.В. Панюкова _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д.т.н., доцент С.В. Мкртычев _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящий момент бизнес операторов связи развивается с большой скоростью, поэтому им необходимы высококачественные и гибкие системы для его ведения. Для решения этой задачи были созданы BSS-системы.

BSS-система покрывает абсолютно все сферы ведения бизнеса с клиентом, что позволяет ускорить подключение различных услуг, их оплату и отключение. Одной из самых сложных и полезных компонент в составе данной системы является PLM (Product Lifecycle Management).

PLM – это жизненный цикл продукта (предложения). В рамках BSS-системы он отвечает за создание, тестирование и выпуск нового предложения (тариф, пакет услуг, оборудование и т.д.), за модификацию и выход из эксплуатации текущего предложения.

Процессы, протекающие в составе PLM-компоненты, должны быть максимально быстрыми, так как рынок операторов связи постоянно развивается и меняется, и чтобы не проиграть конкуренцию необходимо уметь быстро подстроиться под потребности клиентов, а это значит, что каталог услуг и предложений должен всегда находиться в актуальном состоянии.

Актуальность исследования обусловлена тем, что бизнес телеком-операторов растёт большими темпами и для его поддержания необходима качественная BSS-система, которая значительно облегчит большинство бизнес-процессов, в частности, стабильный жизненный цикл продуктов и предложений, за который отвечает PLM-компонента.

Из-за того, что BSS-система это сложная и комплексная информационная система, то возникают сложности при внесении каких-либо изменений в каталог товаров и услуг, поэтому все изменения создаются в рамках запросов на изменение, за которые отвечает опытный и квалифицированный сотрудник.

В рамках данного запроса пользователь производит все необходимые модификации, которые отслеживаются под запросом. После завершения всех необходимых изменений в каталоге пользователю необходимо их протестировать на тестовом окружении, чтобы убедиться в том, верно ли

отображаются новые изменения в системе и корректно ли сработала синхронизация новых каталожных данных с биллинговой системой.

Стоит отметить, что алгоритм работы PLM является очень гибким и сложным процессом, поэтому данная тема в диссертации исследуется в широком научном контексте, включая общую теорию, современные проблемы в работе PLM-компоненты, анализ методов, повышающих эффективность работы PLM-компоненты в системе и моделирование. Данным научным областям посвящено множество работ отечественных и зарубежных научных деятелей. Подробное описание работы PLM-компоненты представлено в исследования Курочкина Л.М., Лихачев М.В., Железнякова М.С и других; основные проблемы в работе современных PLM-компонент – в научных исследованиях Б.А. Позина, И.В. Галахова, Boris Toche, Grant McSorley, Robert Pellerin, Clement Fortin. Серьезный вклад в разработку методов, повышающих эффективность процессов в PLM внесли Charmy Patel, Ravi Gulati, Eduardo De Senzi Zancul, Luiz Fernando C.S. Durão, Alexandre M. Rocha, Gabriel Delage e Silva. Сложность и важность данных проблем подтверждает необходимость в проведении сложных методологических и методических исследований в области совершенствования методов и алгоритмов, применяемых в PLM-компоненте BSS-системы.

Целью диссертационной работы является исследование методов и алгоритмов, позволяющих повысить эффективность процессов, протекающих в PLM-компоненте BSS-системы.

Объект исследования – процессы, протекающие в PLM-компоненте BSS-системы.

Предмет исследования – методы и алгоритмы, которые повышают эффективность работы PLM-компоненты.

Гипотезой исследования является предположение, о возможности применения существующих методов и алгоритмов, повышающих эффективность работы PLM-компоненты в BSS-системе, с целью модернизации текущих процессов, протекающие в PLM, благодаря чему скорость выполнения

большинства операций будет значительно выше текущих показателей. Основываясь на данной гипотезе, применение методов и алгоритмов будут наиболее эффективными, если:

- определены основные проблемы, которые повышают время работы пользователя над запросом на изменение;
- выбраны метод, позволяющие увеличить эффективность работы PLM-компоненты;
- спроектированы и внедрены новые алгоритмы в работу PLM-компоненты, которые повышают скорость выполнения основных процессов в ней.

Для того, чтобы достичь поставленную цель необходимо решить задачи:

1) провести анализ и оценку эффективности текущих процессов работы PLM-компоненты в составе BSS-системы;

2) рассмотреть существующие методы модернизации PLM-процессов, позволяющих повысить эффективность текущих процессов, протекающих в ней;

3) спроектировать новые алгоритмы или инструмент, в основе которого лежат методы, позволяющие повысить эффективность работы PLM-компоненты в BSS-системе;

4) провести оценку эффективности внедренных алгоритмов и инструмента в работе PLM-компоненты.

Научная новизна исследования состоит в том, что в нем обоснована важность модернизации процессов, протекающих в PLM-компоненте, и спроектированы новые алгоритмы работы PLM-компоненты, которые позволили увеличить эффективность и расширить функциональные возможности существующих процессов PLM-компоненты.

Практическая значимость исследования заключается в повышении эффективности работы PLM-компоненты BSS-системы с помощью спроектированных алгоритмов и инструмента.

Основой для теоретических исследований выступили труды научных и зарубежных деятелей в информатике, посвященные методам и алгоритмам, которые позволяют повысить эффективность работы PLM-компоненты.

В процессе выполнения данного исследования были использованы практические положения, методы и алгоритмы: структурирование информации, анализ научной и методической литературы по теме выполняемого исследования, проведен анализ и оценка эффективности текущих процессов, протекающих в PLM-компоненте, проведен анализ существующих методов, повышающих эффективность работы PLM-компоненты, обработка полученной информации, моделирование и проектирование алгоритмов и инструмента, проведение оценки эффективность спроектированных алгоритмов и инструмента.

Основные этапы исследования: исследование проводилось с 2017 по 2019 года в несколько этапов:

На первом этапе (констатирующем этапе) – формулировалась тема исследования, выполнялся сбор информации по теме исследования из различных источников, проводилась формулировка гипотезы, определялись постановка цели, задач, предмета исследования, объекта исследования и выполнялось определение проблематики данного исследования.

Второй этап (поисковый этап) – в ходе проведения данного этапа осуществлялось проектирование алгоритмов и инструмента, основанных на методах, повышающих эффективность работы PLM-компоненты в BSS-системе, проводилось обоснование необходимости в использовании данных алгоритмов и инструмента, проводилось написание и публикация научных статей по теме исследования в сборниках научных статей.

Третий этап (оценка эффективности) – осуществлялась оценка эффективности спроектированных алгоритмов и инструмента, проводилось тестирование функциональных изменений в PLM-компоненте и были сформулированы выводы о полученном результате по проведенному исследованию.

На защиту выносятся:

1. Спроектированные алгоритмы, в основе которых лежат методы повышения эффективность работы PLM-компоненты.

2. Инструмент для PLM-компоненты, который позволяет выполнять группировку запросов на изменение.

3. Результаты оценки эффективности спроектированных алгоритмов, в работе PLM-компоненты BSS-системы.

Спроектированные алгоритмы были реализованы и внедрены в продукт компании ООО «Неткрэкер».

В первой главе рассматриваются основные определения и данные, которые связанны с работой PLM-компоненты в BSS-системе; представлен анализ и оценка эффективности текущих процессов, протекающих в PLM-компоненте, определены основные проблемы в работе данных процессов; проведен анализ методов, позволяющих повысить эффективность работы PLM-компоненты и устранить выявленные проблемы.

Во второй главе описан процесс проектирования новых алгоритмов работы PLM-компоненты, в основе которых лежат методы, направленные на повышение эффективности процессов, протекающих в данной компоненте, и, которые добавляют новые функциональные возможности, позволяющие пользователям выполнять параллельное тестирование запросов на изменение.

В третьей главе представлены результаты оценки эффективности спроектированных алгоритмов, повышающих эффективность работы PLM-компоненты.

В заключении приведены выводы по результату научно-исследовательской работы, достигнутые результаты, решенные задачи и перспективы дальнейшего развития проекта.

Диссертация состоит из введения, трёх глав, заключения, списка литературы. Работа изложена на 85 страницах и содержит 36 рисунков, и 14 таблиц.

Глава 1 Анализ PLM-компоненты в BSS-системе

1.1 Основные определения и данные, относящиеся к PLM-компоненте в BSS-системе

В современном мире бизнес операторов связи развивается с бешенной скоростью, поэтому им необходимы высококачественные и гибкие системы для его ведения. Для решения этой задачи были созданы BSS-системы.

Система поддержки бизнеса (BSS, Business Support System) - это комплексный термин для телекоммуникационных программных решений, используемых для поддержки клиентов. Термин включает программное обеспечение, которое поддерживает биллинг и расходы; управление клиентами; создание и управление продуктом; продажи и маркетинг; заказы и порядок активации [17].

На данный момент отрицать полезность BSS-системы не имеет смысла. Её функциональность покрывает абсолютно все сферы ведения бизнеса с клиентом, что позволило ускорить подключение различных услуг, их оплату и отключение.

Основа BSS – биллинговая система, в которой происходят все финансовые взаиморасчеты с абонентами. Также в этот комплекс может входить система CRM (Customer Relationship Management), отвечающая за отношения с клиентами, в которой хранятся различные данные клиентов, используемые для маркетинговых целей. Также, в BSS может входить система ERP (Enterprise Resource Planning), которая используется для управления ресурсами предприятия, ведения бухгалтерии, финансового учета и т.д [18].

Ниже на рисунке 1.1.1 представлена диаграмма компонентов BSS-системы.

В различных BSS-системах те или иные компоненты могу отсутствовать, так как это напрямую зависит от тех потребностей и нужд, которые должны удовлетворять конкретного телеком-оператора.

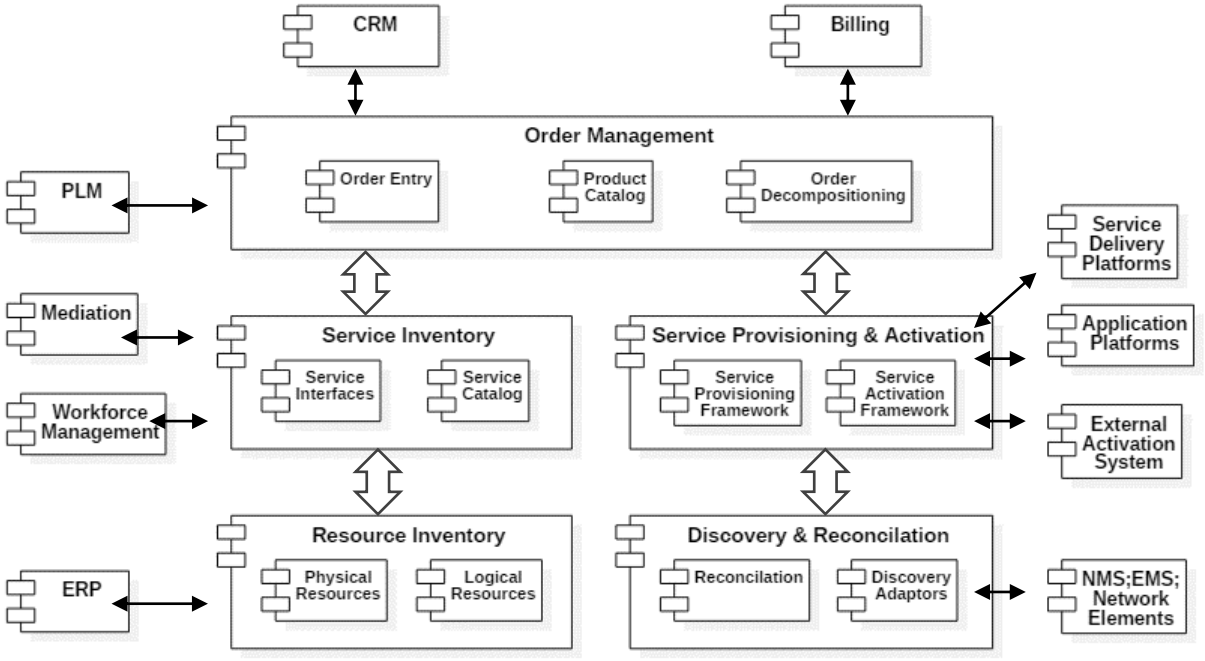

Рисунок 1.1.1 – Диаграмма компонентов BSS-системы

BSS-система включает в себя компоненты, которые поддерживают взаимодействие с клиентами для услуг, предоставляемых OSS-системой. Данная связь отображена на рисунке 1.1.2.

Система поддержки операций (OSS, Operation Support System) – отвечает за взаимодействие с телекоммуникационной средой: сетями электросвязи, коммутационным оборудованием, АТС, аппаратными комплексами обеспечения связи и т.д., предназначенные для поддержки эксплуатации телекоммуникационных систем предприятия связи [23].

OSS облегчает такие операции как: обслуживание услуг клиентов и генерацию важных данных о состоянии сети.

Решения класса OSS/BSS отвечают за две стороны работы телекоммуникационной компании: управление инфраструктурой и ресурсами, а также взаимодействие с абонентами. То есть основная функция таких решений, работающих в комплексе, заключается в том, чтобы услуги предоставлялись и учитывались [24].

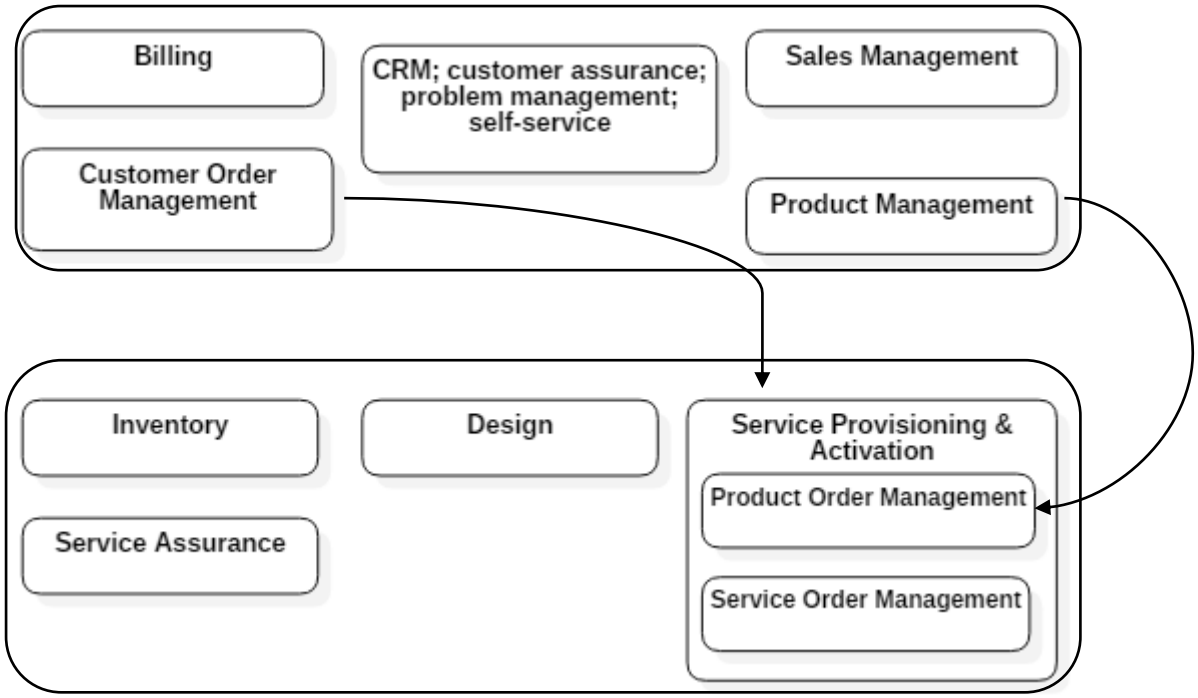

Рисунок 1.1.2 – Взаимодействие компонент между BSS-системой и OSS-системой

Исходя из выше сказанного, можно сделать вывод, что основные области, за которые отвечают BSS-системы это:

1. Product Information Management.
   1.1. Product Lifecycle Management.
2. Sales Management.
3. Customer Information Management.
   3.1. Customer Order Management.
4. Customer Self-Service.
5. Customer SLA/QoS Management.
   5.1. Customer Problem Management.
6. Customer Billing Management.
   6.1. Active Mediation.
   6.2. Online Rating & Charging.

В данном случае необходимо подробно остановится только на областях, связанных с Product Information Management (PIM) и Product Lifecycle Management (PLM).

На рисунке 1.1.3 изображена диаграмма зависимостей компонент модуля PIM между собой.

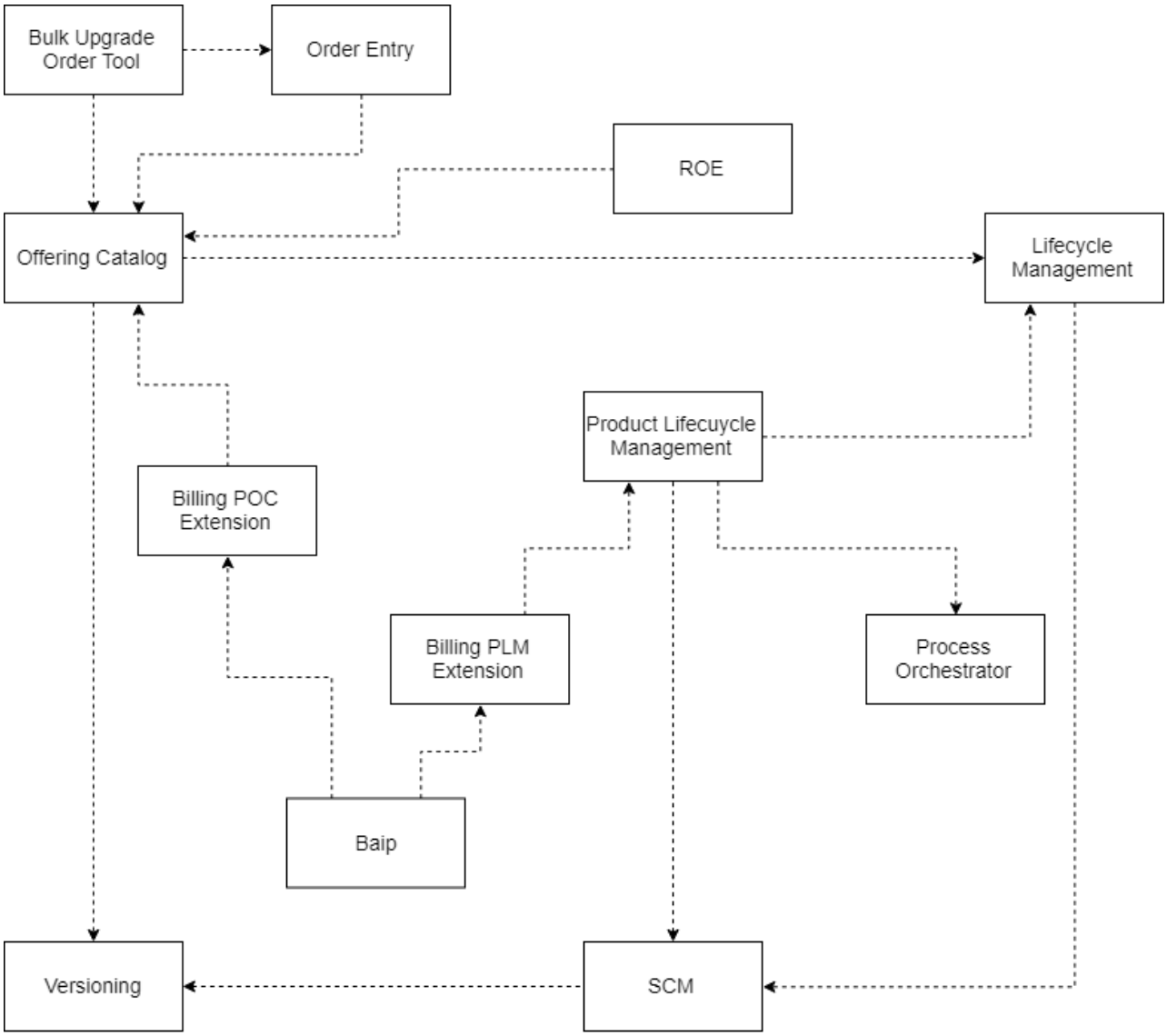

Рисунок 1.1.3 – Диаграмма зависимостей

Проведя разбор данной диаграммы, были выявлены зависимости между всеми компонентами PIM, а также установить их влияние на PLM.

Ниже представлен подробный разбор данных компоненты и их взаимодействий между собой.

Механизм управления жизненным циклом продукта (Product Lifecycle Management engine) является основной частью компонента PLM, который

объединяет всю логику работы с запросом на изменение, каталогом предложений, SCM и внешними интерфейсами [6].

В таблице 1.1.1 описаны зависимости механизма управления жизненным циклом продукта.

Таблица 1.1.1 – Зависимости механизма управления жизненным циклом продукта

Зависимость	Описание
Offering Catalog	Основной механизм PLM должен быть в состоянии работать с: • версионируемостью предложения; • статусами предложений.
SCM	Движок PLM должен иметь возможность взаимодействовать с SCM, чтобы иметь возможность: • начать / остановить отслеживание запроса на изменение • выполнить экспорт / импорт • получить информацию о результатах импорта
Process Orchestrator	Действия Java и общие операции для задач Process Orchestrator должны быть реализованы в основном механизме PLM.
Lifecycle Management	Действия Java и общие операции для переходов LCM должны быть реализованы в основном механизме PLM.

PLM технически состоит из:

1) модуля каталога (Catalog Module). Этот модуль должен отвечать за взаимодействие с каталогом предложений и, возможно, каталогом продуктов в будущем;

2) внешнего интерфейса (External Interface). Этот модуль должен отвечать за взаимодействие с внешними системами. Внешние системные адаптеры будут прослушивать события управления изменениями PLM и реализовывать свою логику того, как обрабатывать их, используя данные запроса на изменение, хранящиеся в каталоге. В настоящее время в области действия находятся только события активации, публикации, отката [9].

3) модуля управления изменениями (Change Management Module). Этот модуль должен содержать всю логику, которая взаимодействует с запросом на изменение.

