ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

ОПТИМИЗАЦИЯ СИСТЕМЫ ДОКУМЕНТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ В КОМПАНИИ ООО «БФТ»

Магистерская диссертация
обучающегося по направлению подготовки
09.04.02 – Информационные системы и технологии
очной формы обучения
группы 12001735
Жукова Романа Владимировича

Научный руководитель
к.т.н., доцент
Сурушкин М.А.

Рецензент
к.т.н., доцент
Старченко Д.Н.

БЕЛГОРОД 2019

РЕФЕРАТ

Оптимизация системы документирования информационных систем в компании ООО «БФТ». – Жуков Роман Владимирович, магистерская диссертация, Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 89, включая приложения 115, количество рисунков 21, количество таблиц 16, количество использованных источников 34.

КЛЮЧЕВЫЕ СЛОВА: информационная система, система документирования, процесс документирования, документация программных продуктов, принцип единого источника.

ОБЪЕКТ ИССЛЕДОВАНИЯ: процесс документирования информационных систем в компании БФТ.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: методы и средства процесса документирования информационных систем.

ЦЕЛЬ РАБОТЫ: оптимизация системы документирования информационных систем в компании БФТ.

ЗАДАЧИ ИССЛЕДОВАНИЯ: изучить предметную область; исследовать существующие методы подготовки документации; проанализировать и описать модель существующего процесса документирования; спроектировать оптимизированную модель процесса документирования; разработать средства реализации оптимизированной модели; провести оценку эффективности разработанной модели.

МЕТОДЫ ИССЛЕДОВАНИЯ: методы системного анализа.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: В результате работы была оптимизирована система документирования информационных систем компании БФТ.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

По мере роста числа создаваемых и вводимых в действие информационных систем все более остро встает вопрос сокращения сроков и снижения трудоемкости разработки и поддержания в актуальном состоянии, в том числе и документации.

Разработка документации на информационную систему предполагает подготовку целого ряда документов: концептуальных, отчетных, проектных, рабочих, эксплуатационных и организационно-распорядительных. Итоговый объем документации системы, включающей в себя текстовую и графическую информацию, как правило, составляет десятки, а иногда и сотни тысяч страниц.

Разработка и поддержание документации в актуальном состоянии на всех стадиях и этапах жизненного цикла информационной системы вызывает немало проблем. В большей степени они связаны, с необходимостью многократного внесения в документацию, состоящую из множества отдельных файлов, всевозможных поправок, дополнений и изменений.

Ситуация отягощается еще и тем, что:

– «государственный» заказчик, финансирующий львиную долю разрабатываемых информационных систем, все чаще и чаще требует от исполнителя строжайшего и неформального соблюдения требований ГОСТ при разработке технической документации;

– требования отечественных ГОСТов несоизмеримо более жесткие по сравнению с требованиями зарубежных стандартов;

– специалистов, способных удовлетворить требованиям заказчика (соблюдение требований ГОСТ), в ВУЗах практически не готовят, так как их подготовка все больше сводится к так называемому «процессному подходу».

Корректировка документов – трудоемкая операция, требующая от технического писателя аккуратности, высокой сосредоточенности, отнимающая массу времени, сил и нервов.

Использование современных технологий позволяющих осуществить переход от «классического» метода документирования к методу документирования по «принципу единого источника» – разумный путь оптимизации процесса документирования, позволяющий получить стабильное качество документации, сократить сроки разработки документации и снизить трудоемкость поддержание комплекта документов в актуальном состоянии.

Предпосылки оптимизации процесса подготовки документации, прослеживаются в формулировках ГОСТов 34-й серии:

– Приложение 1 ГОСТ 34.201–89: «Документация на автоматизированную систему – комплекс взаимоувязанных документов, в котором полностью описаны все решения по созданию и функционированию системы, а также документов, подтверждающих соответствие системы требованиям технического задания и готовность ее к эксплуатации (функционированию)» [1].

– п. 5.1 ГОСТ 34.003–90: «Комплект взаимоувязанных документов, полностью определяющих технические требования к АС, проектные и организационные решения по созданию и функционированию АС» [2].

Объектом исследования является процесс документирования информационных систем в компании БФТ.

Предметом исследования являются методы и средства процесса документирования информационных систем.

Целью данной работы является оптимизация системы документирования информационных систем в компании БФТ.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучить предметную область;

– исследовать существующие методы подготовки документации;

– проанализировать и описать модель существующего процесса документирования;

– спроектировать оптимизированную модель процесса документирования;

– разработать средства реализации оптимизированной модели;

– провести оценку эффективности разработанной модели.

Диссертация состоит из введения, трех разделов, заключения, списка использованных источников и приложений. Работа изложена на 115 страницах машинописного текста, включая 21 рисунок, 16 таблиц, список использованных источников из 34 наименований и двух приложений.

1 Описание предметной области

1.1 Концептуальный подход документирования программных продуктов

В чем же заключается «взаимоувязанность» документов комплекса (комплекта)? Практически каждый документ включает в себя фрагменты, идентичные фрагментам, содержащимся в других документах комплекта. Документация на систему оказывается «насквозь пронизанной» жесткими связями на уровне структурных единиц – разделов, подразделов, пунктов и подпунктов отдельно взятых документов [14].

На рисунке 1 показана связь между подразделами всего лишь трех документов системы: технического задания (ТЗ), пояснительной записки (П2) и общего описания системы (ПД) [1]. Изменения, вносимые в ТЗ, неизбежно повлекут изменения в П2 и ПД. Иначе и быть не может, поскольку документация на систему должна быть согласованной – сведения, содержащиеся в ПД, не должны противоречить решениям, обоснованным в П2, и требованиям, предъявляемым ТЗ. Решения, обоснованные в П2, в свою очередь, не должны противоречить требованиям ТЗ. Согласованность, выражающаяся в непротиворечивости, является наиважнейшим показателем качества технической документации [13].

Каждый документ, создаваемый в отдельном файле, при даже незначительном изменении в требованиях ТЗ потребует ручной корректировки двух других документов. Возникает вопрос, во что выльются трудозатраты, если изменения в ТЗ придется вносить многократно? А если указанный текст содержится не в трех, а в шести (или более) документах, как, например, подраздел «Перечень объектов автоматизации» ТЗ [3], имеющий также место в пяти других документах П2, П4, ПД, И3 и Б1.

В практике технических писателей распространено понятие «принцип единого источника» (single source publishing). Это концепция публикации документов, согласно которой один и тот же фрагмент может быть использован

в разных выпускаемых документах или в разных форматах публикации. Метод документирования по принципу единого источника представлен на рисунке 2.

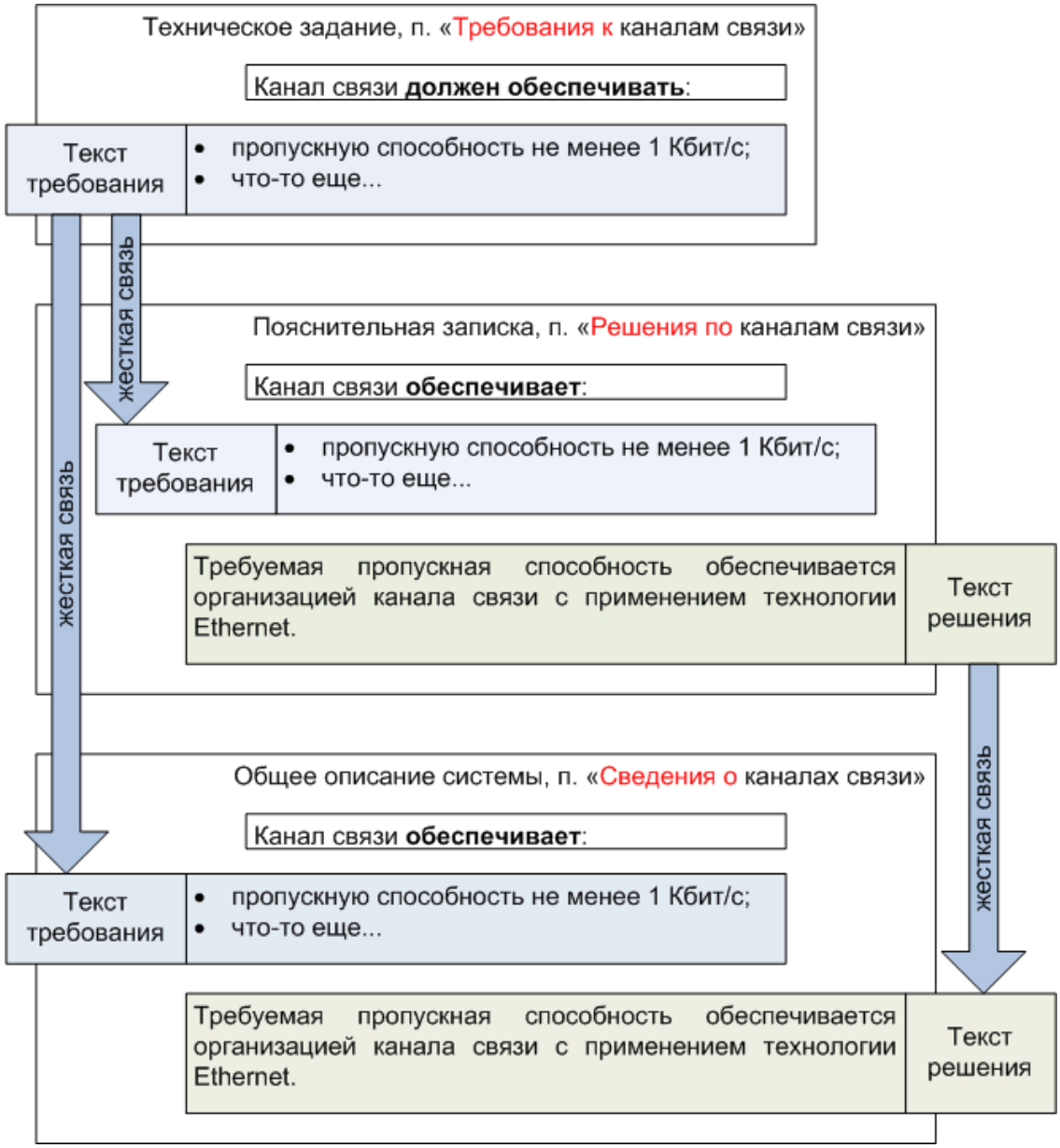

Рисунок 1 – Связь между документами по классическому методу документирования

Все фрагменты, из которых состоят выходные документы, находятся в некотором общем фонде, едином источнике, представляющем собой (в зависимости от конкретной реализации) набор отдельных файлов в файловой системе или некую базу данных. Каждый выходной документ представляет собой упорядоченную выборку из единого источника, который, в принципе, может иметь некоторую структуру, удобную для составителей, однако,

напрямую в структуру какого-либо документа она не трансформируется. Иначе говоря, соотношение между единым источником и документом примерно такое же, как между базой данных и отчетом. Создать документ в такой системе – значит описать его структуру и правила формирования из фрагментов единого источника. Фрагментами могут являться шаблоны страниц, графические элементы, абзацы текста, предложения или отдельные числа.

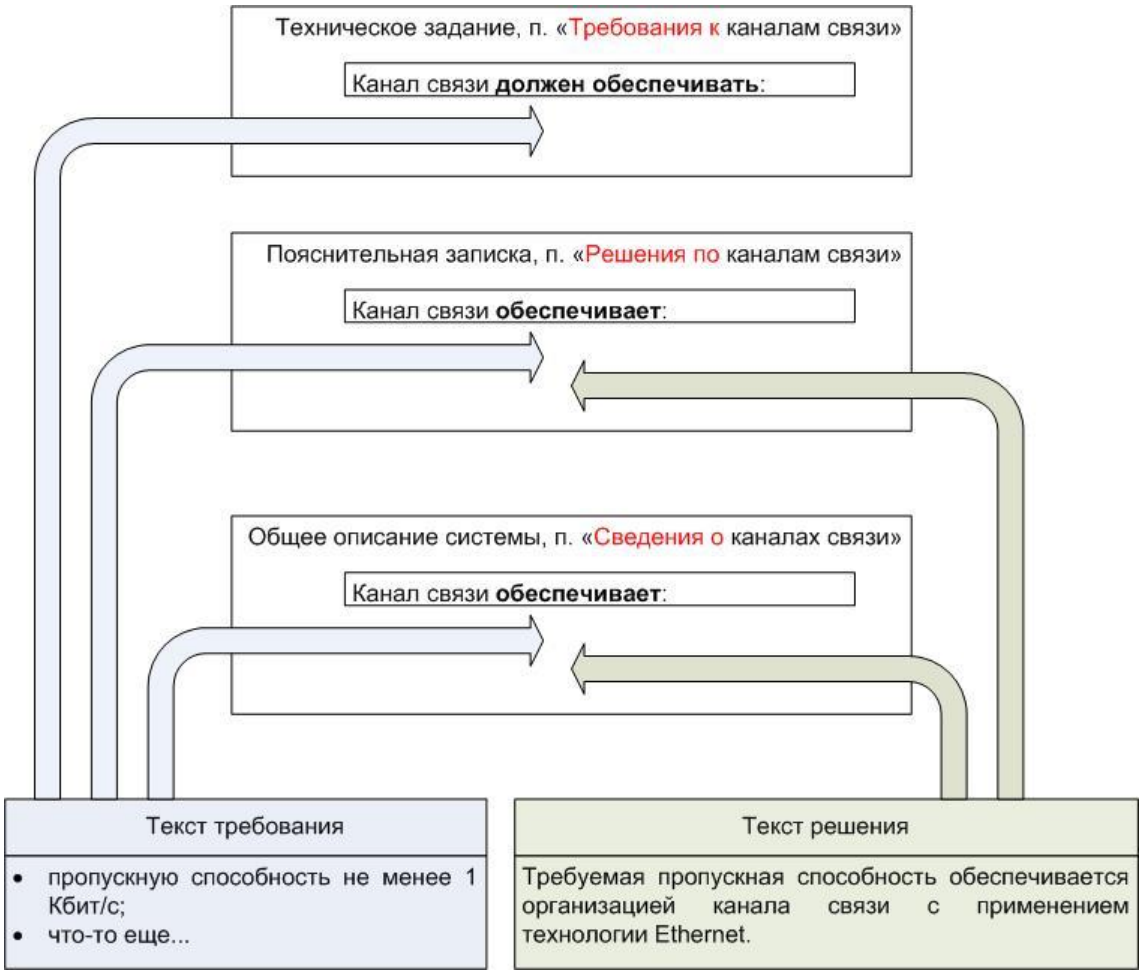

Рисунок 2 – Метод документирования по принципу единого источника

Разработка технической документации сводится к однократной реализации связей между структурными элементами документов комплекта, т.е. к созданию библиотеки взаимоувязанных документов. Связи между структурными элементами реализуются единожды, внедрением одного или множества общих фрагментов в структуры множества документов. Корректировка текста требования и (или) текста решения приводит к синхронному изменению содержимого всех взаимоувязанных документов [13].

В итоге, трудозатратная работа по редактированию проводится только однажды в одном месте (внутри единого источника), дальнейшие преобразования в выходных документах различных форматов выполняются автоматически соответствующим программным средством. Трудозатраты на поддержание документации в актуальном состоянии сводятся к минимуму, снижаются требования к внимательности и аккуратности исполнителя, сокращается повторение ошибок. Исключаются трудозатраты на оформление документов – их публикация может осуществляться с применением единых шаблонов разметки, однажды созданных, к примеру, согласно требованиям ГОСТ 2.104 и ГОСТ 2.105 [4, 5, 13].

Но самое главное состоит в том, что библиотека единого источника с течением времени превращается в настоящую универсальную базу знаний – знания накапливаются в библиотеке от проекта к проекту, документы оптимизируются, качество документации становится выше.

Автоматизация разработки технической документации путем создания каскада взаимоувязанных документов (библиотеки), удовлетворяющих ряду условий создания, жизни, актуализации и утилизации (в том числе требованиям ГОСТов) возможна с использованием одного из современных инструментариев подготовки документации [9].

Возможности современных программ очень обширны и далеко не исчерпываются мощным инструментом создания документации, генерации help-файлов, а также удобной системой управления контентом web-сайтов.

В технической документации приходится описывать сходные функции программ и систем, сходные экранные формы, сходные действия пользователей. Описания одних и тех же объектов приходится полностью или частично дублировать в разных документах [15].

Принцип единого источника позволяет параметризовать почти одинаковое и формировать из этих блоков документы в том виде, в котором их предпочитает получать их аудитория.

Применение технологии единого источника требует подготовки. Сначала проектируется структура единого источника, разрабатываются шаблоны и стили оформления, устанавливается и настраивается инструментарий для формирования документов. Эта стадия отнимает время в начале проекта и предъявляет довольно высокие требования к квалификации разработчика технической документации. Но затем начинается рутинная работа по написанию текста и его загрузке в единый источник. В любой момент на основе введенного текста можно сформировать документы, в большей или меньшей степени готовности. На этой стадии мы получаем отдачу от сделанных вложений. Раздел, рисунок, таблица, абзац, любой другой фрагмент, который должен появляться в нескольких местах, при необходимости достаточно исправить однократно в едином источнике. Это касается не только текста, но и структурных решений. Однократного внесения изменений в шаблон, предположим, руководства пользователя, достаточно, чтобы все документы этого типа после очередной автоматической обработки были изменены нужным образом [15, 16, 17, 18].

Принцип единого источника в документировании помогает организовать работу коллектива соавторов, распределив между ними более-менее изолированные подзадачи. Таким образом, единый источник – это не только техническое, но еще и организационное решение. Взаимодействие между участниками разработки технической документации тоже может быть автоматизировано. Крупные интегрированные среды для автоматизации документирования, такие, как AuthorIT, SiberSafe, RoboHELP, предоставляют для этого различные средства: управления правами доступа, версионный контроль, планирование работ и т.п.

Наиболее поверхностная потребность, которую удовлетворяет единый источник – преобразование одного и того же текста в разные электронные форматы, допустим, в PDF и в HTML. Преобразование документа в разные форматы полезно, если необходимо опубликовать один и тот же текст в разных

средах, например, напечатать книгу, выложить ее в Интернете, подготовить электронную справку из руководства пользователя.

Использование принципа единого источника предоставляет возможность:

– придерживаться строго унифицированного стиля оформления, что, как правило, необходимо при подготовке единого документа или комплекта документации;

– в любой момент придать тексту заданное оформление, что позволяет вести работу над текстом и оформлением параллельно;

– изменить оформление или обеспечить возможность выпускать документы в двух разных видах, скажем, по ГОСТу и в фирменном дизайне.

Чтобы разделить текст и оформление, используют стили. Однако разные инструменты и технологии предусматривают разные степени и формы такого разделения. Так, в текстовом процессоре Microsoft Word стили абзацев, строк и списков можно вынести в шаблон, но формат бумаги, поля, колонтитулы и другие параметры страницы всегда остаются принадлежностью конкретного документа. Технология CSS позволяет хранить многие свойства оформления вне HTML-документа, но не лишает автора возможности применять форматирование непосредственно (хотя бы с помощью атрибута style). Некоторые решения, основанные на XML и XSLT, в том числе, DocBook/XML, позволяют полностью лишить автора контроля над оформлением документа [17].

Автоматическое формирование документов невозможно без унификации структуры единого источника, причем унификация носит многоуровневый характер.

Компоновка документов требует типизации предметов описания и унифицированной рубрикации описаний однотипных предметов. Если у нас есть несколько программных продуктов, то все они должны быть описаны в одних и тех же категориях. Под каждую из них должна быть отведена рубрика, выделенная в едином источнике техническими средствами. В противном случае мы не сможем применять к их описаниям шаблоны. С другой стороны, в силу

различий между отдельными предметами их описания не могут быть тождественными по структуре. Каждая программа имеет индивидуальный набор пользовательских задач, функций, настраиваемых параметров. Следовательно, некоторые рубрики должны представлять собой массивы вложенных описаний. Вложенные описания приходится атрибутизировать, поскольку они могут быть неравнозначны относительно разных аспектов, что важно при компоновке документов. Скажем, часть настраиваемых параметров должна быть описана в руководстве системного администратора, а часть в руководстве пользователя. В случае одного дифференцирующего признака можно было бы обойтись созданием дополнительного уровня структуры в едином источнике, но если требуется различать такие описания по нескольким признакам одновременно, то без атрибутизации не обойтись [16].

Профилирование текста предполагает, что он подразделяется на фрагменты, обладающие четкими границами. Этим фрагментам мы присваиваем профилирующие признаки. Отдачу от профилирования можно получить только в том случае, если фрагменты достаточно мелки. Иначе окажется, что мы в состоянии отфильтровать разделы, описывающие отсутствующие функции, но, скажем, пункты списка или фразы с их упоминаниями в обзорном разделе остаются вне нашего регулирования. Значит, формальная разметка текста не должна заканчиваться на уровне заголовков максимального уровня вложенности. Мы не можем относиться к разделам единого источника как к «черным ящикам», в которых находится неизвестно что. Для компоновки этого было бы достаточно, а для профилирования нет. Отдельные списки, пункты списков, врезки, абзацы и даже отдельные линейные фрагменты внутри абзацев также должны быть элементами формальной структуры.

Подробная разметка текста внутри раздела требуется не только для профилирования, но и для стилевого оформления. Мы можем указать в стилях, что термин следует набирать курсивом, пункты меню полужирным шрифтом, а листинги шрифтом «Курьер». Но для автоматического применения этих правил

к тексту необходимо, чтобы термины, пункты меню и листинги были в нем отмечены явно.

Унификация необходима как на уровне структурных элементов единого источника, так и на уровне «сплошного» текста, неделимого с точки зрения компоновки.

Унификация на уровне структурных элементов единого источника технически может быть обеспечена двумя способами:

а) использованием для ведения единого источника какой-либо СУБД:
   1) AuthorIT;
   2) Help&Manual;
   3) RoboHELP;

б) использованием языка разметки:
   1) XML;
   2) DITA.

Современные инструментальные средства подготовки документов базируются на принципе единого источника. Второе важное свойство технологий единого источника заключается в возможности придать одному и тому же тексту разное оформление и конвертировать его в любой нужный электронный формат. Интегрированные среды автоматизации документирования часто снабжены средствами для организации групповой работы и технического документооборота, но уже побочные, хотя и чрезвычайно полезные функции, для реализации которых вполне можно использовать специализированные программные продукты. Чем больше возможностей по настройке правил формирования документов предоставляет та или иная технология, чем эффективнее она позволяет бороться с повторами на все уровнях, тем она лучше и мощнее [19].

Применение технологии единого источника в особенности целесообразно при разработке и длительном сопровождении комплектов документации на сложные технические решения: программные и программно-аппаратные комплексы, автоматизированные системы, корпоративные ИТ-инфраструктуры.

Документирование таких технических решений в особенности требует эффективности при создании и обновлении наборов типизированных документов, описывающих типизированные предметы. Кроме того, при документировании крупных комплексов и систем исключительно важной становится масштабируемость комплекта документации, возможность быстро распространить его на новые предметы (компоненты, подсистемы, пользовательские роли, операции, процессы) и быстро освоить выпуск новых типов документов [15].

1.2 Рекомендации по стилю изложения информации

Для определения степени возможной унификации рассмотрим рекомендации по стилю изложения информации: структурированности, строгости, детальности, единообразия, однозначности, лаконичности.

1.2.1 Структурированность

Структурированность вытекает из общих требований понятности и простоты обозрения. Требование структурированности предполагает, что изложение в своем линейном развитии делится на составные части по какому-либо очевидному для пользователя принципу [20].

Структурированность изложения проявляется в следующих формах:

– деление документа на структурные элементы;

– раздельное расположение различных типов пользовательской информации;

– структурированное описание однородных объектов, действий, процессов;

– структурированное описание действий пользователя в пользовательском интерфейсе;

– деление текста на абзацы;

– правила включения в текст иллюстраций, таблиц, формул и других специализированных видов подачи информации.

