Реферат

Пояснительная записка содержит 117 страниц, 35 иллюстраций, 18 таблицы и 1 приложение. В приложении приведены исходные тексты основных классов и функций разработанного программного обеспечения.

Графический материал выполнен на 7 листах формата А1.

Ключевые слова: **контроль качества, веб-приложение, 1С:Предприятие, 1С:Управление торговлей, свободное программное обеспечение, веб-сервер, система управления базами данных (СУБД), браузер, тонкий клиент, толстый клиент.**

Целью дипломного проекта является создание информационной системы для отслеживания ошибок, возникающих в процессе разработки программного обеспечения. Приложение разработано на платформе 1С:Предприятие 8.2 в сочетании со стандартной конфигурацией 1С:Управление торговлей. Для работы с приложением на клиентской стороне используется тонкий, толстый клиент 1С:Предприятие или веб-браузер. Серверная часть реализована на платформе 1С:Предприятие, а в качестве веб-сервера – Apache.

Приведено технико-экономическое обоснование проекта, а также рассмотрены вопросы обеспечения безопасности жизнедеятельности человека.

Содержание

Введение

Информационные технологии являются одним из основных элементов инфраструктуры современного общества. Они служат базой для экономической деятельности и социального и культурного развития человечества, обеспечивая людям доступ к огромным массивам разнообразной информации и связывая их друг с другом, где бы они не находились.

Любая информационная система состоит из аппаратного и программного обеспечения (ПО). В начале развития компьютерной техники аппаратная часть была более сложной и значительно более дорогостоящей, стоимость программной части оценивалась примерно в 5% стоимости всей системы. Однако гибкость программного обеспечения и (как оказалось впоследствии, обманчивая) простота внесения в него изменений побуждали использовать его для решения разнообразнейших задач на одном и том же или стандартизированном аппаратном обеспечении. Поэтому постепенно ПО усложнялось, приобретало все большую ценность, и в последние десятилетия его стоимость достигает от 30% до 90% стоимости систем, в зависимости от их типа [6]. Совокупные затраты на создание, развитие и поддержку ПО уже превосходят соответствующие затраты на аппаратное обеспечение [6]. Сложность же современных программных комплексов такова, что многие исследователи считают их самыми сложными системами, созданными человеком [6].

Возрастающая сложность ПО приводит к увеличению количества ошибок в нем, а одновременный рост количества и критичности выполняемых им функций влечет рост ущерба от этих ошибок. Оценки потерь одной экономики США от некачественного программного обеспечения дают около 60 миллиардов долларов в год [6].

При построении систем определенного уровня сложности люди в принципе не могут избежать ошибок, просто потому, что им вообще свойственно ошибаться, а

возрастающая сложность предоставляет все больше возможностей для ошибок, при этом затрудняя их быстрое обнаружение. Для обеспечения корректности и надежности работы таких систем большое значение имеют различные методы верификации и валидации, позволяющие выявлять ошибки на разных этапах разработки и сопровождения ПО, чтобы последовательно устранять их[6].

Таким образом, автоматизация процесса в области контроля качества производства программного обеспечения позволит освободить время специалистов для творческой деятельности и сократить расходы компании.

Актуальность разработки систем контроля качества производства ПО обусловлена следующими преимуществами:

- значительная экономия времени и средств;
- универсальный доступ к учетной системе через браузер;
- накопление и сохранение документов по процессу разработки и контроля качества;
- повышение продуктивности производства.

В настоящем дипломном проекте разработана конфигурация для «1С:Предприятие 8.2» с подсистемой «Качество», которая позволяет формализовать процесс контроля качества ПО и отслеживать уровень качества разработки в разрезе отдельных функциональных групп.

1 Проектно-пояснительная часть

1.1 Организация работы контроля качества производства программного обеспечения и задачи автоматизации

1.1.1 Основные понятия

Под *жизненным циклом программного обеспечения* обычно понимают весь интервал времени от момента зарождения идеи о том, чтобы создать или приобрести программную систему для решения определенных задач, до момента полного прекращения использования последней ее версии. Жизненным циклом этот период назван по аналогии с циклом жизни растения или животного, которое рождается, проходит определенные фазы роста и развития и в итоге погибает, давая жизнь новым существам, проходящим через те же стадии.

Описать общую структуру жизненного цикла произвольной программной системы, по-видимому, невозможно - слишком сильно отличается разработка и развитие ПО, предназначенного для решения разных задач в различных окружениях. Однако можно определить набор понятий, в терминах которых описывается любая такая структура — это, прежде всего, виды деятельности, роли и артефакты.

Вид деятельности в жизненном цикле ПО - это набор действий, направленных на решение одной задачи или группы тесно связанных задач в рамках разработки и сопровождения ПО. Примерами видов деятельности являются анализ предметной области, выделение и описание требований, проектирование, разработка кода, тестирование, управление конфигурациями, развертывание.

Роль в жизненном цикле ПО - это профессиональная специализация людей, участвующих в работах по созданию или сопровождению ПО (или затрагиваемых ими) и имеющих одинаковые интересы или решающих одни и те же задачи по отношению к этому ПО. Примеры ролей: бизнес-аналитик, инженер по требованиям, архитектор, проектировщик пользовательского интерфейса,

программист-кодировщик, технический писатель, тестировщик, руководитель проекта, пользователь, администратор системы.

Артефактами жизненного цикла ПО называются различные информационные сущности, документы и модели, создаваемые или используемые в ходе разработки и сопровождения ПО. Так, артефактами являются техническое задание, описание архитектуры, модель предметной области на каком-либо графическом языке, исходный код, пользовательская документация и т.д. Различные модели, используемые отдельными разработчиками при создании и анализе ПО, но не зафиксированные в виде доступных другим людям документов, не могут считаться артефактами[6].

Тестирование программного обеспечения (software testing) - это процесс анализа или эксплуатации программного обеспечения с целью выявления дефектов. Несмотря на всю простоту этого определения, в нем содержатся пункты, которые требуют дальнейших пояснений. Слово процесс (process) используется для того, чтобы подчеркнуть, что тестирование суть плановая, упорядоченная деятельность.

Согласно этому определению, тестирование предусматривает "анализ" или "эксплуатацию" программного продукта. Тестовая деятельность, связанная с анализом результатов разработки программного обеспечения, называется статическим тестированием (static testing). Статическое тестирование предусматривает проверку программных кодов, сквозной контроль и проверку программы без запуска па машине, т.е. проверку за столом (desk checks). В отличие от этого, тестовая деятельность, предусматривающая эксплуатацию программного продукта, носит название динамического тестирования (dynamic testing). Статическое и динамическое тестирование дополняют друг друга, и каждый из этих типов тестирования реализует собственный подход к выявлению ошибок.

Последний пункт определения, требующий дополнительных пояснений — это понятие *дефекта (bug)*. Говоря простыми словами, программная ошибка— ничто иное, как изъян в разработке программного продукта, который вызывает несоответствие ожидаемых результатов выполнения программного продукта и фактически полученных результатов. Дефект может возникнуть на стадии

кодирования, на стадии формулирования требований или на стадии проектирования, либо же его причина может крыться в некорректной конфигурации или данных. Дефектом может быть также что-то другое, что не соответствует ожиданиям заказчика и что может быть, а может и не быть определено в спецификации программного продукта[14].

1.1.2 Характеристики качества программного обеспечения

Верификация проверяет соответствие одних, создаваемых в ходе разработки и сопровождения ПО, артефактов другим, ранее созданным или используемым в качестве исходных данных, а также соответствие этих артефактов и процессов их разработки правилам и стандартам. В частности, верификация проверяет соответствие между нормами стандартов, описанием требований (техническим заданием) к ПО, проектными решениями, исходным кодом, пользовательской документацией и функционированием самого ПО. Кроме того, проверяется, чтобы требования, проектные решения, документация и код были оформлены в соответствии с нормами и стандартами, принятыми в данной стране, отрасли и организации при разработке ПО, а также - что при их создании выполнялись все указанные в стандартах операции, в нужной последовательности. Обнаруживаемые при верификации ошибки и дефекты являются расхождениями или противоречиями между несколькими из перечисленных документов, между документами и реальной работой программы, между нормами стандартов и реальным процессами разработки и сопровождения ПО. При этом принятие решения о том, какой именно документ подлежит исправлению (может быть, и оба) является отдельной задачей.

Валидация проверяет соответствие любых создаваемых или используемых в ходе разработки и сопровождения ПО артефактов нуждам и потребностям пользователей и заказчиков этого ПО, с учетом законов предметной области и ограничений контекста использования ПО. Эти нужды и потребности чаще всего не зафиксированы документально — при фиксации они превращаются в описание требований, один из артефактов процесса разработки ПО. Поэтому валидация является менее формализованной деятельностью, чем верификация. Она всегда проводится с участием представителей заказчиков, пользователей, бизнес-

аналитиков или экспертов в предметной области — тех, чье мнение можно считать достаточно хорошим выражением реальных нужд и потребностей пользователей, заказчиков и других заинтересованных лиц. Методы ее выполнения часто используют специфические техники выявления знаний и действительных потребностей участников.

Различие между верификацией и валидацией проиллюстрировано на рисунке 1.1.

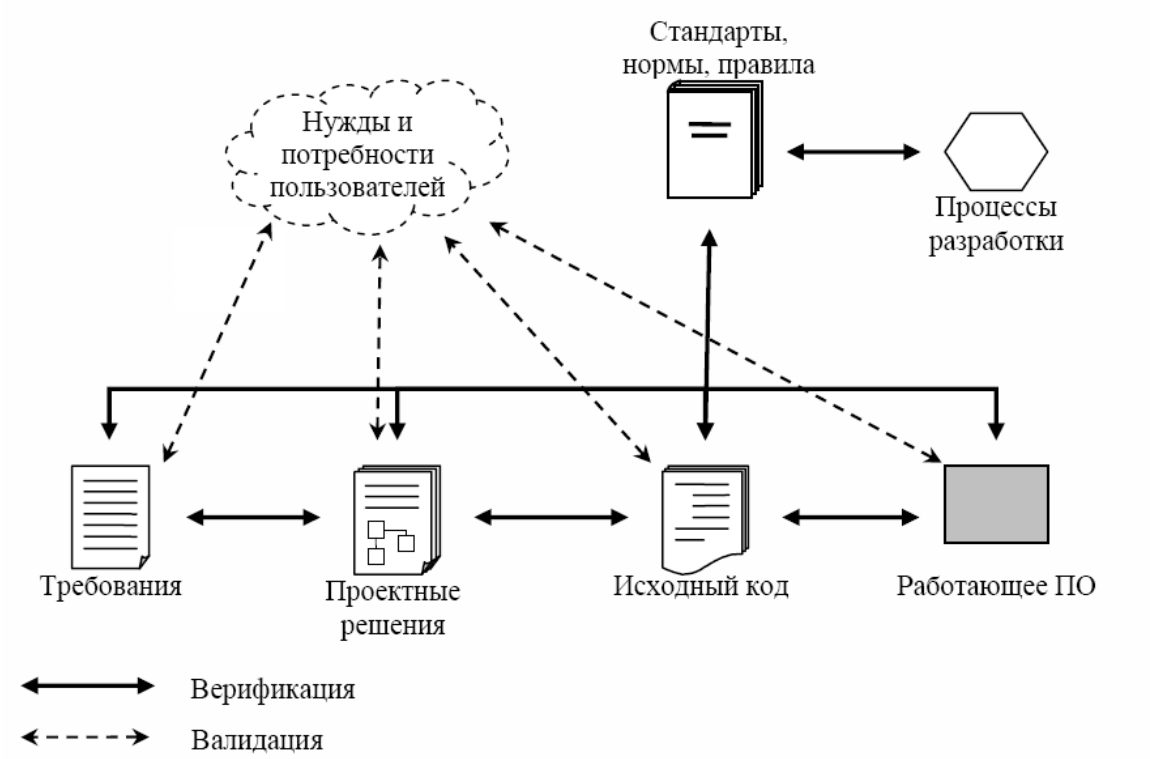

Рисунок 1.1 - Соотношение верификации и валидации

Как видно из рисунка 1.1 верификация и валидация являются видами деятельности, направленными на контроль качества программного обеспечения и обнаружение ошибок в нем. Имея общую цель, они отличаются источниками проверяемых в их ходе свойств, правил и ограничений, нарушение которых считается ошибкой.

Поскольку основной задачей верификации, как и валидации, является контроль качества программного обеспечения, необходимо обратиться к этому понятию. Наиболее широко используется определение качества ПО в виде системы атрибутов

или факторов, которые могут быть оценены с помощью ряда метрик. Такой подход позволяет конструктивно оценивать качество ПО в целом, во всех необходимых аспектах. Ниже описана модель качества ПО, описанная в действующей на сегодняшний день версии этого стандарта ISO 9126, принятой в 2001 году[15]. В ходе идущей переработки этого и других связанных с оценкой качества ПО стандартов по новой схеме SQauRE серьезные изменения в модели качества не планируются.

Стандарт ISO 9126 предлагает учитывать три разных точки зрения при рассмотрении качества ПО: точку зрения разработчиков, которые воспринимают внутреннее качество ПО, точку зрения руководства аттестации ПО на соответствие сформулированным к нему требованиям, в ходе которой определяется внешнее качество ПО, и точку зрения пользователей, ощущающих качество ПО при использовании. Во всех трех случаях для описания качества используется предложенная МакКолом многоуровневая модель, состоящая из целей или факторов, атрибутов или критериев и метрик качества. Цели (факторы) позволяют на верхнем уровне определять основные характеристики, которые ПО должно иметь или уже имеет. Каждый фактор состоит из набора атрибутов (критериев), позволяющих качественно описать желаемые или полученные характеристики более детально. Каждый атрибут поддерживается набором метрик, которые позволяют количественно оценивать наличие соответствующей характеристики.

Для двух точек зрения — внешнего качества и внутреннего качества — в рамках ISO 9126 предложена модель качества, состоящая из 6 факторов и 27 атрибутов и схематически представленная на рисунке 1.2.

Рисунок 1.2 - Факторы и атрибуты внешнего и внутреннего качества ПО по ISO 9126

Факторы и атрибуты качества в этой модели используются для описания качества ПО с точки зрения его разработчиков и их руководства. Для пользовательской точки зрения, т.е. качества ПО при использовании, стандарт ISO 9126 предлагает другую систему факторов:

- Эффективность (effectiveness) - способность решать задачи пользователей с необходимой точностью при использовании в заданном контексте.

- Продуктивность (productivity) - способность предоставлять определенные результаты в рамках ожидаемых затрат ресурсов.

- Безопасность (safety) - способность обеспечивать необходимо низкий уровень риска нанесения ущерба жизни и здоровью людей, бизнесу, собственности или окружающей среде.

- Удовлетворение пользователей (satisfaction) - способность приносить удовлетворение пользователям при использовании в заданном контексте.

В дальнейшем мы будем использовать систему из шести факторов, предназначенную для оценки и описания внешнего или внутреннего качества программного обеспечения[6].

1.1.3 Процесс контроля качества производства программного обеспечения на предприятии

Процесс (process) - это последовательность шагов, в том числе действия, ограничения и ресурсы, которая приводит к желаемому результату определенного рода. Выделим следующие атрибутов процесса:

- в рамках процесса предварительно описываются все основные действия;

- в процессе задействуются ресурсы, с которыми связана некоторая совокупность ограничений (например, расписание), и генерируются промежуточные и конечные результаты;

- процесс может состоять из некоторого числа подпроцессов, связанных между собой определенным образом. Процесс можно определить как некоторую иерархию процессов, организованных так, что каждый подпроцесс описывается собственной моделью процесса;

- с каждым действием процесса связаны критерии входа и выхода, так что известно, когда определенное действие начинается и когда завершается;

- действия выполняются последовательно или параллельно по отношению к другим независимым подроцессам, поэтому легко определить, когда некоторое действие выполняется относительно других действий;

- каждый процесс управляется некоторым набором руководящих принципов, определяющих цели каждого действия;

- к действию, ресурсу или результату могут применяться ограничения или директивы.

Например, бюджет или план накладывает ограничения на промежуток времени, в течение которого выполняется то или иное действие, а некоторое инструментальное средство может накладывать ограничения на способ использования определенного ресурса.

Когда процесс имеет отношение к созданию того или иного продукта, это часто называется жизненным циклом. По той же причине разработка программного продукта называется жизненным циклом программного продукта (software life cycle). Жизненный цикл программного продукта может быть описан разными способами, в то же время для этой цели часто используется модель, которая представляет основные свойства процесса, сопровождаемые определенным сочетанием текста и иллюстраций.

Одной из первых была использована каскадная (или водопадная) модель жизненного цикла программного продукта, показанная на рис. 1.3. Основное свойство каскадной модели заключается в том, что каждая стадия или компонент модели завершается перед началом следующей стадии. Процесс начинается с определения требований к системе. В каскадной модели эти требования выявляются, анализируются и записываются в специальные документы еще до того, как начнутся работы по проектированию.

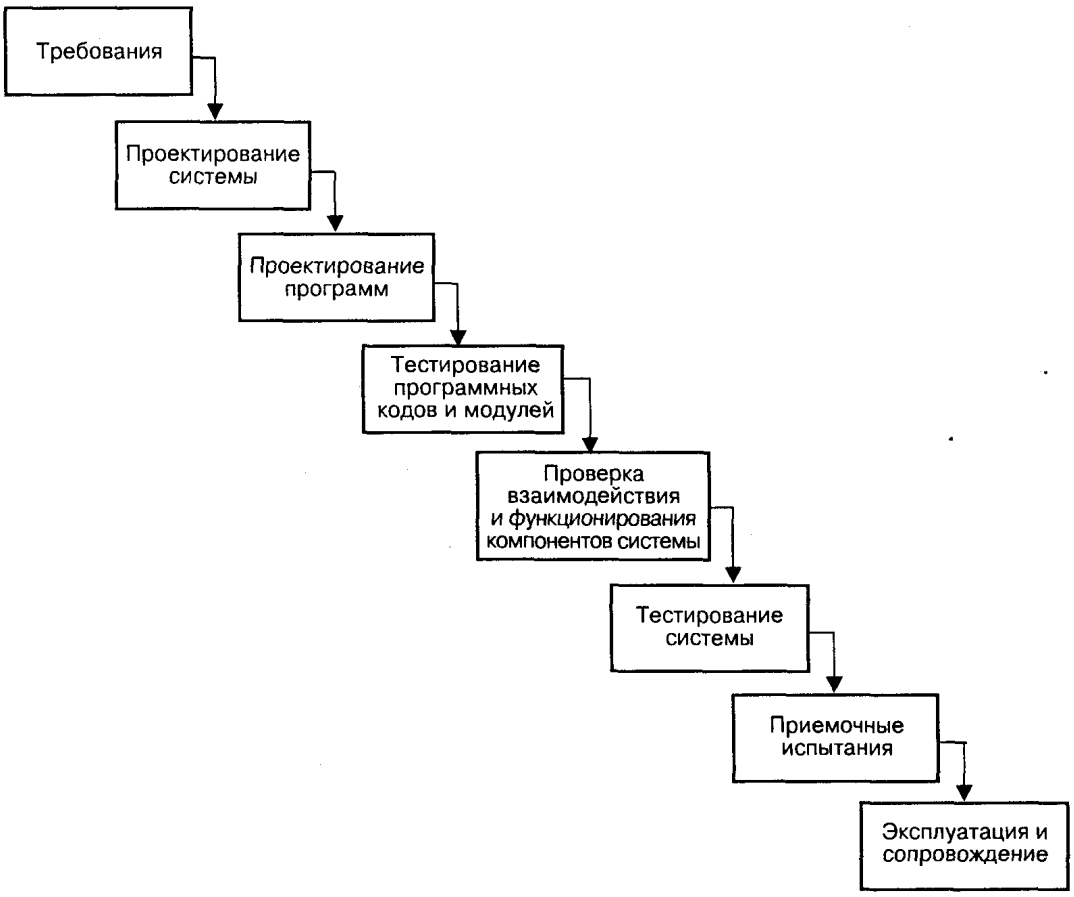

Рисунок 1.3 - Каскадная модель жизненного цикла

Проектирование системы, проектирование программ, кодирование и различные виды тестирования суть самодостаточные и тщательно документированные стадии процесса[14].