4) запрос на изменение и его соответствие жизненному циклу (Change request and offer lifecycle mapping). Небольшой модуль, отвечающий за гибкую настройку отображения состояний. Система будет использовать это сопоставление для автоматической отправки события, чтобы отправлять данные о состоянии жизненного цикла предложений для жизненного цикла запроса на изменение (основной жизненный цикл) [8].

5) REST API:

- управление операциями запроса на изменение жизненного цикла (Отмена / Повторное открытие / Приостановка / Возобновление);
- получение информацию о запросе на изменение.

6) Модуль автоматической синхронизации будет накапливать логику, связанную с автоматической синхронизацией конфигурационной, тестовой и производственной сред.

Каталог предложений (Product Catalog) является компонентом модуля управления продуктом (Product Management module). Все объекты компоненты Business Order Entry хранятся и управляются с помощью каталога предложений.

В таблице 1.1.2 описаны зависимости каталога предложений.

Таблица 1.1.2 – Таблица зависимостей каталога предложений

Зависимость	Описание
Versioning	Каталог предложений должен поддерживать управление версиями для следующих объектов: • товар / услуга; • цена. Пользовательский интерфейс управления версиями: • увидеть историю объекта; • перейти к старой версии; • управлять версионным контекстом.
Lifecycle Manager	Существующий жизненный цикл поддерживает динамические кнопки с предложением жизненного цикла.

Компонент Solution Configuration Management (SCM) представляет собой сложный набор инструментов для поддержки конфигураций в каталоге.

SCM поддерживает все основные типы элементов конфигурации каталога. Эти элементы могут быть изменены и экспортированы в формат XML с контролем версий. Преобразование XML в формат SQL происходит на этапе сборки процесса импорта [7].

SCM предоставляется и устанавливается как компонент, который развертывает функциональность на сервере для отслеживания изменений, экспорта и импорта конфигурации через обобщенный пользовательский интерфейс.

Компонент PLM в основном является расширением компонента SCM, который предоставляет дополнительные функции:

- изменение управления над Process Orchestration;
- гибкое управление жизненным циклом;
- гибкий интерфейс с внешними системами [6].

В таблице 1.1.3 описаны зависимости SCM компоненты.

Таблица 1.1.3 – Таблица зависимостей SCM компоненты

Зависимость	Описание
Versioning	SCM должен иметь возможность экспортировать и импортировать данные версий нового типа для типов объектов, которые поддерживают управление версиями (предложений и цен в PLM).
Lifecycle Manager	Существующий жизненный цикл запроса на изменение поддерживает ограниченное количество состояний по сравнению с PLM. PLM продлит этот жизненный цикл, используя собственные события.

Интерфейс ввода заказа (Order Entry) - обеспечивает полное представление информации о клиенте и продукте на одном экране и облегчает ввод заказа и отслеживание заказа через один из нескольких пользовательских интерфейсов. Процесс ввода заказа является центральной функцией ввода системы.

Используя этот процесс, выполняется следующее [7]:

- создается полевая активность;
- создается биллинг клиента;

- собираются данные для отчетности;
- обновляется информация о клиентах.

В таблице 1.1.4 описаны зависимости SCM компоненты.

Таблица 1.1.4 – Таблица зависимостей Order Entry

Зависимость	Описание
Offering Catalog	Order Entry имеет существующую зависимость от API каталога предложений для получения информации из него. Этот API должен обеспечивает поддержку над: • управлением версиями; • жизненным циклом предложения. Платформа Data Cache будет использоваться в каталоге предложений (LCM) для сброса кэша для конкретного активированного предложения (в настоящее время кэш для всего каталога предложений сбрасывается). Таким образом, дополнительное уведомление для ввода заказа не потребуется.

Адаптер активации биллинга и информации (BAIP) является промежуточным компонентом интеграции между каталогом и биллинговой системой [8].

Компонент BAIP обеспечивает интеграционные возможности для модулей BSS-системы Customer Order Management и Customer Information

Management, чтобы иметь возможность доступа к биллинговым каталогам, для того, чтобы предоставлять и управлять информацией из биллинговой системы. Таким образом, компонент BAIP позволяет реализовать сквозной процесс оплаты заказа в BSS-системе [9].

В таблице 1.1.5 описаны зависимости BAIP компоненты.

Таблица 1.1.5 – Таблица зависимостей BAIP компоненты

Зависимость	Описание
Offering Catalog	Существующая зависимость с каталогом предложений используется для получения информации из него во время синхронизации с биллинговой системой.
PLM (indirect dependency) PLM ← Billing PLM Extension → BAIP)	Billing PLM Extension реагирует на события PLM (активация, размещение, откат) для поддержки размещения и отката изменений в биллинговой системе. Расширение сможет размещать и откатывать изменения на уровне запроса на изменение.

Bulk Upgrade Order Tool (BUOT) - это новый компонент, который предоставляет возможность изменить предложение для существующих клиентов.

Типичный пример использования PLM состоит в следующем: после выпуска новой версии предложения должна быть возможность создавать массовые заказы для массового обновления до последней версии предложения для существующих учетных записей клиентов [9].

В таблице 1.1.6 описаны зависимости BUOT компоненты.

Таблица 1.1.6 – Таблица зависимостей BUOT компоненты

Зависимость	Описание
Order Entry	Bulk Upgrade Order Tool имеет возможность создавать заказы при вводе заказа в массовом режиме. Также реализованы гибкие способы создания заказов: • синхронно / асинхронно путем планирования специальной работы (в непиковое время); • создание заказов для всех клиентов / для набора клиентов. Пользовательский интерфейс (на основе виджетов) должен иметь возможность показывать статистическую информацию о: • количество абонентов, для которых доступно предложение (рынок, категория клиентов, канал сбыта); • статистику отправленных заказов.
Offering Catalog	Инструмент для массового обновления заказа получает информацию из каталога предложений.

Компонента Process Orchestrator BSS-системы позволяет создавать и выполнять процессы, которые представляют собой наборы задач.

Каждая задача в процессе характеризуется параметрами расписания и действием задачи, она также может иметь связи (зависимости) с другими задачами. Параметры расписания задач вместе с зависимостями указывают, когда следует запускать действия этих задач и сколько времени займет их выполнение. Действие задачи определяет, что должно быть сделано в BSS-системе при выполнении задачи. Компонента Process Orchestrator поддерживает Java, асинхронную Java, рабочий процесс, ручное задание, повторное

выполнение, выполнение подпроцесса, прослушиватель событий, завершение процесса и многие другие действия [8].

Lifecycle Manager (LCM) - это механизм, способный управлять состоянием произвольной сущности.

Ключевые особенности:

- Единая структура для управления состояниями объекта.
- Конфигурация жизненного цикла по умолчанию.
- Text User Interface (TUI), позволяющий настраивать пользовательские жизненные циклы;
- Отдельный модуль, не являющийся частью SPF.

Ключевой концепцией LCM является определение модели состояния предприятия с точки зрения: состояния, перехода, события, условия и действия. Данная концепция указана в таблице 1.1.7.

Таблица 1.1.7 – Таблица определения модели состояния предприятия с помощью концепции LCM

Термин	Определение
Состояние (state)	Это объект, который позволяет связать определенный атрибут заказа с набором соответствующих значений (например, «Ввод», «Обработка», «Завершено»).
Переход (Transition)	Это объект, который указывает начальное состояние, к которому применим переход, состояние назначения после перехода и событие, когда переход применим (например, «Ввод-обработка», «Обработка-завершение»).
Событие (Event)	Это объект, который документирует событие, которое может произойти во время обработки заказов («Пуск», «Завершено»).
Условия и действия	Подлежат оценке и выполнению во время

	переходов.

Управление версиями (Versioning) - это новый компонент платформы, который предоставляет функции управления версиями объектов каталога. Для каждой BSS-системы выбираются типы объектов, для которых должны быть доступны функции управления версиями (конфигурация также включает описание доменов бизнес-единиц). Функциональность не влияет на все другие не настроенные типы объектов [7].

В настоящее время компонента включает только API.

Дорожная карта заключается в дополнительном создании виджетов OOB для поддержки основных операций с управлением версиями в пользовательском интерфейсе:

- просмотр списка версий объекта;
- перейти к версии объекта;
- выберите контекст управления версиями.

PLM Cache Management (PCM) предназначен для отслеживания и обработки изменений, внесенных в объекты каталога предложений продуктов, в рамках запросов на изменение, с использованием функции управления жизненным циклом продуктов.

Это необходимо для того, чтобы синхронизировать эти изменения с данными, хранящимися во внешнем кэше [6].

Чтобы синхронизировать изменения, внесенные в каталог предложений продуктов, с данными, хранящимися во внешнем кэше, система должна:

- перезапускать каждый узел PCM после изменений в каталоге предложений;
- перезапускать каждый узел PCM после изменений в SSP.

Для обеспечения PLM Cache Management с возможностью отслеживания и обработки изменений, внесенных в каталог предложений продуктов, необходимо предоставлять управление очередью продуктов для отслеживания

изменений, внесенных в объекты каталога предложений продуктов, и отправки сообщений в очередь JMS [8].

Ключевые функции:

- управление службой очереди для отслеживания изменений, внесенных в объекты каталога предложений продуктов и отправки сообщений в очередь JMS;
- абстрактный слушатель для клиентов на очередь обслуживания;
- абстрактный слушатель для отслеживания изменений, внесенных в объекты каталога продуктов в рамках запроса на изменение с использованием обработчиков объектов;
- абстрактный обработчик для отслеживания изменений, внесенных в объекты каталога продуктов в рамках запроса на изменение через пользовательский интерфейс BSS-системы;
- абстрактное действие для отслеживания изменений, внесенных в объекты каталога продуктов при переходе между состояниями жизненного цикла.

Рассмотрев основные зависимости между компонентами модуля PIM, был проведен также подробный анализ работы данного модуля.

Управление информацией о предложениях (Product Information Management, PIM) является одной из самых важных компонент в составе BSS-системы и включает в себя следующие области [19]:

- каталог товаров (Product Catalog), который служит единой точкой доступа к информации о продукте для всех подразделений, партнеров и клиентов. Он позволяет фильтровать, группировать и представлять информацию о товаре в удобном для пользователя виде;
- механизм управления информацией (Information Management), который позволяет обеспечить централизованное, иерархическое хранилище современных деловых и технических подразделений и партнерских продуктов

логичным и удобным для пользователя способом и позволяет своевременно обновлять информацию;

- механизм управления политикой (Policy Management), который позволяет контролировать использование данных продукта в многопроцессорных, многопользовательских средах. Он поддерживает иерархические правила доступа к данным и позволяет получить информацию о данных, зависящих от пользователя.

PIM непосредственно включает в себя «Управление жизненным циклом продукта» (Product Lifecycle Management, PLM), так как большинство механизмов PLM позволяют пользователю взаимодействовать с каталогом [20].

Рассматривая подробно PLM, стоит отметить, что он также охватывает несколько областей, а именно:

- ориентацию продукта (Product Targeting), которая предоставляет возможности для планирования начальной стратегии продукта, разработки конкурентной дифференциации, определения целевой аудитории и географических регионов для новых продуктов и позиционирования продукта (например, для массового рынка, бизнеса или государственных клиентов);
- поддержку жизненного цикла продукта (Product Lifecycle Support), которая обеспечивает управление жизненным циклом продукта E2E, начиная с разработки концепции маркетинга и конфигурации продукта, заканчивая тестированием продукции, запуском тестирования и последующим массовым производством;
- мониторинг продукции (Product Monitoring), который предоставляет инструменты для отслеживания и анализа состояния разработки продукта (включая уведомления и эскалации), а также мониторинг и анализ производительности и рентабельности продукта;
- связанные продуктовые предложения, которые предоставляют возможность легче связывать предложения вместе, а не определять совершенно новое предложение;

• правила перепродажи и поощрений для переконфигурации и переходов (Cross-sell and Promotional Rules for Reconfigurations and Transitions);

• атрибуты по скидкам (Attributes on Discounts), которые позволяют определять данные продукта как атрибуты, которые могут быть синхронизированы с биллинговой системой для создания скидок для конкретных продуктов.

Начальный PLM-процесс инициализируется в момент создания пользователем «Запроса на изменения» (Change Request, CR), служащего объектом, под которым отслеживаются любые изменения в каталоге товаров.

Момент слежения за пользовательскими действиями начинается тогда, когда пользователь запускает «Задачу разработки» (Development Task) и заканчивается, когда данная задача приостанавливается или завершается пользователем [21].

Слежение необходимо для того, чтобы каталоги товаров были идентичными на разных связанных между собой окружениях.

При использовании PLM-компоненты характерно использование трёх серверов:

1. Конфигурационный (Configuration server) – на данном окружении происходит разработка нового продукта или услуги, или изменение текущих продуктов или услуг. Абсолютно все задачи, связанные с PLM-процессом, создаются и завершаются на данном окружении.

2. Тестовый (Test server) – на данном окружении происходит тестирование разработанного или измененного продукта. Тестирование необходимо, так как изменения, которые произошли в каталоге, были синхронизированы с биллинговой системой прежде, чем оказались на тестовом окружении. Это означает, что при тестировании проверяется перечень E2E кейсов, связанных с различными видами продаж, так как необходимо убедится, что все параметры и атрибуты были успешно перенесены в биллинговую систему.

3. Производственный (Production server) – на данном окружении работают сотрудники телеком-оператора, то есть именно на нём происходят все бизнес-операции, связанные с клиентами.

На данный момент PLM-процесс содержит 5 основных задач, которые существуют только на конфигурационном сервере:

1. Development Task (Задача Разработки) – на данном этапе пользователь имеет возможность модифицировать данные в каталоге. Только во время работы этой задачи допускается возможность изменения данных в каталоге.

2. Check Task (Задача Проверки) – на данном этапе пользователь проверяет правильно ли была выполнена конфигурация новых продуктов или изменения текущих в Residential Order Entry (Интерфейс для добавления новых услуг или отключения старых). Если на данном этапе выявляются проблемы, то данную задачу завершают с отрицательным результатом и снова возвращаются на этап разработке для исправления обнаруженных проблем. Перед проверкой важно убедиться, что на сервере установлена Testing Date (Тестовая дата), которая обеспечивает возможность отображения новых изменений в Order Entry.

3. Testing Task (Задача Тестирования) – во время запуска данной задачи происходит синхронизация с биллинговой системой. Данные о новых продуктах импортируются в биллинговую систему.

Если синхронизация проходит успешно, то пользователь может приступить к тестированию изменений, которые содержатся в запросе на изменения.

Но само тестирование необходимо уже выполнять на тестовом окружении, для этого система генерирует архив, которые содержит запрос на изменения со всеми его модификациям. Данный архив необходимо импортировать на тестовое окружение, причем при импорте также будет выполнена и синхронизация с биллинговой системой.

Во время тестирования пользователь выполняет продажу, отключение, новое подключение и другие операций, которые возможны в рамках Residential

Order Entry. Это необходимо, чтобы быть уверенным, что данные были успешно перенесены в биллинговую систему и они корректны.

Если во время тестирования были обнаружены проблемы конфигурации, которые по каким-либо причинам не были найдены на предыдущем этапе, то необходимо откатить все изменения из биллинговой системы и после исправления снова выполнить синхронизацию.

4. Approve Task (Задача Подтверждения) – на данном этапе пользователь, ознакомившись с результатами тестирования, подтверждает, что запрос на изменения может быть перенесен на производственный сервер.

5. Implementation Task (Задача Имплементации) – на данном этапе система генерирует финальный архив со всеми изменениями, которые содержит запрос на изменения.

Пользователю необходимо выполнить импорт данного архива на производственный сервер. Если импорт был успешен, то задачу можно будет завершить и новые изменения появятся на производственном сервере и вступят в силу согласно дате начала их действия.

Таким образом, PLM-процесс состоит из 5 задач, которые инициализируются в строгом порядке на конфигурационном сервере и должны быть завершены на нём же.

Но стоит отметить, что тестирование запроса на изменение происходит на тестовом окружении, а финальные изменения должны быть импортированы на производственный сервер.

1.2 Анализ и оценка эффективности текущих процессов PLM-компоненты

Во время анализа необходимо выявить наиболее проблемные места, которые замедляют время необходимое запросу на изменение оказаться на производственном сервере.

Для анализа рассмотрим текущий алгоритм, который охватывает все PLM-процессы, представленный на рисунке 1.2.1.

На рисунке отчетливо видны границы всех 5 задач: Development, Check, Test, Approve и Implementation.

Наиболее крупную область в алгоритме занимает Test Task.

На данном рисунке буквенные обозначения C, T и P означают: окружение на котором выполняются данные задачи, то есть на Configuration, Test и Production серверах соответственно.

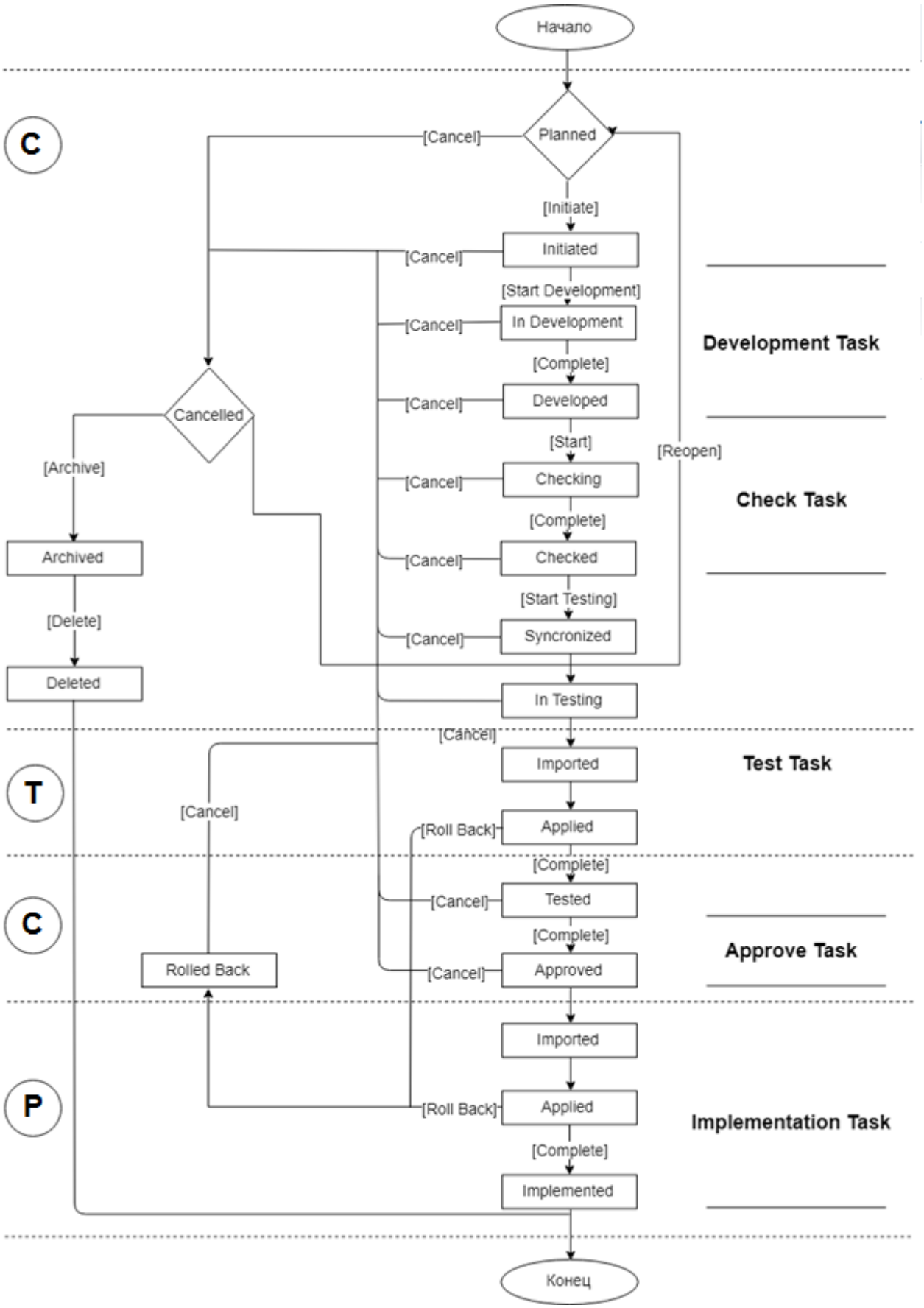

Рисунок 1.2.1 – Текущий алгоритм работы PLM-компоненты BSS-системы

Проведем анализ каждой задачи в отдельности в рамках представленного алгоритма:

1. Development Task – является самой первой задачей, в которую можно попасть после инициализации запроса на изменение. Данная задача выполняется исключительно на конфигурационном сервере. Один пользователь не может одновременно перевести несколько запросов в состояние «In Development», но при этом система поддерживает возможность перевода нескольких запросов разными пользователями, то есть если за каждым пользователем закреплен свой запрос, то каждый из их запросов может быть в состоянии «In Development» в одно и тоже время с другим.

Система также поддерживает возможность отменить запрос во время данной задачи в любом месте её выполнения. Это справедливо для всех задач, но только если они ещё активны на конфигурационном сервере. В ином случае необходимо выполнить откат изменений с тестового или производственного сервера [22].

После завершения данной задачи никого архива сгенерировано системой не будет. Все изменения касательно каталога будут сохранены под запросом. Время необходимое на завершения данной задачи зависит от:

- размера необходимых изменений в каталоге;
- компетентности и опыта пользователя;
- времени, которое требуется для перевода запроса из статуса In Development в статус Developed.

Как можно заметить данный этап не столь сильно зависит от системы, сколько от человека. В данном случае время, которое система тратить на перевод из одного статуса в другой, не имеет столько весомого значения, так как даже если изменений было сделано очень много, то данное действие будет выполнено в рамках 1-5 минут.

2. Check Task – для запуска данной задачи времени нужно гораздо больше чем для Development Task, так как при запуске система меняет статус у всех изменений на конфигурационном сервере с Developed на Testing. В связи с

этим, чем больше имеется модификаций под запросом, тем дольше выполняется запуск. Но стоит отметить, что в данном случае даже один пользователь может инициализировать данную процедуру для несколько запросов.

На время, которое необходимо на выполнение этого этапа, влияет следующие критерии:

- время необходимое для запуска данной задачи и её завершения;
- время для проверки пользователем работы Order Entry на отображение новых изменений, без выполнения продажи.