Предусматривается деление документа на структурные элементы одного или нескольких уровней. Документ допускает деление на структурные элементы первого уровня, структурный элемент первого уровня – на структурные элементы второго уровня и т.д. Предлагается при делении документа на структурные элементы придерживаться требований ГОСТ 2.105–95 и ГОСТ 19.106–78. Структурные элементы первого уровня называются разделами, второго уровня – подразделами, третьего уровня – пунктами, четвертого уровня – подпунктами.

Количество выделяемых в составе структурного элемента структурных элементов следующего уровня составляет два или более. Выделение в составе структурного элемента единственного структурного элемента следующего уровня не допускается. Выделение структурных элементов осуществляется таким образом, чтобы каждый структурный элемент был посвящен определенной частной теме.

При этом:

– структурные элементы соответствуют темам, которые будут чаще всего вызывать интерес у пользователя и которые, таким образом, целесообразно отразить в оглавлении;

– порядок следования структурных элементов соответствует логике изложения.

В документации пользователя только в исключительных случаях допускается выделение структурных элементов пятого уровня (подподпунктов). Рекомендуется избегать подобных решений. Если содержание подпункта таково, что дальнейшее деление на структурные элементы представляется целесообразным, предпочтительнее изменить структуру документа с таким расчетом, чтобы подпункт стал самостоятельным пунктом (если это уместно, подразделом или разделом).

1.2.2 Строгость

Строгость вытекает из общих требований правильности и понятности. Требование строгости предполагает, что выбор лексики и других стилистических средств подчиняется жестким ограничениям, характерным для технических текстов [20].

Строгость изложения проявляется в следующих формах:

– употребление терминологии для обозначения специализированных понятий;

– употребление вспомогательных терминов;

– употребление собирательных и обобщающих наименований;

– соблюдение необходимого уровня конкретности изложения;

– соблюдение нейтрального стиля изложения.

В документации пользователя программного средства для обозначения специализированных понятий употребляются принятые для этих понятий термины.

Терминология подразделяется на специфическую для документируемого программного средства и неспецифическую. К терминологии, специфической для документируемого программного средства, относятся термины, употребляемые для обозначения специфических объектов обработки и процессов, характерных только для документируемого программного средства (либо для серии программных средств, имеющих общую область применения, близкие по своему характеру практические задачи и сходные принципы функционирования).

К специфической терминологии, употребляемой в документации пользователя, предъявляются следующие требования:

– соответствует специфической терминологии, употребляемой в пользовательском интерфейсе программного средства;

– не вступает в противоречие с неспецифической терминологией (т.е. в качестве специфического термина не употребляется слово, уже закрепившееся в качестве неспецифического, широко употребляемого термина).

Терминология, не специфическая для документируемого программного средства, употребляется в соответствии с принятой в профессиональной среде практикой.

Не допускается варьирование терминологии на протяжении отдельных документов и всего комплекта документации пользователя в целом. Употребление терминологии подчиняется правилам единообразного изложения. Не допускается употребление вместо специализированных терминов слов и выражений, заимствованных из профессионального жаргона.

При описании вводимых, хранимых и обрабатываемых данных, объектов обработки, процессов, процедур, отдельных действий пользователя и т.п. соблюдается должный уровень конкретности. Это выражается в том, что для обозначения тех или иных понятий избираются термины, в точности соответствующие этим понятиям, но не более частные и не более общие. Например, если в какой-либо области пользовательского интерфейса выполняется ввод текста, употребляется выражение «введите текст», но не более частное «наберите текст». «Набрать» в данном случае слишком частное понятие, поскольку ввод текста может включать в себя набор текста, его удаление, копирование текста из буфера обмена и т.д. Если файл того или иного формата предназначен для хранения растрового образа, употребляется выражение «растровый образ», но не более общее «графический образ» или «рисунок». «Графический образ» в данном случае слишком общее понятие, поскольку графический образ может быть не только растровым, но и векторным.

Документация пользователя программного средства характеризуется нейтрально-книжным стилем изложения. Нейтрально-книжный стиль изложения проявляется в отказе от образной и субъективно-эмоциональной речи, а также в определенных лексических, грамматических и синтаксических

ограничениях. Нейтрально-книжный стиль изложения предполагает полный отказ от образной и субъективно-эмоциональной речи. Любые слова употребляются в документации пользователя только в своем буквальном значении. Не допускается использование образных выражений, метафор, преувеличений. При необходимости сделать изложение наглядным рекомендуется избегать сравнений и аналогий.

1.2.3 Детальность

Детальность вытекает из общих требований полноты и понятности. Требование детальности предполагает, что все важные для пользователя аспекты функционирования программного средства описаны максимально конкретно, с точным именованием всех объектов, элементов пользовательского интерфейса и компонентов программного средства [20].

Документация пользователя программного средства в обязательном порядке содержит описание конкретных действий пользователя программного средства по решению тех или иных практических задач и по обслуживанию программного средства. Детальность изложения проявляется в следующих формах:

– самодостаточность текста документации пользователя по отношению к пользовательскому интерфейсу программного средства;

– максимальная детализация действий пользователя в пользовательском интерфейсе;

– точная идентификация объектов и элементов пользовательского интерфейса;

– пошаговое описание сложных действий.

Упоминающиеся при описании конкретных действий пользователя элементы пользовательского интерфейса и объекты обработки должны быть исчерпывающим образом идентифицированы: при упоминании каждого из них приводится термин, обозначающий разновидность элемента пользовательского

интерфейса или объекта обработки, а также идентификатор. При описании действий пользователя не допускается исключать из описания те или иные шаги, за их кажущейся очевидностью.

1.2.4 Единообразие

Единообразие вытекает из общих требований непротиворечивости и понятности. Требование единообразия предполагает, что для описания элементов пользовательского интерфейса, объектов обработки, процессов, процедур и т.п. на протяжении всего изложения используются одни и те же слова и выражения [20].

Единообразие изложения проявляется в следующих формах:

– единство терминологии;

– единообразие в употреблении вспомогательных терминов;

– последовательное употребление одних и тех же обобщающих и собирательных наименований;

– единообразие фрагментов текста, описывающих сходные явления.

Единство терминологии предполагает следующие правила:

– каждый термин всегда употребляется в одном и том же значении;

– никакие два (или более) термина не употребляются в одном и том же значении;

– каждый термин сочетается с другими терминами и прочими словами в соответствии с одними и теми же синтаксическими моделями.

Каждый термин на протяжении всей документации пользователя употребляется в одном и том же значении. Не допускается употребление термина в близких, но различающихся значениях. Риск такого смешения значений возникает в тех случаях, когда существуют различные традиции употребления того или иного термина (восходящие, например, к пользовательскому интерфейсу и практике документирования различных операционных систем или программных комплексов). В таких случаях в рамках

процесса документирования вырабатывается единое понимание термина, и в соответствии с этим пониманием термин последовательно употребляется в документации.

Каждый термин на протяжении всей документации образует с другими терминами и прочими словами одни и те же устойчивые выражения, последовательно употребляемые для обозначения или описания тех или иных отношений, процессов, процедур и т.п. Одинаковые отношения между объектами и явлениями описываются одинаковыми лексическими и грамматическими средствами. Например, термин «файл» образует с другими терминами и прочими словами выражение «сохранить данные в файле». Единообразие в употреблении термина «файл» предполагает, в частности, что для описания этого действия на протяжении всей документации будет использоваться только это выражение (с точностью до требуемых законами языка синтаксических трансформаций): «сохраните данные в файле», «данные не были сохранены в файле», «сохранять данные в файле», «сохранение данных в файле», – но не: «сохраните файл с данными», «сохраните данные в файл», «запишите данные в файле» и т.п.

1.2.5 Однозначность

Однозначность вытекает из общих требований правильности и понятности. Требование однозначности предполагает, что любой фрагмент описания допускает однозначное толкование [20].

В документации пользователя программного средства изложение строится таким образом, чтобы пользователь интерпретировал все положения документации однозначным образом. Выражения, допускающие неоднозначное толкование, исключаются из текста [27]. Однозначность изложения проявляется в следующих формах:

– однозначное описание действий, выполняемых лицами, взаимодействующими с документируемым программным средством;

– однозначное описание действий, выполняемых программными и аппаратными средствами;

– упорядоченное употребление указательных местоимений и близких к ним по значению слов и выражений;

– исключение из употребления грамматических форм и синтаксических конструкций, допускающих неоднозначную интерпретацию.

Описание тех или иных действий, выполняемых лицами, взаимодействующими с документируемым программным средством, всегда сопровождается точными указаниями на лицо, выполняющее эти действия. Для обозначения тех или иных действий, выполняемых лицами, взаимодействующими с документируемым программным средством, употребляются следующие формы глаголов:

– изъявительное наклонение, настоящее время;

– повелительное наклонение, множественное число.

В общих описаниях действий, выполняемых лицами, взаимодействующими с документируемым программным средством (при изложении структурной и справочной информации), употребляются глаголы в форме изъявительного наклонения, в 3-м лице единственного или множественного числа настоящего времени. Для обозначения субъектов того или иного действия употребляются обобщающие наименования, принятые в данной документации для обозначения лиц, взаимодействующих с документируемым программным средством, либо местоимения, заменяющие их в контексте связной речи. Например: «Пользователь создает каталог…»; «Администратор запускает программу архивации…».

Для описания действий, выполняемых лицами, взаимодействующими с документируемым программным средством, не допускается употреблять выражения, в которых субъект действия указан неявно или не указан вовсе. Например, не допускается: «Документ открывается щелчком мыши»; правильно: «Откройте документ щелчком мыши», или: «Пользователь открывает документ щелчком мыши».

Описание действий, выполняемых документируемым программным средством либо другими программными и аппаратными средствами, в обязательном порядке содержит явное указание на автоматический характер действий. Например: «Поиск подходящего значения выполняется автоматически». Для описания тех или иных действий, выполняемых документируемым программным средством либо другими программными и аппаратными средствами, употребляются глаголы в форме изъявительного наклонения, в настоящем времени.

Во избежание неоднозначного толкования текста в документации пользователя употребляются с ограничениями следующие грамматические формы и синтаксические конструкции:

– модальные глаголы и выражения;

– имена существительные в форме множественного числа;

– однородные члены предложения, связанные отношением простого перечисления;

– деепричастия.

1.2.6 Лаконичность

Лаконичность вытекает из общих требований понятности и простоты обозрения. Требование лаконичности предполагает, что изложение ведется максимально сжато и кратко [20].

К стилю изложения предъявляются, помимо требований, вытекающих из общих требований к документации пользователя, этикетные требования, вытекающие из этических, культурных и речевых норм, принятых в обществе.

Лаконизм изложения проявляется в следующих формах:

– ограничения, накладываемые на структуру и размер предложений;

– ограничения, накладываемые на содержание документации.

На содержание документации накладываются следующие ограничения:

– не допускаются замечания, не имеющие отношения к предмету документации;

– в пределах одного неделимого структурного элемента не допускаются фразы, содержание которых целиком и полностью исчерпывается предыдущими.

В документации не допускаются описание, не имеющие отношения к предмету документации пользователя.

Относящимися к предмету документации пользователя считаются:

– сведения о решении пользователем свойственных ему практических задач с помощью программного средства;

– сведения о функциональных возможностях, предоставляемых программным средством пользователю;

– важные для пользователя сведения о процессах, происходящих в программном средстве, и процессах, связанных с обменом данными между программным средством и другими программными и аппаратными средствами;

– предписания и инструкции, касающиеся производственной или служебной деятельности пользователя и имеющие прямое отношение к работе пользователя с программным средством (для программных средств, используемых в производственной или служебной деятельности).

Не имеющими отношения к предмету документации пользователя считаются описания внутренних процессов, связанных с работой программного средства и не представляющих интереса для пользователя; детали производственной или служебной деятельности пользователя, не связанные с функционированием программного средства; догадки относительно организации пользователем работы, его случайных предпочтений и эмоциональных реакций и т.п. В минимальном количестве допускаются высказывания, не относящиеся непосредственно к документируемому программному средству, а представляющие собой положения общего плана, относящиеся к программным средствам того же класса, к используемым

информационным технологиям и особенностям пользовательского интерфейса, к предметной области, обслуживаемой программным средством.

1.3 Принципы построения набора стилей для технического документа

Стилем называется набор параметров форматирования, который применяется к абзацам текста, таблицам, спискам и знакам (символам), чтобы быстро изменить их внешний вид. Стили – это удобный механизм для работы с документами большого объема, они полезны, когда создается множество документов с одинаковыми правилами оформления. Удобство механизма стилей заключается в том, что, разработав один раз набор стилей, их можно использовать во всем документе и в новых документах. Даже если требования к форматированию изменятся, изменение параметров стилей приведет к автоматическому переформатированию всего текста [23].

При построении набора стилей рекомендуется руководствоваться следующими принципами:

а) технический документ делится на структурные элементы, каждый из которых несет четко выраженную содержательную нагрузку: объяснение, шаг процедуры, заголовок таблицы, термин и т.п.;

б) в текстовом процессоре структурные элементы представлены абзацами и линейными фрагментами. Каждый стиль должен соответствовать некоторому типу содержания, а не способу его оформления;

в) разным типам содержания должны соответствовать разные стили. Не рекомендуется использовать один и тот же стиль для разных типов содержания на том основании, что эти типы предполагают схожее оформление текста;

г) общие признаки визуально схожих стилей следует выносить в родительские стили, от которых эти признаки будут наследоваться. В результате образуется дерево наследования с одним корнем.

д) для заголовков глав, разделов и т.д., а также для колонтитулов, номеров страниц и других элементов оформления желательно использовать стили, предусмотренные в текстовом процессоре.

Выводы по первому разделу

В данном разделе приводится описание предметной области – документирование программных продуктов.

Рассмотрен концептуальный подход документирования программных продуктов, «взаимоувязанность» различных документов на разных стадиях существования программного продукта. Исследованы существующие методы подготовки документации: «классический» и по «принципу единого источника». Принцип единого источника позволяет унифицировать почти одинаковое описание в блоки, и формировать из этих блоков документы.

Для определения степени возможной унификации рассмотрены рекомендации по стилю изложения информации: структурированности, строгости, детальности, единообразия, однозначности, лаконичности. Приводится обоснование их использования.

Расписаны принципы построения набора стилей для технического документа.

2 Проектирование оптимизированной системы документирования

2.1 Процесс подготовки документации «как есть»

ООО «Бюджетные и Финансовые Технологии» (компания БФТ) – российский разработчик проектных решений на базе собственных методологических и программных продуктов для государственного сектора и бизнеса [25].

Компания БФТ создана в 1997 году, является дочерним предприятием компании IBS. Центры технической поддержки и офисы компании БФТ работают в 21 субъекте Российской Федерации.

Решения компании БФТ входят в Единый реестр российского ПО.

Продуктовая линейка охватывает следующие направления:

- управление государственными финансами;
- управление закупками;
- управление активами;
- труд и занятость;
- оказание государственных и муниципальных услуг.

Линейка решений компании БФТ также включает спектр программных продуктов для коммерческих организаций и консалтинговые услуги.

Результатом многолетнего сотрудничества компании БФТ с органами федеральной и региональной государственной власти стало внедрение централизованных решений масштаба региона в 22 субъектах РФ и успешная реализация более 3870 проектов в 79 регионах и более чем 9500 муниципальных образованиях РФ, а также в Республиках Беларусь и Казахстан.

Проекты федерального уровня реализованы в Федеральном казначействе, Министерстве промышленности и торговли Российской Федерации, Министерстве транспорта Российской Федерации, Министерстве по развитию Дальнего Востока, Федеральной службе по труду и занятости (Роструд).

Компания БФТ входит в состав:

– рабочих групп Минфина России по вопросам совершенствования государственного (муниципального) контроля, по развитию проекта «Бюджет для граждан», а также по повышению доступности качества государственных (муниципальных) услуг.

– экспертной группы при Координационной комиссии по созданию и развитию государственной интегрированной информационной системы управления общественными финансами «Электронный бюджет».

– консультативного совета по вопросам развития инфраструктуры электронного правительства при Министерстве связи и массовых коммуникаций Российской Федерации.

Компания БФТ вошла в ТОП-25 рейтингов крупнейших ИТ-разработчиков России в 2017 году, подготовленных информационно-аналитическими агентствами CNews и Tadviser. По итогам 2013–2015, 2017 годов компании БФТ присвоено звание «Лучший поставщик в сфере информационных технологий».

Компания БФТ имеет очень сложную структуру, множество структурных подразделений, которые показаны на схеме организационного управления, представленной на рисунке 3.

Документированием информационных систем в компаниях-разработчиках программных продуктов, как правило, занимаются специализированные подразделения. В компании БФТ процессами документирования IT-решений занимается отдел технической документации (ОТД) Департамента обеспечения качества. Данное подразделение относится к производственному блоку компании. Все подразделения производственного блока (производственные центры – ПЦ) связаны между собой производственными задачами. Информационные потоки между ПЦ осуществляются в электронном виде.

В функции отдела технической документации входит разработка, обновление и сопровождение эксплуатационной документации на всю ИТ-продукцию компании.

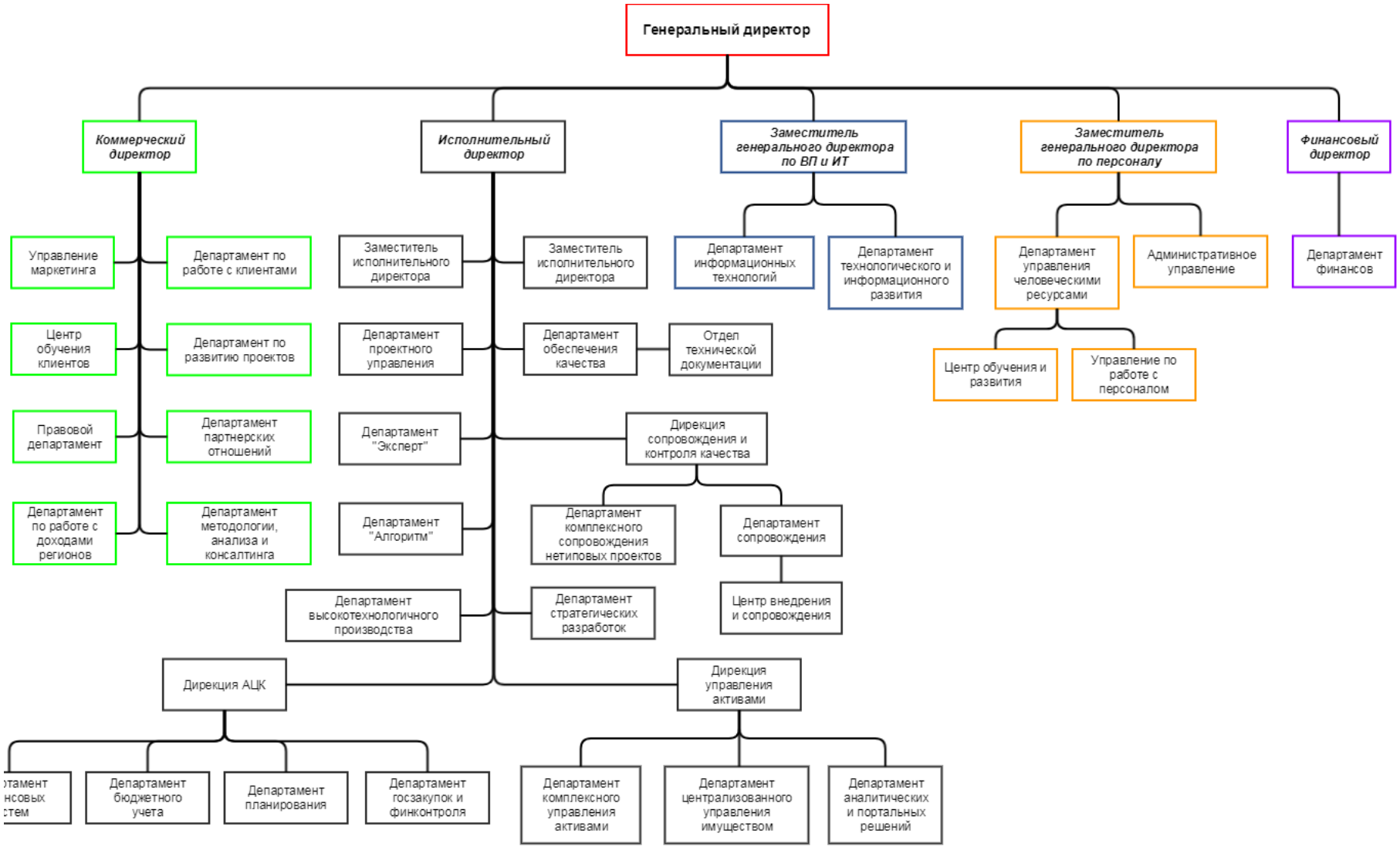

Рисунок 3 – Схема организационного управления компанией

Процесс документирования состоит из следующих этапов:

- получение списка доработок;
- предварительный анализ доработок;
- проектирование списка документов;
- составление плана разработки и обновления документации;
- разработка или обновление документации;
- тестирование документации;
- формирование комплекта документации;
- сопровождение документации.

2.1.1 Получение списка доработок

Список доработок системы предоставляется с указанием приоритета выполнения в системе «АЦК-Контроль» в соответствующем продуктовом проекте (с описанием бизнес-процессов) в ОТД предоставляют представители ПЦ по электронной почте начальнику ОТД.

Описание бизнес-процессов с указанием особенностей выполнения (что создать, последовательность, какие поля являются ключевыми и какие значения должны содержать, какие статусы должны проходить, условия для перехода на другой статус или выполнения того или иного действия и т.п.) для отражения в документации предоставляется в объеме достаточном, полном и однозначно толкуемом. Описание бизнес-процессов может предоставляться с добавлением графического описания представленного материала (в виде схем, наглядных рисунков).

2.1.2 Предварительный анализ доработок

Предварительный анализ доработок системы проводится для определения степени текущей реализации доработок в системе и ориентировочной трудоемкости документирования. Это позволяет приближенно оценить

необходимый трудовой ресурс (время) для разработки новой документации, описания доработок в уже имеющейся документации.

В процессе анализа составляется список пользовательских документов, в которых должны быть отражены изменения, описанные в доработках или же список подготавливаемых документов.

В работу для обновления или разработки документации, а также для включения в план разработки и обновления документации принимаются реализованные доработки программного продукта, прошедшие проверку аналитическим отделом на соответствие функциональным требованиям (с формулировкой в системе «АЦК-Контроль» – ФТ проведено), прошедшие тестирование. Если по доработкам не требуется проверка – доработки, переведенные на статус «проверен» в системе «АЦК-Контроль».

2.1.3 Проектирование списка документов

Проектирования списка документов – определение перечня документов, которые должны входить в комплект документации на конкретную версию программного продукта.

Результатом является информация о документах, входящих в комплект:

- адресат документа (целевая аудитория);
- номер документа;
- название документа;
- общее количество документов, входящих в комплект;
- изменения количества документов с предыдущих версий.

Проект списка документов составляется начальником ОТД и отправляется по электронной почте на согласование представителю ПЦ.

После согласования списка документов на версию, в проекте Документация системы «АЦК-Контроль» создаются записи с задачами по обновлению и разработке документации на статусе «новый». В записях

указываются: версия программы, на которую обновляется документ; дата выпуска версии программы.

2.1.4 Составление плана разработки и обновления документации

На основании предварительного анализа доработок и проекта списка документов составляется план разработки и обновления документации (список задач). В задачах предварительного плана указывается:

– описание задачи;

– приоритет выполнения;

– оценка времени на выполнение;

– первичные сроки выполнения, сроки тестирования документации, сроки исправления ошибок в документации;

– конечный срок получения готовых документов, для включения в комплект документации.