Позже появились итерационная и спиральная модель жизненного цикла ПО.

Итерационная модель предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом. Цель каждой итерации — получение работающей версии программной системы, включающей функциональность, определённую интегрированным содержанием всех предыдущих и текущей итерации. Результат финальной итерации содержит всю требуемую функциональность продукта. Таким образом, с завершением каждой итерации продукт получает приращение — инкремент — к его возможностям, которые, следовательно, развиваются эволюционно[4].

Спиральная модель предполагает, что каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации.

Процесс разработки должен включать проектирование, кодирование и тестирование невзирая на то, выполняются ли они в виде линейной последовательности, что характерно для каскадной модели, или в рамках итеративной последовательности, характерной для моделей эволюционного прототипирования и поэтапной передачи[14]. Каскадная модель содержит основные компоненты, необходимые для разработки программных продуктов. Поэтому мы будем использовать каскадную модель в качестве контекста для решения задач автоматизации процесса контроля качества программного обеспечения.

1.1.3.1 Каскадный процесс тестирования

В традиционной каскадной модели (см. рисунок 1.3) роль организации тестов остается неясной до стадий системного тестирования и приемочных испытаний. Большая часть видов деятельности, характерных для ранних стадий, таких как

проектирование, кодирование и модульное тестирование, в первую очередь связано с коллективом разработчиков программного обеспечения. По этой причине имеет смысл построить соответствующую модель жизненного цикла процесса тестирования. Пример каскадной модели жизненного цикла процесса тестирования приводится на рисунке 1.4.

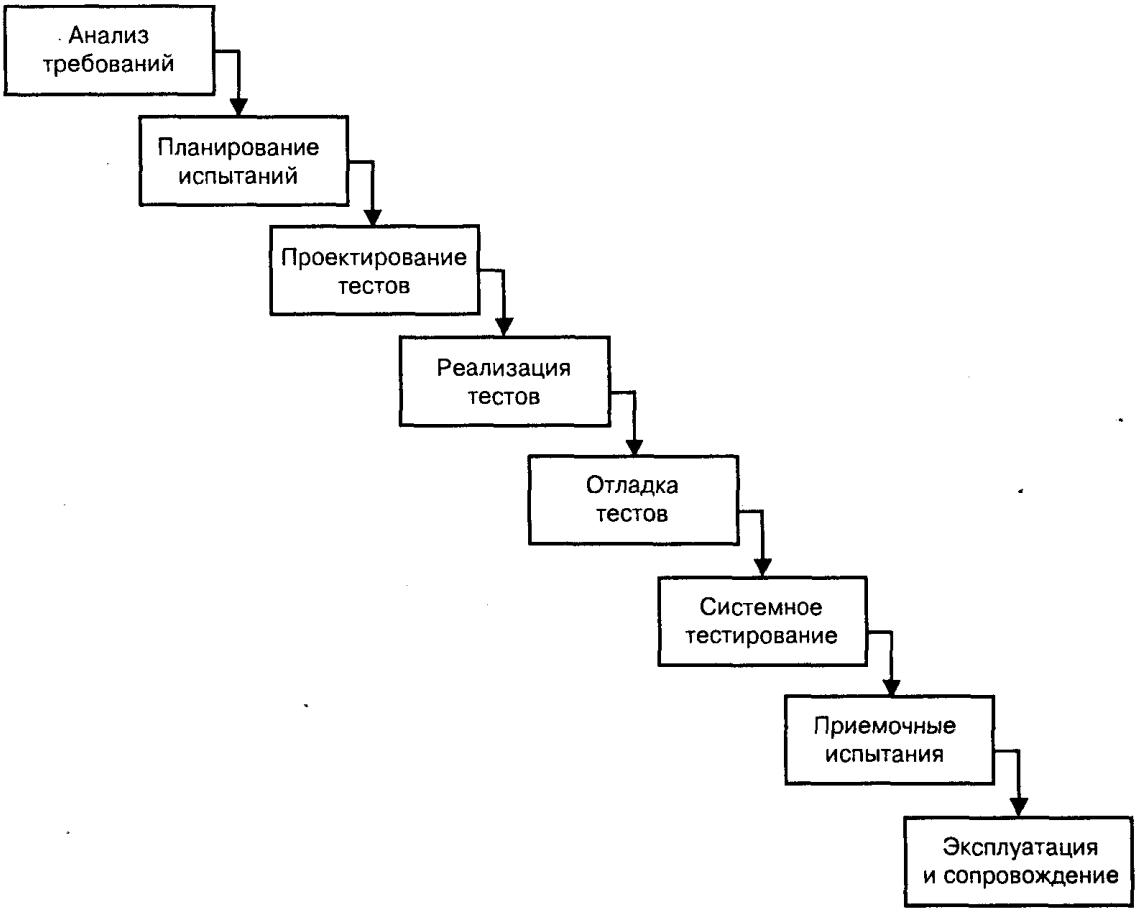

Рисунок 1.4 - Каскадный процесс тестирования

Из того, что разработка динамических тестов является процессом, во многом совпадающим с процессом разработки программного обеспечения, следует, что каскадная модель жизненного цикла динамического тестирования довольно сильно похожа на саму каскадную модель жизненного цикла.

Сводка входных и выходных данных для каждой стадии каскадного процесса отладки представлена в таблице 1.1.

Таблица 1.1 – Входные и выходные данные для каскадного процесса тестирования

Вид деятельности	Входы	Выходы
Анализ требований	Определение требований, спецификация требований	Матрица прослеживаемости требований
Планирование испытаний	Спецификация требований, матрица прослеживаемости требований	План проведения испытаний — стратегия тестирования, испытательная система, оценка объема работ и расписание тестирования
Проектирование тестов	Спецификация требований, матрица прослеживаемости требований, план проведения испытаний	Проектирование тестов — спецификация входных данных тестов, конфигурации тестов
Реализация тестов	Функциональная спецификация программного обеспечения, матрица прослеживаемости требований, план проведения испытаний, проекты тестов	Тестовые случаи — методы проверки и автоматизированные тесты
Отладка тестов	"Ранний вариант" кода, тестовые случаи, рабочая испытательная система	Тестовые случаи для заключительных испытаний
Системное тестирование	План испытания системы, матрица прослеживаемости требований, завершающие тестовые случаи, рабочая испытательная система	Результаты тестирования — сообщения о дефектах, сообщения о состоянии испытаний, суммарный отчет о результатах тестирования
Приемочные испытания	План приемочных испытаний, матрица прослеживаемости требований, бета-вариант программного кода, тестовые случаи для приемочных испытаний, рабочая испытательная система	Результаты тестирования
Эксплуатация и сопровождение	Скорректированный код, тестовые случаи для контроля дефектов, регрессионные тестовые случаи, рабочая испытательная система	Местоположение обнаруженных дефектов

Из таблицы 1.1 видно, что результатом тестирования является сообщения о дефектах и суммарный отчет о результатах тестирования. Согласно техническому заданию, разрабатываемая информационная система должна поддерживать формализацию этих результатов и возможность их дальнейшей обработки.

1.1.3.2 Связь тестирования и разработки

У моделей процесса разработки программного обеспечения и процесса тестирования исходные и конечные точки, однако группы разработчиков и тестировщиков все время заняты различными видами деятельности. В этом разделе

даны описания двух моделей, которые связывают оба эти вида деятельности воедино.

Один из способов продемонстрировать, как соотносятся тестирование и разработка, показан на V-образной диаграмме, изображенной на рисунке 1.5.

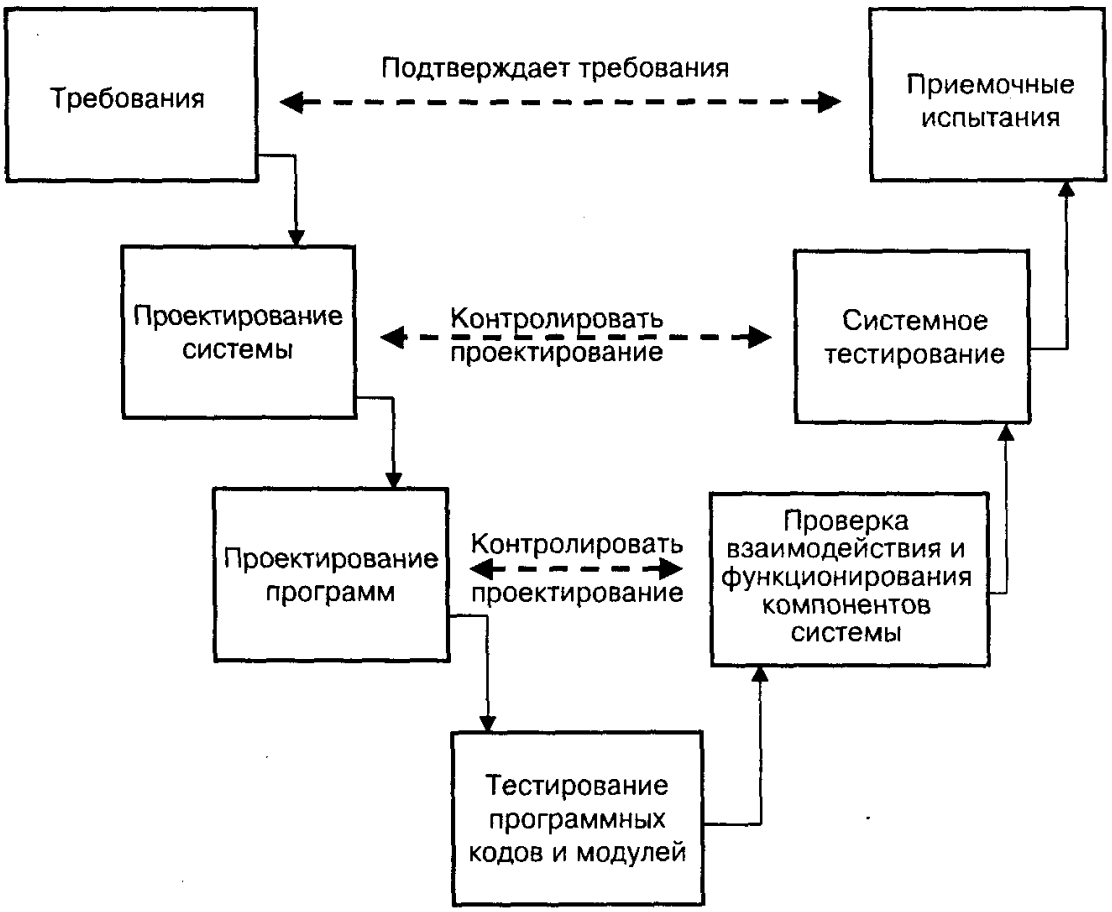

Рисунок 1.5 - Шарнирно-каскадная, или V-диаграмма

В этой модели, на которую также ссылаются как на шарнирно-каскадную, оба вида деятельности, анализ и проектирование, образуют левую сторону V. Кодирование находится в самой нижней точке диаграммы, а тестирование образует ее правую сторону. Для простоты изложения, вид деятельности, подобный сопровождению, на диаграмме не показан.

Пунктирные линии со стрелками на концах определяют отношения между видом тестовой деятельности на правой стороне и видом проектной деятельности на левой. Самая верхняя пунктирная линия со стрелками показывает, что цель

приемочных испытаний заключается в подтверждении требований, а сами приемочные испытания основаны на требованиях. Аналогично, системные испытания служат для проверки проекта системы, а системные тесты получены по результатам проектирования системы. Следовательно, одно из назначений V-диаграммы состоит в том, чтобы показать, какова цель видов тестовой деятельности в терминах контроля и аттестации на ранних стадиях разработки.

Несмотря на то что V-диаграммы служат иллюстрацией отношений, связывающих разработку и тестирование, она не отражает двух параллельных потоков деятельности, которые две эти группы специалистов осуществляют в процессе разработки. Иллюстрацией одного из способов показать отдельные действия, связанные с разработкой и тестированием, служит параллельная каскадная модель на рисунке 1.6.

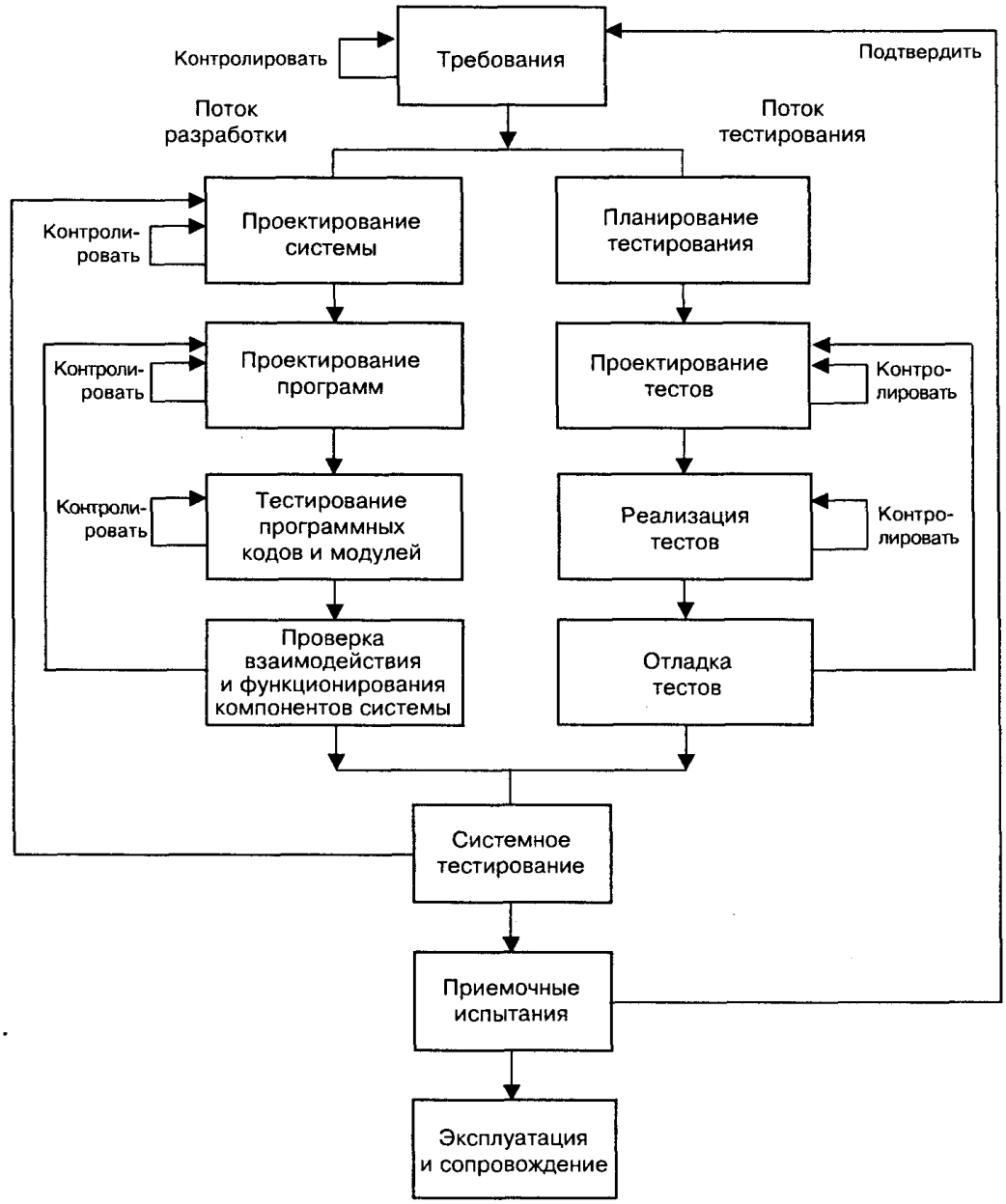

Рисунок 1.6 - Параллельная каскадная модель

Диаграмма на рисунке 1.6 отражает:

- отчетливо видны параллельные потоки разработки и тестирования;

- стрелка, соединяющая системное тестирование и проектирование системы, говорит о том, что цель стадии системного тестирования заключается в проверке проекта системы;

- стрелка, соединяющая приемочные испытания с требованиями, показывает, что назначение приемочных испытаний заключается в подтверждении требований;

- аналогичные стрелки говорят о том, что назначение тестирования модулей и проверки взаимодействия и функционирования компонентов системы заключается в контроле разработки программы, а цель отладки тестов состоит в контроле разработки тестов;

- каждая оставшаяся стадия потоков разработки и тестирования имеет «петли обратной связи», назначение которых заключается в проверке того, что промежуточные результаты системного проектирования, проектирования программ, планирования испытаний и тому подобного, соответствуют требованиям, предъявленных к ним в начале соответствующей стадии. Такая проверка выполняется в рамках статического тестирования.

На рисунке 1.6 видно, почему тестирование программного обеспечения является столь ответственной и трудоемкой задачей. Даже при работе в традиционной каскадной среде группа специалистов по отладке должна уметь выполнять широкий спектр действий, каждое из которых зависит от результатов деятельности коллектива разработчиков. Между потоками разработки и тестирования этого процесса должен надежно поддерживаться обмен данными, и каждая группа должна участвовать в некоторых контрольных действиях другой группы. Например, группа тестировщиков должна участвовать в контроле системного проектирования, а группа разработчиков — принимать участие в проверке плана тестирования.

Поэтому задачи автоматизации процесса контроля качества программного обеспечения имеют особую актуальность в организациях, занимающихся разработкой ПО как для собственных нужд, так и для своих клиентов.

1.2 Анализ известных систем контроля качества производства программного обеспечения

1.2.1 Выбор известных систем отслеживания ошибок для анализа

Известные системы контроля качества производства программного обеспечения представляют собой клиент-серверные приложения. Среди них можно найти как веб-приложения, так и традиционные оконные программы. При этом

стоит отметить, что решения в виде веб-приложений зачастую оказываются более востребованными в силу следующих преимуществ:

- при работе с веб-приложением не требуется установка дополнительных программ – достаточно стандартного веб-браузера;
- веб-приложение работает независимо от операционной системы клиента.

Система контроля качества производства программного обеспечения может быть как отдельным программным продуктом, так и может входить в состав других систем управления проектами производства программного обеспечения. Система контроля качества производства программного обеспечения является неотъемлемой частью любой из систем управления проектами производства программного обеспечения.

Большинство программных продуктов для контроля качества производства программного обеспечения представляют собой коммерческое программное обеспечение, однако также существуют бесплатные программы для контроля качества производства программного обеспечения.

В обзоре мы рассмотрим наиболее распространенные на российском рынке системы отслеживания ошибок, список которых приведен в таблице 1.2.

Таблица 1.2 – Список рассматриваемых приложений для отслеживания ошибок

№	Название	Тип приложения	Тип ПО	Разработчик
1	Bugzilla	веб-приложение	бесплатное, open source	Dave Miller / Mozilla Foundation
2	JIRA	веб-приложение	коммерческое	Atlassian Software Systems
3	Redmine	веб-приложение	бесплатное, open source	Jean-Philippe Lang
4	Bontq	веб-приложение	коммерческое	Bontq LLC
5	DevProject Manager	традиционное приложение	бесплатное	Gaijin.at
6	BugTracker .NET	веб-приложение	бесплатное, open source	Corey Trager

Из таблицы 1.2 видно, что большинство программного обеспечения для отслеживания ошибок является бесплатным и имеет тип веб-приложение.

1.2.2 Bugzilla

Одной из наиболее популярных систем отслеживания ошибок является Bugzilla (Багзилла)[16]. Она представляет собой программную оболочку, обеспечивающую отслеживания ошибок (багтрекинга) с веб-интерфейсом.

В 1998 году Bugzilla была выпущена как открытое программное обеспечение компанией Netscape. В настоящее время система разрабатывается «Mozilla Foundation».

С одной стороны, Bugzilla довольно проста, с другой стороны, там есть всё, что нужно для багтрекинга типичного проекта. Сейчас Bugzilla используют более тысячи компаний и организаций по всему миру, среди них — такие компании, как: NASA, Id Software, IBM и софтверные проекты: Mozilla Firefox, Linux, GNOME, KDE, Apache Project, OpenOffice.org.

По функциональности Bugzilla в 2007 году отставала от многих современных багтрекеров. Разработчики считали, что одна из причин этого — выбор Perl в качестве языка реализации Bugzilla, рассматривалась возможность переписать её на каком-нибудь другом языке программирования.