Как и в случае с Development Task основное время выполнения данной задачи зависит от пользователя, чем он опытнее, тем быстрее завершится данный этап.

3. Test Task – является одной из самых важных задач для запроса на изменение.

Как упоминалось выше, лишь одна тестовая задача может быть активной на сервере. В связи с чем возникает некоторая очередь на возможность перейти в стадию тестирования.

Данное ограничение очень сильно влияет на время тестирования, когда у нас есть несколько запросов готовых к тестированию.

По мимо вышесказанной проблемы, Test Task занимает очень много времени, так как при её запуске сначала инициализируется процесс синхронизация каталога с биллинговой системой.

Во время синхронизации биллинговая система получает данные, связанные с новыми модификациями. Каждый новый объект, который был создан в запросе на изменение после синхронизации с биллинговой системой получает свой персональные и уникальный id, благодаря которому данный объект можно будет найти в биллинговой системе.

Но стоит отметить, что прежде чем данный id будет присвоен, все новые модификации проходят ряд проверок на валидность. К примеру: цена

предложения не может быть отрицательной, модификации под запросом на изменения не должны быть затронуты другим запросом и т.д [19].

Из выше сказанного, можно сделать вывод, что процесс синхронизации занимает достаточного много времени. Чем больше объектов содержит в себе запрос на изменение, тем дольше будет протекать синхронизация.

Особенно это сильно проявляется, если запрос на изменение содержит много новых версий оборудования, например, мобильные телефоны, так как для них приходится создавать достаточно крупное дерево цен, которое может содержать в себе больше 100 цен, которые являются отдельными объектами, которым будет присвоен свой индивидуальный id после синхронизации с биллинговой системой.

Стоит отметить, что система не учитывает, что цены могут быть одинаковые.

Когда процесс синхронизации завершен, система приступает к проверке того, что все объекты получили персональный id, связанный с биллинговой системой и собирает архив, который содержит новые модификации.

Данный архив необходим для импорта на тестовое окружение. После этого статус запроса на изменение становится In Testing.

После успешного выполнения синхронизации и перевода запроса на изменения в статус In Testing архив необходимо импортировать на тестовое окружение для того, чтобы можно было протестировать новые модификации.

Данный процесс имеет большую продолжительность, так как все модификации перемещаются в систему и также в биллинговую систему, которая с ней связана.

Все валидациооные правила, которые применялись на этапе синхронизации при старте Test Task, также справедливы и здесь.

Если импорт был успешен, то статус запроса на изменения на тестовом окружении будет Applied, а все модификации будут иметь статус Testing.

После чего необходимо проверить, что на сервере имеется Test Date, если нет, то её необходимо указать. После чего пользователь может приступить к полноценному тестированию запроса на изменение.

Если импорт оказался не успешен, то необходимо ознакомится с ошибкой и после чего вернутся на конфигурационный сервер и исправить проблемы, связанные с неверной конфигурацией каталога, то есть пользователь должен вернутся на Development Task.

Когда тестирование запроса на изменения завершено, то пользователь должен вернуться на конфигурационный сервер и завершить Testing Task, после чего статус запроса станет Tested.

Таким образом, Test Task является самой продолжительной задачей из всех, участвующих в PLM-процессе, на неё влияют:

- синхронизация каталога с биллинговой системой на конфигурационном сервере;
- размер архива;
- скорость передачи данных;
- проверка новых модификации каталога;
- создание архива новых модификация для импорта на тестовую среду;
- импорт на тестовое окружение;
- синхронизация с каталогом и биллинговой системой во время импорта на тестовое окружение
- тестирование новых модификаций пользователем на тестовом окружении.

В данном случае видно, что прежде чем пользователь приступит к тестированию, системе необходимо выполнить две различные синхронизации с биллиноговой системой, что значительно повышает время ожидания пользователем перед началом тестирования.

Стоит отметить, что основная цель тестовой задачи, это проверить новые модификации, которые находятся в запросе на изменение.

Исходя из вышесказанного следует, что имеет смысл снизить количество времени до момента тестирования пользователем новых модификаций на тестовом окружении.

Этого можно достичь, если выполнять синхронизацию с биллинговой системой только на одном окружении, также имеет смысл вести возможность распараллелить данный процесс, чтобы не создавалась очередь среди запросов на изменение для перевода в In Testing статус.

И не мало важным будет непосредственно уменьшить время необходимое на синхронизацию между каталогом и биллинговой системой.

4. Approve Task – необходима для того, чтобы ответственное лицо за новый запрос на изменение, перед отправкой на производственный сервер, ознакомилось с результатами тестирования и внесло свои поправки.

В данном случае на скорость выполнения данной задачи влияет только пользователь, так как при нажатии на кнопку завершения никаких серьёзных процессов в системе не запускается.

5. Implementation Task – является финальной задачей, внутри которой формируется архив со всеми каталожными изменениями.

Данный архив необходимо импортировать непосредственно на производственный сервер.

При выполнение данной операции также происходит синхронизация с биллинговой системой, но только уже с производственным сервером.

После того как импорт был успешно завершен, то все изменения на производственном сервере имеют статус Testing до тех пор, пока пользователь не завершит задачу на конфигурационном сервере, как только это произошло, то статус у всех новых модификаций становится Released.

В данном случае на скорость выполнения данной задачи влияют следующие факторы:

- размер архива;
- скорость передачи данных;

• синхронизация каталога с биллинговой системой на производственном сервере;

• импорт на производственный сервер.

Из выше сказанного, видно, что факторы, которые влияют на скорость схожи с факторами тестовой задачи.

Важным отличием является только механизм импорта на производственный сервер, хотя даже в нём основная логика остается прежней.

Таким образом, можно сделать вывод, что текущие PLM-процессы могут быть улучшены для обеспечения более быстрой работы с системой, а также для упрощения тестирования запросов на изменения.

Основной же проблемой является долгий перевод запроса на изменение в стадию тестирования, а также невозможность тестировать их параллельно из-за чего возникает очередь из запросов на изменение.

1.3 Анализ методов модернизации PLM-процессов в BSS-системе

Во время анализа основное внимание уделялось методам, которые позволят повысить скорость работы PLM-компоненты и понизить время необходимое на то, чтобы запрос на изменение перешел на стадию тестирования.

Одним из основных методов, который позволит ускорить переход запроса на изменение в тестирование, является метод поддержки нескольких тестовых окружений (Support of multiple test environments), который изображен на рисунке 1.3.1.

Основная идея метода в том, что пользователи, которые ответственны за тестирование, больше не будут друг другу мешать при переходе на этап тестирования, так как будут выполнять импорт модификаций на различные окружения в связи с чем, проблема очереди на импорт будет решена [27].

Данный метод также предлагает формировать архив для импорта на тестовое окружение по завершению Development Task, при этом не нужно будет выполнять синхронизация новых модификация с биллинговой системой

во время импорта архива на тестовый сервер. Синхронизация будет выполняться уже непосредственно после импорта на тестовом окружении. Что также позволяет избежать повторную синхронизацию с биллинговой системой при переходе к этапу тестирования новых модификаций [27].

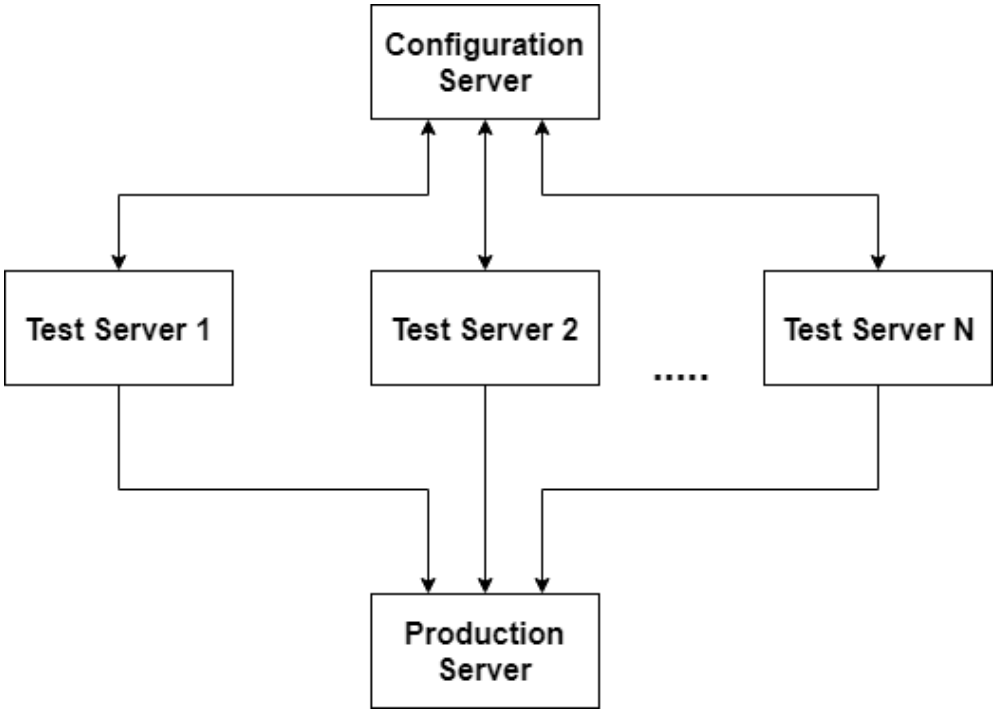

Рисунок 1.3.1 – Метод поддержки нескольких тестовых окружений

Но стоит отметить, что для реализации данного метода необходимо будет выделить дополнительные окружения для тестирования, что не всегда может быть возможно.

Таким образом, плюсы данного метода очевидны и позволяют покрыть практически все недостатки текущих процессов, протекающих в PLM-компоненте. Данный метод позволит:

- избежать повторную синхронизацию каталога с биллинговой системой при переходе к тестированию запроса на изменение;
- устранить проблему «очереди» среди запросов на изменение при переходе к тестированию;
- выполнять параллельное тестирование нескольких запросов на изменение одновременно различными пользователями.

Основным и единственным минусом данного метода является, то что для его реализации необходимо дополнительное окружение. Но несмотря на это, данный минус не столько значительный как может показаться так, как операторы связи располагают достаточными средствами, чтобы позволить себе это.

Следующим методом, который может повысить эффективность PLM-компоненты является метод поддержки параллельного тестирования запросов на изменения на одном окружении (Test of parallel CRs on one environment).

Данный метод стоит обязательно рассмотреть в случае, когда нет возможности добавить в систему дополнительного решения, в связи с чем нельзя будет воспользоваться методом поддержки нескольких тестовых окружений.

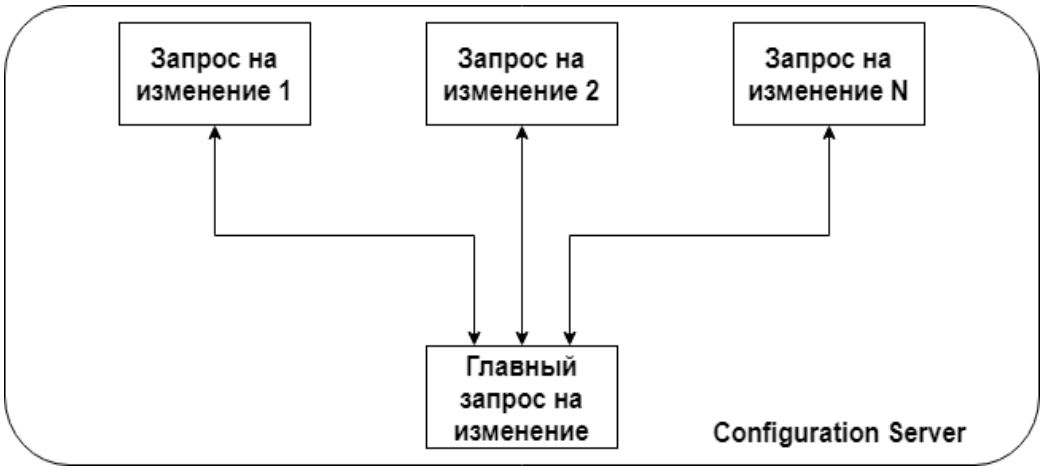

Рисунок 1.3.2 – Метод поддержки параллельного тестирования запросов на изменения на одном окружении

Основная идея данного метода заключается в группировки нескольких запросов на изменения под одним главным, при этом модификации из остальных переместятся под главный, а запросы на изменения изменят свой статус на Cancelled и будут пустыми до тех пор, пока пользователь не инициирует разгруппировку главного запроса [30].

Рассматривая этот метод, стоит учитывать, что он имеет наибольшую полезность тогда, когда имеется несколько небольших запросов на изменение и

которые в сумме не будут превышать допустимый размер конечного архива. В связи с чем, он будет малоэффективен в случае группировку даже двух крупных запросов, так как в таком случае время необходимое для синхронизации с биллинговой системой и импортом на другое окружение будет не оправдано велико [30].

Таким образом, стоит выделить следующие плюсы данного метода:

- возможность параллельно тестировать несколько небольших запросов на изменения одновременно;
- параллельное тестирование будет располагаться на одном окружении;
- возможность импорта нескольких запросов на изменения на производственную среду в одно и тоже время в сгруппированном виде.

Рассматривая минусы данного подхода следует выделить следующее:

- группировка крупных запросов на изменение возможна, но при этом она будет не эффективной, как при синхронизации, так и при импорте;
- если во время тестирования будет обнаружена проблема конфигурации, то придется выполнить разгруппировку запросов.

Как было сказано выше, данный метод стоит рассматривать в том случае, когда отсутствует возможность реализовать метод поддержки нескольких тестовых окружений.

Предыдущие методы предлагали так или иначе решение, которое позволило бы быстрее довести запросы на изменение до состояния тестирования с помощью изменения существующего алгоритма. Метод, о котором пойдет речь ниже, будет взаимодействовать с механизмом синхронизации каталога с биллинговой системой, который называется метод группировки идентичных ценовых значений, который изображен на рисунке 1.3.3.

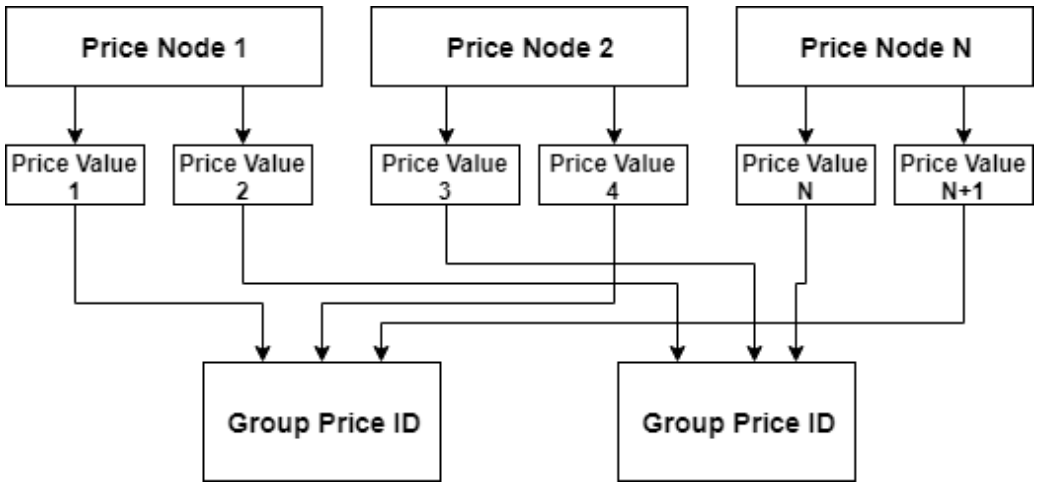

Рисунок 1.3.3 – Метод группировки идентичных ценовых значений

Основная идея данного метода заключается в том, чтобы увеличить скорость синхронизации с биллинговой системой за счет разделение объектов (Price Value) на группы по идентичным показателям и присвоения идентификатора не объекту, а группе (Group Price ID) к которой он принадлежит. Благодаря этому в биллинговую систему при синхронизации будет уходить в десятки раз меньше объектов, что положительно скажется на времени необходимом для этой процедуры [33].

Но у данного метода есть и свои ограничения, он будет полезен только в том случае, если разрабатываемое предложение или услуга имеют большое дерево цен. В другом ином случае эффект от этого метода будет не значительный. Плюсы описанного метода заключаются в следующем:

- увеличение скорости синхронизации каталога с биллинговой системой за счёт группировки идентичных ценовых показателей;
- не требуется дополнительное окружение;
- наиболее эффективен при работе с предложениями, у которых имеются большие деревья цен.

Из минусов стоит выделить следующее:

- становится не эффективным, если модификации, которые находятся в запросе на изменения, не содержат крупных ценовых деревьев
- не предоставляет возможности для параллельного тестирования или импорта.

Таким образом, все три метода, позволяют повысить эффективность работы PLM-компоненты, каждый из которых имеет свои минусы и плюсы. Но использование всех трёх методов одновременно повысит вариативность работы PLM-компоненты и минусы будут не столь значимыми.

Выводы по главе 1

В данной главе дано определение BSS-системе, OSS-системе, PIM и PLM-компонентам как модулям внутри BSS-системы.

Были выделены основные параметры и процессы, которые протекают в PLM-компоненте.

Был проведен анализ и оценка эффективности механизмов работы текущей PLM-компоненты в составе BSS-системы, благодаря которым был выявлен ряд недостатков в текущей работе, которые могут быть исправлены путем модифицирования текущих алгоритмов или внедрения новых инструментариев.

Проведен сравнительный анализ методов, позволяющих улучшить работу PLM-компоненты, которые позволяют устранить текущие недостатки в процессах, протекающих внутри компоненты.

Проведенный анализ позволил выявить основные проблемы в PLM-компоненте BSS-системы на текущий момент, а также определить методы, которые имеют возможность улучшить работу данной компоненты в целом за счёт: сокращения времени необходимого для того, чтобы перевести запрос на изменения в стадию тестированию, увеличить скорость синхронизации каталога с биллинговой системой и повысить удобство при параллельной работе над несколькими запросами на изменения.

Глава 2 Проектирование алгоритмов и инструментов, повышающий эффективность работы PLM-компоненты в BSS-системе

2.1 Проектирование алгоритма, который предоставляет возможность поддержки нескольких тестовых окружений для параллельного тестирования запросов на изменение в PLM – компоненте

В настоящее время система не позволяет выполнять тестирование сразу нескольких запросов на изменение – в среде конфигурации может существовать только один запрос на изменение в фазе «В тестировании». Это замедляет обновление каталога товаров и услуг и увеличивает время выпуска на рынок новых продуктов, поскольку им приходится ждать в очереди перед тестированием [27].

В данном случае отличным вариантом для решения выявленной проблемы является применение метода поддержки нескольких тестовых окружений, которые позволят проводить параллельное тестирование нескольких запросов на изменение, избегая при этом очереди.

Для того, чтобы применить упомянутый выше метод необходимо будет внести ряд важных изменений в текущий жизненный цикл запроса на изменения.

Для достижения параллельного тестирования модификаций, которые относятся к разным запросам на изменение, на отдельных тестовых окружениях в рамках PLM-компоненты необходимо добавить следующие функциональные возможности [27]:

1. Создание архива, который будет содержать в себе все разработанные модификации под текущим запросом на изменения в состоянии «Разработано» (Developed) без перехода в фазу «В тестировании» (In Testing), то есть при генерации архива с модификациями не будет выполняться синхронизация с биллинговой системой, что позволит сильно сэкономить время, что положительно скажется на переходе пользователя непосредственно к тестированию нового запроса на изменения.

Данная функциональная возможность, которая изменяет текущее поведение работы PLM-компоненты, изображена на рисунке 2.1.1.

Рисунок 2.1.1 – Новая функциональная возможность автоматического формирования архива на этапе завершения Development Task

2. Выполнение импорта сформированного архива, содержащего все модификации запроса на изменение, из состояния «Разработано» (Developed) на TEST-сервер без выполнения синхронизации с биллинговой системой.

После данного импорта статус запроса на изменения на TEST-сервере будет «Проверяется» (Checking). Данное изменение в работе текущего алгоритма PLM-компоненты изображено на рисунке 2.1.2.

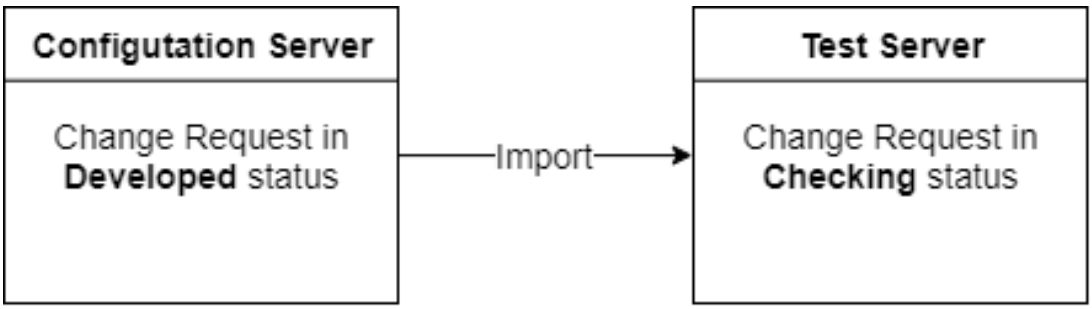

Рисунок 2.1.2 – Возможность выполнения импорта на TEST-сервер из состояния «Разработано» (Developed)

3. Перевод из статуса «Проверяется» (Checking) в статус «В тестировании» (In Testing) запроса на изменения, который находится на TEST-сервере, с обязательным выполнением синхронизации с биллинговой системой. В данном случае до инициализации перехода в тестирование пользователь также имеет возможность сначала убедиться в корректном отображение новых модификаций в окне продаж. И уже после положительного результата перевести запрос на изменения в тестирование. Данная функциональная возможность представлена на рисунке 2.1.3.