Подготовленный предварительный план направляется на согласование представителю ПЦ, для которого разрабатывается или обновляется документация. На данном этапе возможна установка новых приоритетов задач.

После согласования плана разработки и обновления документации со стороны производственного центра он доводится до сотрудников отдела технической документации. В соответствии с планом в проекте Документация системы «АЦК-Контроль» для записей фиксируется согласованные приоритеты и трудоемкость (при переводе записей на статус «принят»). В записях указывается список доработок, которые должны быть описаны в документе и утвержденные сроки выполнения задач (при акцептовании записей на сотрудников).

2.1.5 Разработка или обновление документации

Ответственный сотрудник ОТД (на которого акцептована запись) в

установленные сроки проводит работу по актуализации документации в редакторе MS Word 2013. Для снятия скриншотов экранных форм и подготовки графических материалов используется Hyper Snap DX.6.01.00. При разработке или обновлении документации сотрудник может консультироваться с сотрудниками производственного центра по функционированию различных подсистем и модулей программного продукта. Обращение за консультацией может осуществляться по e-mail, Lync, телефону, ICQ, Skype или иными способами.

ПЦ предоставляет ОТД стенд (сборку, обновление, дистрибутив) на оттестированную версию программного продукта, на основе которой будет проводиться разработка или обновление документации. Выдача сборки, обновления или дистрибутива осуществляется через FTP-сервер компании.

При разработке или обновлении документации ответственный сотрудник проводит детальный анализ доработок, а также собственноручно корректирует список доработок, которые должны быть описаны в документе. В процессе детального анализа доработок сотрудник тщательно изучает постановки на доработку или разработку программного продукта. Постановки на доработку или разработку программного продукта предоставляются отделу технической документации аналитическими отделами посредством прикрепления в соответствующих записях системы «АЦК-Контроль» или предоставлением доступа к специальным серверам с файлами аналитических данных. На основе постановок на доработку или разработку программного продукта и функционирования оттестированной версии ведется разработка или обновление пользовательской документации.

Разработка и обновление пользовательской документации осуществляются на основе ГОСТ 19.503–79 «Руководство системного программиста. Требования к содержанию и оформлению» и ГОСТ 19.505–79 «Руководство оператора. Требования к содержанию и оформлению».

2.1.6 Тестирование документации

Тестирование документации проводится после ее разработки или обновления на версию. Ошибки, обнаруженные по результатам тестирования, исправляются до формирования комплекта документации.

Тестирование документации осуществляется в соответствии с ГОСТ Р ИСО/МЭК 12119–2000 «Информационная технология. Пакеты программ. Требования к качеству и тестирование» [22].

Ошибки, обнаруженные в документации, фиксируются задачах «АЦК-Контроль» и передаются ответственному исполнителю по разработке или обновлению тестируемого документа. Автор исправляет ошибки в документе и возвращает его на повторную проверку. Процесс повторяется до тех пор, пока в документе не будут исправлены все ошибки.

2.1.7 Формирование комплекта документации

После завершения разработки, обновления и тестирования документации отдел технической документации формируется комплект документации по соответствующему программному продукту. Для комплекта документации составляется «Спецификация».

Комплект документации, в соответствии с согласованными сроками выпуска новой версии документации, размещается на ftp-сервере компании. О размещении документации на ftp-сервере отправляется информационное письмо: представителям ПЦ, сотрудникам Департамента эксплуатации.

2.1.8 Сопровождение документации

После выдачи комплекта документации с объектов через Департамент эксплуатации могут поступать заявки на доработку и исправление ошибок в документации.

ОТД рассматривает заявку и принимает решение о необходимости в доработке или исправлении документации. В случае наличия необходимости в доработке или исправления документации запись копируется сотрудниками отдела технической документации из проектов Сопровождение в проект Документация системы «АЦК-Контроль». В проекте Документация запись с указанием срока и приоритета акцептуется на ответственного сотрудника. После доработки и исправления документация выкладывается на ftp-сервере компании. В комментарии к записи в проекте Документация указывается ссылка на исправленную документацию. При отсутствии необходимости в доработке или исправлении документации заявка отказывается с указанием причины.

2.1.9 Описание модели «как есть» и ее анализ

По результатам исследования производственных процессов и описания этапов подготовки документации была спроектирована модель процесса «как есть» и представлена на рисунке 4 в нотации IDEF0.

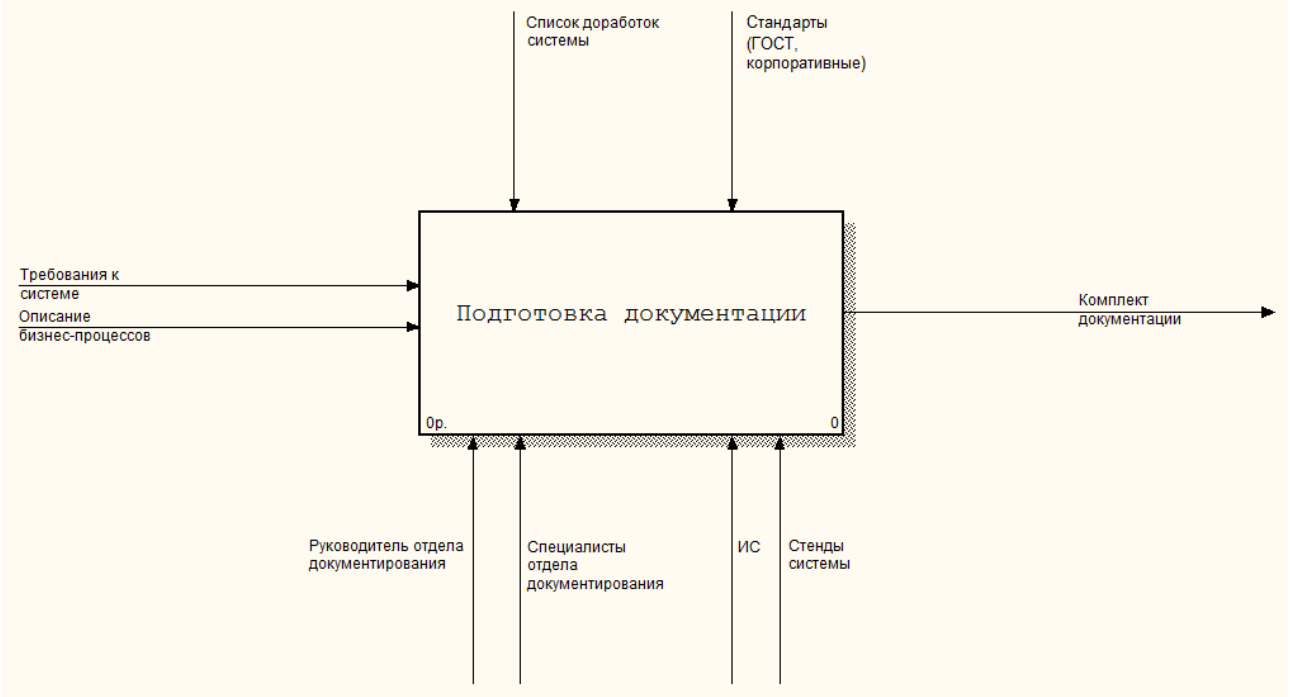

Рисунок 4 – Контекстная модель процесса «Подготовка документации»

Чтобы показать процессы подготовки документации более подробно, приведена декомпозированная модель процесса до уровня подпроцессов, представленная на рисунке 5.

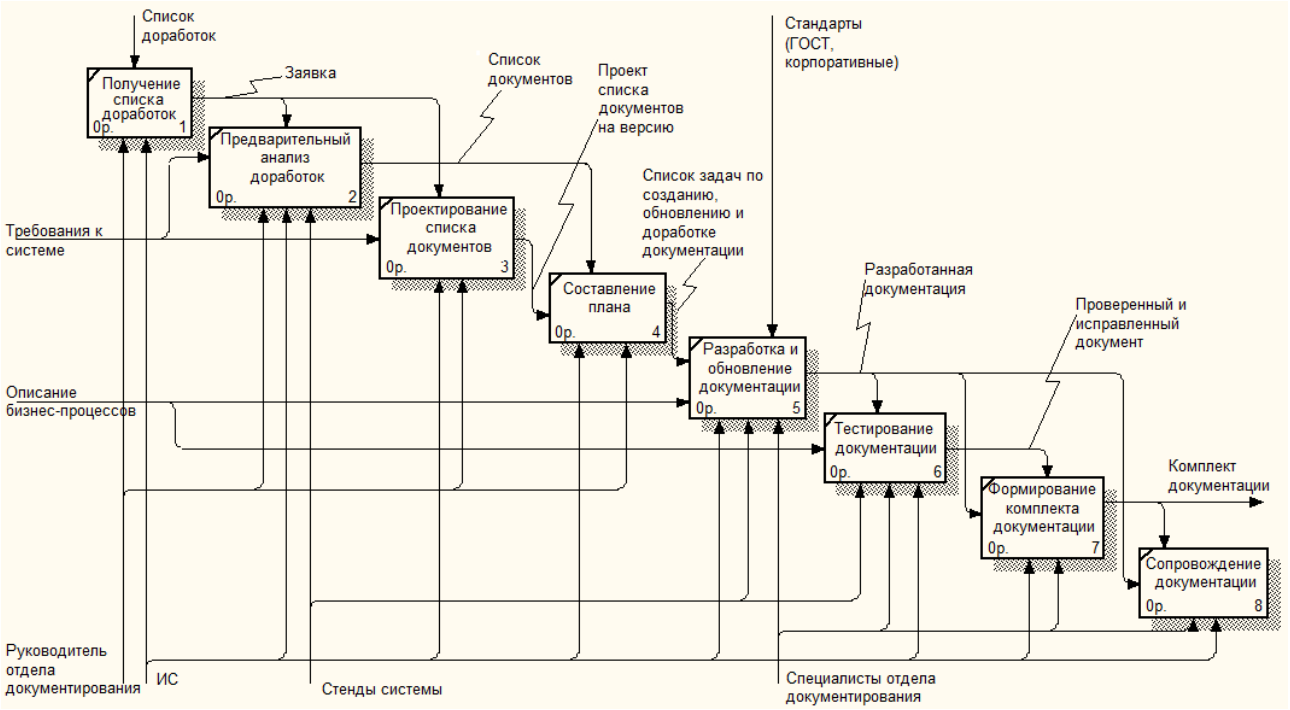

Рисунок 5 – Декомпозированная модель до уровня подпроцессов

Процесс «Подготовка документации» содержит в себе много сложных подпроцессов, которые затрудняют анализ, подготовку и написание документации, что приводит к возникновению ошибок в документации и увеличению трудозатрат. В частности, были выявлены следующие недостатки в процессе подготовки документации и используемых инструментариях:

– использование в ПЦ разных систем учета задач (доработок);

– использование в ОТД системы учета задач «АЦК-Контроль», содержащей устаревший и неудобный функционал;

– исходная аналитическая информация хранится в разрозненных местах, содержит нетиповую структуру, детальность информации зависит от знаний и опыта аналитика;

– предварительный анализ доработок требует изучения всей информации исходных данных, а не краткого релиза;

– существенные сложности в переходе от одного перечня документов к другому;

– сложности в составлении и актуализации сетевых планов (диаграмм Ганта) работы по задачам документирования в разрезе сотрудников, продуктов;

– отсутствие возможности многопользовательской работы над рабочим материалом, сводимое к работе над одним документом;

– высокая трудоемкость подготовки одной и той же документации в разных выходных форматах: в печатном виде, формате электронных книг, в форматах, встраиваемых в клиентское приложение системы;

– отсутствие возможности автоматического импорта аналитических данных в инструментарий подготовки документации.

2.2 Проектирование оптимизированной модели процесса подготовки документации

Для устранения недостатков в процессе подготовки документации было осуществлено переструктурирование и сокращение подпроцессов с восьми до пяти.

Процесс «Подготовка документации» после оптимизации будет состоять из пяти основных подпроцессов [21]:

– получение списка доработок;

– предварительный анализ доработок;

– составление и согласование плана работ;

– выполнение задач по документированию;

– выдача документации.

В результате была составлена декомпозированная диаграмма в нотации IDEF0 (показанная на рисунке 6) представляющая собой модель процесса «Подготовка документации» после оптимизации.

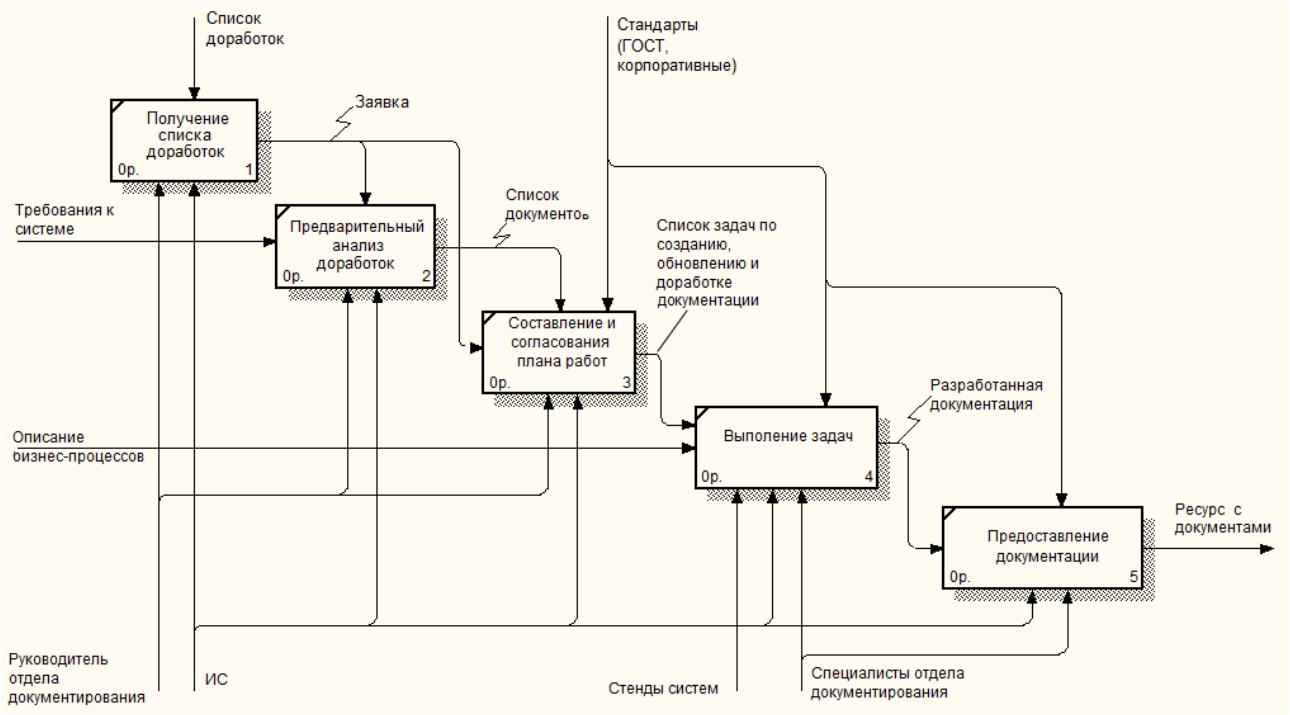

Рисунок 6 – Декомпозированная диаграмма в нотации IDEF0

2.2.1 Получение списка доработок

ПЦ приводит перечень задач (в виде фильтра отбора) в советующем продуктовом проекте JIRA с типом «Доработка» в разрезе версий с указанием даты выдачи.

В соответствующих полях каждой задачи по доработке функционала программного продукта или системы ПЦ указывает следующую информацию:

– приоритет выполнения задачи;

– версия системы, содержащая доработку;

– бюджет проекта для списания трудозатрат;

– желаемый срок выполнения доработки;

– краткое описание доработки;

– ссылка на постановку доработки в Confluence.

Общий план работ по выпуску версии со списком доработок, которые должны быть реализованы на версию, создаются и ведутся ПЦ на странице «Информация по выпуску версий» в Confluence.

При возникновении потребности у ПЦ в срочном документировании или при выявлении ошибок в документации начальнику ОТД назначается задача в соответствующем продуктовом проекте Jira с типом «Документирование» с высоким приоритетом.

2.2.2 Предварительный анализ доработок

Начальник ОТД осуществляет:

– проверку статуса реализации подзадач доработки: аналитики, разработки, тестирования. Если работы по подзадачам не завершены (не имеют статус «Закрыт») – в поле «Метка» подзадачи по документированию ставится: Ожидание_аналитики или Ожидание_разработки или Ожидание_тестирования. Если работы по подзадачам завершены – осуществляется анализ задачи: определяется перечень документов для внесения доработок, добавляется ссылка на группирующую запись, заполняется данные в подзадаче «документирования»: по приоритету выполнения, версии реализации, оценивается время на выполнение документирования;

– определение наличия свободного трудового ресурса в требуемом объеме и в сроки, определенные задачей. Если трудового ресурса нет – документирование откладывается до следующего интервала планирования работ или получения новых вводных от ПЦ (пересмотр приоритетов по задачам), в поле «Метка» подзадачи по документированию проставляется метка «Нет_ресурсов». Контроль изменений в задаче доработки осуществляется еженедельно в конце рабочей недели. Если трудовой ресурс на выполнение задачи имеется – в задачах документирования в поле «Срок» указываются сроки, к которым будет подготовлена документация; устанавливаются сроки промежуточного контроля, сроки финишного контроля, сроки проверки документации, сроки исправления ошибок в документации после проверки (указывается в поле «Комментарий»).

2.2.3 Составление и согласование плана работ

Перечень задач документирования передается начальником ОТД на согласование в ПЦ – в задаче в поле «Метка» ставится значение «Согласование». Согласование задач осуществляется еженедельно по пятницам. Согласованными задачи считаются при указании ПЦ метки – «Согласовано». Согласованный перечень задач со сроками и приоритетами образует план работы. Несогласованные задачи в план работы не включаются и работы по задачам не ведутся.

2.2.4 Выполнение задач по документированию

Начальник ОТД назначает ответственному исполнителю задачу по документированию при переводе на статус «ПРИНЯТ».

Ответственный исполнитель ОТД осуществляет:

– перевод задачи на статус «В РАБОТЕ»;

– в установленные сроки (поле «Срок») проводит работы по документированию:

Основной подпроцесс документирования по разработке эксплуатационной документации адресованной конечному пользователю компьютерной системы определяется следующим порядком операций:

– анализ технических требований к выходному формату готовой документации (печатной, электронной или интерактивной);

– настройка параметров проекта документа;

– определение требуемой структуры документа (скелета) в соответствии с заданным стандартом или выходным форматом;

– формирование структуры документа (скелета) документа;

– изучение целевой аудитории документа, выяснение ее задач, потребностей в информации, уровне подготовки;

– изучение основ предметной области;

– подбор источников исходного материала;

– отбор исходного материала из источников (аналитических данных);

– импортирование исходного материала в редактор;

– переработка исходного текстового материала для включения в имеющийся контекст с целью составления новой содержательной части документа;

– подготовка графических материалов в качестве иллюстраций;

– компоновка переработанного текстового и графического материала в соответствии со структурой документа (скелетом);

– применение к текстовому и графическому материалу средств форматирования в соответствии со стилями и другими средствами оформления;

– задание параметров разметки, ссылок в рамках одного документа и комплекта документации;

– составление вводной (аннотации) и заключительной частей документа;

– создание в документе информационно-поискового аппарата;

– формирование документа требуемого выходного формата;

– вычитка документа, устранение ошибок в оформлении и опечаток;

– отладка работы документа выходного формата (электронной справочной системы) отдельно от программного средства или системы и в составе программного средства или системы.

Формирование готовой документации осуществляется в выходном формате, оговоренном с Заказчиком при постановке задачи. Существует возможность формирования документов в следующих выходных форматах: Adobe PDF, HTML Help, WebHelp, MS Word DOCX, Windows EWriter eBooks, Apple iBooks®/ePUB eBooks.

В процессе выполнения задания ответственный исполнитель в задаче Jira в окне «Сделать запись о работе» (пункт меню *Другие действия → Сделать запись о работе*) фиксирует:

– затраченное время – в поле «Затрачено»;

– краткую достаточную информацию, на что потрачено время – в поле «Сделать запись о работе».

В случае невозможности выполнения задания (по объективным причинам) в задаче JIRA в поле «Метки» заполняется одно из значений и указывается информация (причина) в поле «Комментарий»**:**

– ожидание_консультации – при отсутствии четкого понимания реализации, выявлении несоответствия постановки и фактической работы функционала: ожидание получения консультаций по вопросам функционирования различных подсистем и модулей, либо получение визуальной демонстрации (обучения) работы данного функционала;

– ожидание_аналитики – если подзадача по аналитике не реализована или переоткрыта;

– ожидание_разработки – если подзадача по разработке не реализована или переоткрыта;

– ожидание_тестирования – если требуется окончание тестирования по задаче.

По окончании выполнения работ в записи JIRA осуществляется:

а) перевод на статус:

1) «СДЕЛАН» для задач с типом «Документирование»;

2) «ЗАКРЫТ» для задач с типом «Документирование (Доработка)»;

б) внесение одного из значений в поле «Резолюция» и указание информации в поле «Комментарий» при обработке задачи в зависимости от результатов работы:

1) «Решен» – если работы по задаче выполнены полностью (в комментарии указываем что, где создано или изменено, краткий итог);

2) «Не будет выполняться» – если работы по задаче не будут производиться (в комментарии указываем причину невыполнения задачи);

3) «В соответствии с постановкой» – если изменения документации не производились, так как документация соответствует постановке (указываем задачу, по которой ранее были внесены изменения);

4) «Не полностью описан» – если работы по задаче выполнены не полностью (указываем информацию, что и где создано или изменено, что не сделано, краткий итог по задаче).

2.2.5 Предоставление документации

В Confluence в пространстве Продуктов компании по шаблону создается страница с названием: «Документация системы ... версии X.X.X.X от <Дата>». На странице указывается информация о готовых документах (спецификация документов приведена в таблице 1), прикрепляются документы, ссылки на задачи, дополнительная информация.

Таблица 1 – Спецификация документов

Обозначение	Наименование	Примечание
код документа	наименование документа	Прилепляется файл документа, ссылка на задачу, дополнительная информация

О готовых документах отправляется информационное письмо представителям ПЦ. Отправка осуществляется средствами Confluence по действию «Поделиться этой страницей», диалоговое окно представлено на рисунке **7**.

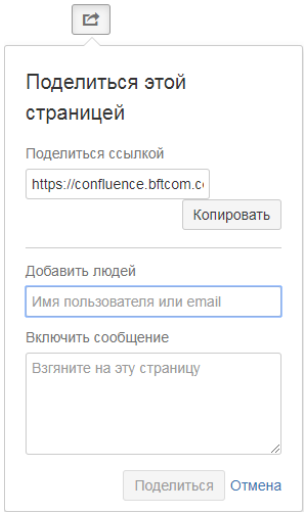

Рисунок 7 – Диалоговое окно отправки информационного письма

2.3 Оптимизация производственных процессов

Для устранения недостатков и оптимизации производственных процессов были приняты решения:

– о переходе на единую базу учета и хранения всех корпоративных знаний, в том числе аналитических данных – Confluence;

– о переходе на централизованную систему учета всех производственных задач – JIRA и автоматизации процесса учета заданий документирования в ней;

– о выборе и внедрении специализированного программного обеспечения для подготовки документации.

2.3.1 Учет заданий документирования в едином инструментарии учета всех производственных задач

Atlassian JIRA – платформа с открытым кодом, предназначенная для управления жизненным циклом рабочих процессов [26].