Ключевым понятием системы является «баг» — некоторое задание, запрос, рекламация по поводу ошибки в системе, или просто сообщение, требующее обратной связи.

Для работы Bugzilla требуются:

- веб-сервер с поддержкой CGI (рекомендуется Apache);
- поддержка языка Perl;
- база данных MySQL, PostgreSQL, или Oracle (экспериментально)[17].

1.2.3 JIRA

Atlassian JIRA — коммерческая система отслеживания ошибок, предназначена для организации общения с пользователями, хотя в некоторых случаях систему можно использовать для управления проектами. Разработана компанией Atlassian

Software Systems[18]. Платная. Имеет веб-интерфейс. Название системы (JIRA) было получено путём модификации названия конкурирующего продукта — Bugzilla. JIRA создавалась в качестве замены Bugzilla и во многом повторяет архитектуру Bugzilla. Система позволяет работать с несколькими проектами. Для каждого из проектов создаёт и ведёт схемы безопасности и схемы оповещения.

До версии 3.13.5 (включительно) различались Enterprise, Professional и Standard версии. После — осталась только Enterprise[19].

1.2.4 Redmine

Redmine — открытое серверное веб-приложение для управления проектами и отслеживания ошибок[20]. Redmine написан на Ruby и представляет собой приложение на основе широко известного веб-фреймворка Ruby on Rails. Распространяется согласно GNU General Public License.

Данный продукт предоставляет следующие возможности:

- ведение нескольких проектов;
- гибкая система доступа, основанная на ролях;
- система отслеживания ошибок;
- диаграммы Ганта и календарь;
- ведение новостей проекта, документов и управление файлами;
- оповещение об изменениях с помощью RSS-потоков и электронной почты;
- вики для каждого проекта;
- форумы для каждого проекта;
- учёт временных затрат;
- настраиваемые произвольные поля для инцидентов, временных затрат, проектов и пользователей;
- лёгкая интеграция с системами управления версиями (SVN, CVS, Git, Mercurial, Bazaar и Darcs);
- создание записей об ошибках на основе полученных писем;
- поддержка множественной аутентификации LDAP;
- возможность самостоятельной регистрации новых пользователей;
- многоязыковой интерфейс (в том числе русский);

- поддержка СУБД MySQL, PostgreSQL, SQLite, Oracle.

Пользователи являются одним из центральных понятий предметной области. Модель пользователя является основой для идентификации и аутентификации работающего с системой персонала и клиентов, а также для авторизации их в разных ролях, проектах и т. п.

Роли пользователей определяются гибкой моделью определения прав доступа пользователей. Роли включают в себя набор привилегий, позволяющих разграничивать доступ к различным функциям системы.

Пользователям назначается роль в каждом проекте, в котором он участвует, например «менеджер в проекте по разработке сайта А», «разработчик в проекте по поддержанию интранета компании» или «клиент в проекте по рефакторингу информационной системы компании Б». Пользователь может иметь несколько ролей. Назначение роли для отдельной задачи (issue) в данный момент невозможно.

Проект является одним из основных понятий в предметной области систем управления проектами. Благодаря этой сущности возможно организовать совместную работу и планирование нескольких проектов одновременно с разграничением доступа различным пользователям (см. выше). Проекты допускают иерархическую вложенность.

Трекеры являются основной классификацией, по которой сортируются задачи в проекте. Само по себе понятие «трекер» восходит к системам учёта ошибок (англ. Bug tracking tool), представлявшим каждая в отдельности один проект.

По сути, в «Redmine» трекеры представляют собой аналог подклассов класса «Задача» и являются основой для полиморфизма разного рода задач, позволяя определять для каждого их типа различные поля. Примерами трекеров являются «Улучшение», «Ошибка», «Документирование», «Поддержка»,

Задачи являются центральным понятием всей системы, описывающим некую задачу, которую необходимо выполнить. У каждой задачи в обязательном порядке есть описание и автор, в обязательном порядке задача привязана к трекеру.

Каждая задача имеет статус. Статусы представляют собой отдельную сущность с возможностью определения прав на назначение статуса для различных ролей

(например, статус «отклонен» может присвоить только менеджер) или определение актуальности задачи (например, «открыт», «назначен» — актуальные, а «закрыт», «отклонен» — нет).

Для каждого проекта отдельно определяются набор этапов разработки и набор категорий задач. Среди других полей интересны также «оцененное время», служащее основой для построения управленческих диаграмм, а также поле выбора наблюдателей за задачей (см. «Получение уведомлений»). К задачам имеется возможность прикреплять файлы (имеется отдельная сущность «Приложение»).

Значения других перечислимых свойств (например, приоритетность) хранятся в отдельной общей таблице.

За отслеживание изменений параметров задач пользователями в системе отвечают две сущности: «Запись журнала изменений» и «Измененный параметр». Запись журнала отображает одно действие пользователя по редактированию параметров задачи и/или добавление комментария к ней. То есть служит одновременно инструментом ведения истории задачи и инструментом ведения диалога.

Сущность «Измененный параметр» привязана к отдельной записи журнала и предназначена для хранения старого и нового значения измененного пользователем параметра.

Задачи могут быть взаимосвязаны: например, одна задача является подзадачей для другой или предшествовать ей. Эта информация может быть полезна в ходе планирования разработки программы, за её хранение в Redmine отвечает отдельная сущность.

Система поддерживает учет затраченного времени благодаря сущности «Затраченное время», связанной с пользователями и задачей. Сущность позволяет хранить затраченное время, вид деятельности пользователя (разработка, проектирование, поддержка) и краткий комментарий к работе. Эти данные могут быть использованы, например, для анализа вклада каждого участника в проект или для оценки фактической трудоемкости и стоимости разработки.

Некоторые недостатки Redmine:

- Управление файлами и документами в Redmine сводится к их добавлению, удалению и редактированию. Правами доступа ни к файлам, ни к отдельным документам управлять нельзя.

- Отсутствуют оповещения об изменении документов.

- В Redmine нельзя управлять правами доступа на уровне отдельных полей задачи. Например, на данный момент от клиентов нельзя скрыть оценки времени работы над проектом или информацию о потраченном времени.

- В Redmine можно управлять правами доступа на уровне проектов, но нельзя назначить права на какую-то версию проекта или отдельную задачу. Это значит, что если пользователю нужен доступ всего к одной задаче, то придется давать доступ ко всему проекту.

- Если пользователь Redmine получил доступ к проекту, то сейчас нельзя ограничить его активность какими-то отдельными типами задач (трекерами). Например, нельзя разрешить просматривать или создавать задачи только какого-то определенного типа.

- В Redmine все дополнительные поля доступны всем пользователям, все участники проекта смогут их видеть и изменять. Это ограничение может привести к сложностям при наличии неоднородной команды, когда доступ к проекту имеют и менеджеры, и разработчики, и клиенты.

- В Redmine нет прав на отдельные типы переходов в workflow. Например, сейчас нельзя указать, что когда кто-то заканчивает исправлять ошибку, он должен выбрать ответственным тестировщика и должен указать номер билда. Также нельзя скрыть внутреннюю переписку между программистами от клиента.

- В Redmine в список задач не выводится общая трудоемкость задач, а в отчетах по трудоемкости нельзя делать отборы, в том числе и по исполнителю.

- В Redmine не реализовано делегирование задач — нельзя передать задачу другому исполнителю, отметив, что задача должен исполнять он, но оставив себе наблюдение за задачей[21].

1.2.5 Bontq

Bontq — веб-приложение для управления проектами и отслеживания ошибок[22]. Bontq написан на PHP и Java, основным его отличием от конкурентов является кросс-платформенный клиент, который может делать скриншоты и записывать видео для составления визуальных отчётов об ошибках.

Возможности:

- отслеживание ошибок в ПО;
- управление проектами и задачами;
- захват скриншотов и видео через Java клиент;
- интеграция с Google Docs;
- импорт данных из Basecamp и FogBugz[23].

1.2.6 DevProject Manager

DevProject Manager представляет собой портативную программу с широкими возможностями управления проектами разработки приложений для Freeware, Shareware и Open-Source разработчиков[24].

DevProject Manager позволяет управлять общей информацией проекта, информацией о версии, список дел и журналы изменений для каждого проекта. Могут быть добавлены пользовательские файлы, такие, как скриншоты, файлы справки и другие важные файлы, которые могут быть открыты для диспетчера DevProject. Может быть добавлена персональная информация для переводчиков, программистов, художников и других заинтересованных лиц. Кроме того, программа может быть расширена собственными плагинами.

Отрывки кода могут быть сохранены в отдельный раздел, так что важные исходные коды в иерархической форме доступны для обработки. Программа поддерживает выделение исходных кодов 13 различных языков программирования (Assembler, C #, C + +, HTML, Java, JScript, Pascal, Perl, PHP, Python, SQL, VBScript и Visual Basic).

DevProject менеджер портативный и предназначен для запуска на устройствах USB. Дополнительные сервера баз данных MySQL, как и MS SQL не требуется[24].

1.2.7 BugTracker .NET

BugTracker .NET — это свободная реализация системы баг-трекинга на платформе .NET[25].

BugTracker .NET является бесплатной системой Free Software с открытым исходным кодом Open Source и распространяется под лицензией GNU General Public License.

Основными преимуществами этой системы является простота и удобство работы, минимизация использования программирования при настройке и в то же время открытый исходный код.

Базовая функциональность системы содержит функциональность баг-трекинга. Проект основался в 2002 году и поддерживается автором и на текущий момент.

Основные особенности BugTracker .NET:

- открытый исходный код;
- низкая стоимость внедрения, владения и технического сопровождения системы за счёт отсутствия затрат на приобретение лицензий;
- полнофункциональный веб-интерфейс;
- общая информационная база для всех модулей системы;

Архитектура системы представляет из себя стандартную трехзвенную архитектуру клиент-сервер. В качестве СУБД используется MS SQL Server. Язык реализации C#.

Функциональные возможности:

- ведение списка проектов;
- интеграция с email;
- различные механизмы аутентификации пользователей;
- трекинг багов и задач;
- гибкая возможность настройки категорий для issue-трекинга;
- функциональность отчетов;
- функциональность Dashboard;
- возможность добавления к категориям новых полей.

1.2.8 Выводы по анализу информационных систем для контроля качества

Рассмотренные в анализе приложения для управления качеством производства программного обеспечения обладают похожим набором функций для отслеживания ошибок в приложениях, поэтому с этой точки зрения нельзя выделить лучшее приложение. Однако стоит признать, что наиболее перспективным направлением развития систем отслеживания ошибок является организация их в виде клиент-серверного приложения с веб-интерфейсом, которое использует свободное программное обеспечение в качестве веб-сервера и СУБД. Учитывая эти требования, приходим к выводу, что наиболее перспективными коммерческими программными продуктами для отслеживания ошибок на современном российском рынке являются JIRA, а среди свободного программного обеспечения – Bugzilla и Redmine или другие подобные разработки систем отслеживания ошибок. Но в них отсутствует формирование сводной таблицы оценки качества реализации проекта по версиям программы, предусмотренное техническим заданием.

Очевидно, что при разработке собственной системы контроля качества производства программного обеспечения нужно стремиться к тому, чтобы она обладала достоинствами существующих программ этой области, а также учитывала их недостатки.

Еще одной особенностью существующих систем отслеживания ошибок является отсутствие полноценной интеграции с 1С: Предприятие, которая используется в большинстве коммерческих фирм в России. Это имеет ряд недостатков, основными из которых являются:

- неудобство в обслуживании и администрировании разных видов баз данных;
- необходимость создания промежуточных механизмов для синхронизации информации о продуктах, заказчиках, и их требованиях.

Гораздо удобнее использовать платформу 1С: Предприятие в качестве основы для построения системы отслеживания ошибок.

1С: Предприятие версии 8.2 представляет возможность работать в режиме тонкого клиента и веб-клиента.

Тонкий клиент и веб-клиент – это два новых клиентских приложения. Кроме привычного файлового доступа и подключения к серверу по локальной сети они позволяют подключаться к информационной базе по протоколу HTTP через специально настроенный веб-сервер. Тонкий клиент и веб-клиент обеспечивают работу пользователей в новом режиме – режиме управляемого приложения.

Тонкий клиент устанавливается на компьютер пользователя. При этом он имеет значительно меньший объем дистрибутива, чем старое клиентское приложение, и использует меньше аппаратных ресурсов. Тонкий клиент поставляется как в составе полного дистрибутива платформы, так и отдельным дистрибутивом.

Веб-клиент - это одно из клиентских приложений системы «1С:Предприятие 8». В отличие от «привычных» клиентских приложений (толстого клиента и тонкого клиента), его не нужно предварительно устанавливать на компьютер пользователя. У веб-клиента нет исполняемого файла. Веб-клиента вы не найдете ни в меню, ни среди исполняемых файлов. Потому он и веб-клиент, что ему для начала работы не нужно иметь никаких файлов на компьютере пользователя.

Веб-клиент, в отличие от толстого и тонкого клиентов, исполняется не в среде операционной системы компьютера, а в среде интернет-браузера (Windows Internet Explorer, Mozilla Firefox, Google Chrome или Safari). Поэтому любому пользователю достаточно всего лишь запустить свой браузер, ввести адрес веб-сервера, на котором опубликована информационная база, – и веб-клиент сам "приедет" к нему на компьютер и начнет исполняться.

Веб-клиент использует технологии DHTML и HTTPRequest. При работе веб-клиента клиентские модули, разработанные в конфигурации, компилируются автоматически из встроенного языка 1С:Предприятия 8.2 и непосредственно исполняются на стороне веб-клиента.

Таким образом, независимо от клиентского приложения (толстый, тонкий, веб-клиент), вся разработка прикладного решения ведется полностью в конфигураторе 1С:Предприятия, серверный и клиентский код пишется на встроенном языке 1С:Предприятия[3].

Подводя итог обзору достоинств и недостатков существующих систем отслеживания ошибок, можно утверждать, что создаваемая в рамках данного дипломного проекта система управления базой данных должна использовать программное обеспечение «1С:Предприятие» версии 8.2, выполнять основные функции известных программ-аналогов и предоставлять возможности по работе с приложением через веб-клиент. В качестве основы для разработки конфигурации мы выберем стандартную конфигурацию «Управление торговлей» а также веб-сервер Apache. Apache веб-сервер стал весьма популярным, в первую очередь из-за своей низкой стоимости (она является свободным программным обеспечением).

1.3 Формализация поставленной задачи

Определим объект автоматизации в виде последовательности действий участников процесса создания программного обеспечения:

а) заказчик предъявляет требования;

б) менеджер проекта создает проект в справочнике проектов и словесно описывает требования;

в) менеджер проекта создает функциональные группы будущего приложения в соответствующем справочнике;

г) разработчик опираясь на требования разрабатывает программу и создает релиз в соответствующем справочнике; словесно описывает внесенные изменения и реализованные функциональности;

д) тестировщик проводит проверку релиза, найденные неисправности заносят в специальные документы, выставляют статус релизу, процент протестированности функциональных групп и оценку качества этих функциональных групп;

е) разработчик исправляет найденные неисправности и выставляет соответствующим документам статусы, добавляя комментарии;

ж) разработчик создает следующий релиз, содержащий исправления ранее найденных багов и реализацию нового функционала;

з) тестировщик проводит тестирование релиза, найденные неисправности заносят в специальные документы, исправленные ошибки проверяют на невоспроизводимость (если проблема осталась, то разработчику предоставляется

информация об условиях ее повторного воспроизведения), выставляют статус релизу, процент протестированности функциональных групп и оценку качества этих функциональных групп;

и) пункты ж, з проводятся до тех пор, пока менеджер проекта не увидит, что все требования заказчика реализованы и продукт имеет хорошее качество;

к) по окончании продукт передается заказчику, проекту выставится соответствующая отметка о завершении стадии разработки.

Определим сущности, которые необходимо будет создать в конфигурации «1С:Предприятие».

Сущность (класс объекта, информационный объект) – это элемент предметной области, обозначающий реальный объект, событие, процесс, явление.

Экземпляр сущности – это один элемент данной сущности.

Сущность характеризуется множеством экземпляров с одинаковым набором атрибутов.

Атрибуты (реквизиты) – это набор характеристик, описывающих каждую сущность.

Для предметной области «контроль качества производства программного обеспечения» выделим следующие сущности:

- пользователи информационной базы;
- проекты;
- неисправности программы;
- версии программы;
- функциональные группы проекта;
- качество версии программы.

Определим наборы атрибутов для каждой из сущностей.

В сущности «Качество версии» должны быть такие атрибуты как «Версия программы», «Функциональная группа» и «Уровень качества». Данная сущность не представлена в стандартной конфигурации «1С:Управление торговлей».

В сущности «Пользователи информационной базы» могут быть такие атрибуты как «Идентификатор пользователя информационной базы» (ключевой атрибут),

«Физическое лицо», «Текущее подразделение», «Контактная информация». Данная сущность представлена в стандартной конфигурации «1С:Управление торговлей».

В сущности «Проекты» есть такие атрибуты как «Порядковый номер проекта» (ключевой атрибут), «Ответственный», «Плановая дата начала», «Дата начала», «Плановая дата окончания», «Дата окончания», «Завершен», «Коментарий». Данная сущность также представлена в стандартной конфигурации «1С:Управление торговлей».

В сущности «Функциональные группы проекта» должны быть такие атрибуты как «Порядковый номер функциональной группы» (ключевой атрибут), «Название» и «Проект». Данная сущность не представлена в стандартной конфигурации «1С:Управление торговлей».

В сущности «Неисправности программы» должны быть такие атрибуты как «Порядковый номер неисправности» (ключевой атрибут), «Версия программы», «Дата и время обнаружения», «Заголовок», «Описание», «Серьезность», «Приоритет», «Статус», «Резолюция», «Целевая версия», «Комментарий», «Автор», «Исполнитель» и «Функциональная группа». Данная сущность не представлена в стандартной конфигурации «1С:Управление торговлей».

В сущности «Версии программного обеспечения» должны быть такие атрибуты как «Порядковый номер версии программы» (ключевой атрибут), «Проект», «Название», «Описание», «Путь к файлу с приложением», «Дата версии» и «Статус». Данная сущность не представлена в стандартной конфигурации «1С:Управление торговлей».

Схема структуры базы данных представлена на рисунке 1.8.