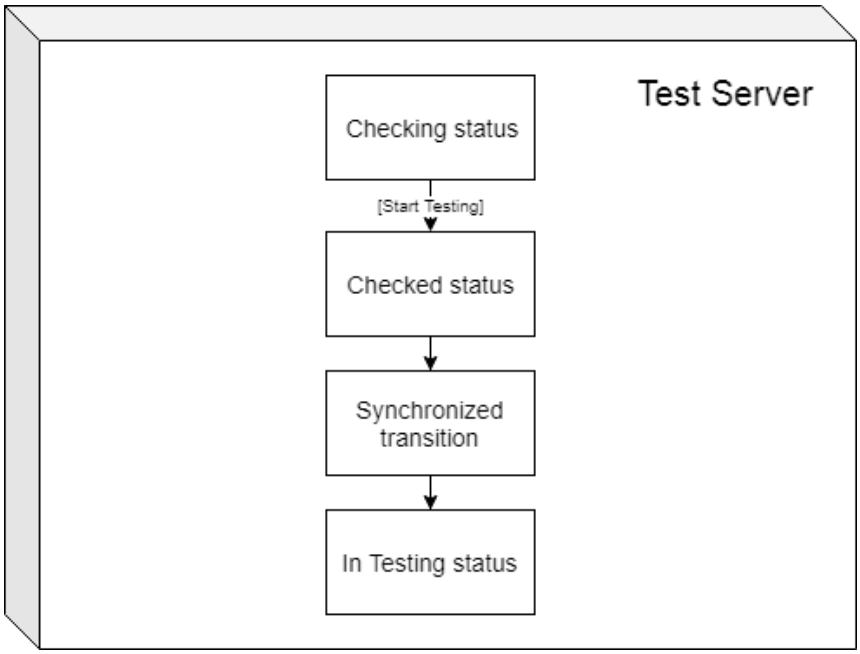

Рисунок 2.1.3 – Перевод запроса на изменения из статуса «Проверяется» (Checking) в статус «В тестировании» (In Testing) на Test-сервере

4. Выполнение отката импортированного запроса на изменение после выполнения тестирования новых модификаций. Данная функциональность необходима для того, так как идентификаторы, полученные от биллинговой системой на TEST-сервере, необходимы только для тестирования и не могут быть использованы при выполнении импорта на Production-сервер. После выполнения отката пользователь должен выполнить стандартные шаги, то есть запустить Test-задачу на Configuration-сервере, во время которой будет выполнена синхронизация с биллинговой системой, а затем сгенерированный архив повторно импортировать на Test-сервере с синхронизированными идентификаторами биллинговой системы, которые были определены на Configuration-сервере. Благодаря этому биллинговая система на Test-сервере будет иметь те же самые идентификаторы, что и на Configuration-сервере. Данная функциональная возможность изображена на рисунке 2.1.4.

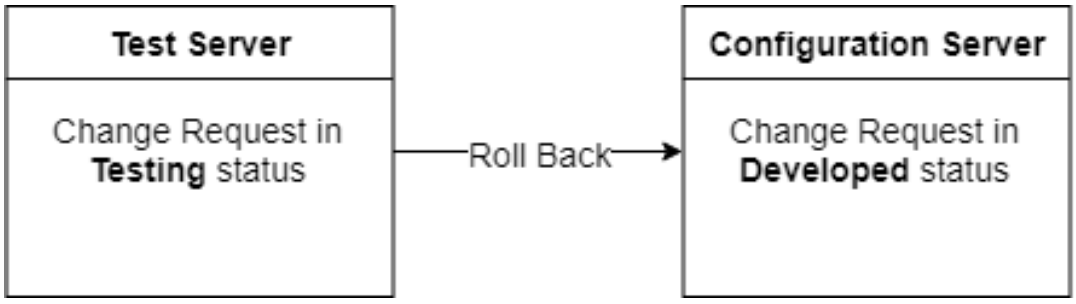

Рисунок 2.1.4 – Возможность отката с Test-сервера импортированного запроса на изменение после выполнения тестирования

Первая и вторая возможности будут реализованы в компоненте PLM; третья и четвертая требуют изменений не только в PLM, но и в компонентах, которые отвечают за синхронизацию каталога с биллинговой системой «Управление жизненным циклом лояльности» (Loyalty Lifecycle Management) и «Интеграция управления оценки выставления счётов» (Rating Billing Management Integration), поскольку объекты «Loyalty Management» и объекты, которые были внесены в биллинговую систему при синхронизации также необходимо откатить для обеспечения согласованной функциональности.

Несмотря на ускорение процесса тестирования, иногда рекомендуется после отдельного тестирования собрать все проверенные запросы на изменение в одной среде и проверить поведение всей измененной области перед переходом на рабочий сервер. Особенно это стоит выполнять, если при выполнении тестирования возникали проблемы с биллинговой системой при синхронизации с каталогом или, когда пользователь проверял новые модификации путем их подключения (продажи) клиентам, которые являются тестовыми данными [27].

Новый алгоритм работы PLM-компоненты с внедрением метода поддержки нескольких тестовых серверов для параллельного тестирования представлен на рисунке 2.1.5.

Как видно из рисунка 2.1.5 изменения были внесены в работу Development Task, так теперь по её завершению она возвращает архив, который содержит в себе все разработанные под запросом на изменение модификации. При чем данные изменения не были синхронизированы с биллинговой системой.

Теперь пользователи после завершения разработки новых модификаций в каталоге могут непосредственно приступить к тестированию. Для этого каждый из них выполняет импорт на различные не занятые ещё другими пользователями Test-сервера.

Если импорт был выполнен успешно, то пользователи могут параллельно каждый на своём окружении инициализировать процедуру перевода запроса на изменение в состояние «В тестировании».

При выполнении это процедуры статус запроса на изменение меняется в следующем порядке:

1. Checked.
2. Synchronization.
3. In Testing.

Стоит отметить, что если во время синхронизации каталога с биллинговой системой произойдет сбой, то статус запроса на изменения останется в состоянии Checked.

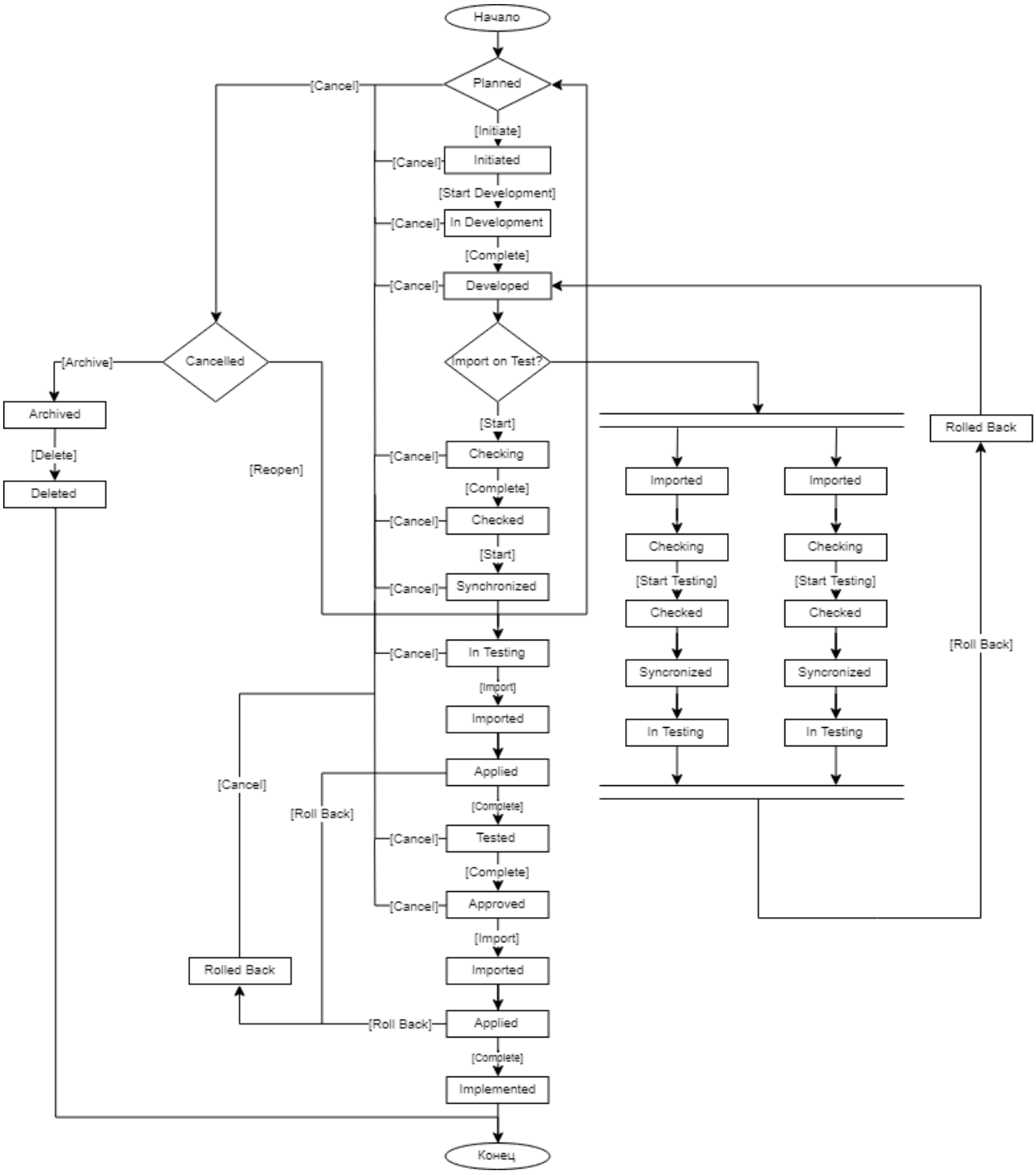

Рисунок 2.1.5 – Новый алгоритм работы PLM-компоненты, поддерживающий несколько тестовых серверов для параллельного тестирования запросов на изменение

На данном этапе пользователь выполняет тестирование запроса на изменение на тестовом окружении в то время, как другие пользователи также имеют возможность — это делать без нахождения в очереди, так как они находятся на других тестовых окружения.

После того как тестирование было успешно завершено или пользователь обнаружил во время него ошибки в конфигурации услуги или продукта, то он должен выполнить процедуру отмены запроса на изменение, то есть все изменения, которые были импортированы на сервере, будут удалены как из каталога, так и из биллинговой системы.

В случае, когда это было выполнено из-за обнаружения конфигурационной ошибки, пользователь должен вновь начать с Development Task, так как только в ней и в запущенном состоянии пользователь может изменять каталог. Как только необходимые исправления будут выполнены пользователь вновь должен повторить импорт на Test-сервер для тестирования.

Если тестирование было успешным, то пользователь, как упоминалось выше, должен выполнить отмену запроса, тем самым инициализировать удаления всех модификаций как из каталога, так и из биллинговой системы, так как идентификаторы, которые были получены из биллинговой системы не пригодны для дальнейшего жизненного цикла продукта.

В данном подходе стоит помнить, что если мы хотим синхронизировать TEST-сервера между собой, то необходимо, чтобы запрос на изменение, утвержденный для импорта на Production-сервер, был в обязательном порядке импортирован пользователем на основной TEST-сервер, иначе каталоги предложений буду отличаться друг от друга.

Исходя из того, что используемый для распараллеливания тестирования запрос на изменение с помощью нескольких TEST-серверов не синхронизирован с биллинговой системой до тех пор, пока не будет инициализирован процесс синхронизации данного запроса на изменение, который находится на тестовом окружении, то есть, не содержит необходимых идентификаторов, следует отметить, что для синхронизации сред и импорта

запроса на производственный сервер необходимо использовать архив, который будет сгенерирован при переходе запроса в состояние In Testing. После этого он уже будет синхронизирован с биллинговой системой конфигурационного сервера, который присвоит необходимые идентификаторы нужным объектам каталога. И в таком случае при импорте на основной Test-сервер, биллинговая система, которая связанна с каталогом сервера не будет генерировать новые идентификаторы, а получит уже готовые идентификаторы, которые были сгенерированы на конфигурационном сервере [27].

По окончанию же импорта на Production-сервер, пользовать должен будет также выполнить импорт модификаций, которые были импортированы на производственный сервер, на другие тестовые окружения, чтобы каталоги всех серверов совпадали друг с другом и не было разногласий в объектах, хранящихся в каталогах на различных серверах.

Данный подход не содержит в себе цели полностью отказаться от прежнего тестирования, в связи с чем периодически рекомендуется тестировать разработанные запросы на изменение не только параллельно, но и в той же среде, особенно если временные сроки это позволяют.

Для того чтобы использовать вышеизложенные изменения в работе PLM-компоненты необходимо внести следующие изменения в работу нескольких модулей, а именно:

- модуль «Управление запросами на изменение» (Change Request Management);
- модуль «Управление жизненным циклом лояльности» (Loyalty Lifecycle Management).

Рассмотрим подробнее изменения, которые будут внесены в работу модуля «Управление запросами на изменение»:

1. PLM-компонента должна предоставить пользователю возможность автоматически экспортировать запрос на изменение и все связанные с ним каталожные модификации после разработки.

2. PLM-компонента должна предоставлять возможность создавать упакованный архив файл, который содержит текущий запрос на изменение и все связанные с ним данные каталожных изменений и прикреплять их к текущему запросу на изменение в результате завершения Development Task.

2.1. Процесс, отвечающий в PLM-компоненте за создание архивов для импорта запросов на изменение, также должен автоматически создать архив с запросом на изменение в состоянии «Разработано» (Developed).

2.2. Архив должен храниться в отдельном параметре. Один параметр должен использоваться для хранения архива с запросом на изменение в состоянии «Разработано» (без идентификаторов объектов, полученных от биллинговой системы), другой для сохранения архива с запросом на изменение в состоянии «В тестировании» (с идентификаторами объектов, полученных от биллинговой системы во время синхронизации).

2.3. Должна быть реализована возможность импортировать архив с запросом на изменения из состояния «Разработано» или «В тестировании» на TEST-сервера.

3. Инструментарий PLM-компоненты должен предоставить пользователю возможность импортировать экспортированный запрос на изменение с применением модификаций на импортируемом сервере.

3.1. Test-сервер должен применяться в качестве импортируемого для архива, сформированного по завершению Development Task.

3.2. Синхронизация с биллинговой системой должна начинается с транзакции «Проверено» (Checked) → «В тестировании» (In Testing) на Test-сервере.

3.3. TEST-сервер должен поддерживать два вида импорта:

3.3.1. Когда импортируемый архив не имеет идентификаторов биллинговой системы, присвоенных на конфигурационном сервере, и после тестирования импортируемый запрос на изменение должен быть откачен с сервера. Процесс тестирования выполняется по завершению транзакции «Синхронизировано» (Synchronized) → «В тестировании» (In Testing).

3.3.2. Когда импортируемый архив имеет идентификаторы биллинговой системы, присвоенных на конфигурационном сервере, а также запрос на изменение должен быть применен на сервере по завершению успешного импорта на производственном сервере. Процесс тестирования в таком случае выполняется сразу после импорта запроса на изменение на Test-сервер. В данном случае запрос имеет статус «Применен» (Applied)

3.4. Инструментарий PLM-компоненты должен предоставлять возможность завершения тестирования на Test-сервере при использовании второго варианта импорта запроса на изменение.

3.5. Состояние запроса на изменение должно автоматически переводиться в «Применено» (Applied) при успешном завершении импорта на Test-сервере при втором варианте импорта.

4. Инструментарий PLM-компоненты должен предоставить пользователю возможность откатить импортированный запрос на изменение и все примененные модификации вручную.

Также стоит отметить, что будет необходимо внести некоторые изменения в конфигурацию PLM-компоненты, так как теперь система должна предоставлять пользователю возможность указать более одной среды с типом Test.

Далее рассмотрим изменения, которые будут внесены в работу модуля «Управление жизненным циклом лояльности».

В данном случае все изменения, которые необходимо будет внести в данном модуле, будут относится к операции отката запроса на изменения с тестового сервера и биллинговой системы.

При откате запроса на изменение с тестового сервера система должна применить следующие операции к объектам, затронутым в рамках данного запроса на изменение:

1. Отменить изменения, примененные к сущностям «Loyalty Offering Involvement» и «Loyalty Price Value», и установите для них статус «Активный» на тестовом сервере, если эти сущности были изменены. Данные сущности

хранятся в биллинговой системе и являются объектами начисления баллов лояльности, которыми можно будет оплатить часть покупки. Они начисляются при за едино разовые покупки. Например, за покупку телефона или роутера.

2. Удалить сущности «Loyalty Offering Involvement» и «Loyalty Price Value» с тестового сервера, если эти сущности были созданы.

3. Отменить изменения, примененные к сущностям «Loyalty Programs» и «Loyalty Categories», и устанавливать для них статус «Активный» на Test-сервере, если эти сущности были изменены.

4. Отменить изменения, примененные к сущностям «Loyalty Programs» и «Loyalty Categories» на Test-сервере и биллинговой системе, если эти сущности были изменены.

5. Удалить объекты «Loyalty Programs» и «Loyalty Categories» с Test-сервера и биллинговой системы, если они были созданы.

6. Отменить изменения, примененные к сущностям «Loyalty Offering Involvement» и «Loyalty Price Value», и установить для них статус «Активный» на Test-сервере, если эти сущности были изменены.

7. Изменить статус сущностей «Loyalty Offering Involvement» и «Loyalty Price Value» на «В разработке» (Under Development) на конфигурационном-сервере, если эти сущности были созданы.

8. Отменить изменения, примененные к сущностям «Loyalty Programs» и «Loyalty Categories», и установить для них статус «Активный» на конфигурационном-сервере, если эти сущности были изменены.

9. Отменить изменения, примененные к объектам «Loyalty Programs» и «Loyalty Categories» на конфигурационном сервере и связанной с ним биллинговой системе, если эти объекты были изменены.

10. Устанавливать статус объектов «Loyalty Programs» и «Loyalty Categories» на «В разработке» на конфигурационном-сервере, если эти объекты были созданы.

11. Удалить объекты «Loyalty Programs» и «Loyalty Categories» с конфигурационного сервера и связанного с ним биллинговой системы, если эти объекты были созданы.

12. Откат должен быть доступен для запроса на изменения, который был импортирован без идентификаторов биллинговой системы в следующих статусах:

- Checking (Проверяется);
- Checked (Проверен);
- In Testing (В тестировании).

13. Откат должен быть доступен для запроса на изменения, который был импортирован с идентификаторами биллинговой системы только в статусе Applied (Применен).

Данный откат изменений должен инициализироваться с помощью специального инструментария, который должен быть применен на всех Test-серверах. В данном случае после отката объекты из биллинговой системы также будут удалены.

Внедрение данного алгоритма в работу PLM-компоненты добавила новые функциональные возможности, которые представлены на рисунках 2.1.6 – 2.1.12.

Теперь при завершении Developed Task будет генерироваться архив, который будет прикреплен к Development Task и к самому запросу. На рисунке 2.1.6 представлено отображение архива на Development Task.

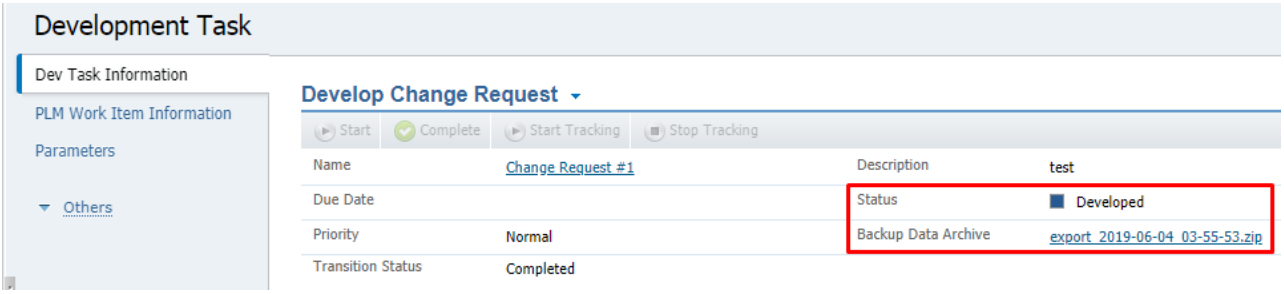

Рисунок 2.1.6 – Создание архива для импорта на Test-сервер в статусе Developed

После выполнения импорта данного архива на Test-сервер запрос на изменение будет иметь статус Checking. На рисунке 2.1.7 изображено данное состояние запроса.

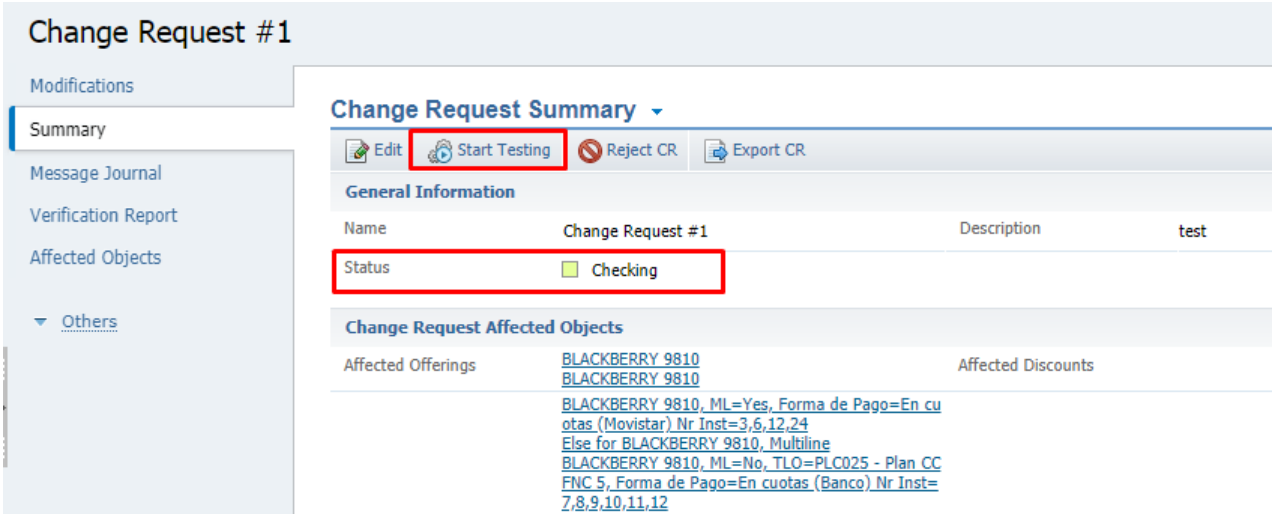

Рисунок 2.1.7 – Запрос на изменение на Test-сервере после выполнения импорта архива из Developed статуса

Для того чтобы приступить к тестированию пользователю необходимо нажать кнопку «Start Testing». После этого статус запроса на изменение изменит сначала статус на Checked и начнется синхронизация с биллинговой системой. Пока процесс не будет завершен или прерван из-за обнаружения ошибки во время синхронизации, кнопки, влияющие на статус запроса, будут не доступны. На рисунках 2.1.8 – 2.1.9 продемонстрирована смена статуса запроса на изменение и запуск процедуры синхронизации с биллинговой системой.