Платформа предоставляет функционал для управления проектами и позволяет разбивать их на этапы, настраивать типы задач, связывать задачи между собой, назначать ответственных по различным направлениям, настраивать роли участников процессов, формировать отчеты. Работа в Atlassian JIRA происходит через веб-браузер без установки клиентских приложений на рабочих местах. Система масштабируема и подходит как для организаций с небольшим количеством сотрудников (менее 10 человек), так и для более крупных предприятий (от 200 человек).

Основным объектом учета и управления в Atlassian JIRA является Задача, которая содержит информацию о том, что нужно сделать, о том, что сделано, кем сделано, какой путь уже прошла задача, и сколько времени было затрачено каждым участником процесса во время работы. В поле «описание» задачи указывается, для чего она нужна, и каким образом ее рассматривать и выполнять[26].

Atlassian JIRA может работать через защищенное соединение с применением SSL. Имеется возможность просмотра хранилища файлов CVS (система конкурентных версий). JIRA имеет опубликованный программный интерфейс, обеспечивающий программный доступ к основным функциям системы (SOAP API), расширения, позволяющие дополнять систему собственными сервисами для решения специфических задач предприятия.

Решение может интегрироваться не только с распространенными системами версионного контроля Subversion и Git, но и с другими системами (Perforce, VSS, Mercurial, Bazaar), а также с Salesforce.com, Agile, инструментами Rally Software, системой управления тестами Zephyr и десятками других. Благодаря архитектуре с поддержкой плагинов, возможности расширения системы практически безграничны. Существует более 100 готовых бесплатных расширений, не говоря уже о возможности написания собственных с помощью инструмента SDK, предоставляемого Atlassian [26].

В JIRA компании БФТ существует следующие типы проектов:

- Software (продуктовый);
- Business (подразделения).

Выполнение задач по документированию в рамках выпуска различных программных продуктов ведется в продуктовых проектах. Проекты подразделений используются для выполнения внутренних задач подразделения (разработки регламентов, методологии работы и т.д.) и ведутся для каждого года в отдельности.

Ознакомиться со списком проектов JIRA SW БФТ можно перейдя в меню *Проекты → Просмотр всех проектов* (представленный на рисунке 8). Перейдя в проект можно просмотреть информацию по проекту и его задачам (представлен на рисунке 9): Обзор, Очередь работ, Активные спринты, Релизы, Запросы, Timesheet, Коммиты Git, Subcomponents, Component Versions, Тесты, Плагины. Данная информация предоставляется на закладках проекта и является настраиваемой (в том числе и отображение закладок).

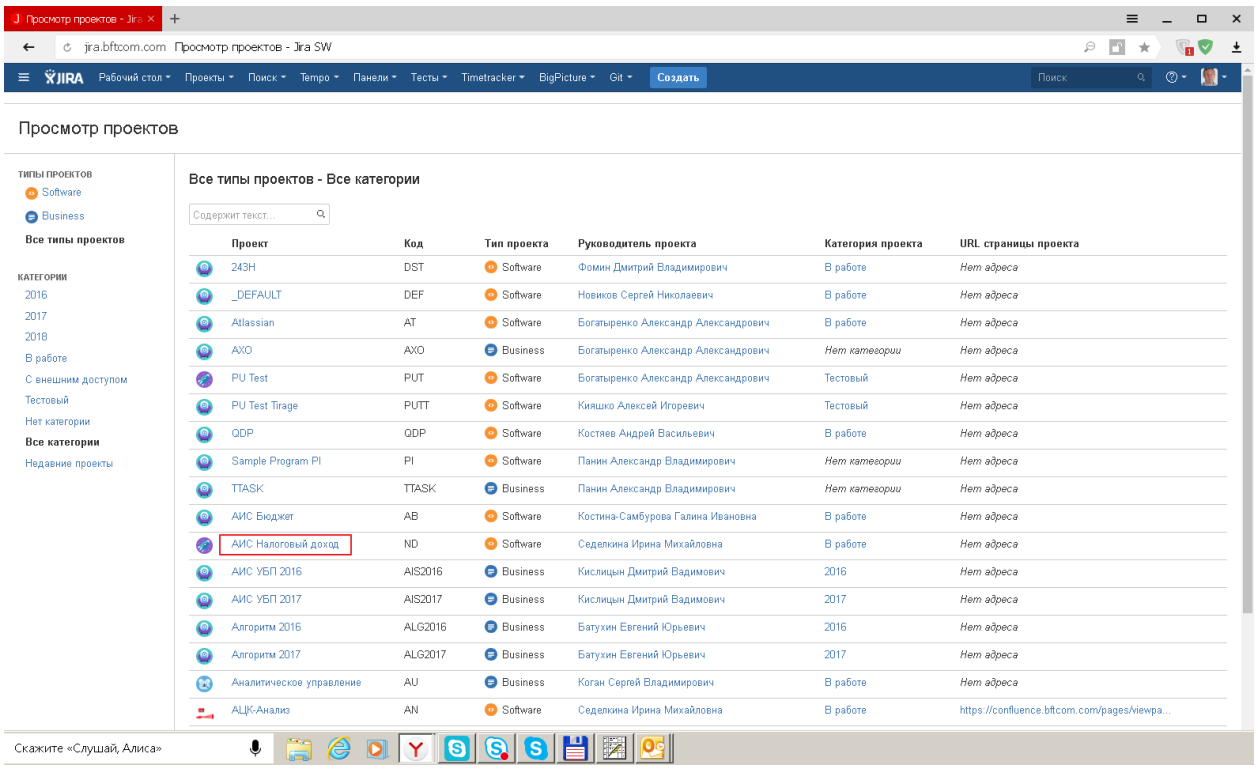

Рисунок 8 – Просмотр списка проектов

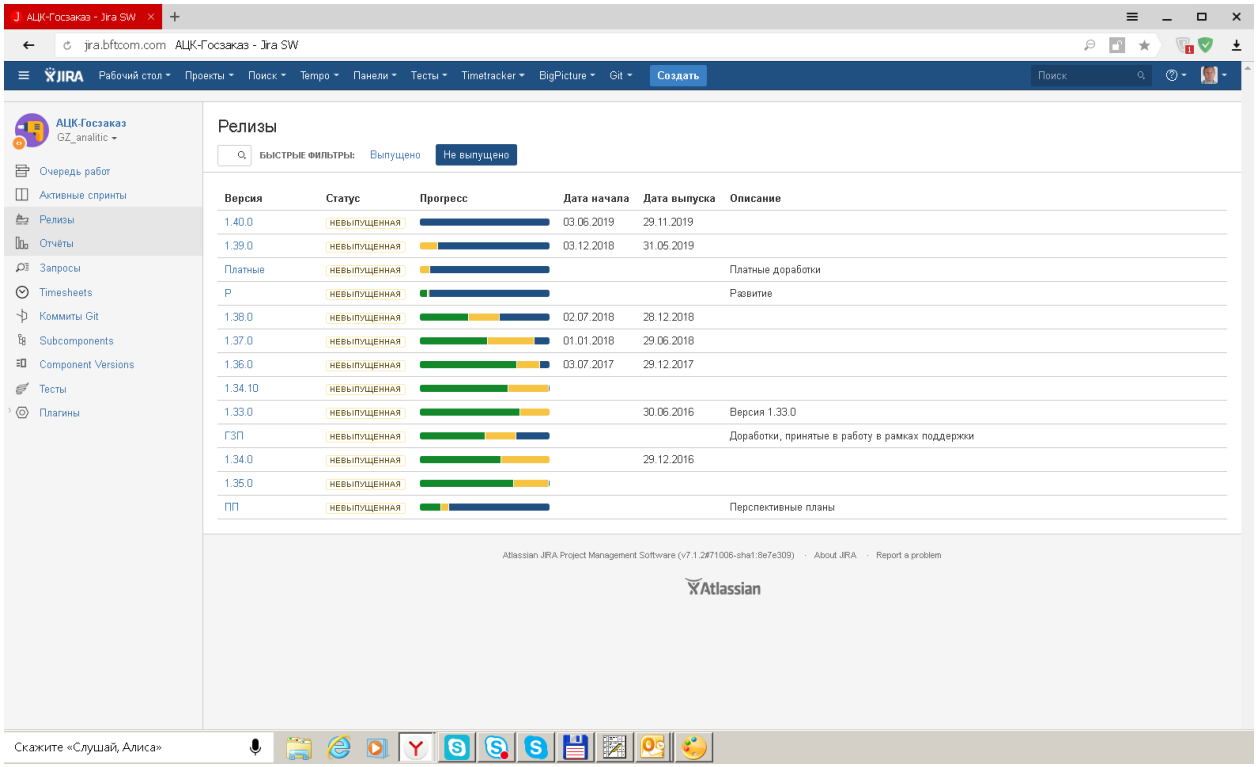

Рисунок 9 – Информация по проекту

Выполнение работ по документированию осуществляется в задачах типов «Документирование (Доработка)» и «Документирование».

Задачи документирования доработок, являются подзадачами задачи типа «Доработка», единственного типа задачи, в рамках которой могут быть выполнены работы по разработке новой функциональности продукта. Задача может быть создана любым пользователем, у которого есть доступ к продуктовому проекту в JIRA. Исходный статус задачи – «НОВЫЙ». Пользователи из группы доступа «Managers» анализируют задачи на статусе «НОВЫЙ» и передают их в работу нажатием кнопки «В работу». Далее задача переходит на статус «ПРИНЯТ» и автоматически создается 5 последовательных подзадач в рамках «Доработки»:

а) «Аналитика (Доработка)» – написание постановки на доработку;

б) «Разработка (Доработка)» – реализация программного кода на функционал;

в) «Тестирование (Доработка)» – тестирование доработки сотрудниками управления тестирования ДОК;

г) «Документирование (Доработка)» – добавление информации по доработке в эксплуатационную документацию отделом технической документации ДОК;

д) «Функциональное тестирование» – финальное тестирование функционала сотрудниками аналитического отдела.

После того, как все 5 подзадач перешли на статус «ЗАКРЫТ» исполнитель по «Доработке» нажимает кнопку «Сделано» и задача переходит на статус «СДЕЛАН». Пользователь из группы доступа «Managers» подтверждает исполнение задачи и ее закрытие, нажимая кнопку «Закрыть», что переводит задачу на статус «ЗАКРЫТ».

Жизненный цикл подзадачи «Документирование (Доработка)» (представлен на рисунке 10): работа начинаются со статуса «НОВЫЙ» после перехода подзадач «Аналитика (доработка)» и «Разработка (Доработка)» на статус «ЗАКРЫТ». На статусе «НОВЫЙ» выполняется предварительный анализ задач, составление и согласование плана работ. Со статуса «НОВЫЙ» пользователь из группы доступа «Doc Lead» (Начальник отдела) нажимает

кнопку «Принять в работу», статус подзадачи изменяется на «ПРИНЯТ». Ответственный исполнитель, начиная работы с проектом документации, переводит задачу на статус «В РАБОТЕ». После выполнения работ по подзадаче Исполнитель завершает работы с проектом документирования и переводит статус подзадачи на «ЗАКРЫТ». Закрытие подзадачи по документированию доработок, возможно, только после того, как подзадача «Тестирование» перешла на статус «ЗАКРЫТ».

Рисунок 10 – Жизненный цикл задачи типа «Документирование (Доработка)»

Задачи верхнеуровневого документирования (тип задачи «Документирование») используются для выполнения работ по подготовке документов. Жизненный цикл задачи (представлен на рисунке 11): исходный статус задачи – «НОВЫЙ». На статусе «НОВЫЙ» выполняется предварительный анализ задач, составление и согласование плана работ. Со статуса «НОВЫЙ» пользователь из группы доступа «Doc Lead» (Начальник отдела) нажимает кнопку «Принять в работу», статус подзадачи изменяется на «ПРИНЯТ». Ответственный исполнитель, начиная работы с проектом документации, переводит задачу на статус «В РАБОТЕ». После выполнения работ по задаче Исполнитель завершает работы с проектом документирования и переводит статус задачи на «СДЕЛАН». Затем пользователь из группы

доступа «Managers» подтверждает исполнение задачи и ее закрытие, нажимая кнопку «Закрыть», что переводит задачу на статус «ЗАКРЫТ».

Рисунок 11 – Жизненный цикл задачи типа «Документирование»

2.3.2 Анализ и выбор оптимального специализированного программного обеспечения подготовки документации

Первоначальный выбор анализируемого специализированного программного обеспечения определялся использованием в нем концепции единого источника. Набор исследуемых инструментариев был ограничен современными, наиболее популярными и широко используемыми системами: AuthorIT, Help&Manual, DocBook, RoboHelp. Анализ был проведен на основе функционирования инсталлированных программ относящихся к данным системами, а также многочисленных публикаций по данной тематике [14].

В результате проведенного анализа наиболее подходящей системой для создания документации в условиях компании БФТ является Help&Manual. Выбор данной системы обусловлен рядом факторов:

а) интерфейс схож с Microsoft Office 2013, имеет большие значки, всплывающие подсказки для быстрого доступа и доступные кнопки команд

(которые не спрятаны где-то внутри меню), при этом наиболее интуитивно понятен по сравнению с другими системами;

б) исходным форматом является XML, это позволяет осуществить настройку обмена данными через интеграционную шину с другими компонентами системы документирования;

в) является WYSIWYG редактором XML, который не требует никакого знания XML или навыков XML-кодирования; работа осуществляется в знакомом текстовом редакторе Word, но также в случае необходимости есть доступ к документу в формате XML;

г) многопользовательская работа над одним проектом, при этом единовременно редактирование топика может осуществлять только одним автор, другой автор этот топик может только просматривать;

д) может редактировать несколько проектов одновременно, что позволяет максимально использовать принцип единого источника;

е) поддержка полного цикла подготовки документации; возможность автоматизированного создания оглавлений, индексов, списков иллюстраций, списков таблиц, возможность гибкой настройки стилей документов; включены утилиты для создания скриншотов и редактирования графических файлов;

ж) разработчики Help&Manual реализовали систему для обработки больших проектов (с многотысячными страницами в разделах);

з) возможность импортирования во внутренний формат файлов наиболее распространенных форматов: HTML Help, Winhelp, HTML files, Text files, Word Documents, RoboHelp;

и) существует возможность динамической загрузки в систему проектов готовых справок, поиск и замена, настройка статуса раздела (топика), многократное использование стилей и настроек;

к) возможность повторного использования текста, в качестве источников унифицированного описания в документах можно использовать разделы и главы из других проектов, различные фрагменты из импортируемых файлов и

фрагменты из центрального хранилища данных; при изменении источников подтянутые фрагменты обновляются автоматически;

л) для вывода готовых документов предусмотрено создание шаблонов HTML, PDF и Word-документов;

м) поддерживает вывод в наиболее распространенные форматы (HTML Help, Cross-platform Help, Adobe PDF, Visual Studio Help, Winhelp, Word, e-Books) нажатием на 1 кнопку.

2.4 Проектирование модели оптимизированной системы документирования

Все проведенные работы по оптимизации производственных процессов, выбор и внедрение современных инструментариев, способных обеспечить данные процессы, позволяют нам спроектировать оптимизированную систему документирования.

Эффективная система документирования, обеспечивающая возможность подготовки документации для ИТ-решений любых прикладных отраслей, должна включать специализированное программное обеспечение документирования, единую базу хранения всех корпоративных знаний, в том числе аналитических данных, систему учета задач и автоматизированный интеграционный механизм для передачи данных между компонентами системы.

Потребность в интеграционном механизме основывается на необходимости автоматизации:

– передачи аналитических данных из единой базы хранения всех корпоративных знаний в специализированное программное обеспечение документирования;

– обработки задач и учет времени, затраченного на процесс документирования, в централизованной системе учета производственных задач;

– учета информации о задаче документирования в проекте документации специализированное программное обеспечение документирования;

– шаблонного формирования страниц выдачи готовой документации и размещения готовой документации в единой базе хранения всех корпоративных знаний из специализированного программного обеспечения документирования.

Решение комплекса задач по передаче данных осуществляется на основе концепции сервис-ориентированной архитектуры (Service Oriented Architecture, SOA), ключевым элементом которой является Интеграционная сервисная шина. Шина – это программное обеспечение, позволяющее объединять большое число платформ и приложений, а также организовать взаимодействие между ними на основе сервисов. При этом технологии, на которых реализованы системы и их сервисы не имеют значения, это может быть JAVA, .NET или другая платформа [34].

Интеграционная шина, как правило, предоставляет следующие функции:

– преобразование сообщений, а также их передача, алгоритмическое перенаправление, постановка в очередь и отслеживание;

– работа с сообщениями в режимах: синхронном, асинхронном, «точка-точка», «публикация-подписка»;

– поддержка XML и SOAP сообщений;

– возможность подключения множества систем через готовые адаптеры и API для написания новых адаптеров;

– оркестровка (автоматическое размещение, координация и управление) служб.

Модель системы документирования с использованием интеграционной шины слияния представлена на рисунке 12.

При внедрении интеграционной шины чрезвычайно облегчается интеграция новых систем – как покупных, так и самостоятельно разработанных. Сервисы перестают быть монолитными приложениями, а разбиваются на единичные службы.

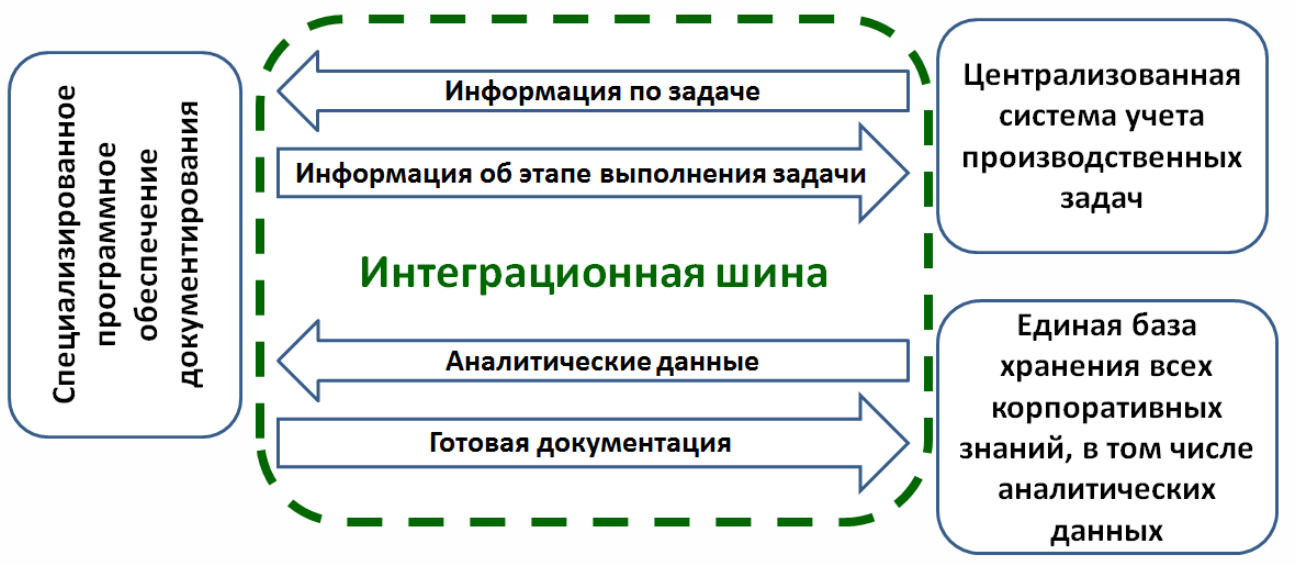

Рисунок 12 – Модель системы документирования

В результате внедрения интеграционной шины достигается прозрачность обмена данными в рамках существующих и внедряемых бизнес-процессов, удается увеличить эффективность и продуктивность работы сотрудников, а также добиться повышения качества услуг документирования, снизить издержки на интеграционные процессы.

Выводы по второму разделу

В данном разделе рассмотрен процесс подготовки документации в компании БФТ «как есть». Описана модель процесса в нотации IDEF0 в укрупненном и декомпозированном виде, базирующаяся на классическом методе составления документации. Установлено, что процесс подготовки документации содержит в себе много сложных подпроцессов, которые затрудняют анализ, подготовку и написание документации, что приводит к возникновению ошибок в документации и увеличению трудозатрат. Были выявлены недостатки в процессе подготовки документации и используемых инструментариях.

Составлена и описана модель процесса подготовки документации после оптимизации, представлена декомпозированная диаграмма в нотации IDEF0 с переструктурированием и сокращением бизнес-процессов с восьми до пяти.

Составлен перечень операций основного подпроцесса документирования, нацеленного на разработку эксплуатационной документации, адресованной конечному пользователю компьютерной системы.

Для устранения недостатков и оптимизации производственных процессов были осуществлены переходы: на единую базу учета и хранения всех корпоративных знаний, в том числе аналитических данных и централизованную систему учета всех производственных задач, проведена автоматизация процесса учета заданий документирования в ней. Осуществлен анализ и выбор специализированного программного обеспечения по подготовке документации основывающего на принципе единого источника и включающего полный цикл работы с документами, от написания до формирования готовых документов.

Осуществлено создание модели оптимизированной системы документирования, базирующейся на принципе единого источника, способной в реальном времени и с минимальными трудозатратами предоставлять актуализированную документацию на любой стадии существования информационных продуктов. Эффективная система документирования, обеспечивающая возможность подготовки документации для ИТ-решений любых прикладных отраслей, включает специализированное программное обеспечение документирования, единую базу хранения всех корпоративных знаний, в том числе аналитических данных, систему учета задач и интеграционную шину слияния системы документирования.

3 Разработка и исследование оптимизированной системы документирования

3.1 Используемые инструментальные средства разработки

В наше время инструментальные средства разработки играют одну из ключевых ролей в создании программного обеспечения. Выбор был обусловлен используемыми инструментариями в компании БФТ. Ниже приведено описание программных средств, которые использовались для разработки интеграционной шины.

Язык программирования. Java – является языком программирования, который предназначен для того, чтобы позволить разработчикам приложений «написать один раз, будет работать везде», а это означает, что код, который выполняется на одной платформе не нужно перекомпилировать для работы на другой. Java-приложения обычно компилируется в байт-код (класс-файл), который может работать на любой виртуальной машине Java (JVM) независимо от компьютерной архитектуры [28, 32, 33].

Библиотеки, фреймворки и технологии. Swing – библиотека, предназначенная для создания пользовательского интерфейса для программ на Java. Она включает в себя множество графических компонентов, например кнопки, поля ввода, таблицы, различные панели и т.д. [29].

Apache Maven – фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML [30]. Maven обеспечивает декларативную сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения.

Java Architecture for XML Binding (JAXB) позволяет Java разработчикам ставить в соответствие Java-классы и XML-представления. JAXB предоставляет две основные возможности: маршаллирование Java-объектов в XML и наоборот, то есть демаршализация из XML обратно в Java-объект. Другими

словами, JAXB позволяет хранить и извлекать данные в памяти в любом XML-формате, без необходимости выполнения определённого набора процедур загрузки и сохранения XML.

Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (object-relational mapping – ORM). Данная библиотека предоставляет легкий в использовании каркас (фреймворк) для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных.

Среда разработки. IntelliJ IDEA – программное средство от компании JetBrains. На данный момент оно развивается гораздо быстрее других IDE и поэтому опережает своих конкурентов. Обладает большим количеством функций, которые позволяют быстро писать высококачественный код. Огромное количество плагинов, которые позволяют настроить систему для решения разного рода задач, в том числе и специфических. Новые версии Idea выходят довольно таки часто, что позволяет очень быстро устранить все выявленные недостатки [31].

Описание аннотаций JAVA. Java-аннотация – в языке Java специальная форма синтаксических метаданных, которая может быть добавлена в исходный код. Аннотации используются для анализа кода, компиляции или выполнения [32].

Аннотации представляют из себя дескрипторы, включаемые в текст программы, и используются для хранения метаданных программного кода, необходимых на разных этапах жизненного цикла программы.