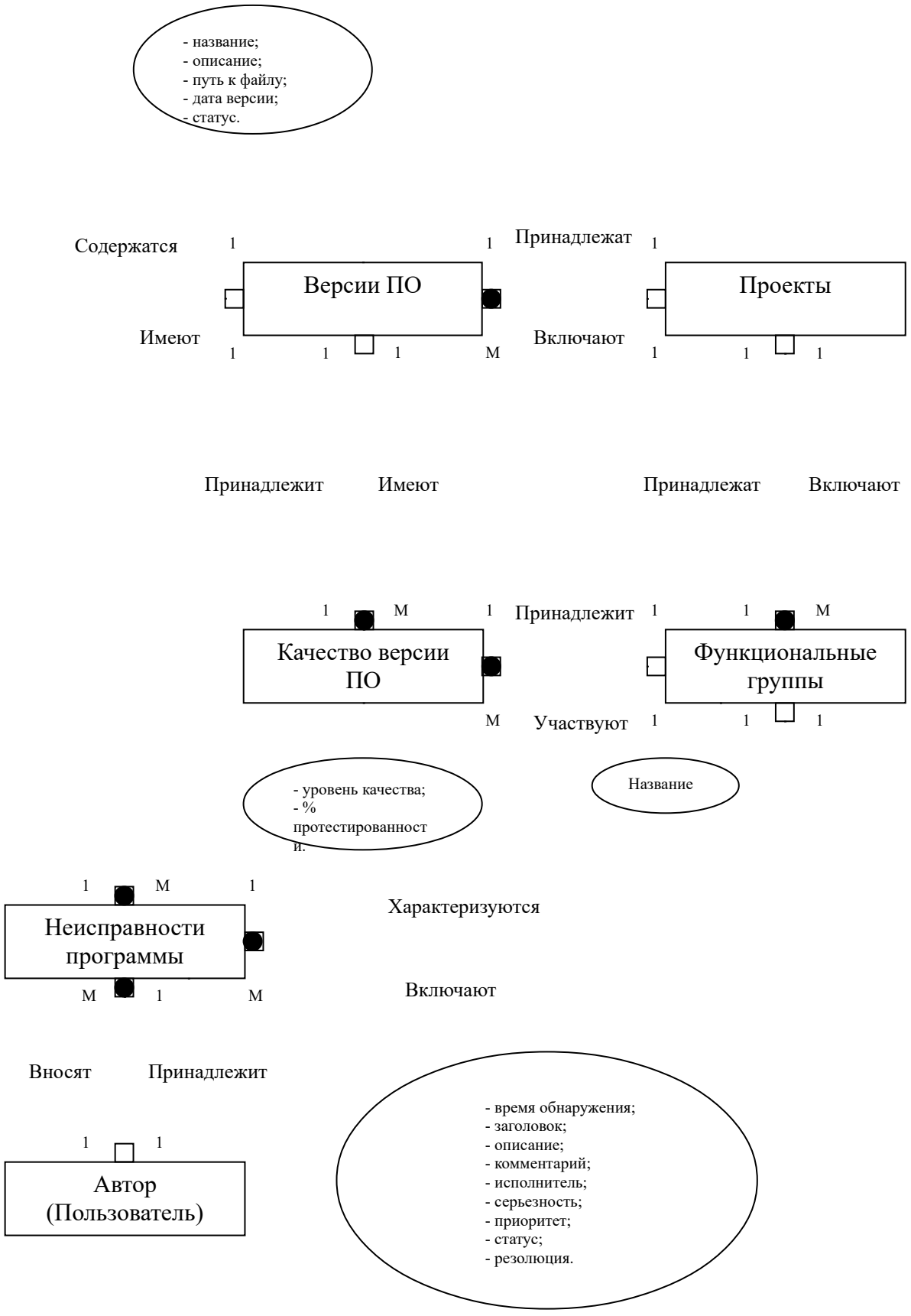

Рисунок 1.8 – Инфологическая модель базы данных

На схеме (см. рисунок 1.8) изображены сущности проектируемой конфигурации с основными атрибутами и характер связи между экземплярами этих сущностей. На ее основе будет определяться окончательное количество необходимых таблиц. Надо отметить, что некоторые атрибуты сущностей (серьезность, приоритет и др.) имеют характер перечисления, и в отдельные сущности здесь не вынесены.

Таблицы «Пользователи» и «Проекты» уже содержатся в стандартной конфигурации «1С:Управление торговлей», что позволит сократить время разработки конфигурации.

Определим перечень основных и наиболее важных функций, которыми должна обладать разрабатываемая система контроля качества производства программного обеспечения. Анализируя возможности существующих программ данного класса и их недостатки, можно составить следующий список функций:

- аутентификация и авторизация администраторов и пользователей системы отслеживания ошибок;
- многопользовательский режим работы приложения;
- хранение данных об обнаруженных дефектах программ;
- формирование форм для ввода и редактирования данных об обнаруженных дефектах;
- формирование форм для выбора проекта и версий программ;
- формирование форм для представления отчетов.

С помощью «1С: Предприятие» можно спроектировать конфигурацию базы данных для хранения информации о проектах по созданию программного обеспечения, версиях этих программ, обнаруженных дефектах, об администраторах и участниках процесса разработки, о состоянии качества разработанных программ и т.д.

Язык программирования 1С, используемый для создания конфигураций, идеально подходит для решения большинства перечисленных выше задач, которыми должна обладать разрабатываемая система отслеживания ошибок.

«1С: Предприятие» призвано решать следующие задачи:

- система должна обеспечить высокий уровень адаптируемости прикладных решений под требования заказчика;

- система должна обеспечивать изменение готового прикладного решения разработчиком, не участвовавшем в его создании;

- система должна обеспечивать эффективное использование компьютерных технологий и платформ, не требуя, при этом, глубоких специальных знаний от разработчика;

- система должна обеспечивать стандартизацию разработки[26].

Таким образом, все функции, которыми должна обладать проектируемая система отслеживания ошибок, можно реализовать, используя набор стандартных функций 1С.

2 Проектно-расчетная часть

2.1 Разработка общей архитектуры конфигурации

2.1.1 Основные элементы конфигурации

Конфигурация для контроля качества производства программного обеспечения, разработанная в данном дипломном проекте, представляет собой дополнительную подсистему «Качество» к стандартной конфигурации «1С:Управление торговлей 8.2». Основными элементами данной подсистемы являются справочники, отчеты и регистр сведений.

Справочник является списком возможных значений того или иного реквизита. Они используются в тех случаях, когда необходимо исключить неоднозначный ввод информации. Каждый справочник представляет собой список однородных объектов. Каждый такой объект называется элементом справочника[13].

Регистр сведений позволяют хранить в прикладном решении произвольные данные в разрезе нескольких измерений. Информация в регистре сведений хранится в виде записей, каждая из которых содержит значения измерений и соответствующие им значения ресурсов. Измерения регистра описывают разрезы, в которых хранится информация, а ресурсы регистра непосредственно содержат хранимую информацию.

Основными функциональными возможностями, которые предоставляет регистр сведений разработчику, являются:

- создание, изменение и удаление записей;
- выбор записей в заданном интервале по заданным критериям;
- выбор записей по регистратору;
- получение значений ресурсов записей, соответствующих указанному периоду и значениям измерений;
- получение значений ресурсов наиболее ранних и наиболее поздних записей регистра, соответствующих указанному периоду и значениям измерений.[3]

В подсистеме «Качество» не используются документы, т.к. в ней не вносятся изменения в какие-либо регистры бухгалтерского и складского учета стандартной

конфигурации.

2.1.2 Краткая характеристика подсистем конфигурации

В связи с тем, что конфигурация «1С:Управление Торговлей 8.2» позволяет автоматизировать практически каждое из направлений деятельности торгового предприятия, для работы с каждым из них для удобства система разделена на несколько подсистем:

- управление запасами;
- ценообразование;
- управление закупками;
- управление взаимоотношениями с клиентами (CRM);
- управление продажами;
- управление правилами продаж;
- управление финансами;
- управление торговыми представителями;
- учет НДС.

В процессе разработки в конфигурацию «1С:Управление торговлей» была добавлена подсистема «Качество», включающая как элементы интерфейса на основном окне программы, так и дополнительные объекты конфигурации, о которых будет сказано ниже.

В стандартной конфигурации уже присутствуют справочники, которые необходимы для реализации описанной в разделе 1.3 функциональности, и которые не требуют отдельной реализации. К ним относятся:

- проекты;
- пользователи информационной системы.

2.1.3 Особенности работы конфигурации

2.1.3.1 Обработка ошибок

Обработка ошибок является одной из самых важных задач любого приложения. В «1С:Предприятие» имеется встроенная поддержка обработки ошибок. Однако для контроля корректного ввода данных созданы дополнительные обработки

позволяющие предупредить ввод пустых или некорректных значений полей. В случае возникновения значимой ошибки будут выведены сообщения об этом на интерфейсе пользователя.

Ниже представлен фрагмент кода в отчете:

```
Если НЕ Отчет.Проект.Пустая() тогда
      ТабДок = ПолучитьОтчетНаСервере();
      Если НЕ (ТабДок = неопределено) тогда
            ТабДок.Показать();
      КонецЕсли;
Иначе
      соб = новый СообщениеПользователю;
      соб.Текст = "Выберите проект.";
      соб.Сообщить();
КонецЕсли;
```

В данном фрагменте видно, что в случае, если обязательное поле «Проект» не заполнено в форме отчета, то пользователь получит сообщение «Выберите проект.».

2.1.3.2 Функции, используемые конфигурацией

Приложение для отслеживания ошибок в ходе своей работы использует большое число функций языка 1С. Список всех используемых в приложении функций представлен в таблице 2.1[27].

Таблица 2.1 – Встроенные функций 1С, используемые в приложении

Функция	Описание
Запрос (Query) УстановитьПараметр (SetParameter)	Устанавливает параметр запроса. Параметры доступны для обращения в тексте запроса. С помощью этого метода можно передавать переменные в запрос, например, для использования в условиях запроса.
Табличная часть (Tabular section) Очистить (Clear)	Удаляет все строки табличной части.
Запрос (Query) Выполнить (Execute)	Выполняет запрос к базе данных. В случае, если запросу установлен пакетный запрос, метод последовательно выполнит все запросы из пакета и вернет результат последнего запроса пакета, который не создает и не уничтожает временную таблицу. Если такого запроса нет, то будет возвращен результат исполнения последнего запроса.

РезультатЗапроса (QueryResult) Пустой (IsEmpty)	Определяет, есть ли в результате записи (без учета общих итогов).
РезультатЗапроса (QueryResult) Выбрать (Choose)	Формирует выборку записей из результата запроса.
ВыборкаИзРезультатаЗапроса (QueryResultSelection) Следующий (Next)	Получает следующую запись из результата запроса. Для обхода результата запроса нужно после получения выборки вызвать данный метод для позиционирования на первый элемент и далее вызывать до тех пор, пока не будет возвращено значение Ложь.
Табличная часть (Tabular section) Добавить (Add)	Добавляет строку в конец табличной части.
РегистрСведенийМенеджер.<Имя регистра сведений> (InformationRegisterManager.<Имя регистра сведений>) Получить (Get)	Получает ресурсы регистра сведений по указанным ключевым полям.
РегистрСведенийМенеджер.<Имя регистра сведений> (InformationRegisterManager.<Имя регистра сведений>) СоздатьНаборЗаписей (CreateRecordSet)	Создает набор записей регистра сведений. Набор записей создается пустым.
ЭлементОтбора (FilterItem) Установить (Set)	Устанавливает значение отбора и флаг использования. При выполнении данного метода вид сравнения устанавливается в значение Равно.
РегистрСведенийНаборЗаписей.<Имя регистра сведений> (InformationRegisterRecordSet.<Имя регистра сведений>) Добавить (Add)	Добавляет новую запись в набор.
РегистрСведенийНаборЗаписей.<Имя регистра сведений> (InformationRegisterRecordSet.<Имя регистра сведений>) Записать (Write)	Записывает набор записей в базу данных. В зависимости от переданного параметра, может быть выполнено добавление записей или их замещение.
ОтчетОбъект.<Имя отчета> (ReportObject.<Имя отчета>) ПолучитьМакет (GetTemplate)	Получает макет отчета.
ТабличныйДокумент (SpreadsheetDocument) ПолучитьОбласть (GetArea)	Получает область табличного документа как табличный документ. Область может состоять из нескольких расположенных подряд строк или колонок, либо прямоугольной областью таблицы.
Встроенные функции языка (Script functions) ТекущаяДата (CurrentDate)	Определяет текущую (системную) дату на компьютере.
ТабличныйДокумент (SpreadsheetDocument) Вывести (Put)	Выводит табличный документ в результирующий табличный документ, добавляя его со следующей строки вслед за самой нижней выведенной строкой, начиная с первой колонки.
Встроенные функции языка (Script functions) Формат (Format)	Формирует удобное для чтения представление значений. Полезно использование в отчетах и при прочем визуальном отображении значений.
Табличная часть (Tabular section)	Получает количество строк табличной части.

Количество (Count)	
Глобальный контекст (Global context) Сообщить (Message)	Выводит текст сообщения в окно сообщений. Если в момент вызова окно сообщений отсутствует, то будет открыто новое окно сообщений. Сообщение, в зависимости от его смысловой нагрузки, можно пометить одной из пиктограмм, входящих в предопределенный набор.
ТабличныйДокумент (SpreadsheetDocument) Показать (Show)	Открывает окно для показа и редактирования табличного документа.
ДиалогВыбораФайла (FileDialog) Выбрать (Choose)	Открывает окно диалога выбора файла.

Из таблицы 2.1 видно, что для реализации контроля качества производства программного обеспечения достаточно мало встроенных функций 1С.

Подробное описание использования вышеперечисленных функций в конфигурации для контроля качества производства программного обеспечения представлено в пункте 2.2 пояснительной записки.

2.2 Разработка основных классов конфигурации

2.2.1 Справочники

Для реализации функций контроля качества производства программного обеспечения используются справочники, приведенные в таблице 2.2.

Таблица 2.2 – Справочники «1С:Предприятие», используемые для контроля качества производства программного обеспечения

Справочник	Описание
ФункциональныеГруппы	Предназначен для хранения информации о функциональных группах, по которым будет отслеживаться уровень качества создаваемого программного обеспечения.
Релизы	Справочник содержит информацию о создаваемых версиях программного обеспечения в разрезе каждого проекта
Баги	Предназначен для хранения информации о выявленных в ходе тестирования неисправностях в программном обеспечении
Пользователи	Предназначен для хранения списка пользователей, которым разрешена работа с информационной базой.
Проекты	Предназначен для ведения списка долгосрочных проектов, которые используются в торговом предприятии.

Некоторые из перечисленных в таблице 2.2 справочников уже реализованы в стандартной конфигурации «1С:Управление торговлей», которые можно

использовать для реализации контроля качества производства программного обеспечения.

Справочник «Пользователи» в стандартной конфигурации «1С:Управление торговлей» предназначен для хранения списка пользователей, которым разрешена работа с информационной базой. Основное назначение справочника – идентификация пользователя в начале работы с конфигурацией. Заполнение справочника может производиться как вручную, так и автоматически - при первом запуске 1С:Предприятия от имени нового пользователя, созданного в режиме «Конфигуратор». Пользователей можно объединять в группы (подгруппы). Для каждого пользователя могут быть заданы настройки, оптимизирующие рабочий процесс в зависимости от индивидуальных пожеланий и предпочтений, а также от особенностей предприятия.

Паспортные данные пользователя (дата рождения, удостоверение личности задаются в справочнике «Физические лица».

Карточка пользователя позволяет просматривать и редактировать свойства пользователя информационной базы. В частности, здесь можно настроить параметры аутентификации (внешний вид формы повторяет форму редактирования параметров аутентификации пользователя информационной базы в режиме «Конфигуратор»), назначать роли, а также редактировать контактную информацию пользователя.

Справочник «Проекты» в стандартной конфигурации «1С:Управление торговлей» предназначен для ведения списка долгосрочных проектов, которые используются в торговом предприятии. В справочнике хранится список проектов предприятия с указанием темы проекта, плановых и фактических сроков работы по проекту, ответственного по проекту, привлеченных к проекту лиц. В рамках проекта можно вести список взаимодействий с участниками проекта, оформлять и контролировать задания сотрудникам на различных этапах проекта, хранить файлы документации по проекту[27].

Справочник «Функциональные группы» не содержит дополнительных реквизитов. Поля стандартных реквизитов данного справочника перечислены в

таблице 2.3.

Таблица 2.3 – Поля справочника «Функциональные группы»

Поле	Тип	Описание
Код	Строка	Содержит код элемента справочника. Число или Строка в зависимости от настроек справочника в конфигураторе.
Наименование	Строка	Содержит наименование элемента справочника.
Владелец	СправочникСсылка.<Имя справочника>; ПланВидовХарактеристикСсылка.<Имя плана видов характеристик>; ПланСчетовСсылка.<Имя плана счетов>; ПланВидовРасчетаСсылка.<Имя плана видов расчета>; ПланОбменаСсылка.<Имя плана обмена>	Содержит ссылку на владельца элемента справочника.
ПометкаУдаления	Булево	Содержит признак пометки на удаление элемента справочника. Истина - пометка удаления установлена.
Ссылка	СправочникСсылка	Содержит ссылку на элемент справочника. Это значение может быть записано в базу данных для полей соответствующего типа.
Предопределенный	Булево	Указывает, что данный элемент справочника является предопределенным элементом. Истина - предопределенный.

В таблице 2.3 видно, что справочник имеет шесть стандартных полей, которых достаточно для работы подсистемы «Качество».

Справочник «Версии программного обеспечения» содержит поля как стандартных реквизитов, так и дополнительных. Поля реквизитов данного справочника перечислены в таблице 2.4.

Таблица 2.4 – Поля справочника «Версии программного обеспечения»

Поле	Тип	Описание
Код	Строка	Содержит код элемента справочника. Число или Строка в зависимости от настроек справочника в конфигураторе.
Наименование	Строка	Содержит наименование элемента

		справочника.
Владелец	СправочникСсылка.<Имя справочника>; ПланВидовХарактеристикСсылка.<Имя плана видов характеристик>; ПланСчетовСсылка.<Имя плана счетов>; ПланВидовРасчетаСсылка.<Имя плана видов расчета>; ПланОбменаСсылка.<Имя плана обмена>	Содержит ссылку на владельца элемента справочника.
ПометкаУдаления	Булево	Содержит признак пометки на удаление элемента справочника. Истина - пометка удаления установлена.
Ссылка	СправочникСсылка	Содержит ссылку на элемент справочника. Это значение может быть записано в базу данных для полей соответствующего типа.
Предопределенный	Булево	Указывает, что данный элемент справочника является предопределенным элементом. Истина - предопределенный.
Описание	Строка	Содержит словесное описание версии программы.
ПутьКФайлу	Строка	Содержит путь к файлу или папке с версией программы.
ДатаВремяВерсии	Дата	Содержит дату и время, когда была создана версия программы.
Статус	ПеречислениеСсылка.СтатусыРелизов	Содержит статус, выставленный для данной версии программы.

В таблице 2.4 видно, что справочник имеет шесть стандартных полей и четыре дополнительных, которых достаточно для работы подсистемы «Качество».

Справочник «Неисправности программы» содержит поля как стандартных реквизитов, так и дополнительных. Поля реквизитов данного справочника перечислены в таблице 2.5.

Таблица 2.5 – Поля справочника «Неисправности программы»

Поле	Тип	Описание
Код	Строка	Содержит код элемента справочника. Число или Строка в зависимости от настроек справочника в конфигураторе.
Наименование	Строка	Содержит наименование элемента справочника.
Владелец	СправочникСсылка.<И	Содержит ссылку на владельца элемента

	мя справочника>; ПланВидовХарактеристикСсылка.<Имя плана видов характеристик>; ПланСчетовСсылка.<Имя плана счетов>; ПланВидовРасчетаСсылка.<Имя плана видов расчета>; ПланОбменаСсылка.<Имя плана обмена>	справочника.
ПометкаУдаления	Булево	Содержит признак пометки на удаление элемента справочника. Истина - пометка удаления установлена.
Ссылка	СправочникСсылка	Содержит ссылку на элемент справочника. Это значение может быть записано в базу данных для полей соответствующего типа.
Предопределенный	Булево	Указывает, что данный элемент справочника является предопределенным элементом. Истина - предопределенный.
Релиз	СправочникСсылка.Релизы	Содержит ссылку на версию программы, в которой найдена неисправность.
ДатаВремяОбнаружения	Дата	Содержит дату и время обнаружения неисправности.
Серьезность	ПеречислениеСсылка.УровниСерьезностиНеисправностей	Содержит указание уровня серьезности обнаруженной неисправности.
Приоритет	ПеречислениеСсылка.УровниПриоритетовНеисправностей	Содержит указание на приоритет для решения данной неисправности.
Статус	ПеречислениеСсылка.СтатусыНеисправностей	Содержит указание на актуальный статус неисправности.
Резолюция	ПеречислениеСсылка.РезолюцииДляНеисправностей	Содержит указание на присвоенную резолюцию.
ВерсияПрограммыСИсправлением	СправочникСсылка.Релизы	Содержит ссылку на версию программы, в которой исправлена неисправность.
Описание	Строка	Содержит словесное описание обнаруженной проблемы.
Автор	СправочникСсылка.Пользователи	Содержит ссылку на автора, создавшего элемент справочника.
Исполнитель	СправочникСсылка.Пользователи	Содержит ссылку на пользователя, которому поручено решение указанной проблемы.
ФункциональнаяГруппа	СправочникСсылка.ФункциональныеГруппы	Содержит ссылку на функциональную группу, где была обнаружена неисправность
Комментарии	ТабличнаяЧасть	Содержит записи комментариев для указанной неисправности.

В таблице 2.5 видно, что справочник имеет шесть стандартных полей и тринадцать дополнительных, которых достаточно для работы подсистемы «Качество». Также в этом справочнике есть табличная часть, которая имеет свои поля, перечисленные в таблице 2.6.

Таблица 2.6 – Табличная часть «Комментарии» справочника «Неисправности программы»

Поле	Тип	Описание
НомерСтроки	Число	Номер строки табличной части.
Комментарий	Строка	Содержит словесное описание проблемы и другие заметки.
ДатаВремя	Дата	Дата и время комментария.
Автор	СправочникСсылка.Пользователи	Содержит ссылку на автора, создавшего элемент табличной части.