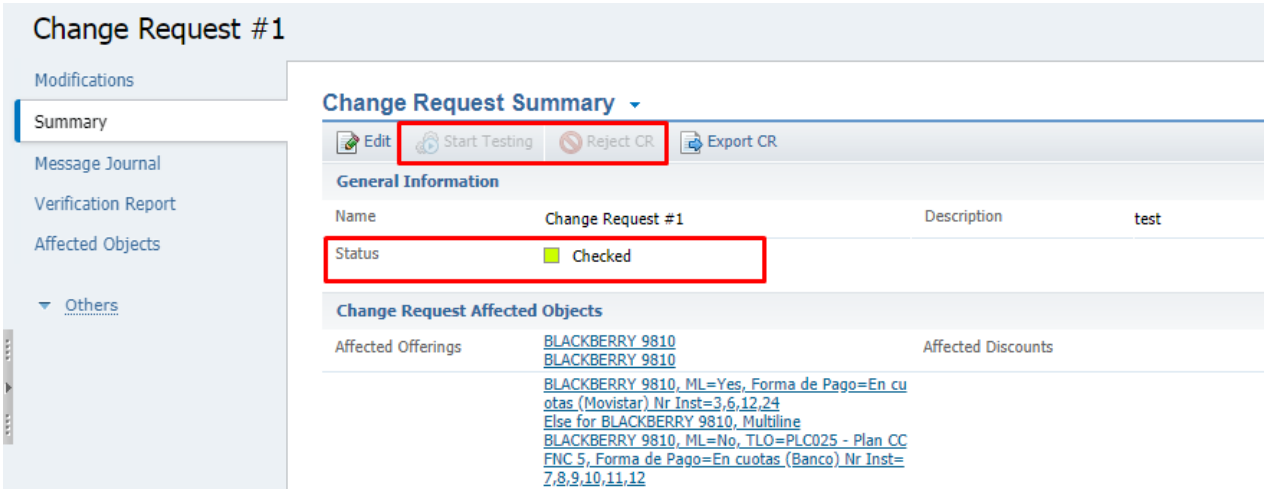

Рисунок 2.1.8 – Перевод запроса на изменение в тестирование

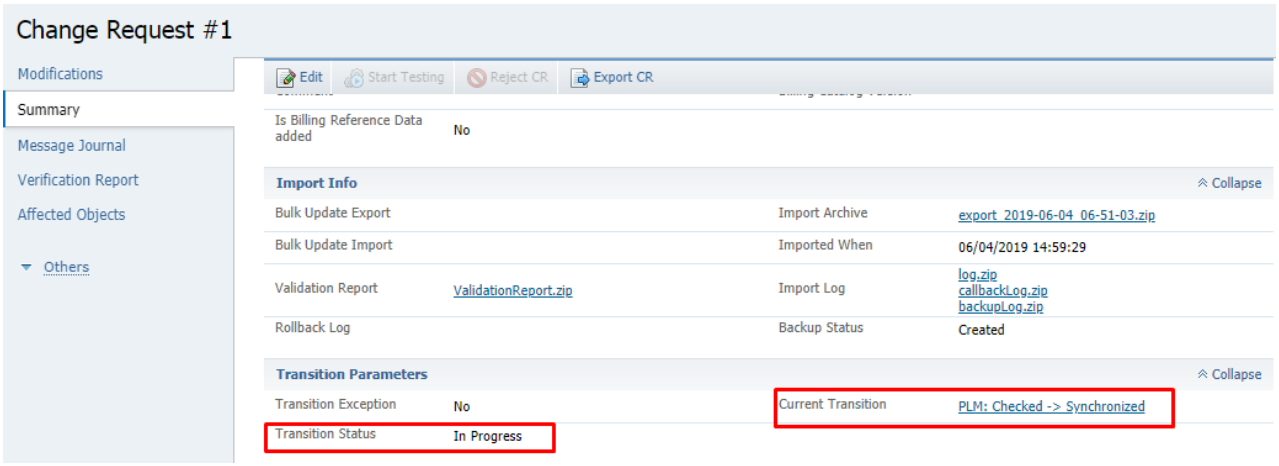

Рисунок 2.1.9 – Выполнение синхронизации каталожных модификаций, принадлежащих запросу на изменение, с биллинговой системой

После того как процесс синхронизации был успешно выполнен, статус запроса на изменение изменяется на In Testing. В данном состоянии пользователь может приступить к тестированию новых каталожных модификаций.

На рисунках 2.1.10 – 2.1.11 представлен запрос на изменение в состоянии In Testing.

После того как пользователь завершит тестирование запроса на изменение, он обязан откатить данный запрос на изменение с Test-сервера с помощью «Reject CR». После чего статус запроса на изменение станет Cancelled. На рисунках 2.1.12 – 2.1.13 изображена данная смена статуса.

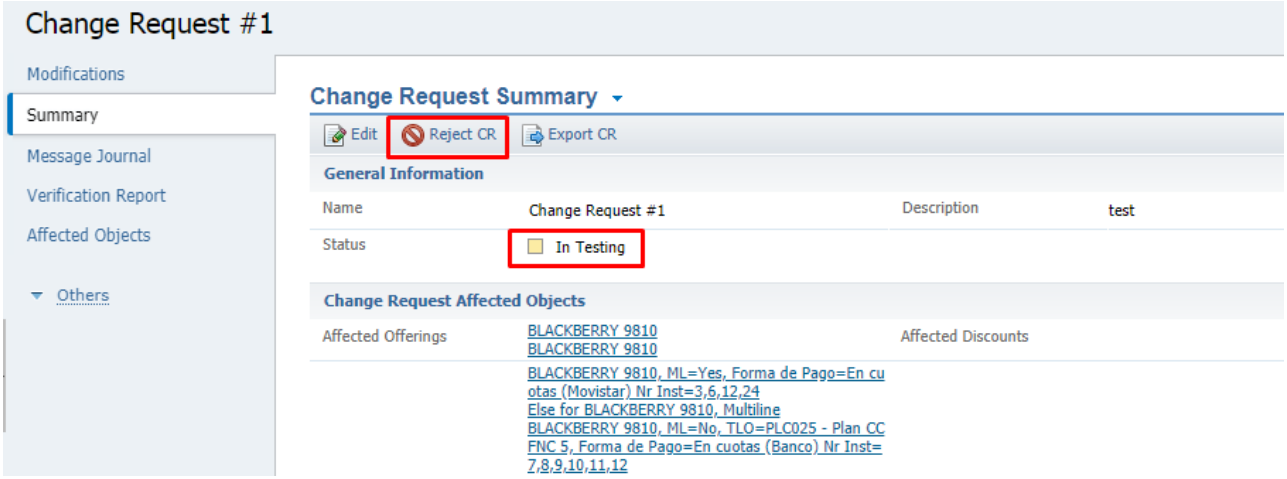

Рисунок 2.1.10 – Запрос на изменение в состоянии In Testing

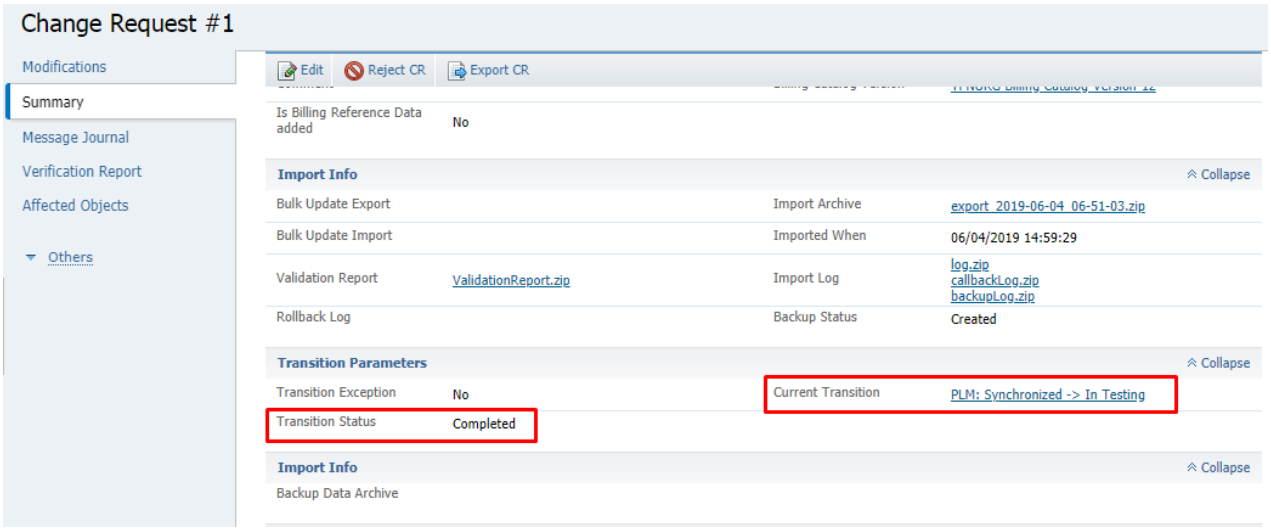

Рисунок 2.1.11 – Процесс синхронизации с биллинговой системой завершен успешно

Рисунок 2.1.12 – Запрос на изменение в статусе Cancelled

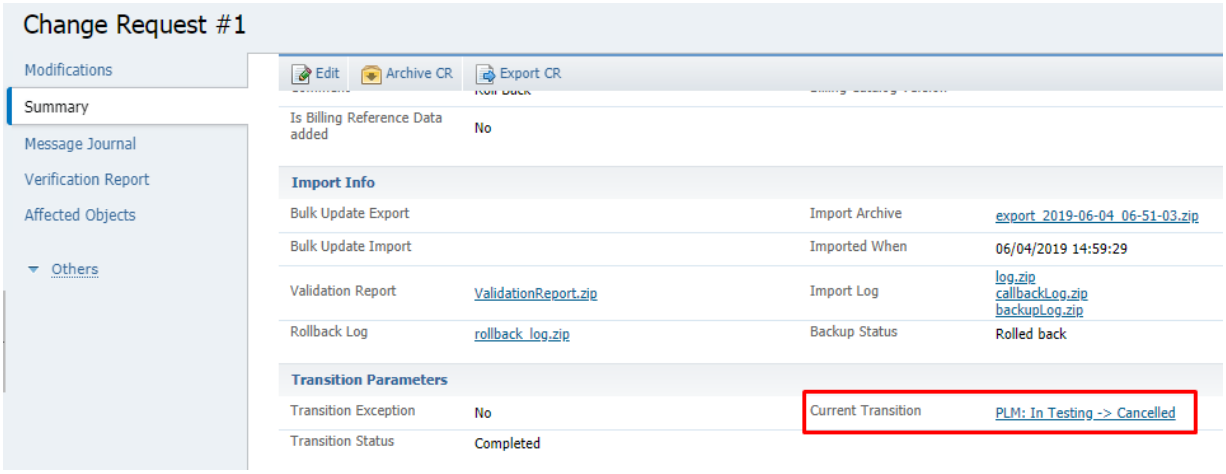

Рисунок 2.1.13 - Изменение статуса запроса на изменение с In Testing на Cancelled

Таким образом, новая функциональность делает PLM-процессы более эффективным и быстрым, потому что несколько пользователей смогут параллельно тестировать несколько запросов на изменение. Благодаря этому тестирование других запросов на изменение не блокируется в тестовой среде, а активация каталога в биллинговой системе происходит только на Test-серверах, при чем на конфигурационном сервере этого не происходит.

Но стоит отметить, что для реализации этого подхода необходимо иметь несколько сред тестирования для параллельного тестирования запросов на изменение. А идентификаторы биллинговой системы, сгенерированные в тестовой среде во время тестирования, не могут быть использованы как ссылка для любых новых объектов, импортируемых в дальнейшем на основной тестовое или производственное окружение.

2.2 Проектирование инструмента, позволяющего выполнить группировку нескольких модификаций из разных запросов на изменение под одним главным в PLM – компоненте

На данный момент каждый запрос на изменение тестируется отдельно. В тестировании на сервере может быть только один запрос на изменение. Если на сервере имеется много запросов на изменение, которые готовы к синхронизации с биллинговой системой и тестированию, может быть не

практично и долго тестировать запросы на изменение и их модификации по одному. Таким образом, на текущий процесс требует больше времени [30].

Данный инструментарий позволит выполнять тестирование нескольких запросов благодаря их группировки на одном сервере. В связи с чем при данном подходе не нужно будет вводить дополнительные окружения.

В связи с чем можно предложить следующий подход к решению проблемы с невозможностью параллельного тестирования нескольких запросов на изменение: группировать модификации нескольких запросов на изменение в рамках одного из существующих, если все они имеют одинаковый статус.

Все запросы на изменения должны быть подготовлены и не иметь синхронизированных изменений. Отдельные запросы на изменение, модификации которых сгруппированы под главным запросом, будут обрабатываться до состояния «Разгруппировать» (Ungrouping) [30].

Такой подход может помочь более эффективно протестировать модификации каталога продуктов с точки зрения их времени и усилий.

Используя новый подход, следует отметить, что логика переходов между статусами для сгруппированных запросов на изменение будет немного изменена.

Рассматривая детально новую функциональную возможность для PLM-компоненты, стоит отметить следующее:

1. Система должна предоставлять возможность группировать существующие запросы на изменение под другим запросом, если эти запросы разработаны и их изменения еще не синхронизированы с биллинговой системой.

1.1. При группировке нескольких запросов система должна перемещать модификации всех запросов под тот запрос, который был пользователем выбран как главный. Статус запросов, чьи модификации были сгруппированы, меняется на «Отменен» (Canceled).

1.2. Система должна запрещать группировку, если изменения выбранных запросов на изменение связаны с различными биллинговыми каталогами выставления счетов или запросы имеют разные статусы.

2. Система должна предоставлять возможность разгруппировать запрос на изменение из-под главного запроса, если модификации под главным запросом на изменение еще не были синхронизированы с биллинговой системой.

3. При выполнении операции разгруппировки запроса на изменение система должна удалять модификации других запросов на изменение из-под главного запроса и переместить их обратно в запросы, в которых они были разработаны изначально.

Главный же запрос на изменение при этом будет переведен в состояние «Инициирован» (Initiate).

Ниже представлен рисунок 2.2.1, который описывает группировку и разгруппировку запросов на изменение и их поведение во время этого процесса.

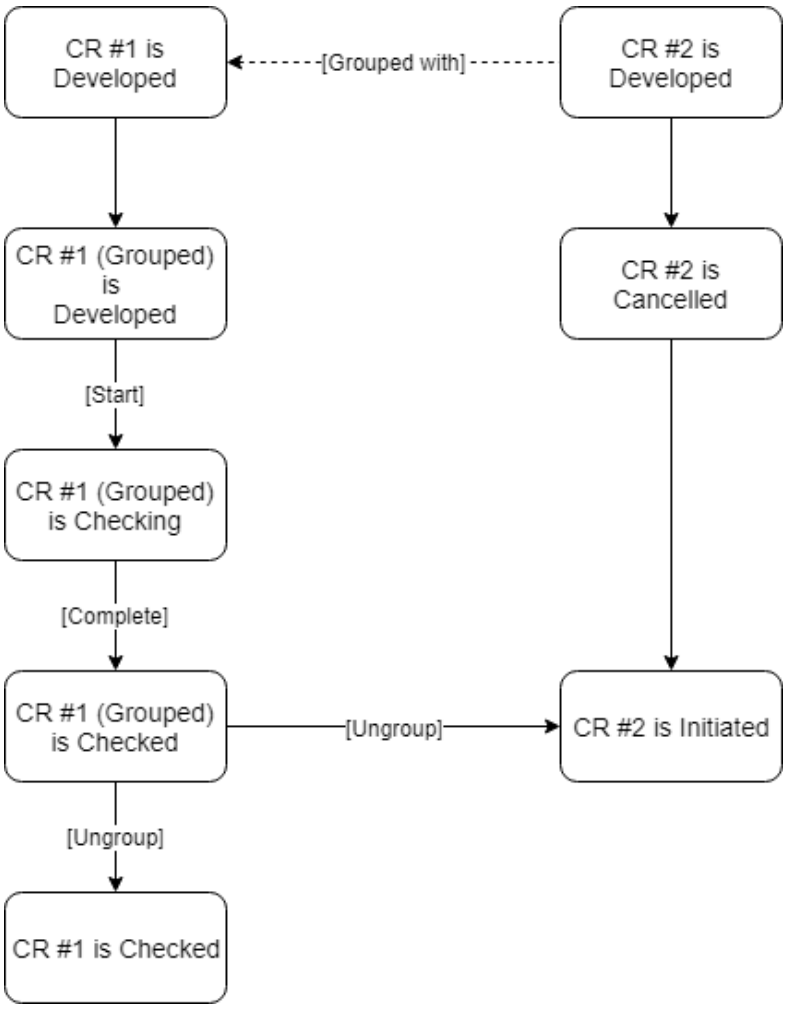

Рисунок 2.2.1 – Процесс группировки и разгруппировки «Запросов на изменение»

4. Система должна предоставлять возможность просмотра списка сгруппированных запросов на изменение из-под главного запроса.

Стоит отметить, что сгруппировать можно как минимум два запроса.

При этом их статус должен быть одинаковым и их модификации не должны быть синхронизированы с биллинговой системой.

В результате группировки запросов на изменение: их модификации буду перемещены под выбранный пользователем в качестве главного запрос, который будет иметь такой же статус, что и все запросы перед началом данной процедуры.

Главным же запросом на изменение пользователь может выбрать как совершенно новый запрос, который не имеет своих собственных модификаций, так и один из запросов, который подготовлен к группировке.

Однако, в качестве главного запроса следует выбирать отдельный запрос, который не содержит каких-либо модификаций. Это позволит минимизировать вероятность ошибок при синхронизации и импорте на другие окружения [30].

После группировки пользователь готов к параллельному тестированию нескольких запросов в составе одного запроса, который в себе содержит модификаций сгруппированных запросов.

Система позволяет выполнить группировку и разгруппировку только для тех запросов на изменение, которые находятся в следующих статусах:

- Разработан (Developed).
- Проверяется (Checking).
- Проверен (Checked).

На рисунке 2.2.2 наглядно продемонстрирована данная логика поведения системы при инициализации данных процедур.

5. Если тестирование сгруппированного запроса на изменение было неудачным или возникла ошибка при синхронизации с биллинговой системой или импорте на тестовое или производственное окружение, то запрос необходимо откатить с этого окружения.

В связи с чем его модификации будут удалены из импортируемых окружений и на конфигурационном сервере он будет переведен в состояние «Инициирован» (Initiated) с использованием переходов общего жизненного цикла запроса на изменение.

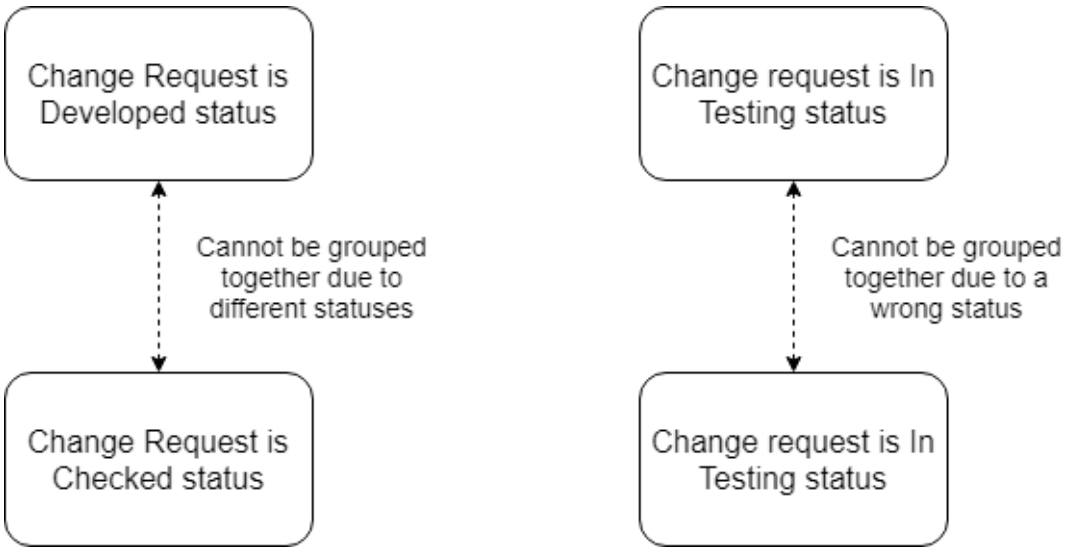

Рисунок 2.2.2 – Некорректные статусы «Запросов на изменение» при инициализации процедуры их группировки

6. Список модификаций для просмотра должен быть доступен как в главном запросе на изменение (включает в себя модификации из всех сгруппированных запросов), так и в исходном запросе (включает только собственные модификации).

Данный инструмент был внедрен в текущую работу PLM-компоненты. На рисунках 2.2.3 – 2.2.8.