Информация, хранимая с соответствующими обработчиками вспомогательных файлов или для маркировки классов, полей и т.д. Выглядит как *@ИмяАннотации*, предваряющее определение переменной, параметра, метода, класса, пакета (описание приведено в таблицах 2 и 3).

Таблица 2 – Описание используемых аннотаций библиотеки Hibernate

@ИмяАннотации	Описание
@Entity	определяет класс, как сущность БД
@Table(name= «NameTable»)	задает имя таблицы, в которой он будет храниться
@Id	говорит о том, что следующий реквизит является ключом для текущего объекта
@GeneratedValue(strategy = GenerationType.AUTO)	используется для того, чтобы Hibernate при создании новых объектов сам генерировал значение ключа
@Column(name= «NameColumn»)	хранения атрибута сущности

Таблица 3 – Описание используемых аннотаций библиотеки JAXB

@ИмяАннотации	Описание
@XmlRootElement	указывает на то, что этот объект может быть «корнем дерева» элементов в XML, т.е. быть элементом самого верхнего уровня, все остальные элементы лежат в нем
@XmlType(name = «cat»)	указывает на то, что класс участвует в JAXB сериализации, в ней же задано имя, которое будет у XML-тега для этого класса
@XmlElement(name)	Ставится около поля. Поле будет представлено в XML-элементом. Позволяет задать имя для тэга
@XmlAttribute(name)	Ставится около поля. Поле будет представлено в XML-атрибутом. Позволяет задать имя для атрибута
@XmlElementWrapper(nillable = true)	Ставится около поля. Позволяет задать «обрамляющий тег» для группы элементов
@XmlJavaTypeAdapter	Ставится около поля. Позволяет задать класс, который будет преобразовывать данные поля в строку

3.2 Описание форматов обмена данными

Все данные в интеграционную шину передаются в виде SOAP-сервисов, в формате xml. В таблицах 4 и 5 приведено описание реквизитного состава для передачи информации о задаче из JIRA в Интеграционную шину и из Интеграционной шины в Help&Manual.

Таблица 4 – Описание реквизитного состава для передачи информации о задаче из JIRA в Интеграционную шину

Тег	Родительский тег	Атрибуты	Описание
rss	–	version	Служебная информация, не используется для интеграции
channel	rss	–	
title	channel	–	
link		–	
description		–	
language		–	
build-info		–	
version	build-info	–	
build-number		–	
build-date		–	
item	channel	–	Информация о задаче
title	item	–	Название задачи
link		–	Ссылка на задачу в Jira
project		id – идентификатор проекта	Информация о проекте, в рамках которого создана задача
		key – ключ проекта	
description		–	Описание задачи
key		id – идентификатор задачи в БД	Ключ задачи
summary		–	Краткое резюме о задачи
type		id – идентификатор типа задачи	Название типа задачи – Документирование
priority		id – идентификатор приоритета задачи	Приоритет задачи
status		id – идентификатор статуса задачи	Статус задачи
resolution		id – идентификатор резолюции по задаче	Резолюция по задаче
assignee		username – логин исполнителя задачи	Исполнитель задачи
labels		–	Метки

Продолжение таблицы 4

Тег	Родительский тег	Атрибуты	Описание
reporter	item	username – логин автора задачи	Автор задачи
created		–	Дата создания задачи
updated		–	Дата изменения
fixVersion		–	Версия проекта
component		–	Модуль проекта
due		–	Срок по задаче (дата)
watches		–	Количество наблюдателей
timeestimate		seconds – оценка по задаче в секундах	Оценка по задаче в часах
timeoriginalestimate		seconds – время первоначальной оценки в секундах	Время первоначальной оценки
timespent		seconds – потраченное время в секундах	Потраченное время в часах
aggregatetimespent		seconds – общее время, потраченное в секундах	Общее время, потраченное
aggregatetimeoriginalestimate		seconds – Первоначальная оценка совокупного времени в секундах	Первоначальная оценка совокупного времени
aggregatetimeremainingestimate		seconds – оставшееся время в секундах	Оставшееся время в часах
comments	item	–	Комментарии
comment	comments	id – идентификатор комментария author – логин автора комментария created – дата создания комментария	Описание одного комментария
attachments	item	–	Прикрепленные файлы

Окончание таблицы 4

Тег	Родительский тег	Атрибуты	Описание
attachment	attachments	id – идентификатор файла в БД name – название файла с расширением size – размер файла author – логин автора файла created – дата создания	Информация о файле
subtasks	item	–	Подзадачи
subtask	subtasks	id – идентификатор подзадачи	Описание одной подзадачи
customfields	item	–	Пользовательские поля
customfield	customfields	id – идентификатор в БД	Описание пользовательского поля
customfieldname	customfield	–	Название поля
customfieldvalues	customfield	–	Значение поля

Таблица 5 – Описание реквизитного состава для передачи информации о задаче из Интеграционной шины в Help&Manual

Тег	Родительский тег	Атрибуты	Описание
task	–	id – идентификатор задачи в БД	Описание задачи
link	task	–	Ссылка на задачу в Jira
name		–	Название задачи без номера
description		–	Полное описание задачи
number		–	Ключ задачи (номер)
title		–	Название задачи с номером
projects		–	Проекты
project	projects	–	Описание проекта
id		–	Идентификатор проекта в БД
key		–	Ключ проекта

Продолжение таблицы 5

Тег	Родительский тег	Атрибуты	Описание
name	projects	–	Название проекта
fixVersion		–	Версия проекта
module		–	Модуль проекта
typetask	task	–	Тип задачи
id	typetask	–	Идентификатор типа задачи
name		–	Название типа задачи
prioritytask	task	–	Приоритет задачи
id	prioritytask	–	Идентификатор приоритета задачи
name		–	Название приоритета задачи
resolution	task	–	Резолюция по задаче
id	resolution	–	Идентификатор в БД
name		–	Название резолюции
creator	task	–	Создатель задачи
username	creator	–	Логин создателя
name		–	ФИО создателя
executor	task	–	Исполнитель задачи
username	executor	–	Логин исполнителя
name		–	ФИО исполнителя
datecreate	task	–	Дата создания
dateend	task	–	Срок по задаче, дата
time	task	–	Оценка по задаче
value	time	–	Оценка по задаче в часах
seconds		–	Оценка по задаче в секундах
timeleft	task	–	Оставшееся время
value	timeleft	–	Оставшееся время в часах
seconds		–	Оставшееся время в секундах

В таблице 6 приведен маппинг полей при передаче данных из Jira в Help&Manual.

Таблица 6 – Маппинг полей при передаче данных из Jira в Help&Manual

Поле в Help&Manual	Поле в Jira	Описание
тег task	тег item	Описание по задаче
тег link	тег link	Ссылка на задачу в Jira
тег name	тег summary	Название задачи без номера
тег description	тег description	Описание задачи
тег number	тег key	Ключ задачи
тег title	тег title	Название задачи с номером
тег projects	–	Добавляется в интеграционной шине
тег project	тег project	Описание проекта
тег id	атрибут id, тега project	Идентификатор проекта
тег key	атрибут key, тега project	Ключ проекта
тег name	Значение тега project	Название проекта
тег fixVersion	тег fixVersion	Версия проекта
тег module	тег component	Модуль проекта
тег typetask	тег type	Тип задачи
тег id	атрибут id, тега type	Идентификатор типа задачи
тег name	Значение тега type	Название типа задачи
тег prioritytask	priority	Приоритет задачи
тег id	атрибут id, тега priority	Идентификатор приоритета задачи
тег name	Значение тега priority	Название приоритета задачи
тег resolution	resolution	Резолюция по задаче
тег id	атрибут id, тега resolution	Идентификатор в БД
тег name	Значение тега resolution	Название резолюции
тег creator	reporter	Создатель задачи
тег username	атрибут username, тега reporter	Логин создателя
тег name	Значение тега reporter	ФИО создателя
тег executor	assignee	Исполнитель задачи
тег username	атрибут username, тега assignee	Логин исполнителя
тег name	Значение тега assignee	ФИО исполнителя

Продолжение таблицы 6

Поле в Help&Manual	Поле в Jira	Описание
тег datecreate	created	Дата создания
тег dateend	due	Срок по задаче, дата
тег time	timeestimate	Оценка по задаче
тег value	значение тега timeestimate	Оценка по задаче в часах
тег seconds	атрибут seconds, тега timeestimate	Оценка по задаче в секундах
тег timeleft	aggregatetimeremainingestimate	Оставшееся время по задаче
value	значение тега aggregatetimeremainingestimate	Оставшееся время в часах
seconds	атрибут seconds, тега aggregatetimeremainingestimate	Оставшееся время в секундах

В таблицах 7 и 8 приведено описание реквизитного состава для передачи информации об этапе работы над задачей из Help&Manual в Интеграционную шину и из Интеграционной шины в JIRA.

Таблица 7 – Описание реквизитного состава для передачи информации об этапе работы над задачей из Help&Manual в Интеграционную шину

Тег	Родительский тег	Атрибуты	Описание
step	–	id – идентификатор этапа в БД	Этап выполнения задачи
name	step	–	Наименование этапа
datestart		–	Дата начала работы над этапом
dateend		–	Дата окончания работы над этапом
description		–	Полное описание этапа работы
timespent		seconds – потраченное время в секундах	Потраченное время в часах
task		–	Идентификатор задачи в БД

Таблица 8 – Описание реквизитного состава для передачи данных об этапе выполнения задачи из Интеграционной шины в JIRA

Тег	Родительский тег	Атрибуты	Описание
rss	–	version	Служебная информация
channel	rss	–	
title	channel	–	
link		–	
description		–	
language	channel	–	Служебная информация
build-info		–	
version	build-info	–	
build-number		–	
build-date		–	
item	channel	–	Информация о задаче
title	item	–	Название задачи
link		–	Ссылка на задачу в Jira
project		id – идентификатор проекта	Информация о проекте, в рамках которого создана задача
		key – ключ проекта	
key		id – идентификатор задачи в БД	Ключ задачи
summary		–	Краткое резюме о задачи
namestep		–	Название этапа задачи.
descriptionstep		–	Описание этапа задачи
assignee		username – логин исполнителя задачи	Исполнитель этапа задачи
timespent		seconds – потраченное время в секундах	Потраченное время в часах
datestart		–	Дата начала работы над этапом
dateend		–	Дата окончания работы над этапом

В таблице 9 приведен маппинг полей при передаче данных из Help&Manual в Jira.

Таблица 9 – Маппинг полей при передаче данных из Help&Manual в Jira

Поле в Jira	Поле в Help&Manual	Описание
тег rss	Нет соответствия	Служебная информация, достается из БД
атрибут version, тега rss		
тег channel		
тег title		
тег link		
тег description		
тег language		
тег build-info		
тег version		
тег build-number		
тег build-date		
тег item	тег step	Информация о задаче
тег title	Достается из БД по значению тега task	Название задачи
тег link		Ссылка на задачу в Jira
тег project		Информация о проекте, в рамках которого создана задача
атрибут id, тега project		Идентификатор проекта
атрибут key, тега project	Достается из БД по значению тега task	Ключ проекта
тег key	Нет соответствия	Ключ задачи
атрибут id, тега key	Значение тега task	Идентификатор задачи в БД
тег summary	Достается из БД по значению тега task	Краткое резюме о задачи
тег namestep	тег name	Название этапа задачи.
тег descriptionstep	тег description	Описание этапа задачи
тег assignee	Достается из БД по значению тега task	Исполнитель этапа задачи
атрибут username, тега assignee		Логин исполнителя этапа задачи
тег timespent	тег timespent	Потраченное время в часах

Продолжение таблицы 9

Поле в Jira	Поле в Help&Manual	Описание
атрибут seconds, тега timespent	атрибут seconds, тега timespent	Потраченное время в секундах
тег datestart	тег datestart	Дата начала работы над этапом
тег dateend	тег dateend	Дата окончания работы над этапом

В таблицах 10 и 11 приведено описание реквизитного состава для передачи данных о постановке задачи из Confluence в Интеграционную шину и из Интеграционной шины в Help&Manual.

Таблица 10 – Описание реквизитного состава для передачи данных о постановке задачи из Confluence в Интеграционную шину

Тег	Родительский тег	Атрибуты	Описание
technicaltask	–	id – идентификатор в БД	Информация о постановке
name	technicaltask	–	Название постановки
author		username – логин автора постановки	Автор постановки
createdate		–	Дата создания
task		id – идентификатор в БД	Наименование задачи в JIRA
text		–	Постановка задачи

Таблица 11 – Описание реквизитного состава для передачи данных о постановке задаче из Интеграционной шины в Help&Manual

Тег	Родительский тег	Описание
formulationproblem	–	Информация о постановке
key	formulationproblem	Идентификатор постановки в БД
title		Название постановки
creator		Автор постановки
username	creator	Логин автора постановки
date		Дата создания
document	formulationproblem	Постановка
task		Информация о задаче

Продолжение таблицы 11

Тег	Родительский тег	Описание
id	task	Идентификатор задачи
name		Наименование задачи в JIRA

В таблице 12 приведен маппинг полей при передаче данных из Confluence в Help&Manual.

Таблица 12 – Маппинг полей при передаче данных из Confluence в Help&Manual

Поле в Confluence	Поле в Help&Manual	Описание
тег technicaltask	тег formulationproblem	Информация о постановке
атрибут id, тега technicaltask	тег key	Идентификатор постановки в БД
тег name	тег title	Название постановки
тег author	тег creator	Автор постановки
атрибут username, тега author	тег username	Логин автора постановки
тег createdate	тег date	Дата создания
тег text	тег document	Постановка
тег task	тег task	Информация о задаче
атрибут id, тега task	тег id	Идентификатор задачи
Значение тега task	тег name	Наименование задачи в JIRA

В таблицах 13 и 14 приведено описание реквизитного состава для передачи готовой документации из Help&Manual в Интеграционную шину и из Интеграционной шины в Confluence.

Таблица 13 – Описание реквизитного состава для передачи готовой документации из Help&Manual в Интеграционную шину

Тег	Родительский тег	Атрибуты	Описание
documentation	–	–	Информация о документации
id	documentation	–	Идентификатор файла в хранилище
name		link – ссылка на файл	Название файла

Продолжение таблицы 13

Тег	Родительский тег	Атрибуты	Описание
size	documentation	–	Размер файла
author		–	Логин автора файла
created		–	Дата создания

Таблица 14 – Описание реквизитного состава, для передачи готовой документации из Интеграционной шины в Confluence

Тег	Родительский тег	Атрибуты	Описание
document	–	name – идентификатор файла в хранилище size – размер файла created – дата создания	Информация о документе
text	document	–	Текст документа
author	document	–	Логин автора файла

В таблице 15 приведен маппинг полей при передаче данных из Help&Manual в Confluence.

Таблица 15 – Маппинг полей при передаче данных из Help&Manual в Confluence

Поле в Confluence	Поле в Help&Manual	Описание
тег document	тег documentation	Информация о документе
тег text	Получаем из хранилища	Текст документа
тег author	тег author	Логин автора файла
атрибут name, тега document	значение тега name	Идентификатор файла в хранилище
атриут size, тега document	значение тега size	Размер файла
атриут created, тега document	значение тега created	Дата создания

3.3 Описание классов и методов интеграционной шины

Для организации взаимодействия компонентов системы по форматам обмена данными и на основе приведенного мапинга была реализована интеграционная шина посредством вызова сервисов, оркестровки и автоматизированной передачи данных. Реализация данного функционала осуществлялась на языке программирования Java, при помощи ключевых слов и конструкций языка с использованием библиотек. Были спроектированы и реализованы классы интеграционной шины.

Основным классом является – класс Integration. Основные методы класса:

a) Метод IntegrationFromJIRAToHM().

Заголовок: public static void IntegrationFromJIRAToHM();

Назначение: Интеграция данных о задаче из JIRA в Help&Manual.

Входные данные: Нет.

Выходные данные: Нет.

Внутри себя метод использует следующие методы:

1) Метод ImportJira.getDataFromJIRA(Context context).

Заголовок: public ImportJiraObject getDataFromJIRA(Context context);

Назначение: Получение данных о задаче из JIRA.

Входные данные: context – контекст шины.

Выходные данные: данные о задаче.

2) Метод mappingDataFromJIRAToHM(ImportJiraObject importJiraObject).

Заголовок: public ExportHMTaskObject mappingDataFromJIRAToHM (ImportJiraObject importJiraObject);

Назначение: Маппинг данных о задаче из JIRA в HM.

Входные данные: importJiraObject – объект, который хранит в себе информацию о задаче.

Выходные данные: данные о задаче в формате Help&Manual.

3) Метод ExportHM.sendDataTaskToHM(ExportHMTaskObject exportHMTaskObject).

Заголовок: public void sendDataTaskToHM(ExportHMTaskObject exportHMTaskObject);

Назначение: Отправка данных о задаче в Help&Manual.

Входные данные: exportHMTaskObject – данные о задаче в формате HM.

Выходные данные: нет.

b) Метод IntegrationFromHMToJIRA().

Заголовок: public static void IntegrationFromHMToJIRA ();

Назначение: Интеграция данных об этапе задачи из Help&Manual в JIRA.

Входные данные: Нет.

Выходные данные: Нет.

Внутри себя использует следующие методы:

1) Метод ImportHM.getDataStepTaskFromHM(Context context).

Заголовок: public ImportHMStepTaskObject getDataStepTaskFromHM (Context context);

Назначение: Получение данных об этапе задачи из Help&Manual.

Входные данные: context – контекст шины.

Выходные данные: данные об этапе задачи.

2) Метод mappingDataFromHMToJIRA (ImportHMStepTaskObject importHMStepTaskObject).

Заголовок: public ExportJIraObject mappingDataFromHMToJIRA (ImportHMStepTaskObject importHMStepTaskObject);

Назначение: Маппинг данных о задаче из Help&Manual в данные формата JIRA.

Входные данные: importHMStepTaskObject – объект, который хранит в себе информацию об этапе задачи.

Выходные данные: данные о задаче в формате JIRA.

3) Метод ExportJira.sendDataToJira(ExportJIraObject exportJIraObject).

Заголовок: public void sendDataTaskToHM(ExportJIraObject exportJIraObject);

Назначение: Отправка данных об этапе задачи в Jira.

Входные данные: exportJIraObject – данные об этапе задачи в формате Jira.

Выходные данные: нет.

c) Метод IntegrationFromHMToConfluence().

Заголовок: public static void IntegrationFromHMToConfluence();

Назначение: Интеграция документации в Confluence.

Входные данные: Нет.

Выходные данные: Нет.

Внутри себя использует следующие методы:

1) Метод ImportHM.getDocumentatuionFromHM(Context context).

Заголовок: public ImportHMDocumentationObject getDocumentatuionFromHM(Context context);

Назначение: Получение документации из Help&Manual.

Входные данные: context – контекст шины.

Выходные данные: документация.

2) Метод mappingDataFromHMToConfluence (ImportHMDocumentationObject importHMDocumentationObject).

Заголовок: public ExportConfluenceObject mappingDataFromHMToConfluence (ImportHMDocumentationObject importHMDocumentationObject)

Назначение: Маппинг информации о документации из Help&Manual в данные формата Confluence.

Входные данные: importHMDocumentationObject – объект, который хранит в себе документацию.

Выходные данные: документация в формате Confluence.

3) Метод ExportConfluence.sendDataToConfluence (ExportConfluenceObject exportConfluenceObject).

Заголовок: public void sendDataToConfluence (ExportConfluenceObject exportConfluenceObject);

Назначение: Отправка данных об этапе задачи в Jira.

Входные данные: exportConfluenceObject – документация в формате Confluence.

Выходные данные: нет.

d) Метод IntegrationFromConfluenceToHM ()

Заголовок: public static void IntegrationFromConfluenceToHM();

Назначение: Интеграция информации о постановке из Confluence.

Входные данные: Нет.

Выходные данные: Нет.

Внутри себя использует следующие методы:

1) Метод ImportConfluence.getDataFromConfluence(Context context).

Заголовок: public ImportConfluenceObject getDataFromConfluence (Context context);

Назначение: Получение постановки задачи из Confluence.

Входные данные: context – контекст шины.

Выходные данные: Постановка задачи.

2) Метод mappingDataFromConfluenceToHM (ImportConfluenceObject importConfluenceObject).

Заголовок: public ExportFormulationProblemHMObject mappingDataFromConfluenceToHM (ImportConfluenceObject importConfluenceObject)

Назначение: Маппинг информации о постановке задачи из Confluence в данные формата Help&Manual.

Входные данные: importConfluenceObject – объект постановки задачи.

Выходные данные: данные о постановке в формате Help&Manual.

3) Метод ExportHM.sendDataFormulationProblemToHM (ExportConfluenceObject exportConfluenceObject).

Заголовок: public void sendDataToConfluence (ExportFormulationProblemHMObject exportFormulationProblemHMObject);

Назначение: Отправка постановки задачи в Help&Manual.

Входные данные: exportFormulationProblemHMObject – постановка задачи в формате Help&Manual.

Выходные данные: нет.

Помимо перечисленных выше методов класс Integration использует следующие классы:

– Context – хранит контекст шины; например, пул коннектов к БД или к внешним системам, количество подключений;

– ExportHM – используется для передачи данных в Help&Manual;

– ImportHM – используется для получения данных из Help&Manual;

– ExportJira – используется для передачи данных учета времени для каждого из этапов выполнения задачи в JIRA;

– ImportJira – применяется для получения данных о задаче из JIRA;

– ExportConfluence – отправляет данные в Confluence;

– ImportConfluence – получает данные из Confluence;

– ParamsHM – хранит различные системные параметры, которые используется для получения/передачи данных в/из системы Help&Manual;

– ParamsJira – хранит различные системные параметры, которые используется для получения/передачи данных в/из системы Jira;

– ParamsConf – хранит различные системные параметры, которые используется для получения/передачи данных в/из системы Confluence;

Листинг программы интеграционной шины представлен в приложении А.

3.4 Описание работы интеграционной шины в оптимизированной системе документирования

Для выгрузки аналитических данных из Confluence в Help&Manual по связанным страницам в Jira задача переводится на статус «ПРИНЯТ» по действию «Принять в работу». Форма задачи представлена на рисунке 13.

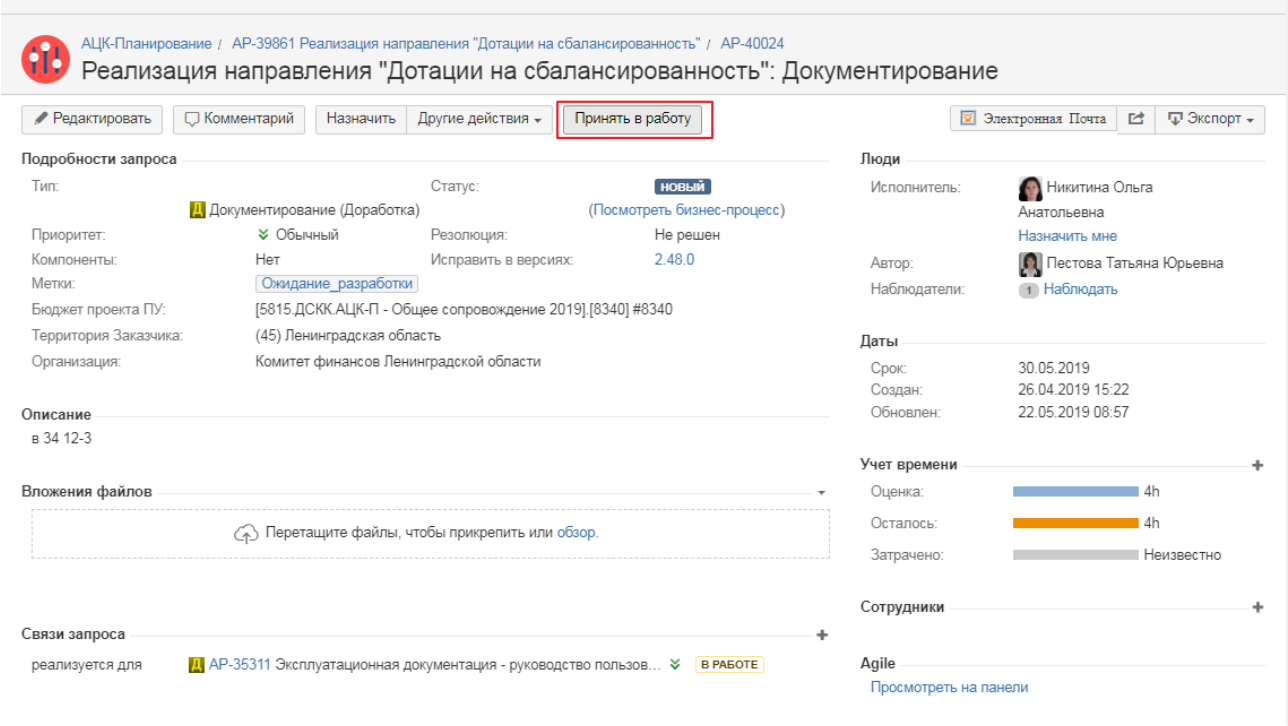

Рисунок 13 – Принятие в работу задачи по документированию

В открывшемся окне «Принять в работу» (представленном на рисунке 14) на закладке «Основное»:

а) устанавливается признак «Начать документирование»;

б) заполняется путь к проекту документации Help&Manual;

в) и нажимается кнопка «Принять в работу».