В таблице 2.6 видно, что табличная часть имеет четыре поля, необходимых для работы подсистемы «Качество».

2.2.2 Отчеты

Для отображения качественных характеристик производимых версий программного обеспечения необходимо реализовать два отчета, «Сравнительное качество версий» и «Текущее состояние версии», которые изображены на рисунках 2.7 и 2.8.

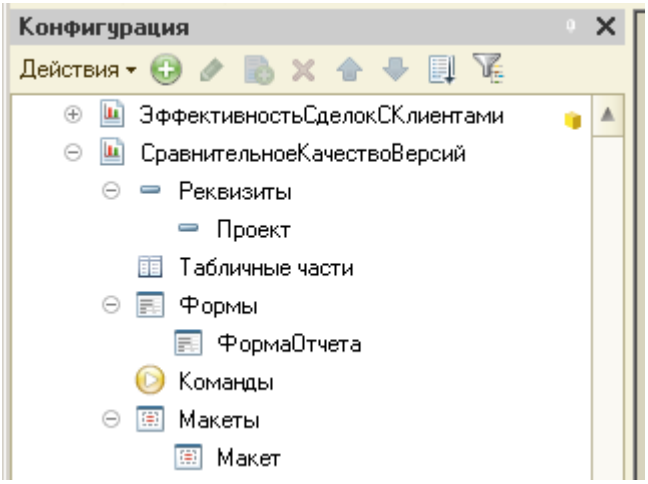

Рисунок 2.7 – Вид отчета «Сравнительное качество версий»
в конфигураторе «1С:Предприятие»

На рисунке 2.7 видно, что отчет «Сравнительное качество версий» использует всего один реквизит «Проект» типа «СправочникСсылка.Проекты», который заполняется в форме отчета «ФормаОтчета». Результаты отчета отображаются в виде табличного документа, сформированного на основе макета «Макет».

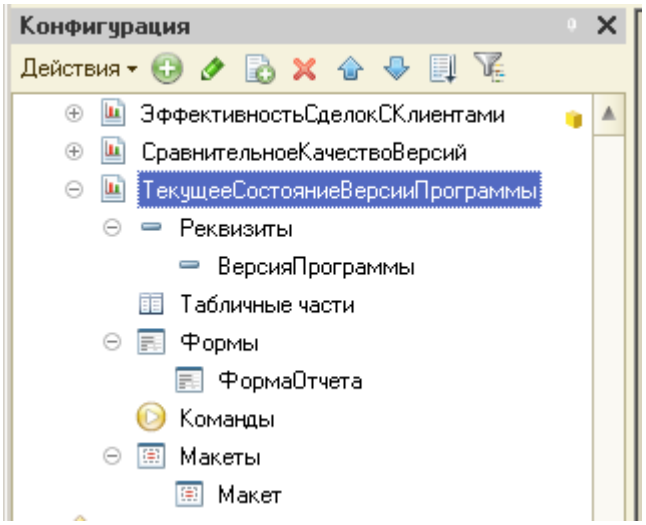

Рисунок 2.8 – Вид отчета «Текущее состояние версии»
в конфигураторе «1С:Предприятие»

На рисунке 2.8 видно, что отчет «Текущее состояние версии» использует один реквизит «ВерсияПрограммы» типа «СправочникСсылка.Релизы», который заполняется в форме отчета «ФормаОтчета». Результаты отчета отображаются в виде табличного документа, сформированного на основе макета «Макет».

2.2.3 Обработки

Для заполнения характеристик качества версий программного обеспечения для пользователей подсистемы контроля качества производства программного обеспечения предусмотрена обработка «Отметки качества». Данная обработка формирует запросы к регистру сведений «Качество версии программного обеспечения» для выборки значений качества для указанной версии программы, а также предоставляет возможность сохранить измененные значения. На рисунке 2.9 изображен фрагмент окна конфигурации с данной обработкой.

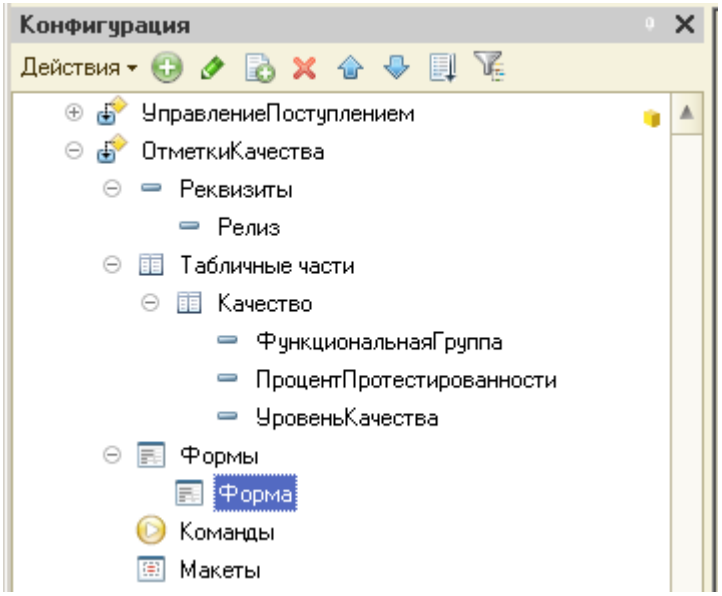

Рисунок 2.9 – Вид обработки «Отметки качества»
в конфигураторе «1С:Предприятие»

На рисунке 2.9 видно, что обработка «Отметки качества» использует один реквизит «Релиз» типа «СправочникСсылка.Релизы», который заполняется в форме обработки «Форма».

2.2.4 Подсистемы

Для реализации функций контроля качества производства программного обеспечения на основе стандартной конфигурации «1С:Управление торговлей» создана подсистема «Качество», изображенная на рисунке 2.10, которая включает в себя все метаданные, необходимые для осуществления процесса контроля качества на предприятии.

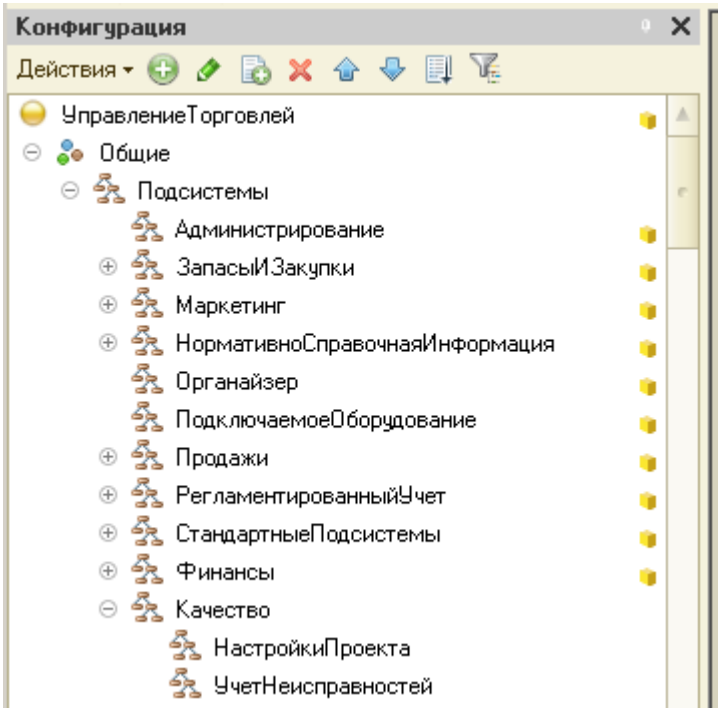

Рисунок 2.10 – Вид подсистемы «Качество»
в конфигураторе «1С:Предприятие»

Как видно из рисунка 2.10, подсистема имеет в своем составе дочернюю подсистему «Настройки проекта» и «Учет неисправностей».

К подсистеме «Учет неисправностей» относятся справочники «Версии программного обеспечения» и «Неисправности программы».

К подсистеме «Настройки проекта» относятся справочники «Функциональные группы» и «Проекты».

2.2.5 Роли

Для удобства администрирования пользователей «1С:Предприятие» и предоставления прав участникам разработки программного обеспечения в приложении предусмотрена отдельная роль «Подсистема контроль качества ППО», которая включает в себя права на редактирование и просмотр всех основных объектов подсистемы «Качество».

2.2.6 Общие картинки

Для обозначения подсистемы «Качество» на главной форме приложения в стандартную конфигурацию «1С:Управление торговлей» была добавлена картинка

«Контроль качества» разрешением 128 х 128 пикселей, представленная на рисунке 2.11.

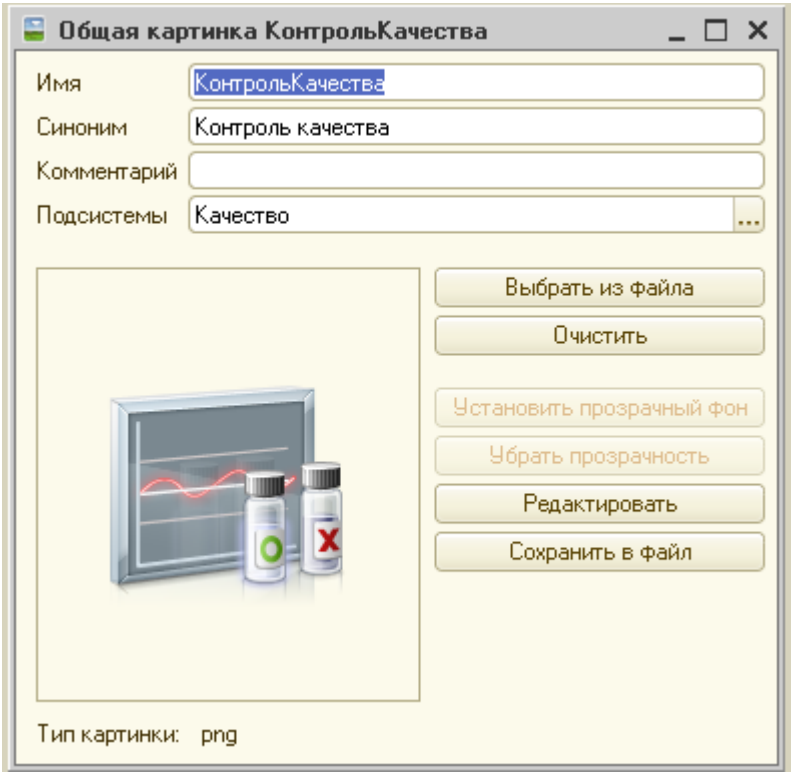

Рисунок 2.11 – Вид общей картинки «Контроль качества»
в конфигураторе «1С:Предприятие»

Как показано на рисунке 2.11, картинка имеет тип «png» и относится к подсистеме «Качество».

2.2.7 Перечисления

Объекты прикладного решения Перечисление позволяют хранить в информационной базе наборы значений, которые не изменяются в процессе работы прикладного решения.

В подсистеме «Качество» используются следующие перечисления:

- статусы релизов;
- уровни серьезности неисправностей;
- уровни приоритетов неисправностей;
- статусы неисправностей;
- резолюции для неисправностей;
- уровни качества.

Перечисление «Статусы релизов» представлено на рисунке 2.12.

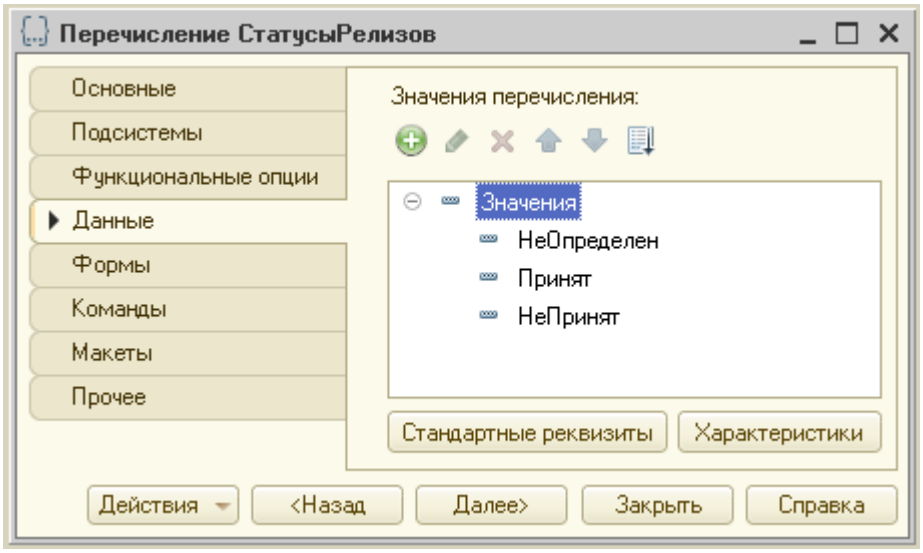

Рисунок 2.12 – Значения перечисления «Статусы релизов»

На рисунке 2.12 видно, что данное перечисление состоит из трех значений. Объект «Перечисление.СтатусыРелизов» использован в «Справочник.Релизы.Реквизит.Статус.Тип».

Перечисление «Уровни серьезности неисправностей» представлено на рисунке 2.13.

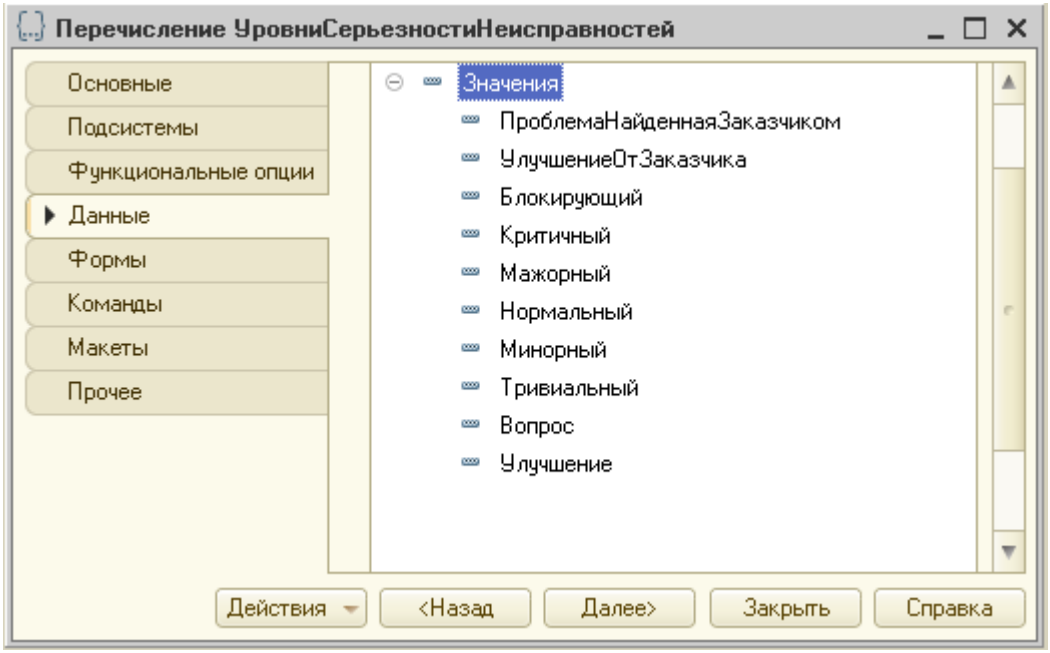

Рисунок 2.13 – Значения перечисления «Уровни серьезности неисправностей»

На рисунке 2.13 видно, что данное перечисление состоит из десяти значений. Объект «Перечисление.УровниСерьезностиНеисправностей» использован в «Справочник.Баги.Реквизит.Серьезность.Тип».

Перечисление «Уровни приоритетов неисправностей» представлено на рисунке 2.14.

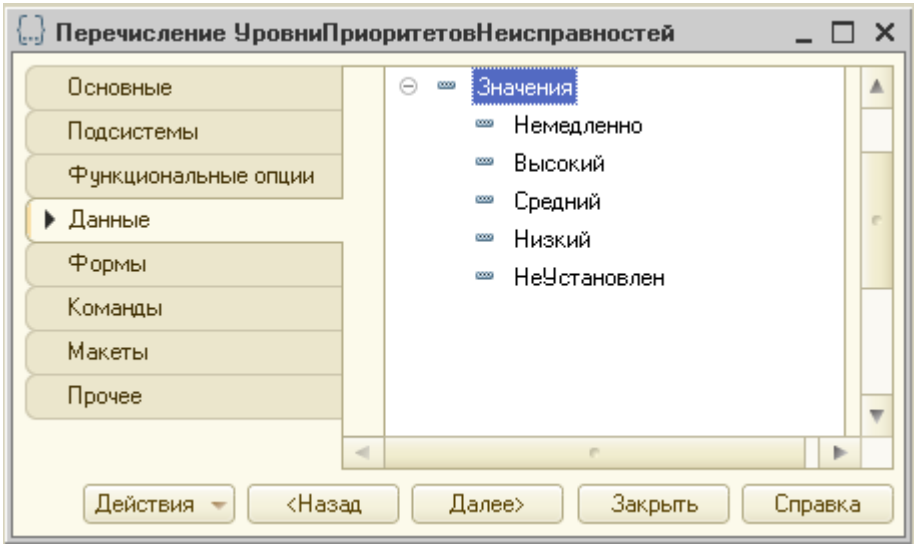

Рисунок 2.14 – Значения перечисления «Уровни приоритетов неисправностей»

На рисунке 2.14 видно, что данное перечисление состоит из пяти значений. Объект «Перечисление.УровниПриоритетовНеисправностей» использован в «Справочник.Баги.Реквизит.Приоритет.Тип».

Перечисление «Статусы неисправностей» представлено на рисунке 2.15.

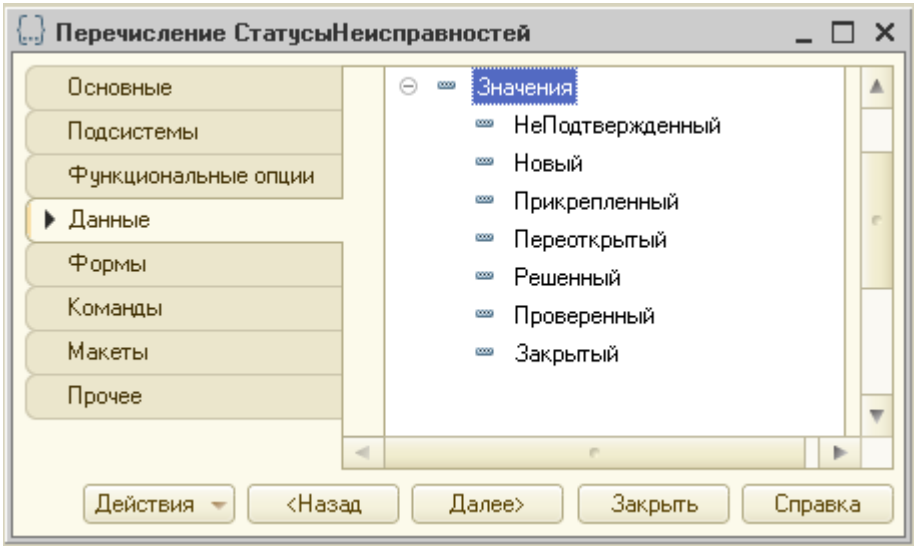

Рисунок 2.15 – Значения перечисления «Статусы неисправностей»

На рисунке 2.15 видно, что данное перечисление состоит из семи значений. Объект «Перечисление.СтатусыНеисправностей» использован в «Справочник.Баги.Реквизит.Статус.Тип».

Перечисление «Резолюции для неисправностей» представлено на рисунке 2.16.