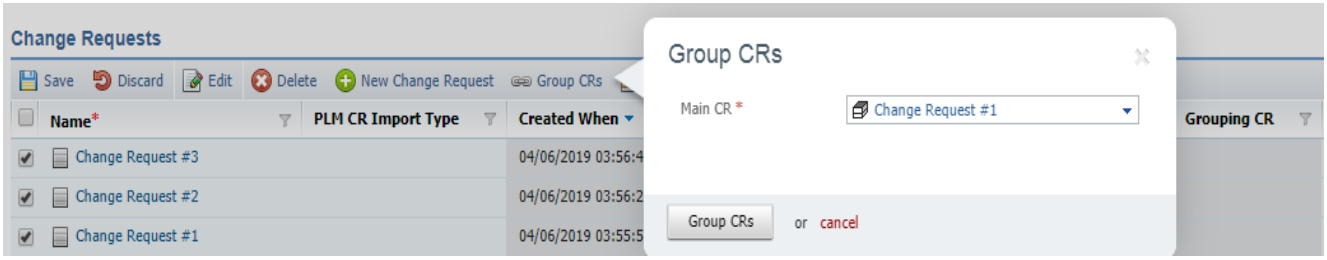

Рисунок 2.2.3 – Группировка запросов на изменение

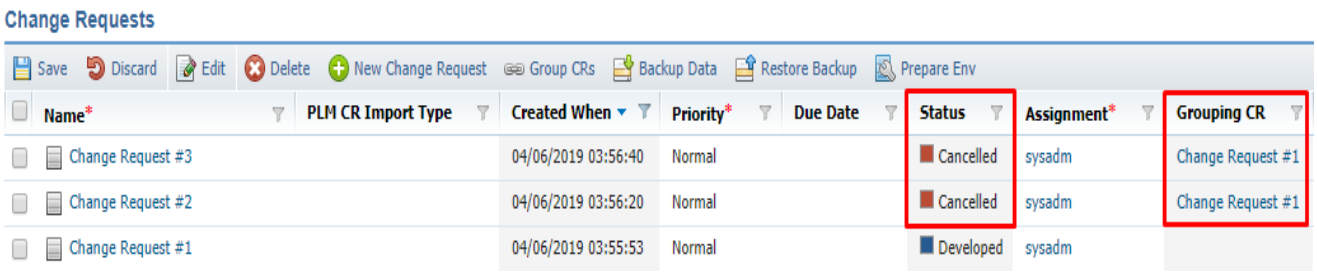

Рисунок 2.2.4 – Сгруппированные запросы

На рисунках 2.2.3 – 2.2.4 изображен основной процесс группировки, который состоит из следующих шагов:

1. Пользователь выбирает запрос на изменение в списке запросов.

2. Пользователь нажимает кнопку «Group CRs».

3. Если пользователь выбирает только один объект, система выводит предупреждение. В противном случае система отображает всплывающее окно «Group CR» со следующими полями:

- основной запрос на изменение (ссылка на запрос, которая содержит отфильтрованные выбранные запросы на изменение).

4. Пользователь выбирает группировку запросов на изменение и нажимает кнопку [ОК].

5. Система проверяет выбранные запросы по их состоянию:

5.1. Если состояние всех сгруппированных запросов на изменение - Developed, Chcking или Checked, то система выполнит группировку.

5.2. В противном случае система отображает предупреждающее сообщение и прерывает процесс.

5.3. Если состояние всех выбранных запросов на изменение не совпадает, система отображает предупреждающее сообщение и прерывает процесс.

6. Система проверяет статус запроса на изменение, которые будет главным при группировке.

6.1. Если какой-либо запрос на изменение уже включен в какую-либо группу, система отображает предупреждающее сообщение.

6.2. Если какой-либо выбранный запрос на изменение уже является групповым, система отображает предупреждающее сообщение.

7. Система перемещает модификации из сгруппированных запросов на изменение в главный запрос.

7.1 Происходит изменение идентификаторов для выбранных запросов на изменение, где новым идентификатором становится значение главного запроса.

7.2 Сохраняет отношения родительский запрос на изменение - дочерний запрос на изменение в плоской таблице «plm_grouping_crs» со следующими столбцами: group_cr, grouped_cr.

7.3 Сохраняет модификации, сгруппированную под основным запросом на изменение, в плоской таблице «plm_grouped_modifications» со следующими столбцами: modif_id, grouped_cr

7.4 Изменяет состояние дочерних запросов на изменение с Developed, Checking или Checked на Cancelled.

8. Система отображает перезагруженную вкладку «Изменить запросы».

При группировании также учитывается все изменения, которые происходят с данными, то есть все модификации переносятся под основной и сохраняется история изменений при группировке.

Для просмотра сгруппированных запросов необходимо перейти на вкладку «Group CRs» на основном запросе.

На данной вкладке пользователю доступны следующие функции: добавление нового запроса в группу и удаление запроса из группы. Данные функциональные возможности изображены на рисунках 2.2.5 – 2.2.6.

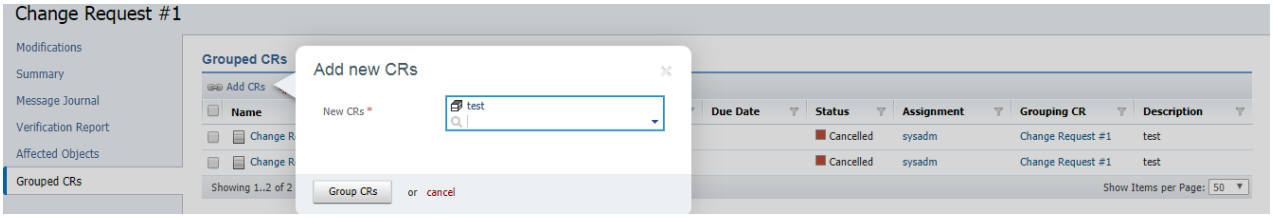

Рисунок 2.2.5 – Добавление нового запроса на изменение к текущей группировке

Для того чтобы добавить новый запрос на изменение в группу, пользователю необходимо выполнить следующие шаги:

1. Пользователь нажимает кнопку «Add CRs».

2. Система отображает всплывающее окно с действительным (не сгруппированным и имеющим тот же статус, что и у основного запроса на

изменение, который является главным, например, «Разработанным») списком запросов на изменение.

3. Пользователь выбирает запрос или запросы и нажимает кнопку [Group CRs].

4. Система определяет текущий запрос на изменение как основной и запускает процесс проверки.

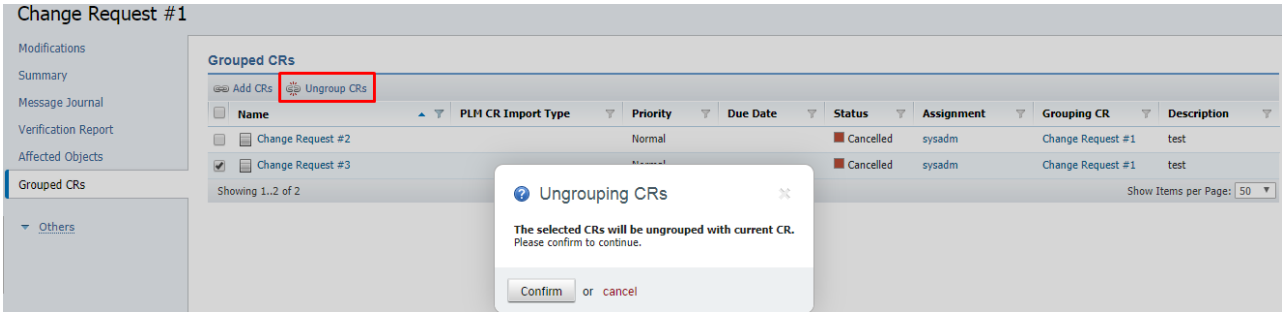

Рисунок 2.2.6 – Удаление запроса на изменение из состава группы

Пользователь также имеет возможность удалить запрос из основного запроса, то есть выполнить разгруппировку. Для этого ему необходимо выполнить следующие шаги:

1. Пользователь выбирает один или несколько запросов на изменение из отображаемого списка запросов, сгруппированных с текущим.

2. Пользователь нажимает кнопку «Ungroup CRs».

3. Система отображает диалоговое окно подтверждения.

4. Пользователь подтверждает или отменяет операцию.

5. В случае подтверждения система проверяет состояния группировки основного запроса на изменение.

5.1 Статус группы запроса должен быть равен Planned, Initiated или In Development для продолжения. В противном случае система отображает предупреждающее сообщение.

6. Система разгруппировывает запросы в следующем порядке:

6.1 Восстанавливает все модификации в разгруппированном запросе на изменение из основного.

6.2 Изменяет состояние разгруппированного запроса на Planned.

7. Система обновляет вкладку «Grouped CRs».

На вкладке «Modifications» теперь для модификаций добавлен дополнительный столбец, который отображает в каком именно запросе было изначально зафиксировано изменение в каталоге.

Также объекты, которые были подвергнуты изменению или созданы в каталоге, отображаются на вкладке «Summary».

Данные изменения отображены на рисунках 2.2.7 – 2.2.8.

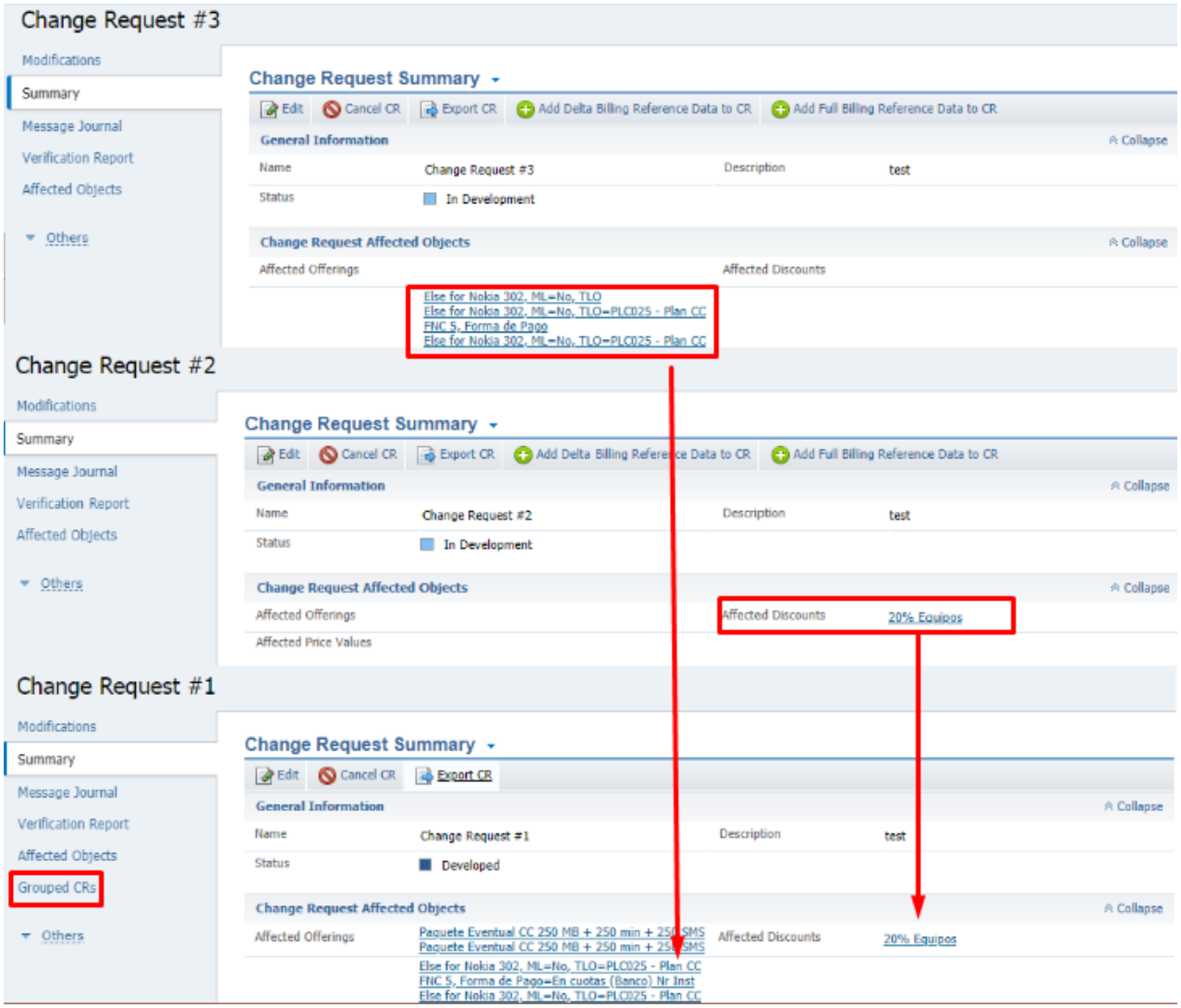

Рисунок 2.2.7 – Перенос объектов под главный запрос на изменение после выполнения группировки

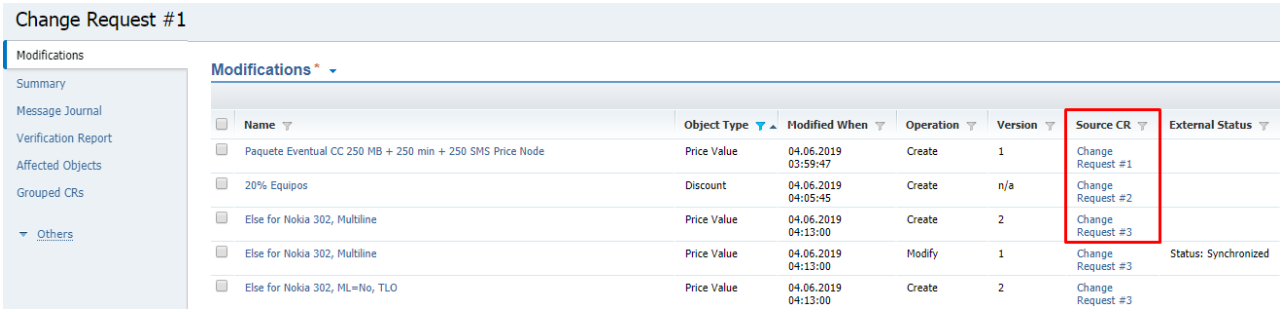

Рисунок 2.2.8 – Отображение сгруппированных модификаций под главным запросом

Таким образом, группировка запросов на изменение поможет пользователям синхронизировать и тестировать каталожные модификации параллельно на одном окружении, что на данный момент невозможно. Новый инструментарий повысит производительность пользователей и сократит их усилия и время на обработку и тестирование изменений в запросах на изменение.

2.3 Проектирование алгоритма, обеспечивающего возможность группировки идентичных значений цен «Одноразового сервиса» при синхронизации с биллинговой системой

В настоящее время во время синхронизации каталога и биллинговой системы, последней приходится создавать один внебиржевой тариф для каждого значения цены одноразового сервиса (One Time Service, OTS), и каждое ценовое значение имеет свой уникальный идентификатор связанного внебиржевого тарифа.

Если в разных ветвях дерева цен в каталоге имеются одинаковые значения цен, то биллинговая система создает дубликаты внебиржевых тарифов, вместо повторного использования одного из них и, таким образом, повышается время выполнения синхронизации, что в свою очередь влияет на скорость начала тестирования разработанного запроса на изменение [37].

В данном случае имеет место быть предложение изменить алгоритм работы текущего процесса синхронизации каталога с биллинговой системой,

внедрив в неё метод группировки идентичных ценовых значений, которые будут определятся в различные группы благодаря схожим значениями определенных критериев, связанных параметрами цен во время синхронизации с биллинговой системой и создавать единый внебиржевой тариф для каждой группы. Затем этот тариф связывается с каждым из сгруппированных ценовых значений [33].

Новый подход в перспективе можно будет использовать для всех типов продуктов и тарифов, которые находятся в каталоге.

Однако, на данный момент наибольшую эффективность данная методика принесет именно для одноразовых сервисов, так как они имеют значительно более сложное дерево цен, в связи с чем группировка даст наиболее лучший результат при проверке данного подхода.

Для того, чтобы убедиться в эффективности данного подхода, необходимо будет провести ряд нагрузочных тестов, которые подтвердят или опровергнут значимость внедрения данной модификации в текущую работу PLM-компоненты.

Не мало важным при применении данного метода будет возможность отказаться от повторной автоматической синхронизации объектов, содержащихся в каталоге, так как это позволит снизить временные затраты в случае ресинхронизации из состояния «В тестировании» на конфигурационном сервере, а также при откате модификаций с тестового или производственного серверов при обнаружении ошибки в конфигурации, которые содержатся в запросе на изменение [33].

На рисунке 2.3.1 изображен процесс группировки идентичных значений цен, который будет применяться только для одноразовых сервисов при их синхронизации с биллинговой системой.

Используя вышеизложенный подход необходимо помнить, что система должна сохранить возможность синхронизировать сущности, существующие в каталоге с биллинговой системой в соответствии с текущими правилами

сопоставления, то есть должна сопоставлять модель цены предложения (продукта или тарифа) с моделью, которая находится в биллинговой системе.

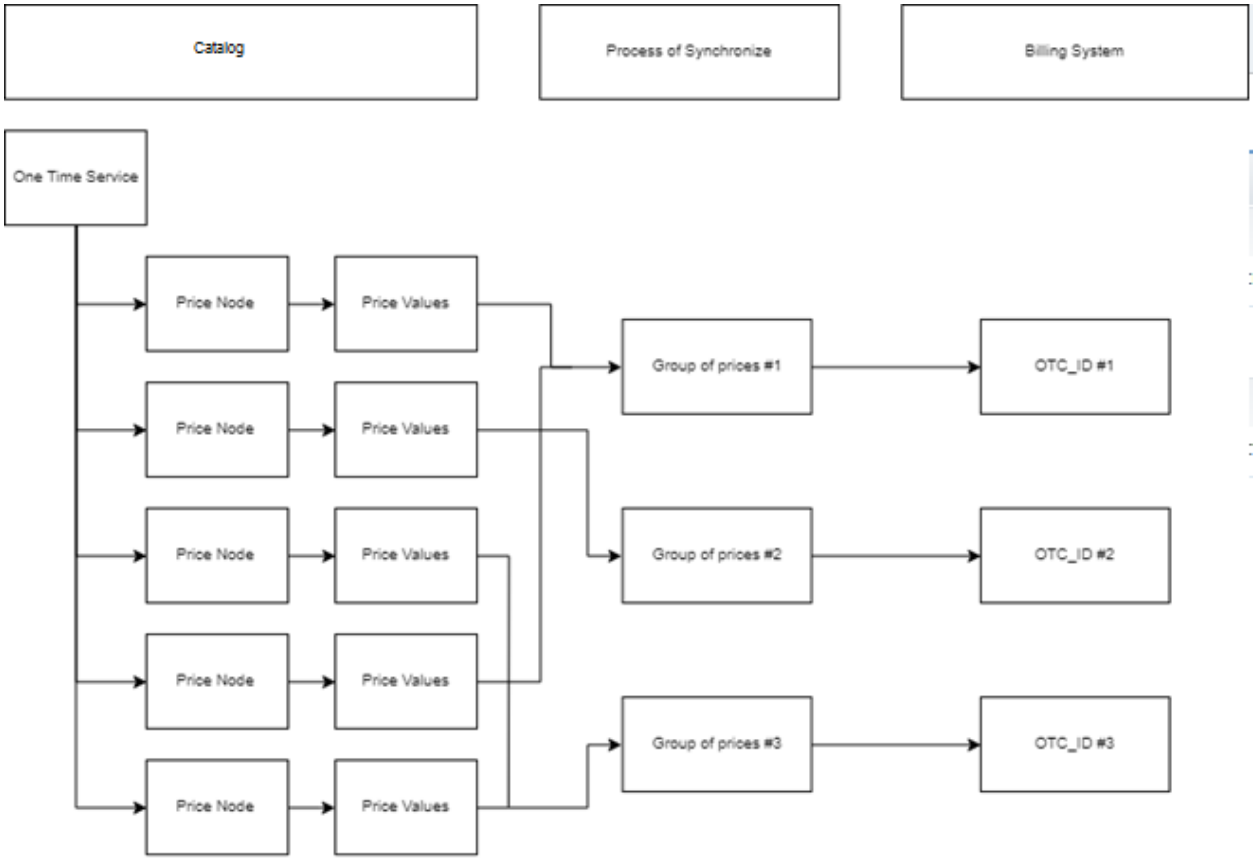

Рисунок 2.3.1 – Процесс группировки идентичных значений цен для одноразовых сервисов при их синхронизации с биллинговой системой

В таком случае под данный подход попадают следующие объекты, чьи критерии уникальности ценовых значений реализованы в модуле каталога предложений, хранящиеся в биллинговой системе [37]:

- код выручки (Revenue Code);
- код выручки по договорной цене (Negotiated Price Revenue Code);
- рассрочка (Installment);
- валюта (Currency).

При сравнении выше упомянутых параметров, которые относятся к ценовым значениям, будет учитываться регистр символов; нулевые значения рассматриваются как одно и то же значение. Таким образом, несколько

параметров будут считаться идентичными, если их наборы значений полностью совпадают.

Во время синхронизации система должна сгруппировать значения цены с одинаковыми значениями критериев и создать один внебиржевой тариф для каждой группы и связать каждое значение цены из группы с созданным внебиржевым тарифом.

При использовании данного подхода модуль «Rating Billing Management extension» компоненты «Product Information Management» должен придерживаться следующего поведения для иерархии экземпляров бизнес-продуктов:

1. Система должна позволять создавать экземпляры скидок в биллинговой системе.

2. Система должна позволять создавать экземпляры бизнес-продукта как единовременные экземпляры обслуживания без специального процесса подготовки и связанных экземпляров скидок в биллинговой системе.

Отдельно стоит уточнить, что в модуле «Rating Billing Management extension» следует рассчитывать внебиржевую цену независимо от того, существует ли переопределение цены, план рассрочки или скидка на значение цены.

Углубляясь в детали реализации предложенного подхода, отдельно стоит упомянуть, что класс «PriceValueDeleteMethod» должен удалить One Time Service (Одноразовый сервис) из биллинговой системы, если ни один идентификатор этого One Time Service не сохранен в значении цены в атрибуте «OTC_ID».

Из выше сказанного можно сделать вывод, что возможность создания одного внебиржевого тарифа для одинаковых ценовых значений позволит избежать создания дублирующих внебиржевых тарифов и сократит продолжительность синхронизации двух систем, что значительно снизит время, которое необходимо для тестирования и реализации запроса на изменение в системе.

Таким образом, при внедрении данных изменений в работу алгоритма синхронизации каталога с биллинговой системой, текущий алгоритм жизненного цикла запроса на изменение не будет переработан, а останется в своём естественном состоянии.

Данный фактор играет большую значимость в том, что пользователем не придется изучать новые функциональные возможности и поэтому данному подходу нужно меньше времени для того, чтобы быть встроенным в текущую работу PLM-компоненты.