Аналитические данные из Confluence по соответствующим шаблонам экспорта (форматам обмена данными) через интеграционную шину передаются в xml-формате по указанному пути, с созданием нового проекта документации для Help&Manual.

В свойства проекта документации Help&Manual из Jira выгружается информация по задаче, представленнная на рисунках 15 и 16.

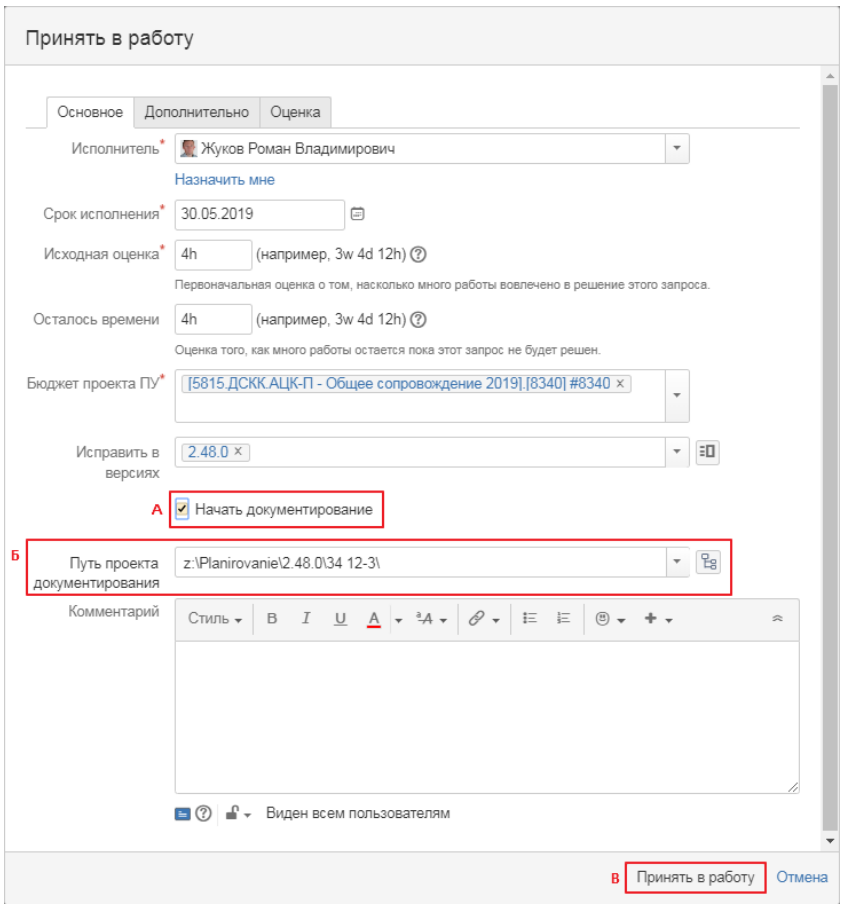

Рисунок 14 – Настройка выгрузки аналитических данных по связанным страницам из Confluence в Help&Manual

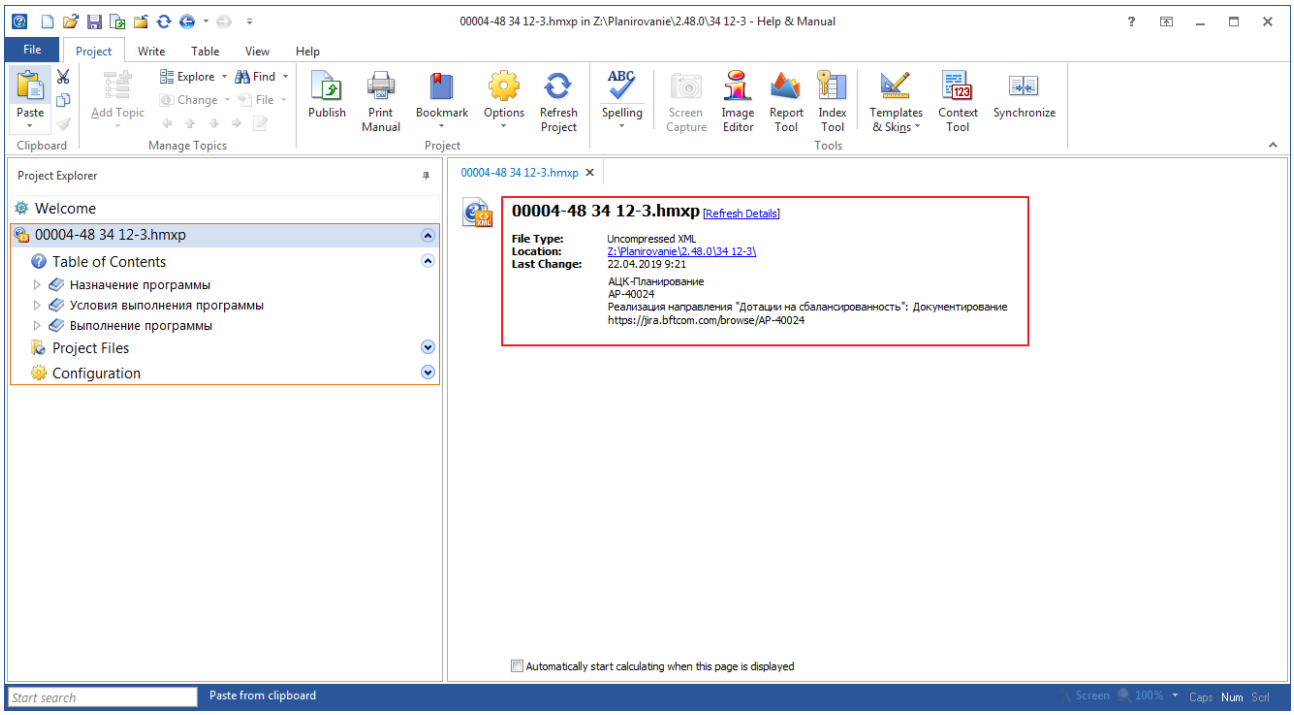

Рисунок 15 – Общая информация по проекту документации

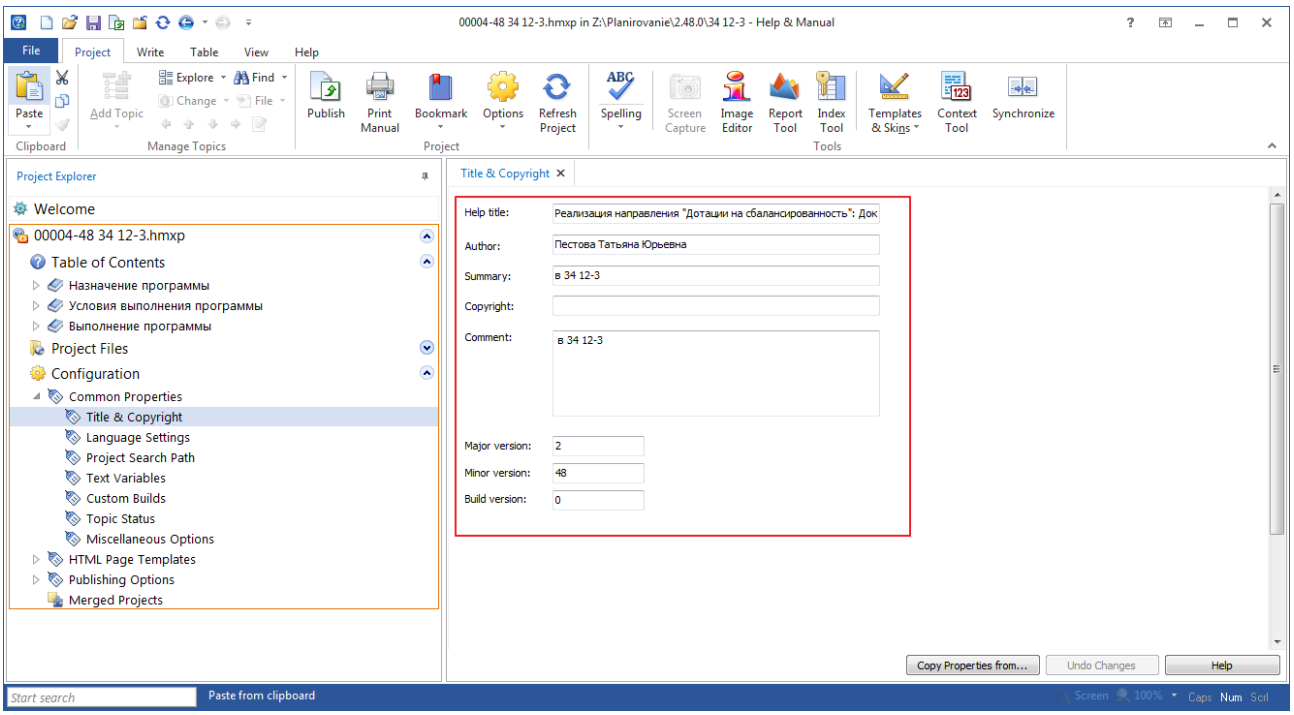

Рисунок 16 – Общие свойства проекта документации

Как только ответственный исполнитель приступает к работе над проектом, Help&Manual отправляет команду в Jira – изменить статус задачи с «ПРИНЯТ» на «В РАБОТЕ» и запустить счетчик времени. Форма задачи представлена на рисунке 17.

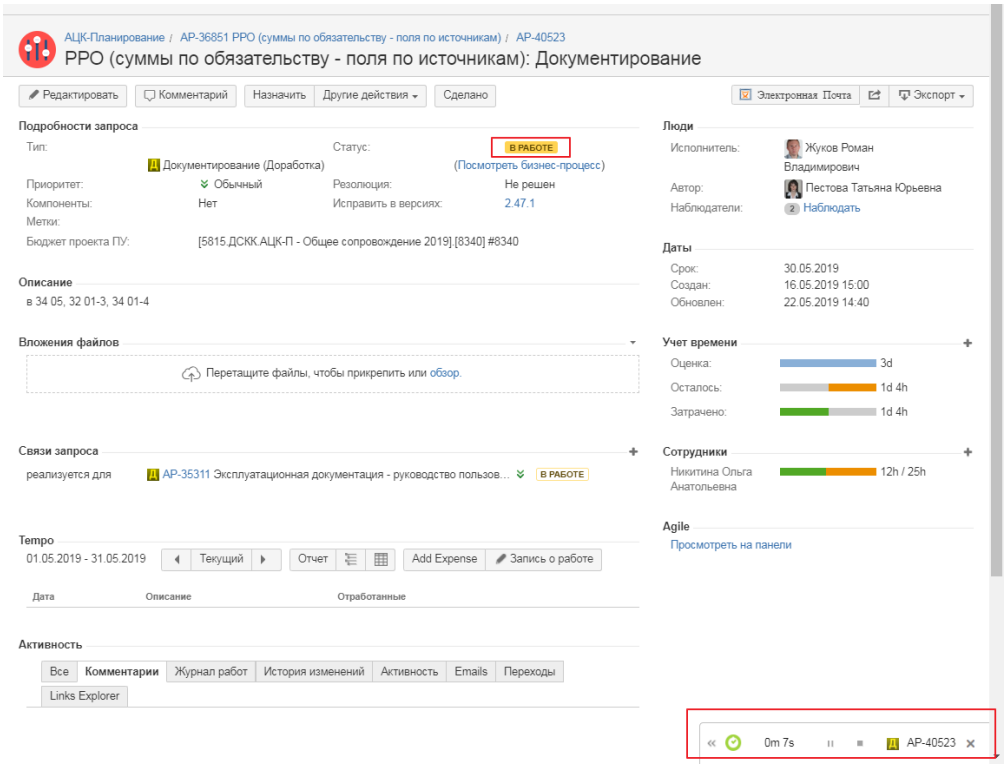

Рисунок 17 – Изменение статуса задачи и запуск счетчика времени

По факту остановки работ над данными (работы окончательно не выполнены), проект документации закрывают в Help&Manual по действию «Close project» (представленном на рисунке 18) счетчик времени по задаче документирования в Jira останавливается, автоматически открывается окно «Запись о работе» (окно представлено на рисунке 19), в котором:

а) автоматически заполняется потраченное время;

б) добавляется описание о проделанной работе;

в) и нажимается кнопка «Запись о работе».

При повторном открытии проекта документации из Help&Manual в Jira отправляется команда – запустить счетчик времени по задаче документирования.

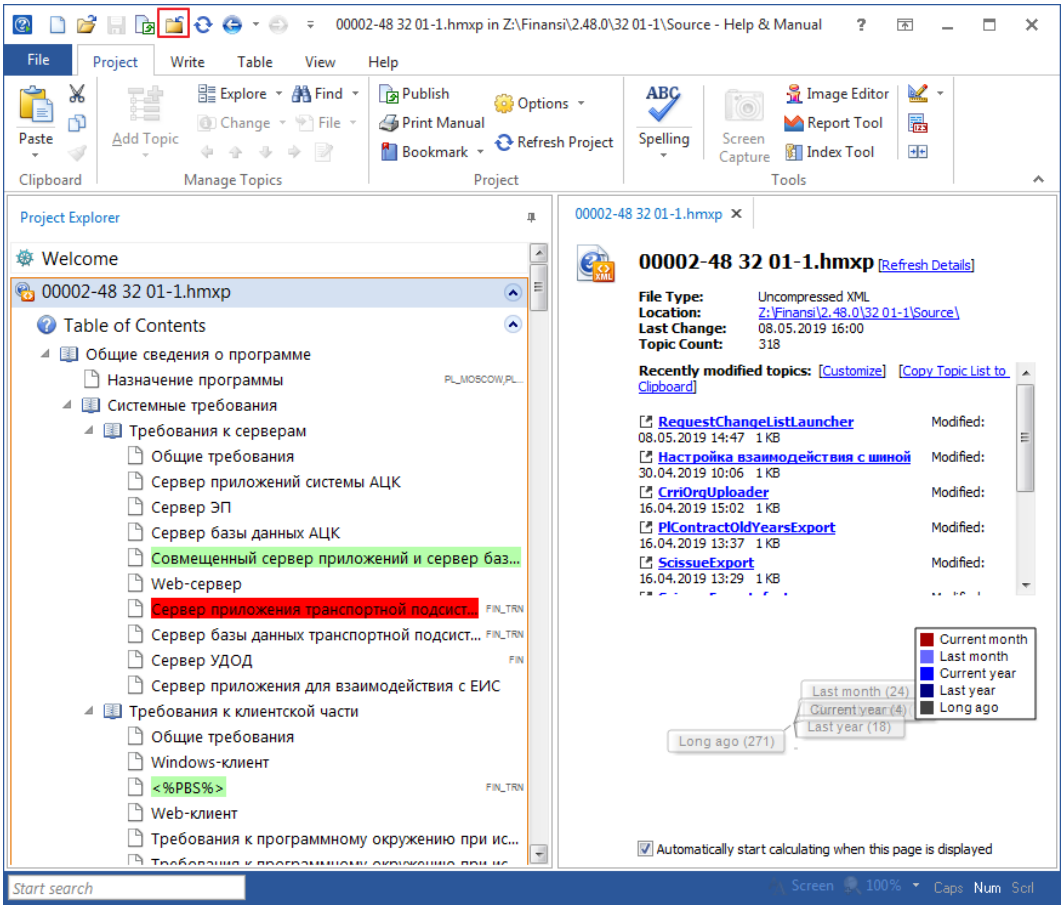

Рисунок 18 – Закрытие проекта документации по действию «Close project»

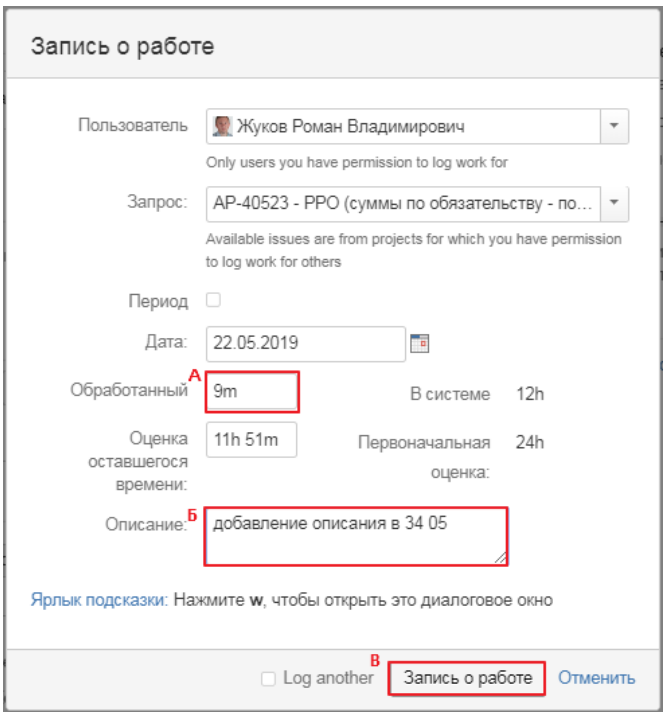

Рисунок 19 – Запись о работе в задаче Jira

При публикации готовых документов Help&Manual отправляет команду в Confluence о создании в пространстве Продукты компании страницы по шаблону выдачи документов с наименованием задачи. На странице, представленной на рисунке 20, указывается информация о готовых документах (спецификация документов приведена в таблице 1), прикрепляются документы, ссылка на задачу.

По факту окончания работ над данными проекта документации в Help&Manual (работы окончательно выполнены) в проекте выполняется действие «Status Project – Completed» (представлено на рисунке 21). Help&Manual в Jira отправляет команду – изменить статус задачи с «В РАБОТЕ» на «ЗАКРЫТ» или «СДЕЛАН» в зависимости от типа задачи документирования.

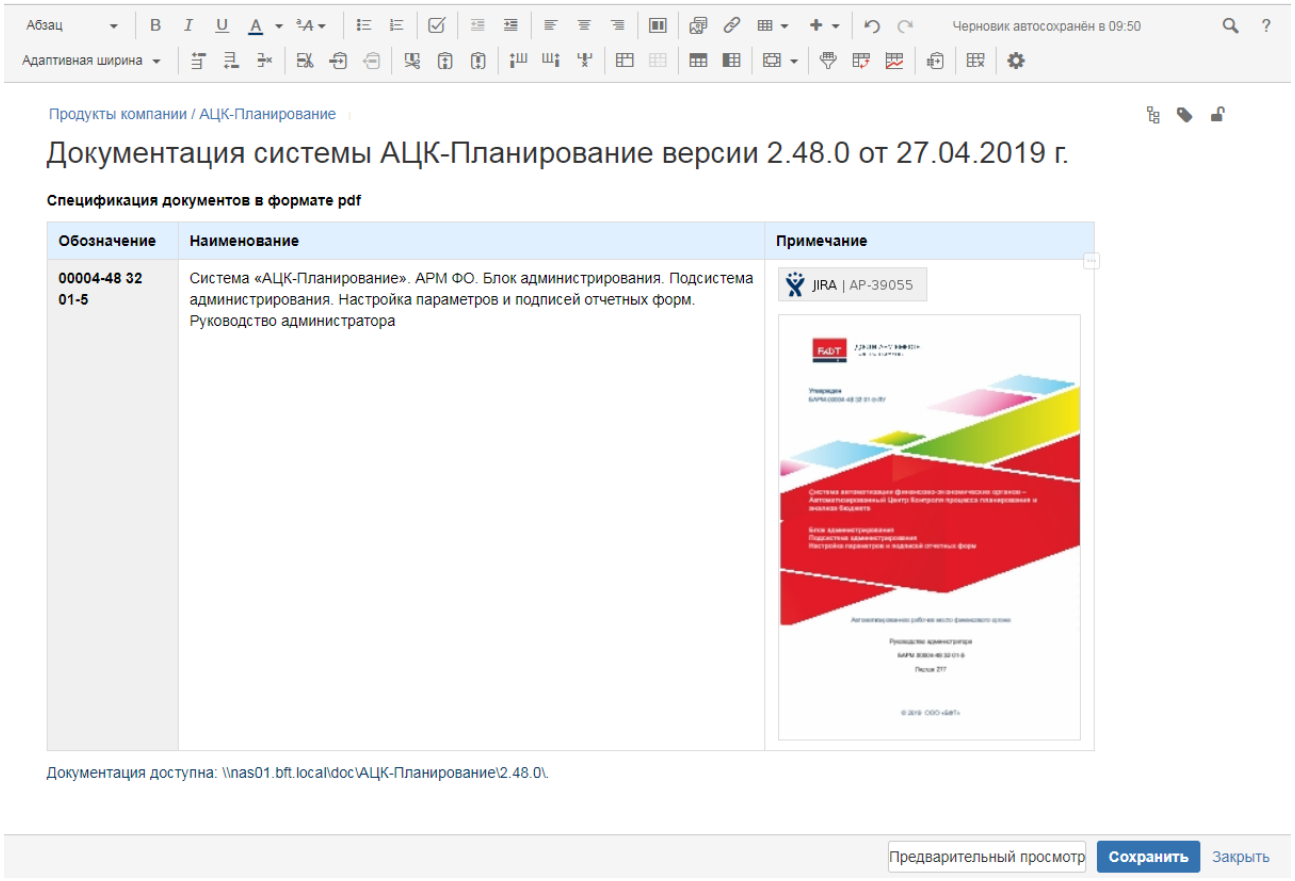

Рисунок 20 – Создание страницы выдачи в Confluence

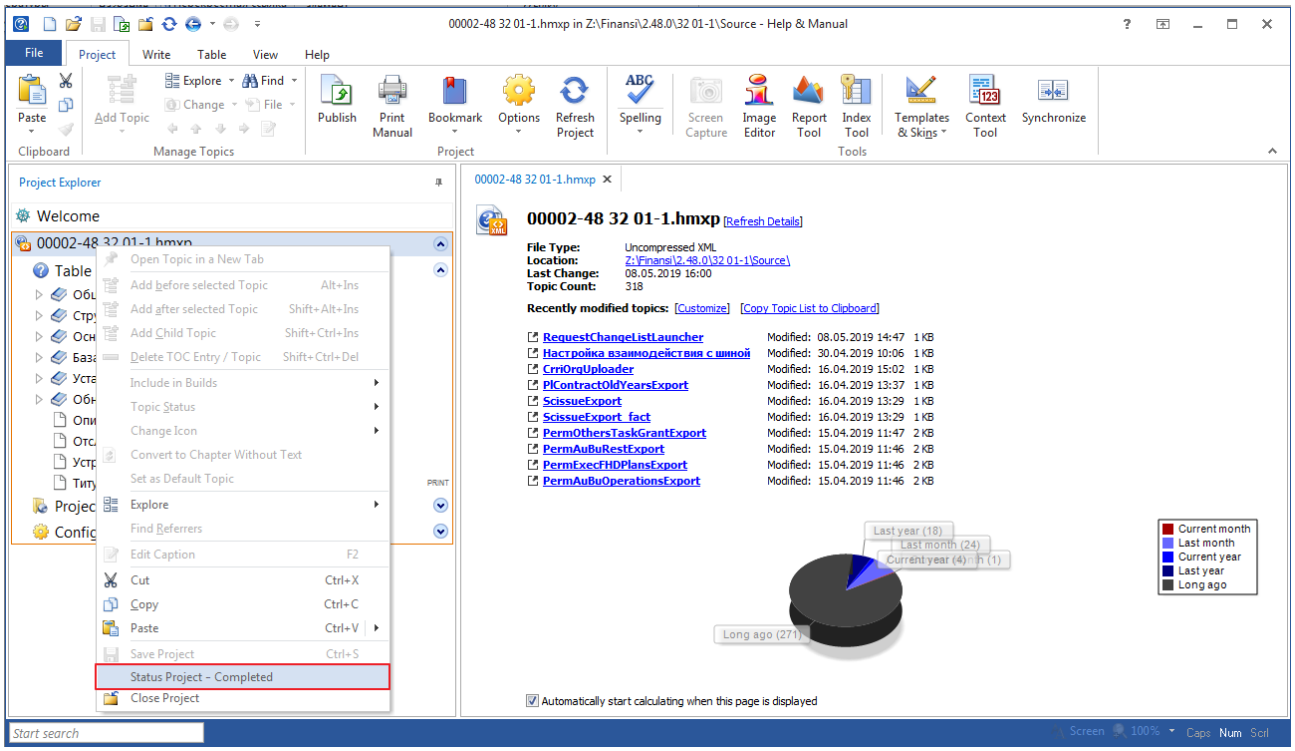

Рисунок 21 – Окончание работ над проектом документации по действию «Status Project – Completed»

3.5 Экономический эффект от оптимизированной системы документирования

В компании БФТ для проверки экономической эффективности оптимизированной системы документирования осуществлен подсчет трудозатрат времени на подготовку документации для типового проекта до оптимизации и после. Комплект типовой документации состоял из следующих видов документов:

- техническое задание;
- общее описание системы;
- пояснительная записка;
- руководство системного программиста;
- руководство оператора.