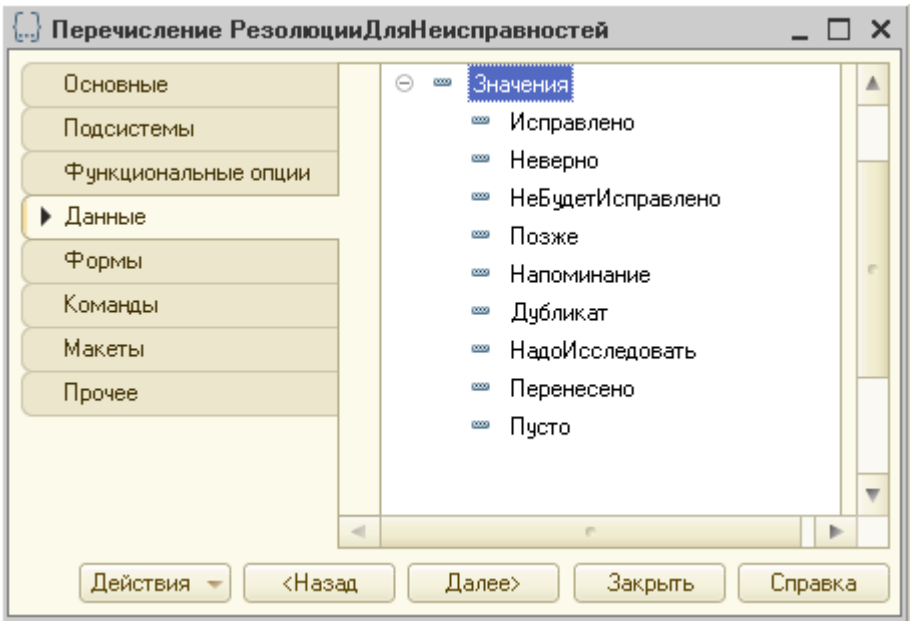

Рисунок 2.16 – Значения перечисления «Резолюции для неисправностей»

На рисунке 2.16 видно, что данное перечисление состоит из девяти значений. Объект «Перечисление.РезолюцииДляНеисправностей» использован в «Справочник.Баги.Реквизит.Резолюция.Тип».

Перечисление «Уровни качества» представлено на рисунке 2.17.

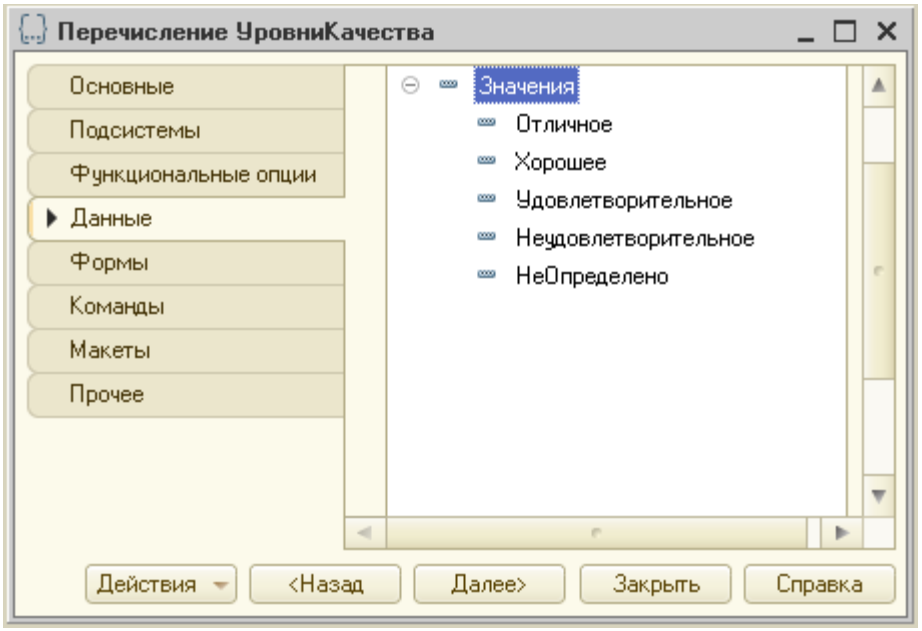

Рисунок 2.17 – Значения перечисления «Уровни качества»

На рисунке 2.17 видно, что данное перечисление состоит из девяти значений. Объект "Перечисление.УровниКачества" использован в «Обработка.ОтметкиКачества.ТабличнаяЧасть.Качество.Реквизит.УровеньКачества.Тип» и «РегистрСведений.КачествоРелиза.Ресурс.УровеньКачества.Тип».

2.2.8 Регистры сведений

Для хранения присвоенных значений уровня качества функциональных групп разных версий приложений разработан регистр сведений «Качество версии программного обеспечения», представленный на рисунке 2.18.

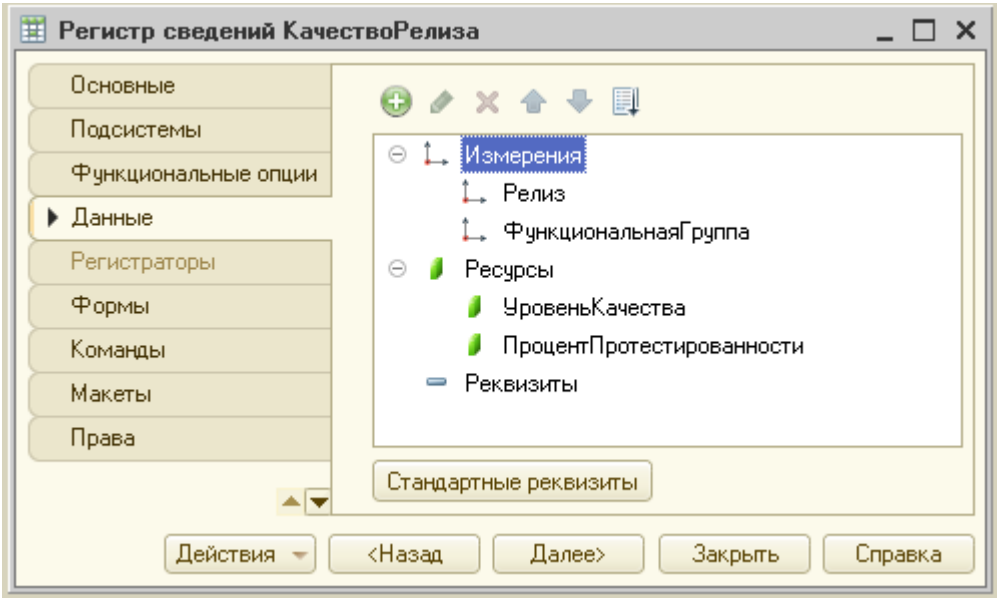

Рисунок 2.18 – Данные регистра сведений «Качество версии программного обеспечения»

На рисунке 2.18 видно, что он состоит из измерений «Релиз» типа «СправочникСсылка.Релизы», «Функциональная группа» типа «СправочникСсылка.ФункциональныеГруппы» и ресурсов «Уровень качества» типа «ПеречислениеСсылка.УровниКачества», «Процент протестированности» типа «Число».

2.3 Детальная разработка алгоритмов отдельных подзадач

2.3.1 Реализация построения отчета «Сравнительное качество версий»

Для формирования отчета «Сравнительное качество версий» был реализован алгоритм, который изображен на рисунке 2.19.

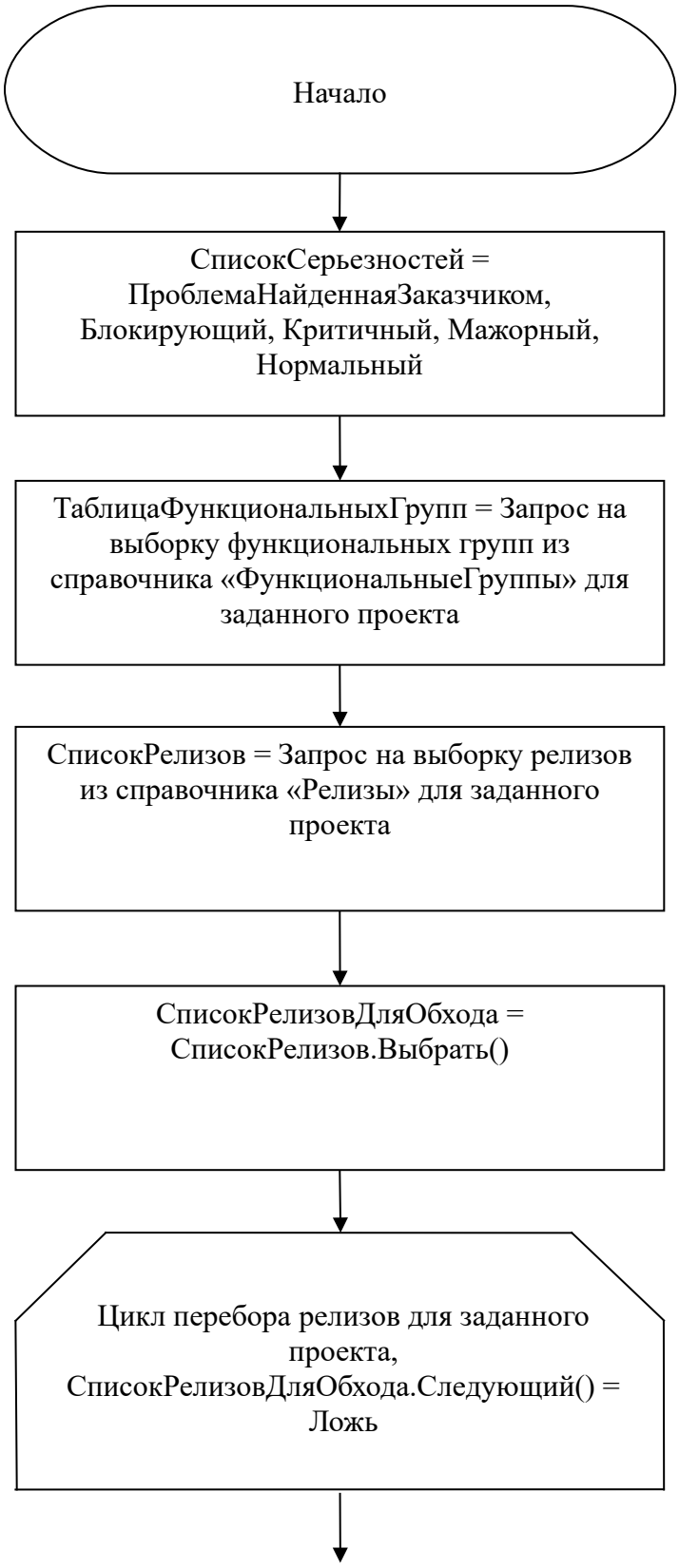

Рисунок 2.19 – Схема алгоритма

отчета «Сравнительное качество версий»

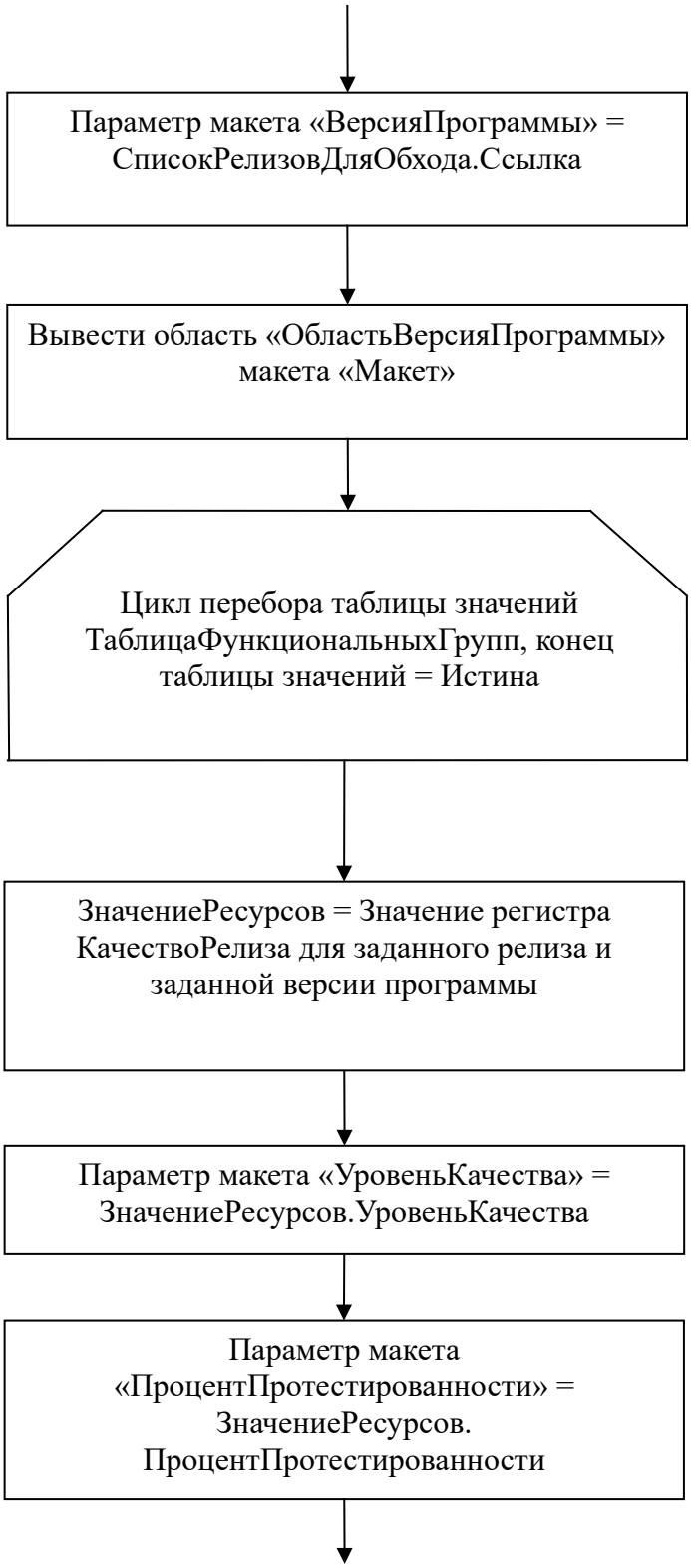

Рисунок 2.19 – Схема алгоритма

отчета «Сравнительное качество версий»

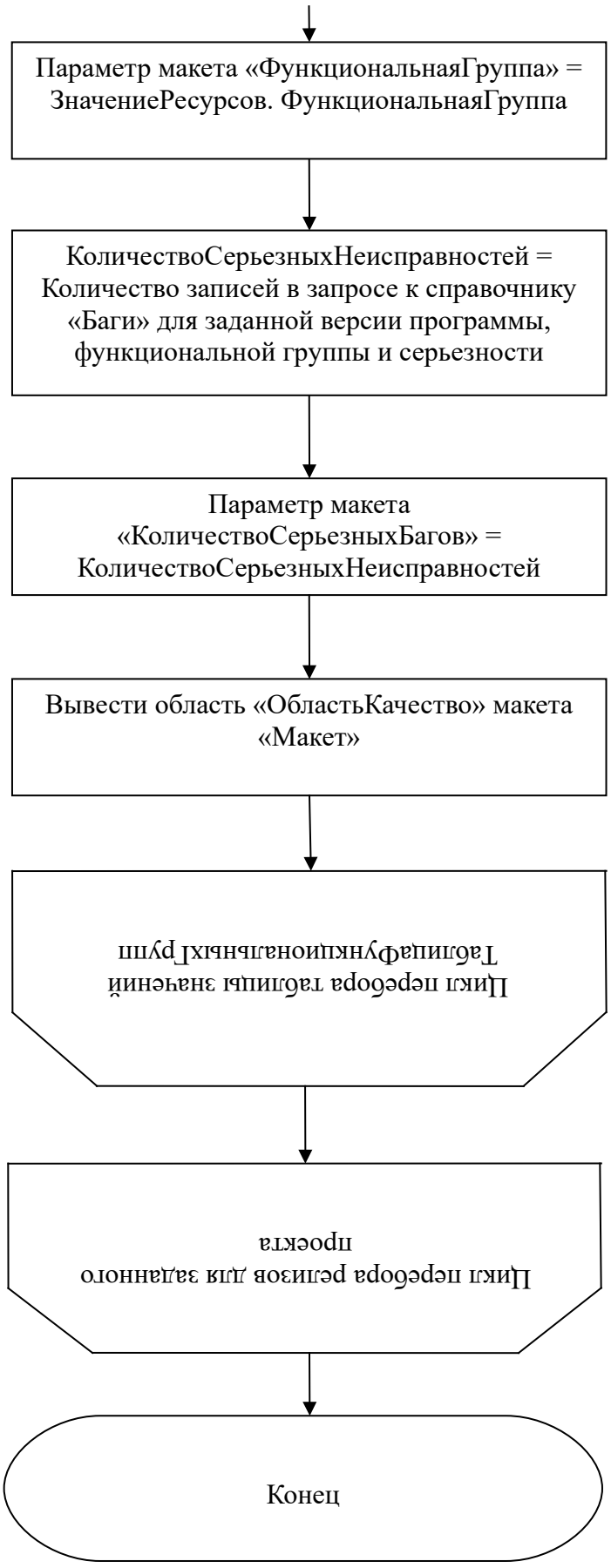

Рисунок 2.19 – Схема алгоритма

отчета «Сравнительное качество версий»

Алгоритм, изображенный на рисунке 2.19, реализован в модуле формы отчета и предназначен для обработки полученных строк таблиц значений качества, выбранных для всех версий программы для заданного проекта, и вывода результата в табличный документ, который потом будет отображен пользователю. Из рисунка 2.19 видно, что в табличный документ выводится информация об уровне качества версий, проценте протестированности и функциональной группе. Листинг модуля, формирующего отчет, приведен в приложении 1 к пояснительной записке.

2.3.2 Реализация построения отчета «Текущее состояние версии»

Для формирования отчета «Текущее состояние версии» был реализован алгоритм, часть которого изображена на рисунке 2.20.

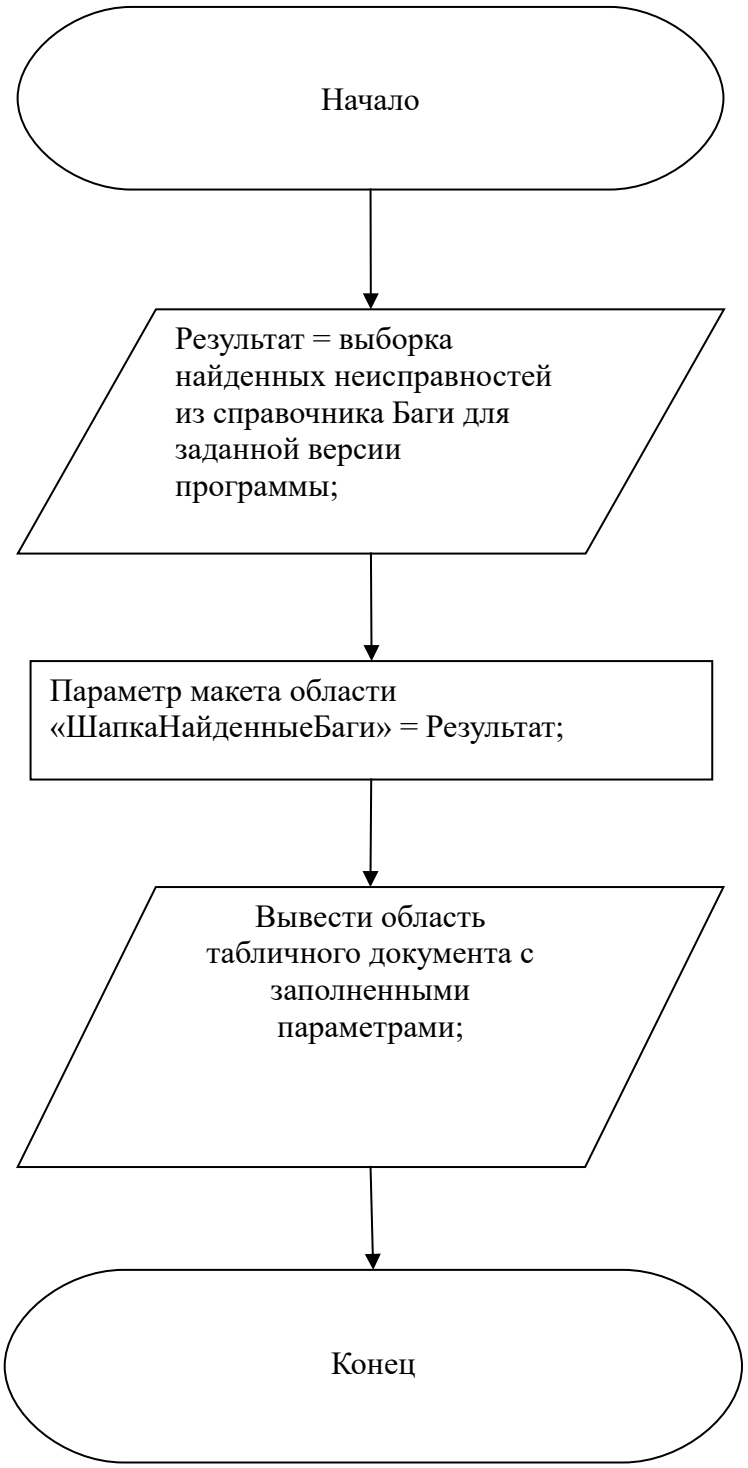

Рисунок 2.20 – Схема алгоритма вывода результата отчета «Текущее состояние версии»

Алгоритм, изображенный на рисунке 2.20, реализован в модуле формы отчета и предназначен для поиска неисправностей, которые были найдены для заданной версии программы, и вывода результата в табличный документ, который потом будет отображен пользователю. Листинг модуля, формирующего отчет, приведен в приложении 1 к пояснительной записке.