Вывод по главе 2

В данной главе были спроектированы алгоритм и инструмент, в основе которых лежат методы, позволяющие сократить время необходимое пользователю для начала тестирования разработанного запроса на изменение и предоставляющие возможность параллельного тестирования нескольких запросов на изменение разными способами.

Был спроектирован алгоритм, который позволяет сократить время синхронизации каталога с биллинговой системой, за счет группировки идентичных ценовых значений одноразового сервиса.

Была описана работа новых алгоритмов и инструмента, которые повышают эффективность работы PLM-компоненты.

Глава 3 Оценка эффективности работы модернизированных процессов PLM-компоненты BSS-системы

3.1 Оценка эффективности работы PLM-компоненты с использованием алгоритма, предоставляющего возможность поддержки нескольких тестовых окружений для параллельного тестирования запросов на изменение в BSS-системы

Перед началом оценки эффективности алгоритма, предоставляющего возможность поддержки нескольких тестовых окружений для параллельного тестирования запросов на изменение, были определены следующие параметры:

- количество запросов на изменение, которые необходимо протестировать;
- количество пользователей, которые занимались тестирование запросов на изменение;
- виды запросов, которые были протестированы (разделение по кол-ву объектов);
- количество тестовых серверов.

Исходя из вышесказанных параметров, в оценке эффективности участвовало: 80 запросов на изменение, из которых 20 было маленькими (от 10 000 до 15 000 объектов), 40 средними (от 15 000 до 30 000 объектов) и 20 большими (свыше 30 000 объектов).

Тестирование данных запросов занималось три пользователя, количество доступных тестовых серверов для нового алгоритма было равно трём.

Проводя оценку эффективности внедрения данного алгоритма в работу PLM-компоненты было решено разбить её на 3 этапы:

1. Измерение времени, которое требуется для перевода трёх запросов на изменение одного вида в состояние «В тестировании».

2. Измерение времени, которое требуется для перевода всех доступных запросов на изменение в состояние «В тестировании» без учета времени на само тестирование данных запросов.

3. Измерение времени, которое требуется для полноценного тестирования всех доступных запросов на изменение.

Сбор данных, необходимых для проведения оценки эффективности нового жизненного цикла запроса на изменение, проводился с середины января по конец мая текущего года, то есть данный период охватывает 133 дня.

Собранные данные, представлены в таблицах 3.1.1 и 3.1.2.

Таблица 3.1.1 – Показатели времени работы PLM-компоненты без внедрения нового алгоритма

Вид запроса	Кол-во запросов	Время перевода 3х запросов в тестирование	Время перевода всех запросов в тестирование (без их тестирования)	Время перевода всех запросов в тестирование (с учетом тестирования)
Маленький	20	~ 145 мин	~ 1025 мин	~ 4625 мин
Средний	40	~ 253 мин	~ 3500 мин	~ 15500 мин
Большой	20	~ 423 мин	~ 2808 мин	~ 13608 мин

Таблица 3.1.2 – Показатели времени работы PLM-компоненты с внедрением нового алгоритма

Вид запроса	Кол-во запросов	Время перевода 3х запросов в тестирование	Время перевода всех запросов в тестирование (без их тестирования)	Время перевода всех запросов в тестирование (с учетом тестирования)
Маленький	20	~ 30 мин	~ 310 мин	~ 1990 мин
Средний	40	~ 55 мин	~ 749 мин	~ 6629 мин
Большой	20	~ 100 мин	~ 940 мин	~ 7660 мин

На время, которое необходимо, чтобы запрос на изменение оказался на тестовом окружении, влияли следующие факторы:

1. Скорость формирования архива с разработанными модификациями.

2. Скорость выполнения Check Task, время которое прямо зависит от количества объектов под запросом на изменение.

3. Скорость синхронизации каталога с биллинговой системой.

4. Скорость импорта запроса на тестовое окружение.

5. Скорость отката запроса на изменение с тестового окружения.

Стоит отметить, что Check Task влияет только на самый первый запрос на изменение, так как для других запросов не имеет смысла учитывать это время, так как их Check Task будет выполнена во время импорта первого запроса на изменение.

Основываясь на данные, которые представлены в таблицах 3.1.1 и 3.1.2 были построены объемные гистограммы, представленные на рисунках 3.1.1 – 3.1.3, которые наглядно отображают эффективность в работе PLM-компоненты с использованием нового алгоритма.

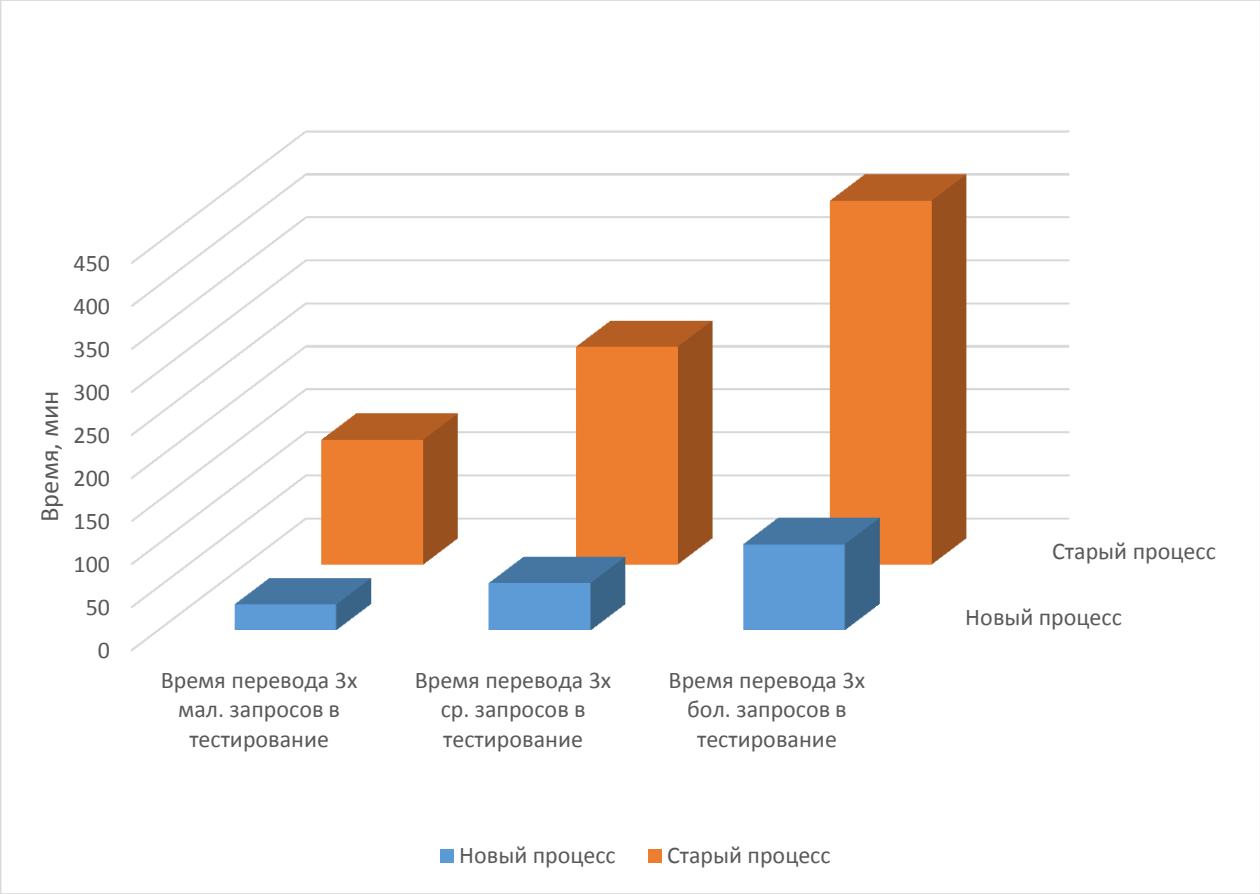

Рисунок 3.1.1 – Время необходимое для перевода трёх запросов на изменение в состояние «В тестировании» с использованием старого и нового процесса

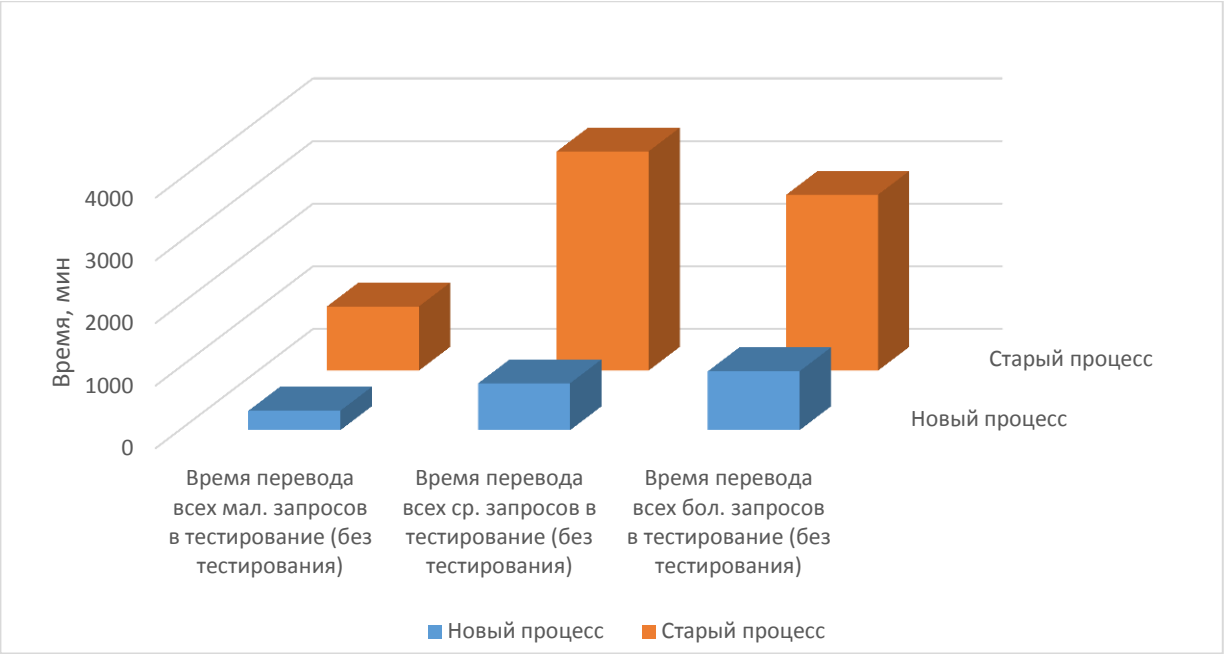

Рисунок 3.1.2 – Время необходимое для перевода всех запросов на изменение в состояние «В тестировании» с использованием старого и нового процесса (без учета тестирования)

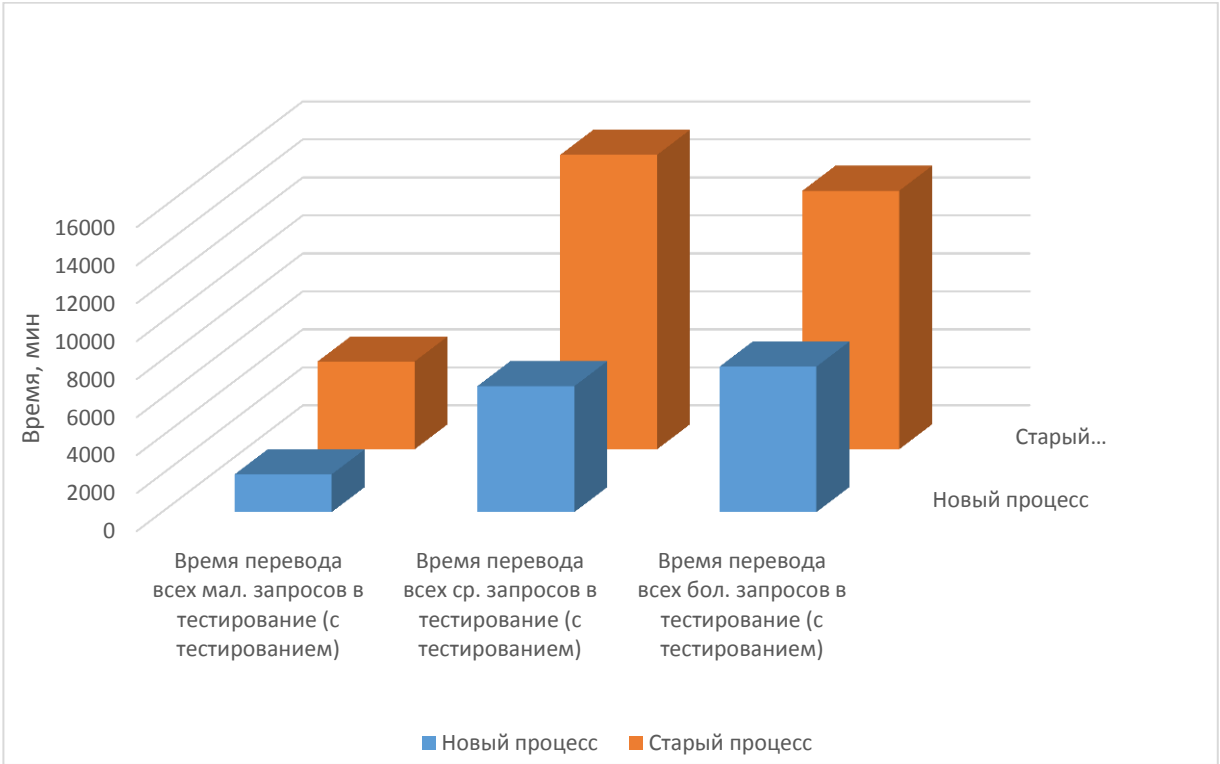

Рисунок 3.1.3 – Время необходимое для перевода всех запросов на изменение в состояние «В тестировании» с использованием старого и нового процесса (с учетом тестирования)

Исходя из выше приведенных показателей, можно оценить эффективность нового подхода для PLM-компоненты.

В случае перевода в тестирование только трёх запросов на изменение: показатель эффективности варьируется в диапазоне от 4 до 5 раз, но уже при переводе всех запросов на изменение в тестирование без учета тестирования, этот же показатель составляет от 3 до 3,5 эффективности, а если рассматривать тот же самый показатель, но с учетом времени на тестирование, то он уже будет располагаться в диапазоне между от 2 до 2,5.

Данный показатель при тестировании запроса увеличивается в следствие того, что пользователи теперь выполняют откат в произвольном порядке, так как каждый из них тратить различное время на тестирование запроса.

Стоит отметить, что значение эффективности возрастает прямо пропорционально размеру запроса на изменение, что является хорошим знаком.

3.2 Оценка эффективности нового инструмента, позволяющего выполнить группировку нескольких модификаций из разных запросов на изменение под одним главным

Перед началом оценки эффективности инструмента, позволяющего выполнить группировку нескольких модификаций из разных запросов на изменение под одним главным, будут использованы те же параметры, что и ранее за исключение количества тестовых окружений, в данном случае этот параметр будет заменен количеством запросов, которые будут участвовать в группировке.

В данной оценке эффективности было использовано: 77 запросов на изменение, из которых 36 было маленькими (от 8 000 до 12 000 объектов), 25 средними (от 12 000 до 20 000 объектов) и 16 большими (свыше 20 000 объектов).

Тестирование данных запросов занималось три пользователя, количество группировок для каждого тип запроса определялось отдельно.

Количество группировок и количество запросов, которые в них участвуют приведены в таблице 3.2.1.

Таблица 3.2.1 – Количество группировок для каждого вида запросов на изменение

Количество запросов в группировке	Общее количество группировок для маленьких запросов	Общее количество группировок для средних запросов	Общее количество группировок для больших запросов
По 3 запроса	12	8	5
По 4 запроса	9	6	4
По 5 запросов	7	5	3

Как видно из таблицы 3.2.1 было проведено 9 замеров при использовании нового инструментария, позволяющего группировать запросы, а также 9 замеров без использования данного инструментария.

Результаты для данных замеров представлены в таблицах 3.2.2-3.2.4.

Таблица 3.2.2 – Показатели времени для перевода маленьких запросов на изменение в состояние «В тестировании»

Количество запросов (разбито на группы)	Время перевода запросов в тестирование без группировки	Время перевода сгруппированных запросов в тестирование
36 (по 3)	~ 1740 мин	~ 612 мин
36 (по 4)	~ 1740 мин	~ 535 мин
35 (по 5)	~ 1595 мин	~ 414 мин

Таблица 3.2.3 – Показатели времени для перевода средних запросов на изменение в состояние «В тестировании»

Количество запросов (разбито на группы)	Время перевода запросов в тестирование без группировки	Время перевода сгруппированных запросов в тестирование
24 (по 3)	~ 2024 мин	~ 864 мин
24 (по 4)	~ 2024 мин	~ 1012 мин
25 (по 5)	~ 2277 мин	~ 1148 мин

Таблица 3.2.4 – Показатели времени для перевода больших запросов на изменение в состояние «В тестировании»

Количество запросов (разбито на группы)	Время перевода запросов в тестирование без группировки	Время перевода сгруппированных запросов в тестирование
15 (по 3)	~ 2155 мин	~ 1978 мин
16 (по 4)	~ 2578 мин	~ 2491 мин
15 (по 5)	~ 2155 мин	~ 2283 мин

Основываясь на данные, которые представлены в таблицах 3.2.2 – 3.2.4 были созданы объемные гистограммы, которые представлены на рисунках 3.2.1 – 3.2.3, на которых наглядно видно на сколько эффективнее становится работа PLM-компоненты с использованием нового инструментария.

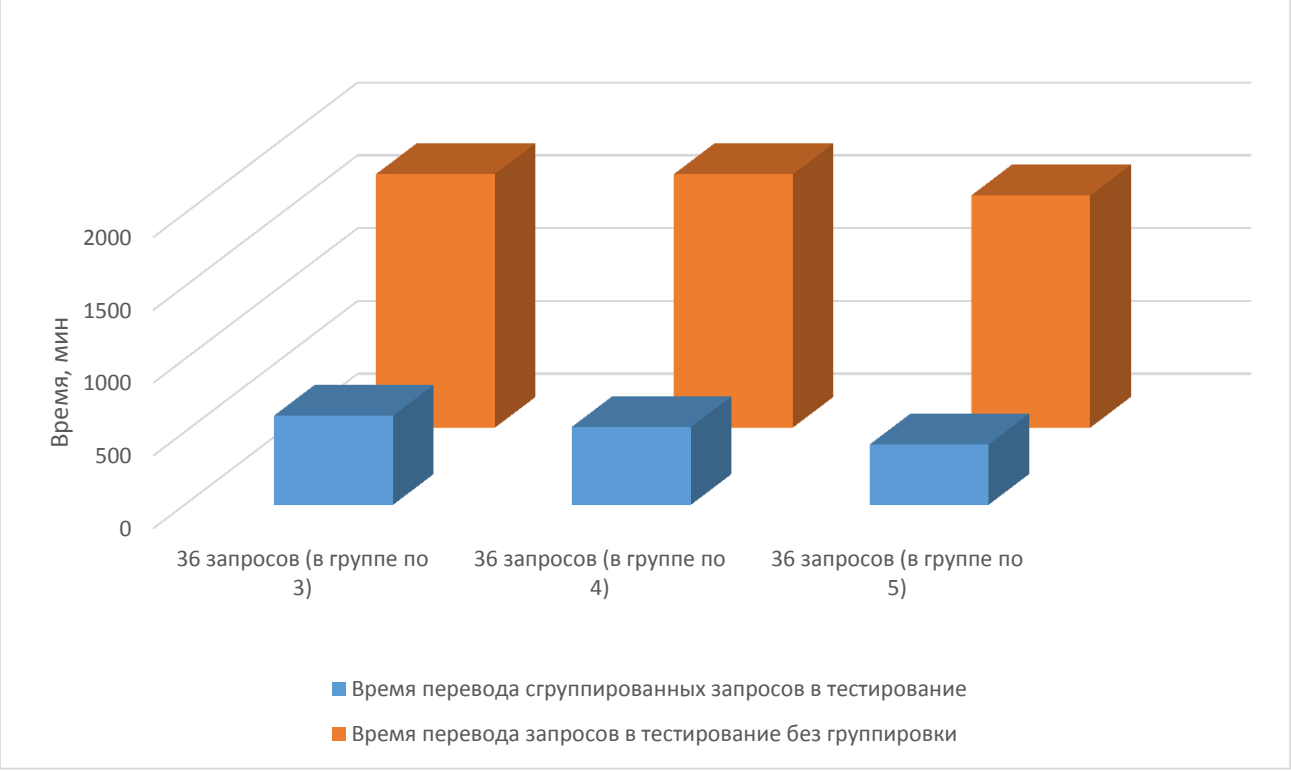

Рисунок 3.2.1 – Время необходимое для перевода всех маленьких запросов на изменение в состояние «В тестировании» с использованием старого и нового процесса