Общий объем документации на проектах составил порядка 1000 л.

Определив общую трудоемкость документирования за 100%, были подсчитаны количественные показатели трудозатрат на различные виды деятельности в процессе первичной подготовки документации и правки документов с учетом дополнений и исправлений по замечаниям. Результаты анализа, представленные в таблице 16, показывают, что возможность использования описания схожих элементов различных документов (унификации) возросла до 30%.

Перераспределение трудоемкости происходит в область подготовки и правки содержательной части документации с уменьшением трудозатрат на структурирование, оформление и создания документации в дополнительном выходном формате.

Реализованная модель документирования позволила повысить эффективность процесса подготовки документации на программные продукты Компании БФТ, о чем свидетельствует полученная справка, представленная в приложении Б.

Таблица 16 – Эффективность подготовки документации на проект

Количественный показатель при подготовке документации на проект	До оптимизации	После оптимизации
Унифицированное описание в разных типах документов	10%	40%
Трудоемкость первичной разработки документации		
составление содержательной части документации	50%	80%
формирование структуры документов	10%	5%
применение различного оформления	20%	10%
подготовка документации дополнительно в формате справки	20%	5%
Трудоемкость правки документов с учетом дополнений и исправлений по замечаниям		
изменение содержательной части документации	30%	80%
изменение структуры документа	25%	5%
применение иного оформления	25%	10%
внесение правок в документацию в формате справки	20%	5%

Выводы по третьему разделу

В данном разделе приведено описание разработки интеграционной шины слияния системы документирования: специализированного программного обеспечения документирования с единой базой хранения всех корпоративных знаний и централизованной системой учета производственных задач, обеспечивающей возможность использования универсальной модели системы документирования для ИТ-решений в любых прикладных отраслях.

Дано описание используемых инстументальных средств разработки, описаны форматы обмена данными, классы и методы интеграционной шины. Дано описание работы интеграционной шины в оптимизированной системе документирования.

Определен экономический эффект от оптимизированной системы документирования. Возможность использования описания схожих элементов различных документов (унификации) возросла до 30%. Перераспределение трудоемкости происходит в область подготовки и правки содержательной части документации с уменьшением трудозатрат на структурирование, оформление и создания документации в дополнительном выходном формате.

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской диссертации были достигнуты следующие результаты:

а) изучена предметная область – документирование программных продуктов, рассмотрен концептуальный подход документирования программных продуктов, «взаимоувязанность» различных документов на разных стадиях существования программного продукта;

б) исследованы методы подготовки документации: «классический» и по «принципу единого источника»; показано, что для повышения качества документации и экономии трудовых ресурсов при разработке и актуализации различных взаимоувязанных документов необходимо использование «принципа единого источника»;

в) рассмотрены рекомендации по стилю изложения информации для определения степени возможной унификации: структурированности, строгости, детальности, единообразия, однозначности, лаконичности; приведено обоснование их использования; расписаны принципы построения набора стилей для технического документа;

г) рассмотрен процесс подготовки документации в компании БФТ «как есть», описана применяемая в настоящее время модель процесса документирования в компании, базирующаяся на классическом методе составления документации;

д) выявлены существующие недостатки в процессе подготовки документации и используемых инструментариях, для устранения которых проведена оптимизация производственных процессов с переходом на единую базу учета и хранения всех корпоративных знаний, в том числе аналитических данных – Confluence и на централизованную систему учета всех производственных задач – JIRA;

е) разработан и описан процесс учета заданий документирования в едином инструментарии учета всех производственных задач компании,

осуществлен анализ и выбор специализированного программного обеспечения по подготовке документации основывающего на принципе единого источника и включающего полный цикл работы с документами, от написания до формирования готовых документов;

ж) составлена и описана декомпозированная диаграмма в нотации IDEF0 представляющая собой модель процесса подготовки документации после оптимизации: с переструктурированием и сокращением бизнес-процессов с восьми до пяти;

з) разработана модель системы документирования, базирующаяся на принципе единого источника, способная в реальном времени и с минимальными трудозатратами предоставлять актуализированную документацию на любой стадии существования информационных продуктов;

и) осуществлена разработка интеграционной шины слияния оптимизированной системы документирования: специализированного программного обеспечения документирования с единой базой хранения всех корпоративных знаний и централизованной системой учета производственных задач, обеспечивающей возможность использования универсальной модели системы документирования для ИТ-решений в любых прикладных отраслях;

к) определен экономический эффект от использования оптимизированной системы документирования, показывающий сокращение трудозатрат на подготовку и поддержание в актуальном состоянии документации систем;

л) осуществлено внедрение оптимизированной модели системы документирования информационных систем в компании БФТ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.201–89 Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем. – М. : ИПК Издательство стандартов, 2002. – 9 с.
2. ГОСТ 34.003–90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения. – М. : Стандартинформ, 2009. – 15 с.
3. ГОСТ 34.602–89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – М. : Стандартинформ, 2009. – 11 с.
4. ГОСТ 2.104–2006 Единая система конструкторской документации. Основные надписи. – М. : Стандартинформ, 2007. – 15 с.
5. ГОСТ 2.105–95 Единая система конструкторской документации. Общие требования к текстовым документам. – М. : Межгосударственный совет по стандартизации, метрологии и сертификации, 2006. – 17 с.
6. ГОСТ 7.1–2003 Библиографическая запись. Библиографическое описание: общие требования и правила составления. – М. : Изд-во стандартов, 2005. – 47 с.
7. ГОСТ 7.32–2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М. : Стандартинформ, 2017. – 27 с.
8. ГОСТ Р 7.0.11–2011 Система стандартов по информации, библиотечному и издательскому делу. Диссертация и автореферат диссертации. Структура и правила оформления. – М. : Стандартинформ, 2012. – 11 с.
9. Жуков, Р.В. Концепция автоматизации процесса подготовки документации на информационные системы / Р.В. Жуков, И.В. Жуков// Сборник докладов XV Ежегодной Международной научно-технической

конференции: IT-технологии: развитие и приложения. – Владикавказ, 2018. – С. 277-283.

10. ГОСТ 19.701–90 Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения – М. : Стандартинформ, 2010. – 158 с.

11. ГОСТ 19.102–77 Единая система программной документации. Стадии разработки. – М. : Стандартинформ, 2010. – 28 с.

12. Муромцев, В.В. Проектирование информационных систем: Учебное пособие для студентов вузов заочной формы обучения по спец. 010502 "Прикладная информатика в экономике" / В.В. Муромцев. – Белгород : Изд-во БелГУ, 2009. – 160 с.

13. Даншина, С.Ю. Технология автоматизированного документирования разработки радиоэлектронной аппаратуры / С.Ю. Даншина, А.В. Денисенко // Открытые информационные и компьютерные интегрированные технологии. – 2012. – №57. – С. 176-186.

14. Красовский, С.П. Средства вопросно-ответного документирования в проектировании автоматизированных систем: диссертация кандидата технических наук: 05.13.12 / Красовский Сергей Павлович; [Место защиты: Ульян. гос. техн. ун-т]. – Ульяновск, 2008. – 192 с.

15. Дмитриев, П.И. Методы и средства управления знаниями в базовых процессах жизненного цикла программных средств: диссертация кандидата технических наук: 05.02.23 / Дмитриев Павел Игоревич; [Место защиты: С.-Петерб. гос. ун-т аэрокосм. приборостроения]. – Санкт-Петербург, 2014. – 242 с.

16. Вершинина, Л.П. CASE-средства поддержки процесса документирования в жизненном цикле программных средств / Л.П. Вершинина, П.И. Дмитриев // Качество. Инновации. Образование. – 2015. – №7. – С. 54-58.

17. Вершинина, Л.П. Внедрение процесса управления знаниями в базовые процессы жизненного цикла программных средств / Л.П. Вершинина, П.И. Дмитриев // Качество. Инновации. Образование. – 2012. – №10. – С. 24-28.

18. Дмитриев П.И. Повышение качества и эффективности процессов жизненного цикла программных средств на основе методов и средств управления знаниями // Качество. Инновации. Образование. – 2013. – №09. – С. 62-66.

19. Липаев В.В. Документирование сложных программных средств. – М. : СИНТЕГ, 2005. – 216 с.

20. Стиль изложения документации пользователя программного средства. [Электронный ресурс]. – Режим доступа : http://docplayer.ru/271717-Stil-izlozheniya-dokumentacii-polzovatelya-programmnogo-sredstva.html

21. ГОСТ Р ИСО/МЭК 15910–2002 Информационная технология. Процесс создания документации пользователя программного средства. – М. : ИПК Издательство стандартов, 2002. – 44 с.

22. ГОСТ Р ИСО/МЭК 12119–2000 Информационная технология. Пакеты программ. Требования к качеству и тестирование. – М. : ИПК Издательство стандартов, 2002. – 16 с.

23. Острогорский, М. Рекомендации по системе стилей в шаблонах для технической документации. [Электронный ресурс]. – Режим доступа : https://philosoft-services.com/content/ru/articles/wordstyles.html

24. Грекул, В. И. Проектирование информационных систем: учебник и практикум для академического бакалавриата / В. И. Грекул, Н. Л. Коровкина, Г. А. Левочкина. – М. : Издательство Юрайт, 2017. – 385 с.

25. компания БФТ – лучший поставщик в сфере информационных технологий 2017 года. [Электронный ресурс]. – Режим доступа : https://www.securitylab.ru/blog/company/bftcom/343951.php

26. Анненков, И.С. Управление процессами и знаниями с помощью Atlassian JIRA / И.С. Анненков // Вестник ЮРГТУ (НПИ). – Новочеркасск, 2013. – Выпуск №6 – С. 63-68.

27. Общие требования к технической текстовой документации. [Электронный ресурс]. – Режим доступа : http://docplayer.ru/27837599-Standart-predpriyatiya-sistema-menedzhmenta-kachestva.html

28. Шилдт, Герберт. Ш57 Java 8. Полное руководство; 9-е изд. : Пер. с англ. – М. : ООО "И.Д. Вильямс", 2015. – 1376 с.

29. Документация Java – Java™ Platform, Standard Edition 7 API Specification. [Электронный ресурс]. – Режим доступа : https://docs.oracle.com/javase/7/docs/api/

30. Apache Maven Projects. [Электронный ресурс]. – Режим доступа : http://maven.apache.org/guides/

31. Давыдов С., Ефимов А. IntelliJ IDEA. Профессиональное програмирование на Java. – БХВ-Петербург, 2005 – 800 с.

32. Роберт Седжвик, Кевин Уэйн. "Алгоритмы на Java", 4-е изд. : Пер. с англ. – М. : ООО "И.Д. Вильямс", 2013. – 848 с.

33. Хорстманн, К. Хорстманн и Г. Корнелл. JAVA2. Том 2. Тонкости программирования. Пер. с англ. – М. : ООО «Вильямс», 2003 – 1120 с.