Данный отчет для заданной версии программы предоставляет пользователю:

- информацию о найденных ошибках;
- информацию об исправленных ошибках;
- список исправленных ошибок, которые были заново обнаружены;
- список проверенных исправленных ошибок;
- информацию о текущем состоянии качества.

3 Эксплуатационно-технологическая часть

3.1 Системные требования

3.1.1 Программные требования

В комплекс программных требований для работы конфигурации контроля качества программного обеспечения входят следующее ПО:

- технологическая платформа 1С:Предприятия;

а) тонкий клиент и толстый клиент;

б) веб клиент;

- сервер 1С:Предприятия;
- сервер баз данных;
- веб сервер.

Тонкий и толстый клиент поддерживают следующие операционные системы семейства Windows:

- Windows 7;
- Windows Server 2008 R2;
- Windows Server 2008;
- Windows Server 2003;
- Windows Server 2000;
- Windows Vista;
- Windows XP;
- Windows 2000.

Для работы веб клиента можно использовать операционные системы как Windows, так и Linux и Mac OS X. На них приложение будет стабильно работать, если пользователь будет использовать следующие поддерживаемые браузеры:

- «Mozilla Firefox» 3.0 и выше для Windows и Linux;
- «Microsoft Internet Explorer» 6.0, 7.0, 8.0, 9.0 для Windows;
- «Google Chrome» 4 и выше для Windows;
- «Safari» 4.0.5 и выше для Mac OS X и Windows.

Перечень поддерживаемых версий операционных систем представлен в таблице 3.1.

Таблица 3.1 – Перечень версий операционных систем, поддерживаемых веб клиентом

Операционная система	Версия
Microsoft Windows	Windows 7
Microsoft Windows	Windows Server 2008 R2
Microsoft Windows	Windows Server 2008
Microsoft Windows	Windows Server 2003
Microsoft Windows	Windows Server 2000
Microsoft Windows	Windows Vista
Microsoft Windows	Windows XP
Microsoft Windows	Windows 2000
Linux	Linux
Mac OS X	Mac OS X 10.5 и выше

Как видно из таблицы 3.1, доступ к конфигурации для осуществления контроля качества производства программного обеспечения может быть осуществлен с широкого круга версий операционных систем, что упрощает доступ для внесения данных в информационную систему в процессе как тестирования, так и разработки.

Для работы приложения в режиме веб-клиента сервер 1С:Предприятие должен использовать веб-сервер Apache, или IIS. Список поддерживаемых веб-серверов приведен в таблице 3.2.

Таблица 3.2 – Перечень поддерживаемых версий веб-серверов

Операционная система	Версия
Microsoft Windows	IIS 5.1
Microsoft Windows	IIS 6.0
Microsoft Windows	IIS 7.0
Microsoft Windows	IIS 7.5
Microsoft Windows	Apache 2.0
Microsoft Windows	Apache 2.2
Linux	Apache 2.0
Linux	Apache 2.2

Из таблицы 3.2 видно, что платформа «1С:Предприятие» версии 8.2 поддерживает работу с веб-серверами как IIS, так и с бесплатным Apache.

3.1.2 Аппаратные требования

Аппаратные требования для работы платформы «1С:Предприятие» версии 8.2 различаются в зависимости от режима использования:

В режиме веб-клиента нет особых аппаратных требований, т.к. для доступа к базе данных на стороне клиента используется браузер, который уже удовлетворяет незначительным аппаратным требованиям. Единственное условие – это наличие доступа в интернет или к локальной сети, для чего необходим сетевой адаптер LAN, Wi-Fi или интернет модем.

В режиме толстого клиента интернет соединение не требуется, однако возрастает и нагрузка на аппаратную часть компьютера. Для стабильной работы приложения в этом режиме необходимо:

- процессор не ниже Intel Pentium 1500;
- оперативная память не менее 512 Мб;
- свободное место на диске не менее 250 Мб.

В режиме тонкого клиента требуется сетевое соединение, однако нагрузка на аппаратную часть компьютера снижается. Для стабильной работы приложения в этом режиме необходимо:

- процессор не ниже Intel Pentium 1500;
- оперативная память не менее 512 Мб;
- свободное место на диске не менее 250 Мб;
- сетевой адаптер.

В режиме толстого и тонкого клиента для корректного отображения информации и нормальной работы с конфигурацией рекомендуется использовать монитор с диагональю 19 дюймов и разрешение экрана не меньше 1024*768.

В режиме сервера необходима следующая минимальная конфигурация:

- Intel Pentium IV/Xeon 2800 МГц, рекомендуется два и более процессоров;
- оперативная память 1024 Мбайт;
- сетевой адаптер.

Для безопасной работы с информационной базой желательно использование на сервере источников бесперебойного питания и регулярного резервированного копирования данных.

3.2 Установка приложения

3.2.1 Установка платформы «1С:Предприятие 8.2»

Так как система программ «1С:Предприятие 8.2» использует для защиты своих авторских прав аппаратные ключи защиты, для работы конфигурации потребуется поддержка лицензии на использование платформы с помощью HASP4 Net, который использует USB-порт для соединения с компьютером пользователя. Если в программе предполагается работать нескольким пользователям, то потребуется использование утилиты HASP License Manager, которая управляет многопользовательскими лицензиями, а также обеспечивает работу в пределах одного сегмента локальной сети нескольких пользователей (при условии одновременного запуска в сегменте не более N экземплярах при наличии N-пользовательской лицензии).

В папке дистрибутива платформы 1С есть папки дистрибутива для Linux и Windows. Мы рассмотрим установку платформы 1С под Windows для режимов толстого клиента и веб клиента, поэтому нужно открыть ее и запустить setup.exe. В установщике нужно выбрать «Модули расширения веб-сервера» и «1С:Предприятие», выбрать путь для установки программы и после успешной установки программы закрыть окно установщика.

3.2.2 Установка конфигурации для контроля качества производства программного обеспечения

Установка подсистемы «Качество» производится в существующую стандартную конфигурацию «1С:Управление торговлей». Для этого необходимо установить сначала ее.

Установка конфигурации «1С:Управление торговлей» производится аналогично установке любой типовой конфигурации «1С:Предприятие 8.2». Для установка шаблона конфигурации запустите с дистрибутивного диска файл setup.exe. Для продолжения установки нажмите «Далее>». Программа предложит указать путь к каталогу шаблонов. Вы можете согласиться с предложенным вариантом или выбрать другой каталог. Для перехода к следующему шагу установки нажмите кнопку «Далее>». Произойдет установка программы, и по завершению будет выведено сообщение об успешной установке конфигурации. Нажмите «Готово». На этом установка конфигурации завершена.

Для установки подсистемы «Качество» необходимо в меню «Конфигурация» выбрать пункт «Сравнить, объединить с конфигурацией из файла…», где нужно будет выбрать файл конфигурации, содержащий разработанную подсистему. Появится окно «Сравнение, объединение» которое представлено на рисунке 3.1, в котором нужно нажать кнопку «Выполнить».

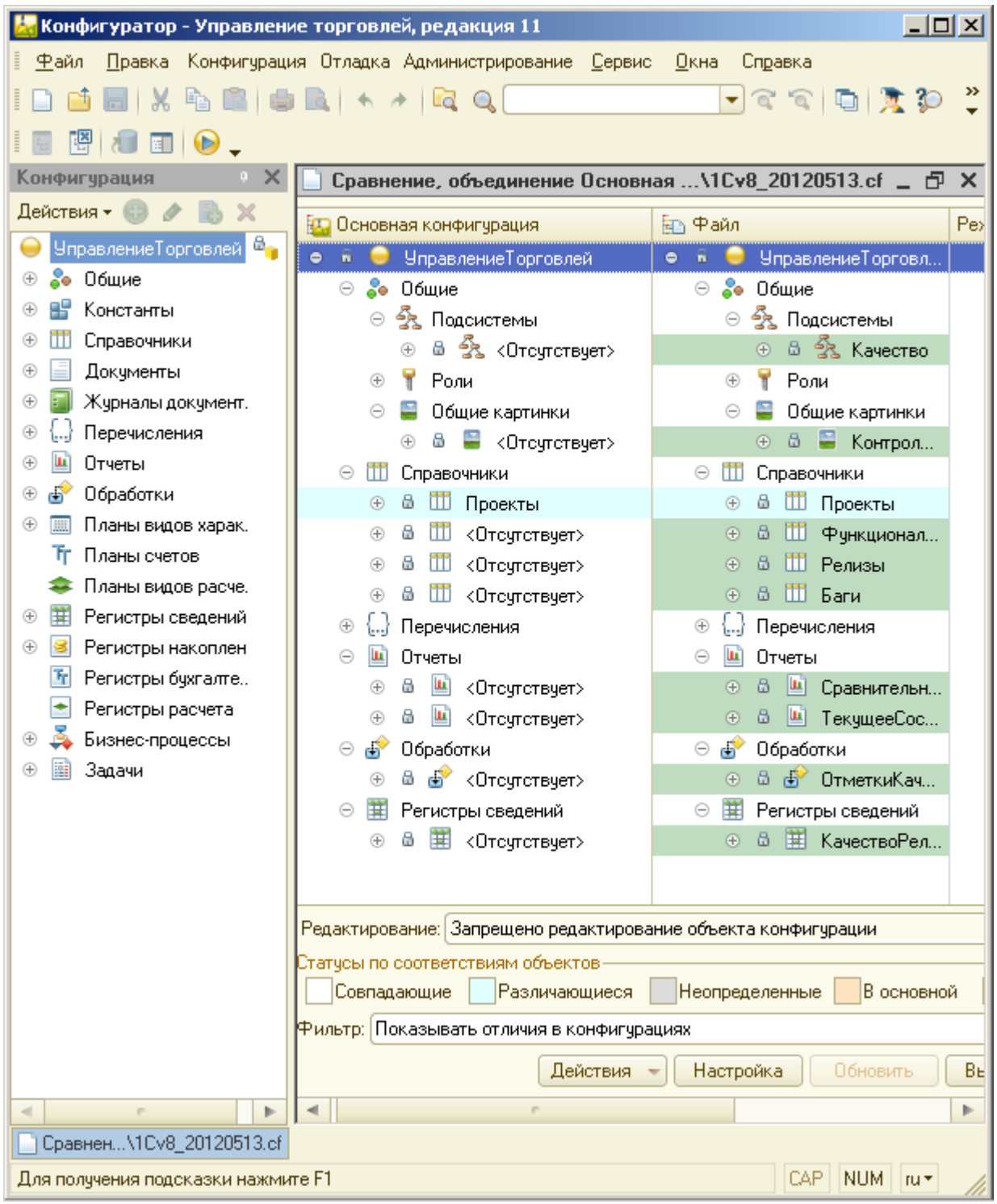

Рисунок 3.1 – Форма «Сравнение, объединение» при
в процессе установки конфигурации

На рисунке 3.1 видно выделение зеленым цветом тех объектов подсистемы «Качество», которые будут добавлены в стандартную конфигурацию «1С:Управление торговлей».

Веб-сервер Apache является свободно распространяемым программным обеспечением и может быть свободно скачен с официального сайта http://httpd.apache.org. Перед загрузкой выберите операционную систему вашего сервера (Windows).

Процедура установки Apache достаточно проста. После скачивания небольшого дистрибутива (5.2 MB) в формате MSI веб-сервера распакуйте дистрибутив и запустите файл «setup.exe».

После начала установки заполните параметры вашего веб-сервера. Укажите адрес электронной почты администратора системы, в полях Network Domain, Server Name укажите имена ваших серверов, если вы не планируете конфигурировать сервер для доступа из внешних сетей, то можно указать произвольные имена, например myServer, 1c_doc и т.д.

После ввода параметров нажмите «Next» и дождитесь окончания установки. Состояние веб-сервер Apache в системе Windows отображается в системном окне рядом с часами, наличие зеленого треугольника говорит о том, что сервер запущен и корректно работает.

Обычно никаких дополнительных настроек Apache в системе Windows нам делать не придется, конфигурация «1С:Управление торговлей» будет опубликована на веб-сервере автоматически, из конфигуратора «1С:Предприятие 8.2».

Для включения возможности работать с конфигурацией в режиме веб-клиента и использовать все возможности управляемых формы версии 8.2 нам необходимо опубликовать (экспортировать) конфигурацию на веб-сервер. Для публикации конфигурации на веб-сервер необходимо открыть базу данных в режиме «Конфигуратор» и в меню «Администрирование» выбрать пункт «Публикация на веб-сервере».

В открывшемся окне «Публикация на сервере» введите необходимые данные.

Имя - это имя вашей базы данных латинскими буквами. Данное имя будет использовать пользователям в веб-броузерах пользователями при вводе URL-адреса.

Веб-сервер - выберите Apache 2.2.

Каталог - место для физического расположения опубликованных системных файлов в директории вашего веб-сервера.

После ввода имени и каталога нажмите кнопку «Опубликовать».

3.3 Руководство пользователя

3.3.1 Введение

Для запуска программы «1С:Управление Торговлей 8.2» войдите в меню Пуск и выберите пункт Программы. В списке программ, установленных на компьютере, выберите 1С:Предприятие 8.2. Щелкните на пункт меню 1С:Предприятие.

В результате откроется диалог «Запуск 1С:Предприятия», как показано на рисунке 3.2.

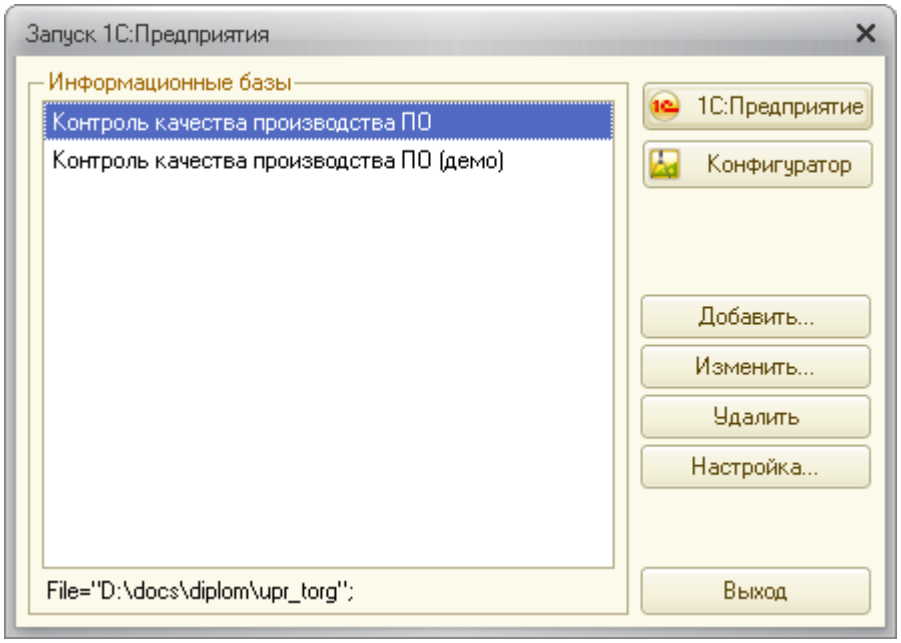

Рисунок 3.2 – Форма «Запуск 1С:Предприятие»

В окне «Запуск 1С:Предприятия» можно выбрать режим, в котором пользователь будет работать с базой 1С. Для выбора режима предназначены кнопки «1С:Предприятие» и «Конфигуратор», запускающие соответствующие режимы.

3.3.2 Работа с пользовательским разделом приложения

Выбрав режим «1С:Предприятие» пользователю необходимо будет ввести имя пользователя и пароль для доступа к базе данных. Если у пользователя есть право

использовать подсистему «Качество», тогда на главном окне появится соответствующая закладка, как показано на рисунке 3.3.

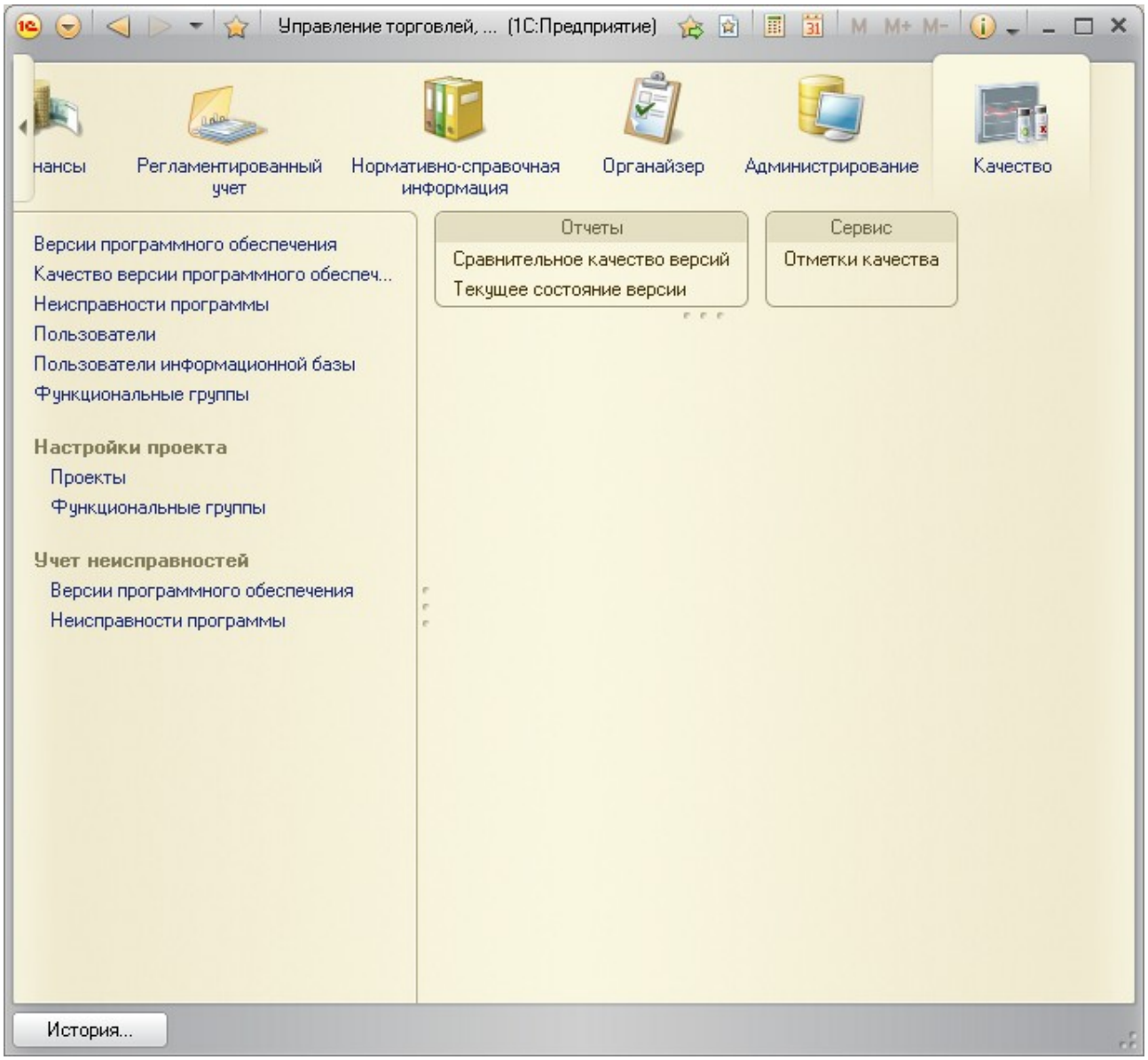

Рисунок 3.3 – Главная форма на вкладке «Качество»

На форме доступны элементы интерфейса для доступа к функциям подсистемы «Качество». Для начала необходимо создать проект, который требует осуществления контроля качества производства. На рисунке 3.4 изображен справочник проектов демонстрационной базы 1С, который используется для этих целей.