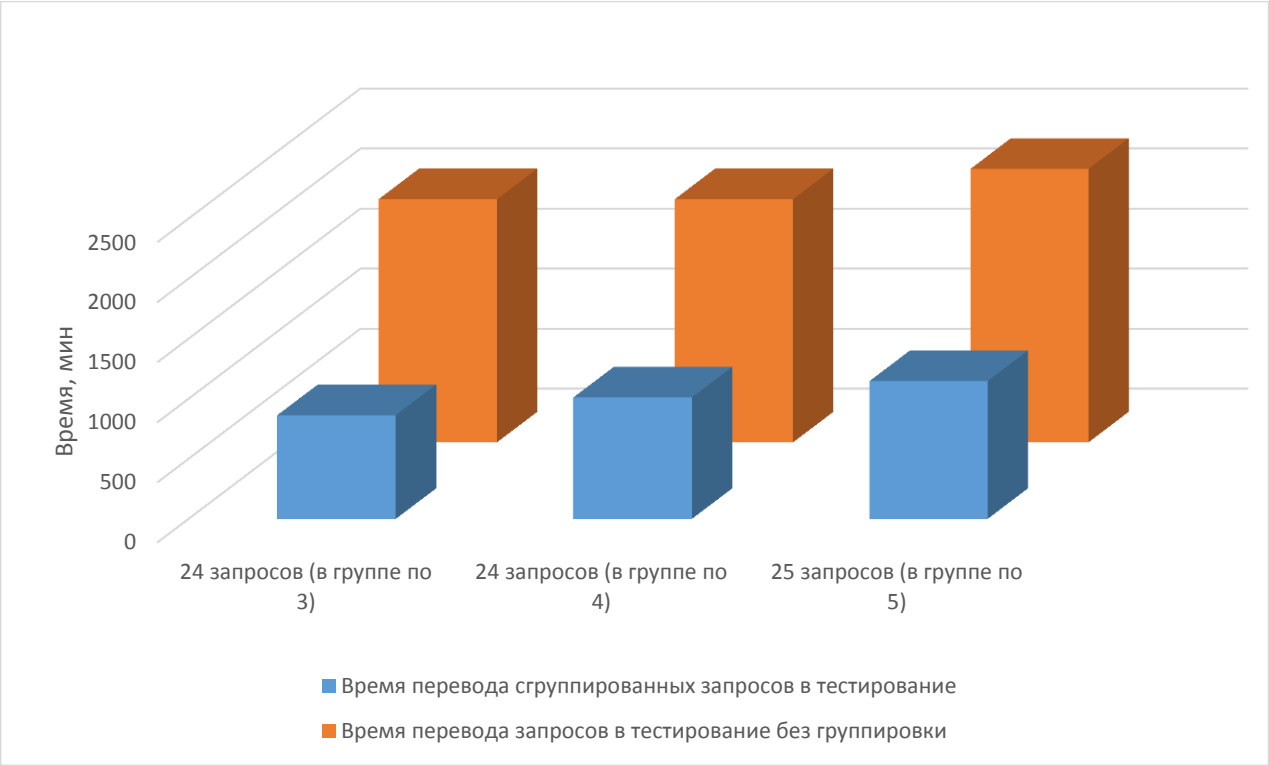

Рисунок 3.2.2 – Время необходимое для перевода всех средних запросов на изменение в состояние «В тестировании» с использованием старого и нового процесса

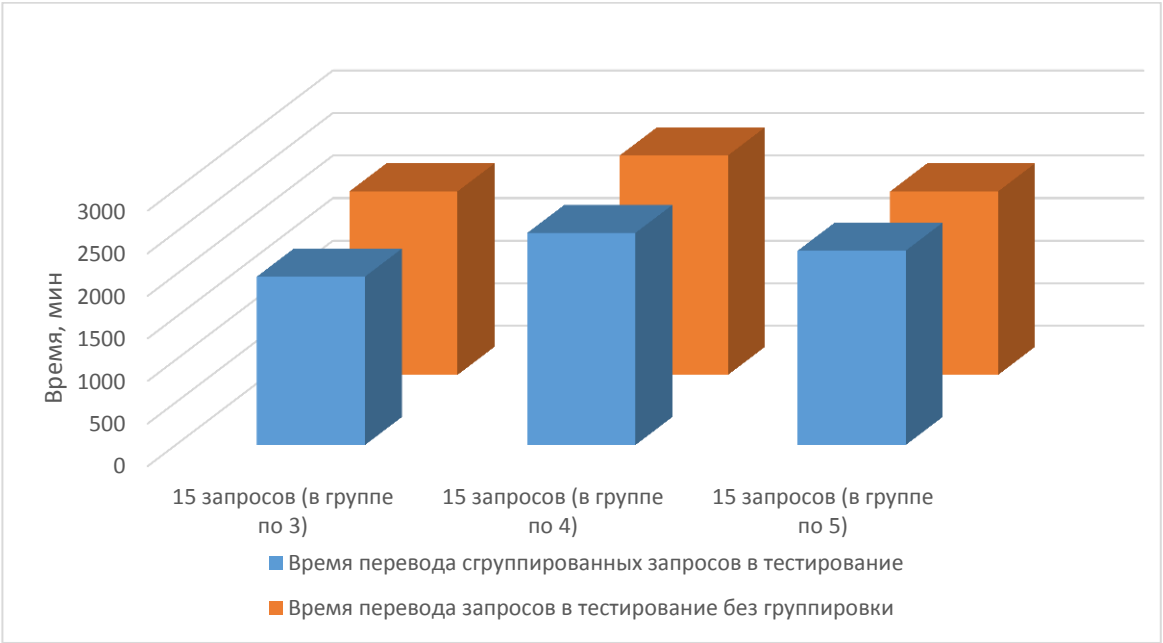

Рисунок 3.2.3 – Время необходимое для перевода всех больших запросов на изменение в состояние «В тестировании» с использованием старого и нового процесса

Как видно из рисунков 3.2.1 – 3.2.3, изначальное предположение о том, что данный метод эффективен только для работы с небольшими запросами, полностью подтвердилось.

Так при работе с маленькими запросами данный инструментарий показывает отличные результаты, которые даже превышают аналогичные показатели метода, поддерживающего несколько тестовых окружений. Так, с использованием данного подхода скорость перевода несколько модификаций на тестовую среду увеличилась в диапазоне от 2,8 до 4 раз. Причем стоит отметить, что скорость увеличивается прямо пропорционально количеству сгруппированных запросов, но это не значит, что чем будет их больше, тем быстрее, так как в определенный момент данная связь будет полностью обратно пропорциональной, из-за чрезмерной группировки под главным запросом может оказать слишком много объектов, и тогда всё преимущество данного подхода будет не актуально.

Стоит отметить, что хорошая тенденция в увеличении скорость перехода к состоянию «В тестировании» нескольких сгруппированных запросов сохраняется и для средних по количеству объектов запросов. Но тут ситуация немного другая, так как видно, что чем больше запросов группируется под одним, тем снижается и скорость их перехода в состояние «В тестировании».

В данном случае при применении группировки скорость перехода средних запросов в состояние «В тестировании» увеличивается в диапазоне от 2 до 2,5 раз.

И совсем отрицательные показатели были получены для группировки больших запросов на изменение и их перевода в состояние «В тестировании». Причем преимущества как такового не наблюдается.

Лишь при группировке трёх запросов, скорость будет увеличена на 0,1. А при группировке более трёх запросов, данный показатель и во все будет отрицательным.

3.3 Оценка эффективности алгоритма, позволяющего группировать идентичные значения цен «Одноразового сервиса» во время синхронизации с каталога с биллинговой системой

Перед началом оценки эффективности алгоритма, позволяющего группировать идентичные значения цен «Одноразового сервиса» во время синхронизации с каталога с биллинговой системой, были определены следующие параметры, которые должны быть определены перед началом сбора данных:

- количество одноразовых сервисов в одном запросе на изменение;
- количество ценовых значений в одном одноразовом сервисе.
- количество программ лояльности, привязанных к одному ценовому значению.
- Количество запрос на изменение.

Исходя из вышесказанных параметров, в оценке эффективности участвовало: 4 вида запросов на изменение, каждый из которых будет содержать по 15, 30, 45 и 60 одноразовых сервисов соответственно. Каждый одноразовый сервис содержит в себе приблизительно от 250 до 300 ценовых значений, к каждому из которых привязано по 4 программы лояльности.

Данные о времени синхронизации каталога с биллинговой системой представлены в таблице 3.3.1.

Таблица 3.3.1 – Показатели времени синхронизации каталога с биллинговой системой

Вид синхронизации	Время синхронизации 15 одноразовых сервисов	Время синхронизации 30 одноразовых сервисов	Время синхронизации 45 одноразовых сервисов	Время синхронизации 60 одноразовых сервисов
Старый процесс	~ 90 минут	~ 185 минут	~ 273 минуты	~ 360 минут
Новый процесс	~ 30 минут	~ 58 мнут	~ 93 минуты	~ 120 минут

Используя полученные данные, была построена объемная гистограмма, изображенная на рисунке 3.3.1. Данный рисунок наглядно демонстрирует

эффективность нового подхода к синхронизации каталога с биллинговой системой.

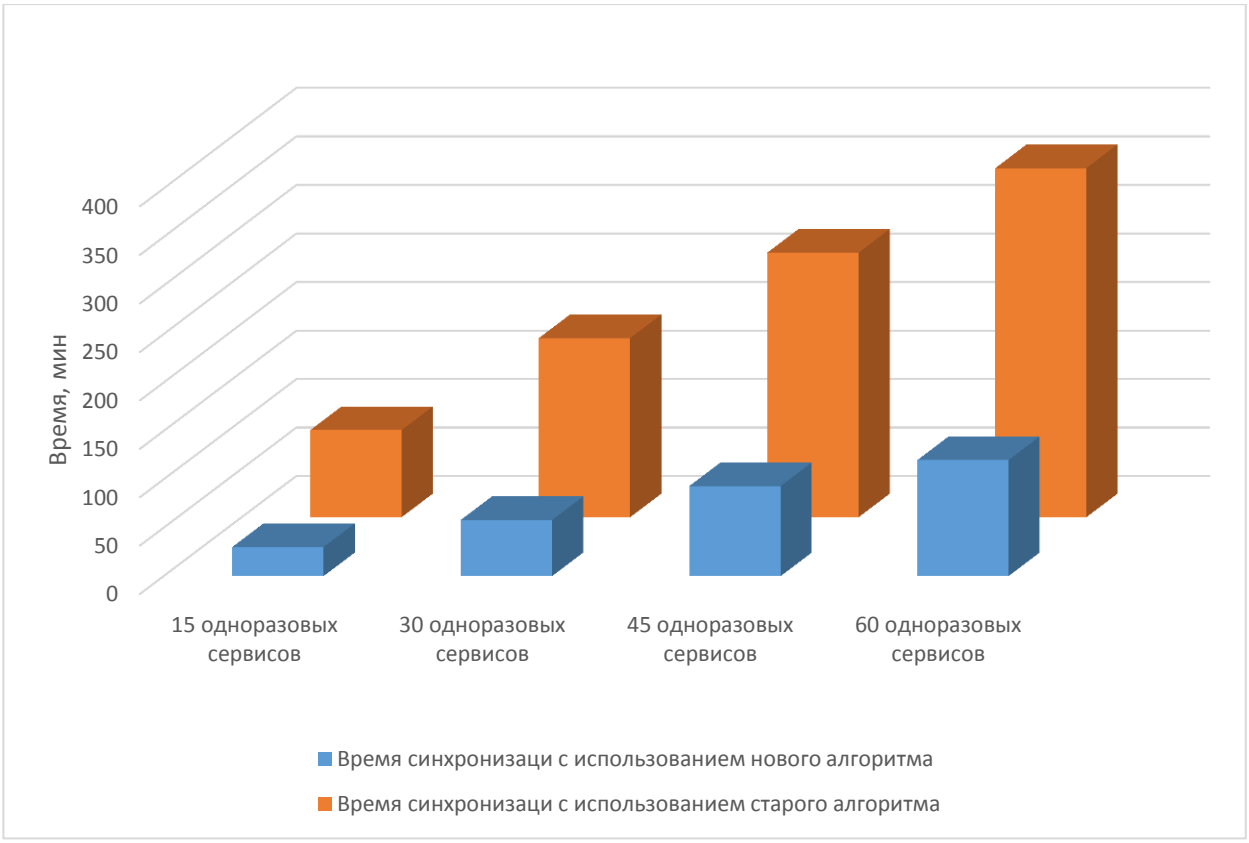

Рисунок 3.3.1 – Время необходимое для синхронизации каталога с биллинговой системой с использованием старого и нового процесса

Как видно из полученных данных, скорость синхронизации каталога с биллинговой системой для одноразовых сервисов уменьшилась в примерно в три раза.

Данные показатели являются отличным результатом, так как при данном подходе не придется обучать пользователей новым функциональным возможностям.

Несмотря на то, что данный метод не позволяет тестировать запросы на изменение параллельно, нельзя исключать его невероятную эффективность относительно синхронизации одноразовых сервисов с биллинговой системой.

В данном случае этот подход является точечным решением для весьма серьезной проблемы, которая эффективно решается за счёт него. А если ещё учесть, что данный метод можно будет внедрить и для других объектов,

находящихся в каталоге, то можно в будущем рассчитывать и на то, что синхронизация перестанет быть на столько долгим и проблемным местом в PLM-компоненте.

Также данный подход может быть легко совмещен с любым другим методом, которые были рассмотрены в этой работе.

Таким образом, если совместить данный подход с метод поддержки нескольких тестовых окружений для параллельного тестирования запросов не изменение, то можно получить весьма впечатляющие результаты. А при комбинации с методом группировки нескольких запросов на изменение под одним главным, данный метод и во все сможет сгладить основной недостаток данного подхода в лице нецелесообразности группировать слишком крупные запросы на изменение.

Результаты сочетаний представленных подходов будут описаны ниже, где также будет представлено сравнение данных подходов между собой.

Вывод по главе 3

В данной главе описаны параметры, которые участвовали в оценке эффективности для каждого из предложенных подходов, призванных повысить эффективность работы PLM-компоненты.

Были представлены и обработаны данные, которые были собраны во время проведения оценки эффективности новых подходов на протяжении 133 дней.

Были построены объёмные гистограммы, которые наглядно отображают эффективность каждого подхода по отдельности и их общую эффективность для работы PLM-компоненты.

Были проанализированы результаты оценки эффективности новых подходов в работе PLM-компоненте, что позволило подтвердить их эффективность.

ЗАКЛЮЧЕНИЕ

За время исследования и выполнения диссертационной работы был проведен анализ литературы о возможности применения методов модернизации процессов, протекающих в PLM-компоненте, и проектировании соответствующих алгоритмов и инструментариев, повышающих эффективность в проведении тестирования запросов на изменение. Данный анализ литературы помог выявить наиболее подходящие методы, которые повышают эффективность работы PLM-компоненты. Данные методы были успешно применены при проектировании новых алгоритмов работы PLM-компоненты, благодаря чему PLM-компонента была модернизирована.

При выполнение данной диссертационной работы были выполнены следующие поставленные задачи, благодаря которым и был достигнут конечный результат:

1) проведен анализ и оценка эффективности текущих процессов работы PLM-компоненты в составе BSS-системы;

2) рассмотрены существующие методы модернизации PLM-процессов, позволяющих повысить эффективность тестирования запросов на изменение и ускорить синхронизацию каталога с биллинговой системой;

3) спроектирован алгоритм, в основе которого лежит метод, предоставляющий возможность поддержки нескольких тестовых окружений для параллельного тестирования запросов на изменение в PLM – компоненте;

4) спроектирован инструмент, позволяющий выполнить группировку нескольких модификаций из разных запросов на изменение под одним главным в PLM – компоненте;

5) спроектирован алгоритм, обеспечивающий возможность группировки идентичных значений цен «Одноразового сервиса» при синхронизации с биллинговой системой;

6) проведена оценка эффективности внедренных алгоритмов и инструмента в работе PLM-компоненты.

Модернизированная PLM-компонента с помощью спроектированных методов и инструментов теперь имеет более эффективный процесс перевода запросов на изменение в состояние «В тестировании», который теперь также позволяет выполнять параллельное тестирование данных запросов на изменение; улучшенный процесс синхронизации каталога с биллинговой системой для одноразовых сервисов значительно сократил время ожидания окончания данной процедуры для пользователя, благодаря чему пользователь имеет возможность раньше приступить к тестированию, что даёт ему больше времени для более качественной проверки запроса на изменение.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения. – Взамен ГОСТ 24.003-84, ГОСТ 22487-77; введ. 1992-01-01. – М.: Издательство стандартов, 1992. – 14 с.

2. ГОСТ 34.601-90. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. – Введ. 1992-01-01. М.: Издательство стандартов, 1992. – 6 с. – (Основополагающие стандарты).

3. ГОСТ 34.602-89. Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 1990-01-01. – М.: Издательство стандартов, 1990. – 12 с. – (Основополагающие стандарты). 4. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения (ИСО 5807-85). Введ. 1992-01-01. – М.: Издательство стандартов, 1992 – 14 с. – (Единая система программной документации).

4. ГОСТ 2.105-95. Общие требования к текстовым документам. – М.: Издательство стандартов, 1996. – 29 с. – (Единая система конструкторской документации).

5. ГОСТ 7.82-2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – 92 Введ. 2002-07-01. – Минск: Издательство стандартов, 2001. – 35 с. – (Система стандартов по информации, библиотечному и издательскому делу).

Учебники и учебные пособия

6. Железнякова М.С. Концепция PLM – управление жизненным циклом продукта // Приволжский научный вестник. – 2015 – №11 – С. 64-67

7. Курочкин Л.М. Автоматизированная система технологической поддержки предприятий малого и среднего бизнеса: дис. Курочкина Л.М. кандидата технических наук: защищена 29.06.2011: утв. 11.09.2011 / С.К.

Лавровский. Спб: Изд-во Санкт-Петербургского государственного политехнический университета, 2011 – 149 с.

8. Левин Д.А., Малюх В.Г., Ушаков Д.И. Энциклопедия PLM. / Д.А. Левин – Эком, 2008. – 445 с.

9. Лихачев М.В. Управление структурой изделия в PLM-системах // Программные средства и информационные технологии. – 2014 – №42 – 262-264 с.

10. Рекс Блэк. Ключевые процессы тестирования. Планирование, подготовка, проведение, совершенствование / Рек Блэк. – Отдельное издание, 2011 – 544 с.

Электронные ресурсы

11. Два подхода к тестированию производительности [Электронный ресурс]: https://habrahabr.ru/post/106493/

12. Нагрузочное тестирование [Электронный ресурс]: https://ru.wikipedia.org/wiki/Нагрузочное_тестирование

13. Нагрузочное тестирование [Электронный ресурс]: http://www.quadrium.ru/solutions/m-testing/m-t-m

14. Нагрузочное тестирование [Электронный ресурс]: http://devopswiki.net/index.php/Нагрузочное_тестирование

15. Нагрузочное тестирование [Электронный ресурс]: http://www.protesting.ru/automation/performance.html

16. Начинающему тестировщику — Нагрузочное тестирование [Электронный ресурс]: http://test-engineer.ru/2016/07/tester-beginner-performance/

17. Роль OSS/BSS в деятельности оператора связи: практический подход [Электронный ресурс]: https://topref.ru/referat/52184.html

18. Связь: OSS/BSS - следующая остановка после биллинга и CRM [Электронный ресурс]: http://www.cnews.ru/articles/svyaz_ossbss__sleduyushchaya_ostanovka

19. Управление жизненным циклом изделия [Электронный ресурс]: http://plmpedia.ru/wiki/Управление_жизненным_циклом_изделия

20. PLM-система [Электронный ресурс]: http://sewiki.ru/PLM-система

21. Product Lifecycle Management. Управление жизненным циклом изделия [Электронный ресурс]: http://www.tadviser.ru/index.php/PLM_-_управление_жизненным_циклом_изделия

22. Product Information Management [Электронный ресурс]: https://go.oracle.com/LP=49409/?elqCampaignId=55582&nm=1

23. OSS/BSS – базовое программное обеспечение в телекоме [Электронный ресурс]: https://www.kommersant.ru/doc/2594410

24. Operations / Business Support Systems. Системы управления и эксплутационной поддержки для операторов связи [Электронный ресурс]: http://www.tadviser.ru/index.php/OSS/BSS

25. Three Tenets of Modern Product Lifecycle Management Cloud [Электронный ресурс]: https://cloud.oracle.com/opc/saas/ebooks/oracle-product-lifecycle-management-cloud-ebook.pdf

Литература на иностранном языке

26. B. A. Pozin, I. V. Galakhov - Models in performance testing, - Programming and Computer Software, vol. 37, no.1, pp. 15-25, 2011.

27. Boris Toche, Grant McSorley, Robert Pellerin, Clement Fortin. A framework to support collaboration during prototyping and testing. International Journal of Product Lifecycle Management, Vol.10, No.4, pp.348 – 374, 2017.

28. C. Prathiba, S. Jayanthi, Dr. P. S. K. Patra N. Ezhil Arasu. Performance Testing with Realistic Load Testing. International Journal of Innovative Research in Science, Engineering and Technology, Volume 3, 2014.

29. Charmy Patel, Ravi Gulati. Software Performance Testing Measures. International journal of management & information technology, Vol. 8, 2014.

30. Eduardo De Senzi Zancul, Luiz Fernando C.S. Durão, Alexandre M. Rocha, Gabriel Delage e Silva. PLM process and information mapping for mass customisation based on additive manufacturing. International Journal of Product Lifecycle Management, Vol.9, No.2, pp.159 – 178, 2016.

31. Eljona Proko, Ilia Ninka. Analyzing and Testing Web Application Performance. International Journal of Engineering and Science, Vol. 3, 2013.

32. Gerald M. Weinberg Perfect Software: And Other Illusions about Testing / Gerald M. Weinberg - Dorset House, 2008 – 200 c.

33. H. Sarojadevi. Performance Testing: Methodologies and Tools. Journal of Information Engineering and Applications, Vol 1, 2013.

34. Ian Molyneaux the Art of Application Performance Testing: From Strategy to Tools 2nd Edition, Kindle Edition / Ian Molyneaux - O'Reilly Media, 2014 – 278 c.

35. James Whittaker Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design 1st Edition / James Whittaker - Addison-Wesley Professional, 2009 – 256 c.

36. Mark Hayward, Shailendra Jain, Sharad Kumar Broadband infrastructure - the ultimate guide to building and delivering OSS / BSS from BusinessEdge Solutions 2nd Edition / Mark Hayward – Kluwer, 2013 – 304 c.

37. Norio Okino, Hiroyuki Tamura, Susumu Fujii. Advances in Production Management Systems. / Norio Okino – Springer US, 2014 – 482.

38. Pratibha Fageria, Dr. Manju Kaushik. Research of Load Testing and Result Based on Loadrunner. SSRG International Journal of Civil Engineering (SSRG-IJCE), Vol. 1, 2014.

39. Rijwan Khan, Mohd Amjad. Performance testing (load) of web applications based on test case management. Perspectives in Science, Vol. 8, 2016.

40. Shikha Dhiman, Pratibha Sharma. Performance Testing: A Comparative Study and Analysis of Web Service Testing Tools. International Journal of Computer Science and Mobile Computing, Vol.5, 2016.

41. Timber Haaker, Harry Bouwman, Wil Janssen, Mark de Reuver. Business model stress testing: A practical approach to test the robustness of a business model. Futures, Vol. 89, 2017.

42. Yogita M. Rasal, Sangeeta Nagpure. Web Application: Performance Testing Using Reactive Based Framework. International Journal of Research in Computer and Communication Technology, Vol 4, 2015.