34. SOA – Service-Oriented Architecture. [Электронный ресурс]. – Режим доступа : http://sewiki.ru/SOA

ПРИЛОЖЕНИЕ А

Листинг программы

```
public class ExportConfluence {
  public static final String MARKER_RA = "RA";
  public static final String DOC_NUMBER = "DOC_NUMBER";
  public static final String MARKER_RAST = "RAST";
  @Override
  protected int getExportDocumentClass() {
    return DocumentCodes.PAYMENTADMINREG;
  }
  @Override
  protected String getMarker() {
    return MARKER_RA;
  }
  @Override
  protected Timestamp getExportDocDate() {
    return getDataStore().getTimestamp("DATE_RA");
  }
  @Override
  protected String getExportDocNumber() {
    return getDataStore().getStringSafe("NUM_DOC");
  }
  @Override
  protected String getObjectDescription() {
    return "Реестр администрируемых платежей";
  }
  @Override
  protected StringBuilder getFillTO(FieldDataStore ds, ArrayList<SchemaItemInfo> objList)
throws SQLException {
    StringBuilder sb = new StringBuilder(MARKER_TO).append(DELIM_FK);
    Long toOrgId = CommonMethod.getTofkId("fk.export.ra", getSofitContext().getContext());
    OrgDict.Org toOrg = toOrgId != null ? OrgDict.getInstance().findByID(toOrgId) : null;
    for (final SchemaItemInfo obj : objList) {
```

```
    switch (obj.getAttrName()) {
      case "KOD_TOFK":
        appendValue(sb, obj, toOrg != null ? toOrg.name : "");
        break;
      case "NAME_TOFK":
        appendValue(sb, obj, toOrg != null ? toOrg.description : "");
        break;
      default:
        appendValue(sb, obj, "");
    }
  }
  return sb;
}
@Override
protected StringBuilder getFillData(Long docId) throws SQLException {
  StringBuilder data = new StringBuilder();
  data.append(getRA(getDataStore()));
  try (PreparedStatement ps = getSofitContext().getContext().prepareStatement(getLineSQL())) {
    ps.setLong(1, docId);
    ResultSet rsLine = ps.executeQuery();
    while (rsLine.next()) {
      data.append(CR).append(getRAST(rsLine));
    }
  }
  return data;
}
@Override
protected FieldDataStore createDataStore(Long docId) throws SQLException {
  try (PreparedStatement ps = getSofitContext().getContext().prepareStatement(getHeadSQL())) {
    ps.setLong(1, docId);
    ResultSet rs = ps.executeQuery();
     if (rs.next()) {
       return new FieldDataStore(rs);
     }
  }
```

```
  return null;
}
private String getRA(FieldDataStore ds) throws SQLException {
  StringBuilder dataRA = new StringBuilder(MARKER_RA).append(DELIM_FK);
  ArrayList<SchemaItemInfo> schemaItems = schemaItemInfoList.get(MARKER_RA);
  for (SchemaItemInfo oneItem : schemaItems) {
    switch (oneItem.getAttrName()) {
      case "GUID_FK":
        appendValue(dataRA, oneItem, "");
        break;
      case "NUM_DOC":
        String numDoc = StringTools.getTrimValue(ds.getString("NUM_DOC"));
        setSofitDocNumber(numDoc);
        appendValue(dataRA, oneItem, numDoc);
        break;
      case "DATE_RA":
        appendValue(dataRA, oneItem, ds.getTimestampStr("DATE_RA"));
        setSofitDocDate(ds.getTimestamp("DATE_RA"));
        break;
      case "NOM_AKT":
        appendValue(dataRA, oneItem, ds.getString("NOM_ACT"));
        break;
      case "DATE_AKT":
        appendValue(dataRA, oneItem, ds.getString("DATE_ACT"));
        break;
      case "DATE_AKT_BEGIN":
        appendValue(dataRA, oneItem, "");
        break;
      case "KOD_GADB":
        appendValue(dataRA, oneItem, ds.getString("KOD_GADB"));
        break;
      case "NAME_GADB":
        appendValue(dataRA, oneItem, ds.getString("NAME_UBP"));
        break;
      case "KOD_UBP_GADB":
```

```
            appendValue(dataRA, oneItem, ds.getString("KOD_UBP"));
            break;
          case "NAME_UBP_ADB":
            appendValue(dataRA, oneItem, "");
            break;
          case "KOD_UBP_ADB":
            appendValue(dataRA, oneItem, "");
            break;
          case "NAME_BUD":
            appendValue(dataRA, oneItem, ds.getString("NAME_BUD"));
            break;
          case "OKTMO":
            final String oktmo = ds.getString("OKTMO");
            appendValue(dataRA, oneItem, !StringTools.isEmpty(oktmo) && oktmo.length() >= 8 ?
oktmo.substring(0, 8) : oktmo);
            break;
          case "KOD_TOFK":
            appendValue(dataRA, oneItem, ds.getString("KOD_TOFK"));
            break;
          case "NAME_TOFK":
            appendValue(dataRA, oneItem, ds.getString("NAME_TOFK"));
            break;
          case "DOL_RUK":
            appendValue(dataRA, oneItem, ds.getString("DOL_RUK"));
            break;
          case "NAME_RUK":
            appendValue(dataRA, oneItem, ds.getString("NAME_RUK"));
            break;
          case "DOL_ISP":
            appendValue(dataRA, oneItem, ds.getString("DOL_ISP"));
            break;
          case "NAME_ISP":
            appendValue(dataRA, oneItem, ds.getString("NAME_ISP"));
            break;
          case "TEL_ISP":
```

```
        appendValue(dataRA, oneItem, ds.getString("TEL_ISP"));
        break;
      case "DATE_POD":
        appendValue(dataRA, oneItem, ds.getTimestampStr("DATE_POD"));
        break;
      case "DOL_RUK_TOFK":
      case "NAME_RUK_TOFK":
      case "DOL_ISP_TOFK":
      case "NAME_ISP_TOFK":
      case "TEL_ISP_TOFK":
      case "DATE_TOFK":
        appendValue(dataRA, oneItem, "");
        break;
     }
    }
    return dataRA.toString();
  }
  private String getStartKbk(ResultSet rs) throws SQLException {
    String startDate = DateTools.date2str(rs.getTimestamp("DATE_START_KBK"),
"dd.MM.yyyy");
    return StringTools.isEmpty(startDate) && rs.getTimestamp("DATE_END_KBK") == null ?
"01.01.2006" : startDate;
  }
  private String getRAST(ResultSet rs) throws SQLException {
    appendInfo("  -=  ОБРАБОТКА БЛОКА " + MARKER_RAST + "  =-  ");
    StringBuilder dataRAST = new StringBuilder(MARKER_RAST).append(DELIM_FK);
    ArrayList<SchemaItemInfo> schemaItems = schemaItemInfoList.get(MARKER_RAST);
    for (SchemaItemInfo oneItem : schemaItems) {
     switch (oneItem.getAttrName()) {
      case "KBK":
        appendValue(dataRAST, oneItem, getKBK(rs));
        break;
      case "NAME_KBK":
        appendValue(dataRAST, oneItem, rs.getString("NAME_KBK"));
        break;
```

```
        case "DATE_START_KBK":
          appendValue(dataRAST, oneItem, getStartKbk(rs));
          break;
        case "DATE_END_KBK":
          appendValue(dataRAST, oneItem,
DateTools.date2str(rs.getTimestamp("DATE_END_KBK"), "dd.MM.yyy"));
          break;
        case "NAME_ADB":
          appendValue(dataRAST, oneItem, rs.getString("NAME_ADB"));
          break;
        case "INN_ADB":
          appendValue(dataRAST, oneItem, rs.getString("INN_ADB"));
          break;
        case "KPP_ADB":
          appendValue(dataRAST, oneItem, rs.getString("KPP_ADB"));
          break;
        case "P_ZACH":
        case "P_VOZVR":
          appendValue(dataRAST, oneItem, "");
          break;
        case "NAME_NPA":
          appendValue(dataRAST, oneItem, rs.getString("NAME_NPA"));
          break;
        case "NOM_NPA":
          appendValue(dataRAST, oneItem, rs.getString("NOM_NPA"));
          break;
        case "DATE_NPA":
          appendValue(dataRAST, oneItem, DateTools.date2str(rs.getTimestamp("DATE_NPA"),
"dd.MM.yyy"));
          break;
      }
    }
    return dataRAST.toString();
  }
  private String getKBK(ResultSet rs) throws SQLException {
```

```
    return rs.getString("kadmd_code").concat(rs.getString("kd_code"));
  }
  protected String getHeadSQL() {
    return "SELECT  budg_lvl.budgtype BUDG_LEVEL, doc.main_admin_id
UBP_ID,doc.main_admin_name NAME_UBP,coalesce(ubp_obd.code_fk, ubp_org.name)
KOD_UBP,doc.ufk_org_code KOD_TOFK,doc.ufk_org_description NAME_TOFK,
doc.doc_number NUM_DOC, doc.doc_date DATE_RA,doc.act_number NOM_ACT,doc.act_date
DATE_ACT,doc.kadmd_code KOD_GADB, budg.caption NAME_BUD, budg_ter.okato
OKTMO,doc.chief_post DOL_RUK,doc.chief_fio NAME_RUK,doc.executor_post
DOL_ISP,doc.executor_fio NAME_ISP,doc.executor_phone TEL_ISP,doc.sign_date DATE_POD
FROM paymentadminreg doc JOIN budget budg on doc.budget_id = budg.id JOIN budglevel
budg_lvl on budg.budglevel_id = budg_lvl.id LEFT JOIN org ubp_org on doc.main_admin_id =
ubp_org.id LEFT JOIN orgbudgetdet ubp_obd on ubp_obd.org_id = ubp_org.id and
ubp_obd.budget_id = doc.budget_id LEFT JOIN territory budg_ter on budg.territory_id =
budg_ter.id WHERE doc.document_id = ? ";
  }
  protected String getLineSQL() {
    return "SELECT line.kadmd_code,line.kd_code,line.kbk_name
NAME_KBK,line.kbk_begin_date DATE_START_KBK,line.kbk_end_date
DATE_END_KBK,line.admin_name NAME_ADB,line.admin_taxcode
INN_ADB,\nline.admin_kpp KPP_ADB,line.act_name NAME_NPA,line.act_number
NOM_NPA,line.act_date DATE_NPA FROM paymentadminreg docJOIN paymentadminregline
line on line.paymentadminreg_id = doc.id WHERE  doc.document_id = ? ";
  }
}
public class ImportConfluence {
  @Override
  public String getSofitDocNumber() {
    return ((TEXPNoticeDO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsNmbrDcmnt();
  }
  @Override
  protected String getSql() {
```

```
    return "select bo.document_id, est_org.id org_id, child_doc.instance_link, trim(bo.doc_number)
from budgorder bo join estimate est on bo.estimate_id = est.id  join org est_org on est.org_id =
est_org.id
left join budget b on b.caption = ? left join orgbudgetdet obd on obd.org_id = est_org.id and
obd.budget_id = b.id left join document child_doc on child_doc.parent_id = bo.document_id and
child_doc.dispstatus_id = 13
where bo.dispstatus_id = 6 and bo.documentclass_id = 258 and bo.doc_date = ? and (obd.code_fk =
? or est_org.name = ?) group by bo.document_id, est_org.id, child_doc.instance_link,
bo.doc_number  having trim(bo.doc_number) = ?";
  }
  @Override
  protected void setPreparedStatementParam(PreparedStatement ps) throws SQLException {
    ps.setString(1, ((TEXPNoticeDO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtNmBdgt());
    ps.setTimestamp(2, new Timestamp(((TEXPNoticeDO) ((TExchangePacket)
schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsDtDcmnt().toGregorianCalendar().getTime().getTime()));
    ps.setString(3, ((TEXPNoticeDO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtCdGRBS());
    ps.setString(4, ((TEXPNoticeDO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtCdGRBS());
    ps.setString(5, ((TEXPNoticeDO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsNmbrDcmnt());
  }
  @Override
  protected void updateDoc() throws SQLException {
    Element updateDocEl = XMLUtils.createElement("SOFIT_DOCUMENT");
    updateDocEl.setAttribute("action", "change_do_number");
    updateDocEl.setAttribute("class", "258");
    updateDocEl.setAttribute("DOCUMENT_ID", String.valueOf(slaveDocId));
    updateDocEl.setAttribute("BO_NUMBER", ((TEXPNoticeDO) ((TExchangePacket)
schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsRegisNmbrDO());
    Database.beginSavepoint(getSofitContext().getContext(), "UPDATE_DOC");
    try {
```

```
      new XMLProcessor().processElementRes(getSofitContext().getContext(), updateDocEl,
XMLUtils.createElement("RESULT"), true);
    } catch (Exception e) {
      Database.undoSavepoint(getSofitContext().getContext(), "UPDATE_DOC");
      appendError("Не удалось изменить документ");
    }
  }
  @Override
  protected String getMarker() {
    return "IQ";
  }
  @Override
  protected String getGuidFK() {
    return ((TEXPNoticeDO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getInfPackGUID();
  }
  @Override
  public Timestamp getSofitDocDate() {
    return new Timestamp(((TEXPNoticeDO) ((TExchangePacket)
schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsDtIDO().toGregorianCalendar().getTime().getTime());
  }
}
public class ImportConfluenceObject {
  public static HashMap<String, CommonImportXML> cashFileProcessor;
  private String marker;
  private String filePath;
  private Object schemaObject;
  private Element docBody;
  public ImportXMLObject(Element docBody, String marker, String filePath, Object schemaObject)
{
    this.marker = marker;
    this.filePath = filePath;
    this.docBody = docBody;
    this.schemaObject = schemaObject;
```

```
  }
  public Element getDocBody() {
    return docBody;
  }
  public Object getSchemaObject() {
    return schemaObject;
  }
  public String getMarker() {
    return marker;
  }
  public CommonImportXML getProcessor() {
    CommonImportXML fileProcessor;
    if (cashFileProcessor == null) cashFileProcessor = new HashMap<>();
    String marker = getMarker();
    fileProcessor = cashFileProcessor.get(marker);
    if (fileProcessor == null) {
      String filePath = ImportFKProcessor.getInfoFromRegFile(marker,
SofitPlanImport.ATTR_SETTINGS_CLASS_PATH,
ImportFKProcessor.XML_FK_REGISTRATION_SETTINGS);
      try {
        Constructor<?> contructor = Class.forName(filePath).getConstructor();
        fileProcessor = (CommonImportXML) contructor.newInstance();
      } catch (ClassNotFoundException | NoSuchMethodException | InvocationTargetException |
InstantiationException | IllegalAccessException e) {
        e.printStackTrace();
      }
      cashFileProcessor.put(marker, fileProcessor);
    }
    return fileProcessor;
  }
  public String getFilePath() {
    return filePath;
  }
}
public class ImportIJira {
```

```
  @Override
  protected String getMarker() {
    return "IU";
  }
  @Override
  protected String getGuidFK() {
    return ((TEXPNoticeBO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsNmbrDcmnt();
  }
  @Override
  public Timestamp getSofitDocDate() {
    return new Timestamp (((TEXPNoticeBO) ((TExchangePacket)
schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtDtDc().toGregorianCalendar().getTime().getTime());
  }
  @Override
  protected void updateDoc() throws SQLException {
    Element updateDocEl = XMLUtils.createElement("SOFIT_DOCUMENT");
    updateDocEl.setAttribute("action", "change_bo_number");
    updateDocEl.setAttribute("class", "10");
    updateDocEl.setAttribute("DOCUMENT_ID", String.valueOf(slaveDocId));
    updateDocEl.setAttribute("BO_NUMBER", ((TEXPNoticeBO) ((TExchangePacket)
schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsRegisNmbrBO());
    Database.beginSavepoint(getSofitContext().getContext(), "UPDATE_DOC");
    try {
      new XMLProcessor().processElementRes(getSofitContext().getContext(), updateDocEl,
XMLUtils.createElement("RESULT"), true);
    } catch (Exception e) {
      Database.undoSavepoint(getSofitContext().getContext(), "UPDATE_DOC");
      appendError("Не удалось изменить документ");
    }
  }
  @Override
  public String getSofitDocNumber() {
```

```
    return ((TEXPNoticeBO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsNmbrDcmnt();
  }
  @Override
  protected String getSql() {
    return "select bo.document_id, \n" +
        "est_org.id org_id, \n" +
        "child_doc.instance_link, \n" +
        "trim(bo.doc_number) \n" +
        " from budgorder bo \n" +
        "join estimate est on bo.estimate_id = est.id \n" +
        "join org est_org on est.org_id = est_org.id \n" +
        "left join budget b on b.caption = ? \n" +
        "left join orgbudgetdet obd on obd.org_id = est_org.id and obd.budget_id = b.id \n" +
        "left join document child_doc on child_doc.parent_id = bo.document_id and
child_doc.dispstatus_id = 13 \n" +
        "where bo.dispstatus_id = 6 \n" +
        "and bo.documentclass_id = 10 \n" +
        "and bo.doc_date = ? \n" +
        "and (obd.code_fk = ? or est_org.name = ?) \n" +
        "group by bo.document_id, est_org.id, child_doc.instance_link, bo.doc_number \n" +
        "having trim(bo.doc_number) = ?";
  }
  @Override
  protected void setPreparedStatementParam(PreparedStatement ps) throws SQLException {
    ps.setString(1, ((TEXPNoticeBO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtNmBdgt());
    ps.setTimestamp(2, new Timestamp(((TEXPNoticeBO) ((TExchangePacket)
schemaObject).getFormularCollection()
        .getFormular()).getInfDcBsDtDcmnt().toGregorianCalendar().getTime().getTime()));
    ps.setString(3, ((TEXPNoticeBO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtCdGRBS());
    ps.setString(4, ((TEXPNoticeBO) ((TExchangePacket) schemaObject).getFormularCollection()
        .getFormular()).getRcpBdjtCdGRBS());
    ps.setString(5, ((TEXPNoticeBO) ((TExchangePacket) schemaObject).getFormularCollection()
```

```
        .getFormular()).getInfDcBsNmbrDcmnt());
    }
}
public class Context {
    private static final Logger log = LoggerFactory.getLogger(Context.class);
    private ContextConnectionPool connectionPool;
    public static final int CONTEXT_POOL_SIZE =
Integer.getInteger("azk.context.subcontextpool.size", 10).intValue();
    private ContextObservers contextObservers = new ContextObservers();
    private boolean readOnly;
    private boolean isSerializable;
    private final boolean isTransaction; // аналог ownsConnection
    private Context parent;
    private Map<String, Object> contextCache = Collections.synchronizedMap(new
HashMap<String, Object>());
    public Session session;
    protected Map cache = Collections.synchronizedMap(new HashMap());
   private Map objects = new TreeMap();
    private ConnectionFactory defaultFactory = Database.getFactory();
    private boolean isClosed;
    private boolean rplMode = false;
    public final Set changedDocClasses = new HashSet();
    private static final String XML_TASK_CACHE_ID = "XML_TASK_CACHE_ID";
    public boolean handlerCall = false;
    private boolean legacySecurity = false;
    private RemoteRequest remoteRequest;
    public final DataObjectChangeLog changeLog = new DataObjectChangeLog(this);
    public RemoteRequest getRemoteRequest() {
      return remoteRequest;
    }
    public void setRemoteRequest(RemoteRequest remoteRequest) {
      this.remoteRequest = remoteRequest;
      handlerCall = remoteRequest != null;
    }
    public String getTaskId() {
```

```
    return remoteRequest == null? null : remoteRequest.getTaskId();
  }
  public Timestamp getCurDate() throws SQLException {
    return BudgetHelper.getCurDate(getConnection(), session.budget_id);
  }
  public boolean isLegacySecurityDisabled() {
    final String appObjName = getAppObjName();
    return handlerCall && !isLegacySecurity() &&
!AppObjFactory.LEGACY_CLIENT_OBJECT.equals(appObjName) &&
!AppObjFactory.TRANSACT_ENGINE.equals(appObjName)  ;
  }
  public boolean isHandlerCall() {
    return handlerCall;
  }
  public boolean isLegacySecurity() {
    return legacySecurity;
  }
  public void setLegacySecurity(boolean enabled) {
    this.legacySecurity = enabled;
  }
  private Context getRootContext() {
    if (parent == null)
      return this;
    else
      return parent.getRootContext();
  }
  public void setDefaultFactory(ConnectionFactory factory) {
    if (this.defaultFactory != factory) {
      this.defaultFactory = factory;
      log.debug("Value of default factory changed. Clone URL is {}", factory.getURL());
    }
  }
  public ConnectionFactory getDefaultFactory() {
    return defaultFactory;
  }
```

```
  public Connection getConnection() throws SQLException {
    return getConnection(defaultFactory);
  }
  public SQLConnection getSQLConnection() throws SQLException {
    return SQLConnection.getSQLConnection(getConnection());
  }
  public Connection getConnection(ConnectionFactory key) throws SQLException {
    Connection result = getConnectionPool().getConnection(key);
    if (readOnly)
      return new ROConnection(result);
    else
      return result;
  }
  public void attach(ContextObserver obj) {
    getRootContext().contextObservers.attach(obj);
  }
  public static Context getSystemContext(boolean readonly, boolean isSerializable)
    throws SQLException, UserException {
    return new Context(SessionManager.getInstance().getSystemSession(), readonly, isSerializable);
  }
  public static Context getSystemContext(ServerRequest request, boolean readonly, boolean
isSerializable)
    throws SQLException, UserException {
    return new Context(request, SessionManager.getInstance().getSystemSession(), readonly,
isSerializable);
  }
  private Context(Context parent, Session session, boolean isReadOnly, boolean isSerializable,
boolean isTransaction) {
    this(parent, session, isReadOnly, isSerializable, isTransaction, null, true);
  }
  private Context(ServerRequest request, Session session, boolean readonly, boolean isSerializable)
{
    this(null, session, readonly, isSerializable, true);
    this.request = request;
  }
```

```
  public Context(ServerRequest request, boolean readonly, boolean isSerializable) {
    this(request, request.getSession(), readonly, isSerializable);
  }
  public Context(Context parent) {
    this(parent, parent.session, false, false, false);
  }
  public Context(Context parent, Session session, boolean readonly, boolean isSerializable) {
    this(parent, session, readonly, isSerializable, false);
  }
  public Context(Context parent, boolean readonly, boolean isSerializable, ConnectionFactory key,
boolean pushInPool) {
    this(parent, parent.session, readonly, isSerializable, false, key, pushInPool);
  }
  private Context(Context parent, Session session, boolean isReadOnly, boolean isSerializable,
boolean isTransaction, ConnectionFactory key, boolean pushInPool) {
    this.parent = parent;
    this.isTransaction = isTransaction;
    this.isSerializable = isSerializable;
    this.readOnly = isReadOnly;
    this.session = session;
    this.isClosed = false;
    if (key != null)
      this.defaultFactory = key;
    if (pushInPool)
      newContext(this);
  }
  public Context(Session session, boolean readonly, boolean isSerializable) throws SQLException {
    this(null, session, readonly, isSerializable, true);
    Context con = Context.getNullableCurrentContext();
    if (con == null) {
      request = new ServerRequest();
      request.getRecordsSnapshot();
    } else
      request = con.getRequest();
  }
```

```
  public final boolean isSerializable() {
    return isSerializable;
  }

  public final boolean isReadOnly() {
    return readOnly;
  }
  public CallableStatement prepareCall(String sql) throws SQLException {
    return getConnection().prepareCall(sql);
  }
  private static final int FETCH_SIZE = Integer.parseInt(System.getProperty("azk.db.fetchsize",
"0"));

  public PreparedStatement prepareStatement(String sql) throws SQLException {
    PreparedStatement preparedStatement = getConnection().prepareStatement(sql);
    if (FETCH_SIZE != 0) {
      preparedStatement.setFetchSize(FETCH_SIZE);
    }
    return preparedStatement;
  }
  public PreparedStatement prepareStatement(String sql,int resultSetType, int resultSetConcurrency)
throws SQLException {
    return getConnection().prepareStatement(sql, resultSetType, resultSetConcurrency);
  }
  public Statement createStatement() throws SQLException {
    return getConnection().createStatement();
  }
    public Statement createStatement(int resultSetType, int resultSetConcurrency) throws
SQLException {
        return getConnection().createStatement(resultSetType, resultSetConcurrency);
    }
  public void close() {
    _close();
  }
  public void closeNested() {
```

```
  if (isTransaction)
    throw new IllegalStateException();
  _close();
}
private void _close() {
  if (isTransaction) {
    getConnectionPool().releaseConnections();
    if (connectionPool != null)
      connectionPool.close();
  }
  isClosed = true;
  removeContext(this);
}
boolean inTransaction = false;
public void setInTransaction(boolean inTransaction) {
  this.getRootContext().inTransaction = inTransaction;
}
public void close(boolean isCommit) throws SQLException, UserException {
  try {
    if (getConnectionPool().inTransaction() || inTransaction) {
      if (isCommit) {
        try {
          commit();
        } catch (SQLException e) {
          rollback();
          throw e;
        } catch (UserException e) {
          rollback();
          throw e;
        } catch (RuntimeException e) {
          rollback();
          throw e;
        }
      } else
        rollback();
```

```
    }
   } finally {
    _close();
   }
  }
  public void setReadOnly(boolean readOnly) {
   if (connectionPool != null)
    throw new RuntimeException("Unable changing readOnly flag until all current connections
closed.");
   this.readOnly = readOnly;
  }
  public void setIsSerializable(boolean isSerializable) {
   if (connectionPool != null)
    throw new RuntimeException("Unable changing isSerializable flag until all current connections
closed.");
   this.isSerializable = isSerializable;
  }
  private void _commit() throws SQLException, UserException {
   DataObjectChangeLog.commitLogData(this);
   if (contextObservers != null)
    contextObservers._commit(this);
   getConnectionPool().releaseConnections();
   if (connectionPool != null)
    connectionPool.commit();
   inTransaction = false;
  }
  public void commit() throws SQLException, UserException {
   if (isTransaction) {
    _commit();
   }
  }
  private void _rollback() throws SQLException {
   DataObjectChangeLog.clearLogData(this);
   if (contextObservers != null)
    contextObservers._rollback();
```

```
  getConnectionPool().releaseConnections();
  if (connectionPool != null)
    connectionPool.rollback();
  inTransaction = false;
  log.warn("Work ROLLED BACK !");
}
public void rollback() throws SQLException {
  if (isTransaction) {
    _rollback();
  }
}
public void write(String source, Throwable e) {
  log.error(source, e);
}
public void trace(String source, String _msg) {
   log.debug(source, _msg);
}
public void trace(String source, LazyValue msg) {
  log.debug(source, msg.getValue().toString());
}
public void hint(String source, String _msg) {
  log.info(source, _msg);
}
public void warning(String source, String _msg) {
  log.warn(source, _msg);
}
public void error(String source, String _msg) {
  log.error(source, _msg);
}
public Dict getDict(String className, Object id) throws UserException, SQLException {
  return DictFactory.getInstance(getRootContext(), className, id);
}
protected void finalize() throws Throwable {
  if (!isClosed) {
    log.warn("Context was not closed explicitly");
```

```
    close(true);
  }
  super.finalize();
}
public void setRplMode() {
  rplMode = true;
}
public boolean getRplMode() {
  return rplMode;
}
public Timestamp parseDate(String strDate) throws SQLException {
  return
    "WORK_DATE".equals(strDate) ? session.work_date :
      "OPER_DATE".equals(strDate) ? getCurDate() :
        DateUtilities.instance().parseDateTime(strDate);
}
public String formatDate(Date date) {
  return DateUtilities.instance().formatUserDateTime(date, true);
}
public interface ContextObserver {
  default void commit() throws SQLException, UserException {};

  default void rollback() throws SQLException, UserException {};

  default void startSavePoint(String savePointName) throws SQLException, UserException {};

  default void rollbackSavePoint(String savePointName) throws SQLException, UserException {};
}
static class ContextObservers {
  private final List contextObservers = new ArrayList();

  void attach(ContextObserver obj) {
    synchronized (this) {
      contextObservers.add(obj);
    }
```

```
  }
  private void _commit(Context con) throws SQLException, UserException {
    synchronized (this) {
      for (Iterator it = contextObservers.iterator(); it.hasNext(); )
        ((ContextObserver) it.next()).commit();
    }
  }
  private void _rollback() {
    synchronized (this) {
      for (Iterator it = contextObservers.iterator(); it.hasNext(); )
        try {
          ((ContextObserver) it.next()).rollback();
        } catch (Exception e) {
          log.error(StringUtils.EMPTY, e);
        }
    }
  }
}
private ContextConnectionPool getConnectionPool() {
  if (isTransaction) {
    if (connectionPool == null) {
      connectionPool = new ContextConnectionPool(this);
    }
    return connectionPool;
  } else
    return parent.getConnectionPool();
}
private static final ThreadLocal currentContext = new ThreadLocal() {
  protected Object initialValue() {
    return new Stack();
  }
};
public static Context getCurrentContext() {
  if (((Stack) (currentContext.get())).empty())
```

```
      throw new IllegalStateException("Метод getCurrentContext не может возвращать пустой
контекст.");
    else
      return (Context) (((Stack) (currentContext.get())).peek());
  }
  public static Context getNullableCurrentContext() {
    if (((Stack) (currentContext.get())).empty())
      return null;
    else
      return (Context) (((Stack) (currentContext.get())).peek());
  }
  public static void newContext(Context con) {
    ((Stack) (currentContext.get())).push(con);
  }
  public static void removeContext(Context con) {
    Stack stack = ((Stack) (currentContext.get()));
    if (!stack.isEmpty() && stack.pop() != con)
      throw new IllegalStateException();
  }
  public ServerRequest getRequest() {
    if (isTransaction) {
      if (request == null)
        request = new ServerRequest(session);
      return request;
    } else
      return parent.getRequest();
  }
  public GenerationSnapShot getRecordsSnapshot() throws SQLException {
    return getRequest().getRecordsSnapshot();
  }
  public TableSnapshot getTableGenSnapshot() throws SQLException {
    return getRequest().getTableSnapshot();
  }
}
public class ExportJira {
```

```
  final static String MARKER_FC = "FC";
  final static String MARKER_FCST = "FCST";

  @Override
  protected String getMarker() {
   return MARKER_FC;
  }
  @Override
  protected String getMarkerLine() {
   return MARKER_FCST;
  }
  @Override
  protected String getReferenceName() {
   return "KCSR";
  }
  @Override
  protected String getObjectDescription() {
   return "Классификаторы целевых статей расходов";
  }
  @Override
  protected void initProcessData() {
   super.initProcessData();
   addNotEmptyListItem(MARKER_FROM + "." + "KOD_UBP");
   addNotEmptyListItem(MARKER_FROM + "." + "NAME_UBP");
   addNotEmptyListItem(MARKER_TO + "." + "KOD_TOFK");
   addNotEmptyListItem(MARKER_TO + "." + "NAME_TOFK");
  }
  @Override
  protected void fillHeadData(FieldDataStore ds, ArrayList<SchemaItemInfo> schemaItems,
StringBuilder data) {
   for (SchemaItemInfo oneItem : schemaItems) {
    switch (oneItem.getAttrName()) {
     case "NAME_BUD":
     case "DOL_ISP":
     case "NAME_ISP":
```

```
        case "TEL_ISP":
          appendValue(data, oneItem, ds.getString(oneItem.getAttrName()));
          break;
        case "DATE_FORM":
          appendValue(data, oneItem, DateTools.date2str(getSofitDocDate()));
          break;
        default:
          appendValue(data, oneItem, "");
      }
    }
  }
  @Override
  protected void fillLineData(FieldDataStore ds, ArrayList<SchemaItemInfo> schemaItems,
StringBuilder data) throws SQLException {
    for (SchemaItemInfo oneItem : schemaItems) {
      switch (oneItem.getAttrName()) {
        case "STATIA":
          appendValue(data, oneItem, ds.getString("CODE"));
          break;
        case "CNAME":
          appendValue(data, oneItem, ds.getString("CAPTION"));
          break;
        case "NAME":
          appendValue(data, oneItem, ds.getString("DESCRIPTION"));
          break;
        case "STATUS":
          appendValue(data, oneItem, "1");
          break;
        case "DATE_BEGIN":
        case "DATE_END":
          appendValue(data, oneItem, DateTools.date2str(ds.getTimestamp(oneItem.getAttrName())));
          break;
        case "DATE_CHANGE":
          appendValue(data, oneItem, DateTools.date2str(ds.getTimestamp(oneItem.getAttrName()),
FORMAT_DATETIME));
```

```
          break;
        default:appendValue(data, oneItem, "");
      }
    }
  }
  @Override
  protected DataBuilder getSQLLine() throws SQLException {
    return getBudgetCodeBuilder(getReferenceName());
  }
  @Override
  protected String getItemTemplateLabel() {
    return "\"%DESCRIPTION%\". Код \"%CODE%\". Период с \"%DATE_BEGIN%\" по
\"%DATE_END%\" (ID справочника \"%ID%\", ID строки справочника \"%LINE_ID%\")";
  }
}
```

ПРИЛОЖЕНИЕ Б

Использование результатов работы

ОБЩЕСТВО
С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«БЮДЖЕТНЫЕ И ФИНАНСОВЫЕ ТЕХНОЛОГИИ»

127018, г. Москва,
ул. Складочная, д. 3, строение 1
+7 (495) 784-70-00,
info@bftcom.com, www.bftcom.com

СПРАВКА

об использовании результатов

дана Жукову Роману Владимировичу в том, что результаты его магистерской диссертации на тему «Модернизация системы документирования информационных систем в Компании БФТ» использованы для модернизации системы документирования в Компании БФТ.

В частности, разработанная Жуковым Р.В., модель документирования позволила повысить эффективность процесса подготовки документации на программные продукты Компании.

Справка дана для предоставления в ГЭК по защите магистерских диссертаций по направлению подготовки 09.04.02 «Информационные системы и технологии».

Директор департамента
(должность руководителя)

Подпись

Гиляян А.М.
(ФИО)

Магистерская диссертационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) (Ф.И.О.)