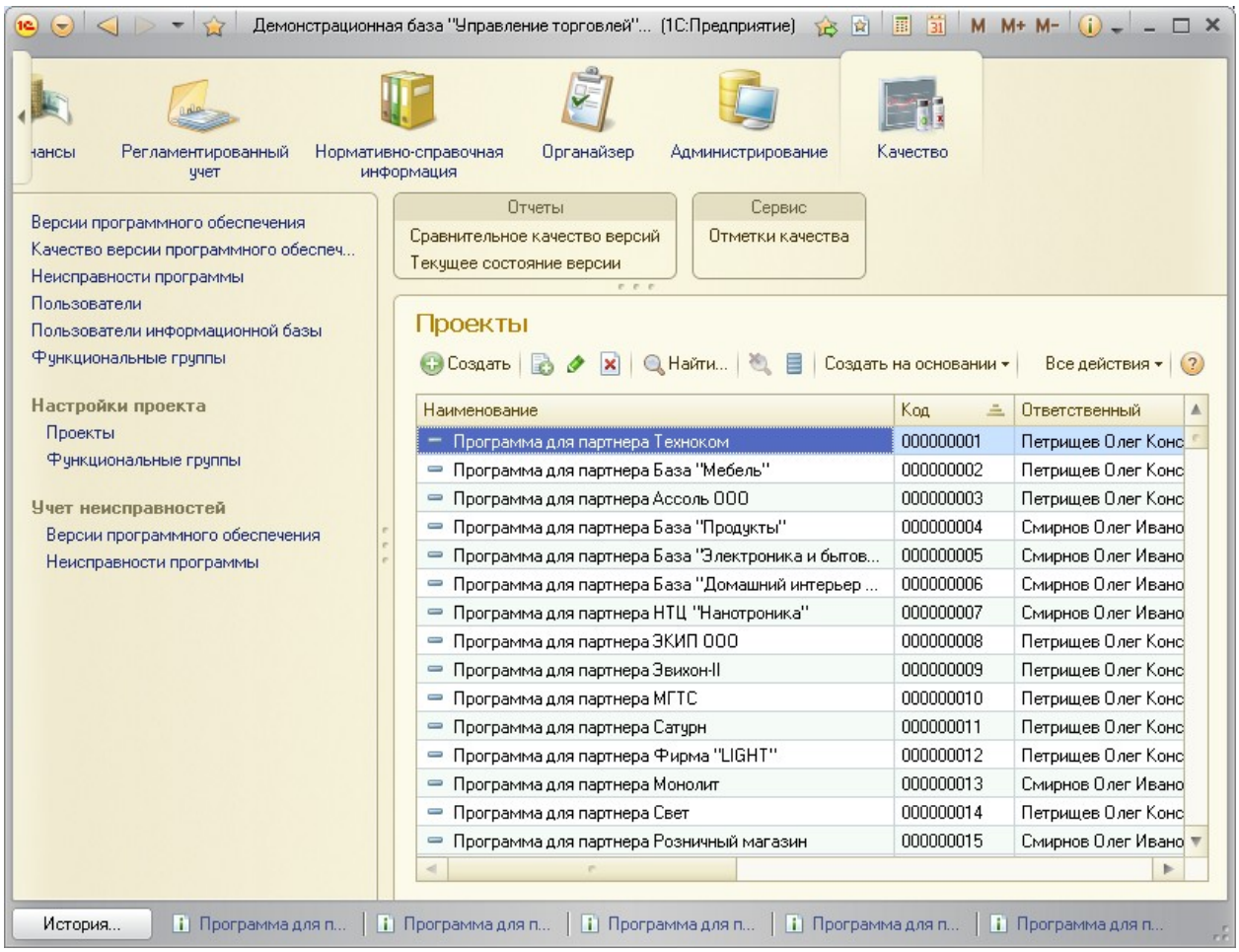

Рисунок 3.4 – Справочник «Проекты», открытый в подсистеме «Качество»

Данный справочник является стандартным для конфигурации «1С:Управление торговлей» и его заполнение описано в документации производителя.

Для детализации характеристик качества программного обеспечения предназначен справочник «Функциональные группы», изображенный на рисунке 3.5.

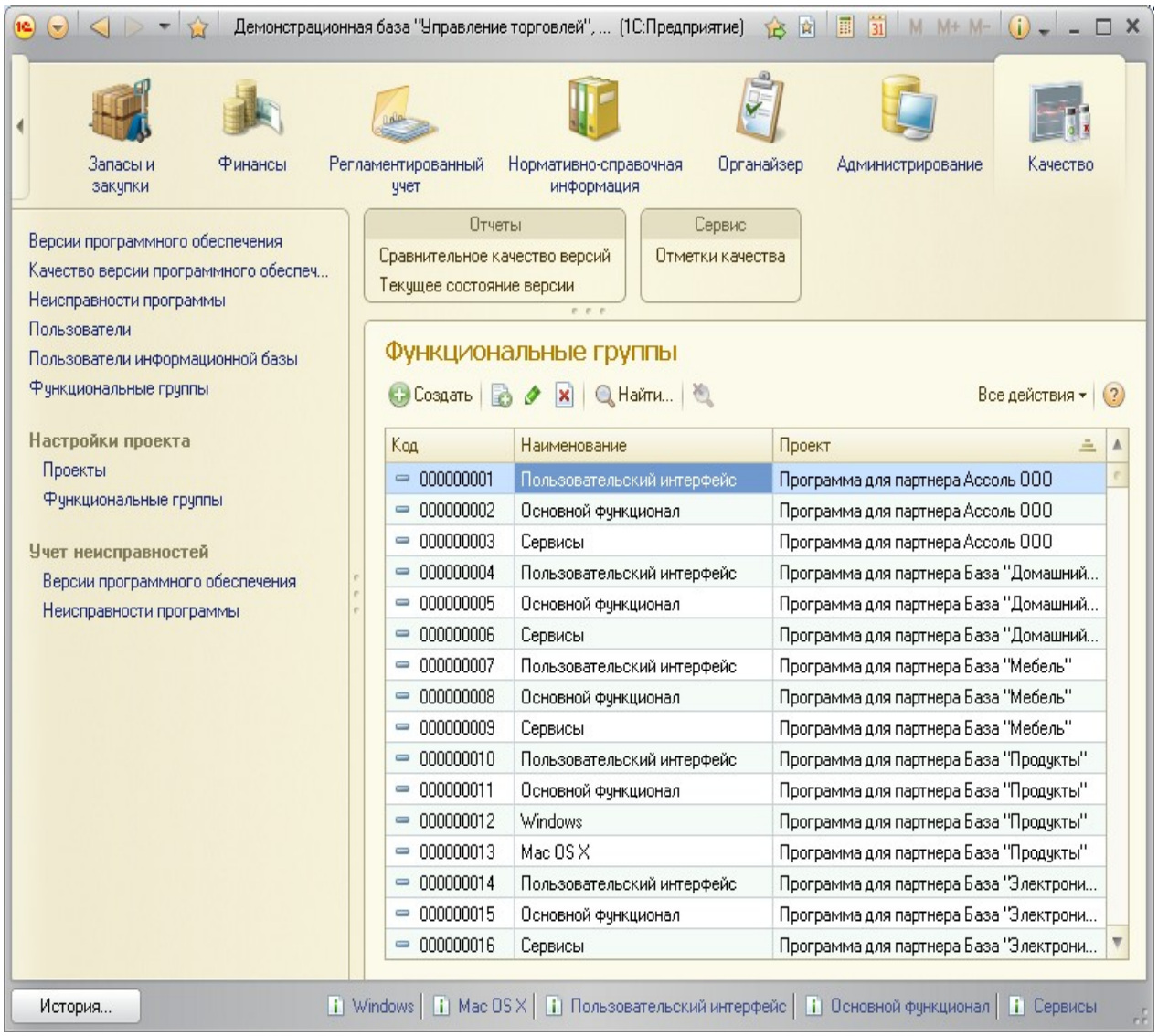

Рисунок 3.5 – Справочник «Функциональные группы» подсистемы «Качество»

На рисунке видно, что по умолчанию справочник отображает все ранее введенные функциональные группы, чтобы пользователь мог выбрать для новых проектов ранее создаваемые группы и скопировать их в новый проект. Для поиска функциональных групп для заданного проекта можно воспользоваться стандартным поиском по справочнику, окно которого приведено в рисунке 3.6.

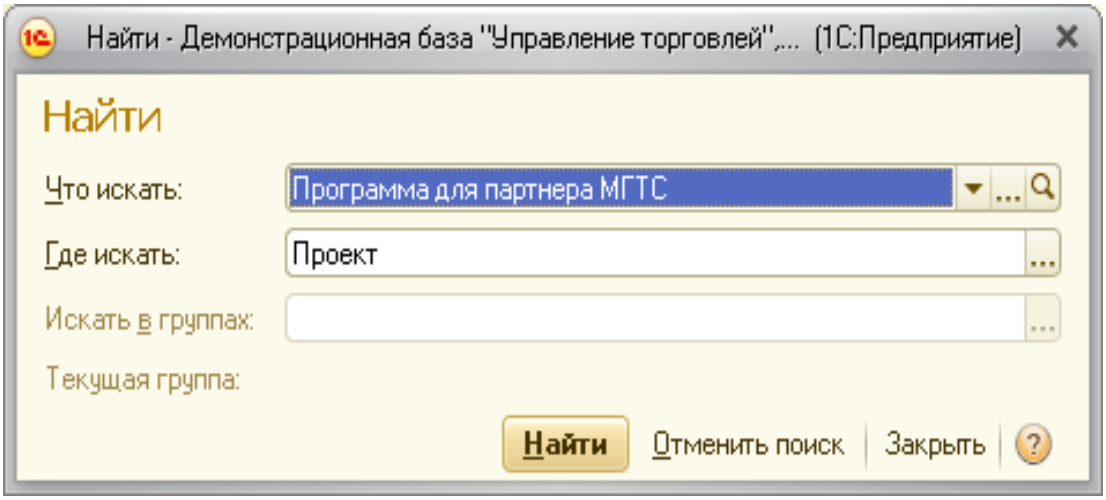

Рисунок 3.6 – Окно параметров поиска по справочнику «Функциональные группы»

В данном окне необходимо заполнить поля «Что искать» и «Где искать». Как видно на рисунке, для поиска функциональных групп по проекту нужно выбрать в поле «Где искать» Проект, а в поле «Что искать» - интересующий нас проект. По нажатию на кнопку «Найти» в окне справочника «Функциональные группы» отобразятся только интересующие пользователя элементы.

Для ввода новой функциональной группы необходимо выполнить команду «Создать» и ввести название функциональной группы и проект, к которому она относится, как показано на рисунке 3.7.

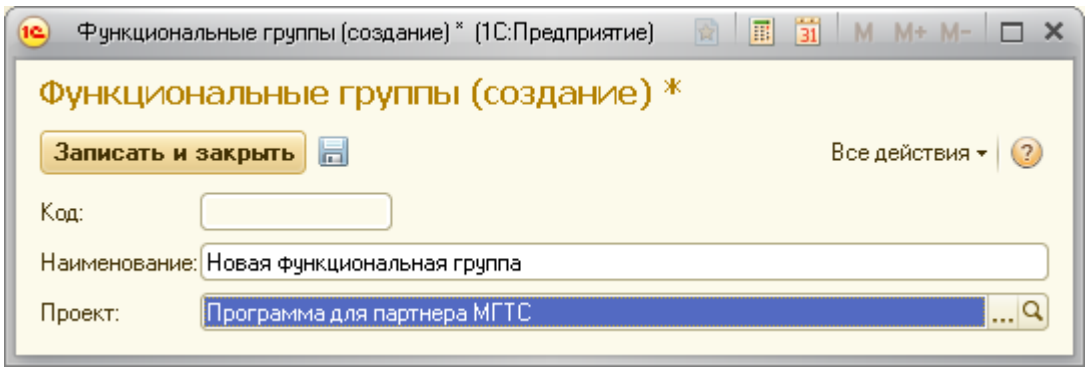

Рисунок 3.7 – Окно ввода нового элемента справочника «Функциональные группы»

Для сохранения введенной информации необходимо нажать кнопку «Записать и закрыть», либо закрыть окно крестиком для отмены ввода. Для редактирования

элементов справочника появляется окно, подобное изображенному на рисунке 3.7, в котором после введенных изменений нужно будет нажать кнопку «Записать и закрыть».

Для учета созданных версий программного обеспечения предназначен справочник «Версии программного обеспечения», изображенный на рисунке 3.8.

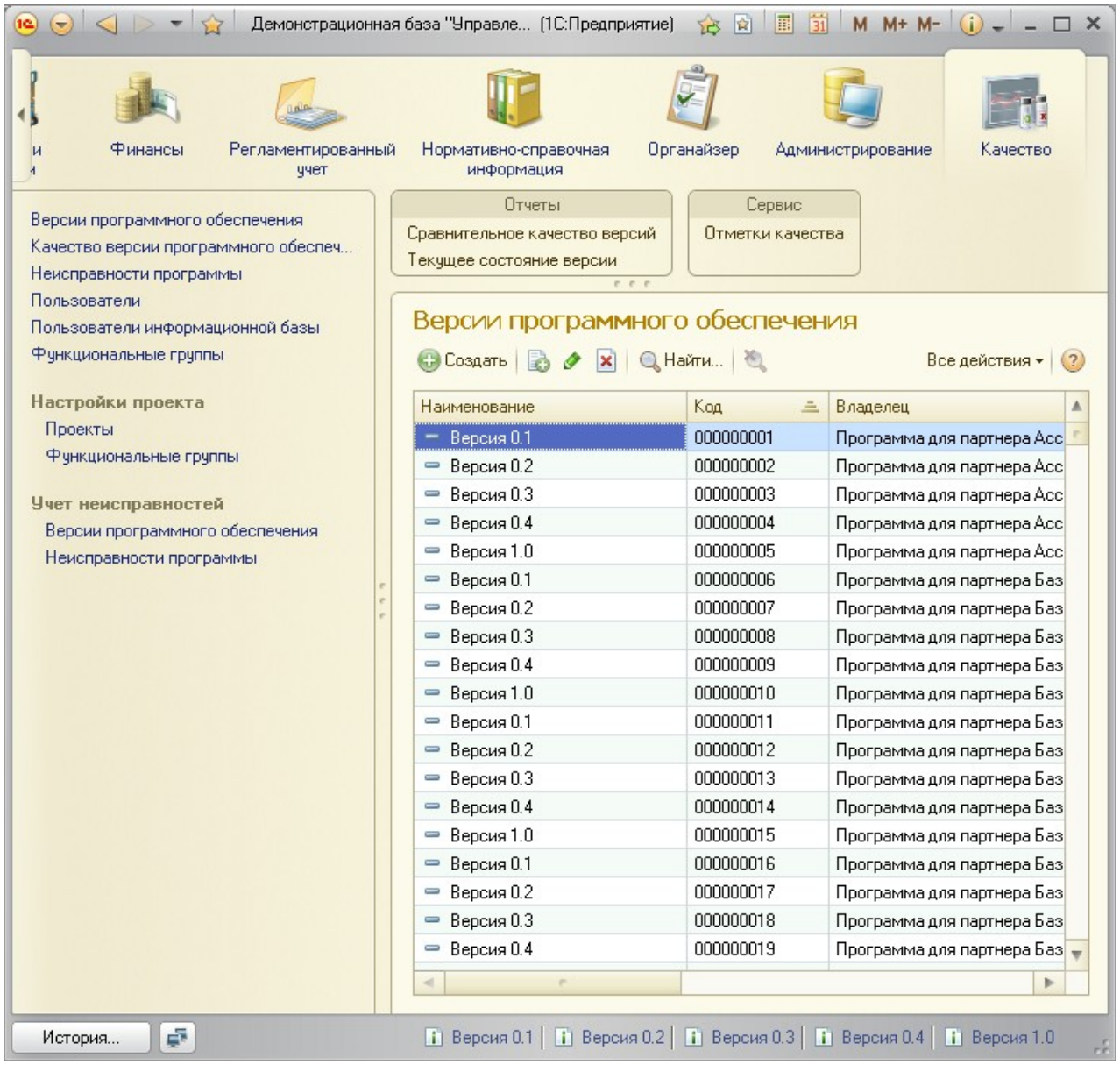

Рисунок 3.8 – Справочник «Версии программного обеспечения» подсистемы «Качество»

На рисунке видно, что по умолчанию справочник отображает все ранее введенные версии программ. Для поиска версий программ для заданного проекта можно воспользоваться стандартным поиском по справочнику, окно которого

аналогично приведенному в рисунке 3.6. В нем окне необходимо заполнить поля «Что искать» и «Где искать», выбрав в поле «Где искать» Владелец, а в поле «Что искать» - интересующий нас проект. По нажатию на кнопку «Найти» в окне справочника «Версии программного обеспечения» отобразятся только интересующие пользователя элементы.

Для ввода нового элемента справочника необходимо нажать на кнопку «Создать», после чего появится форма ввода нового элемента, представленная на рисунке 3.9.

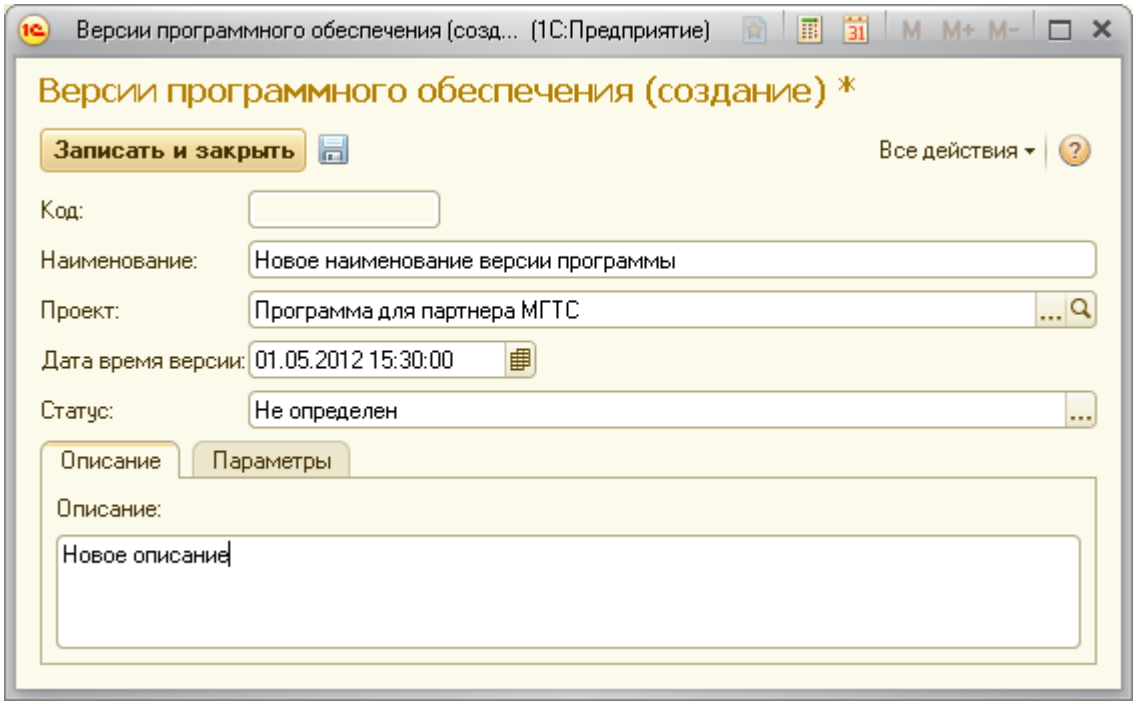

Рисунок 3.9 – Форма ввода нового элемента справочника «Версии программного обеспечения»

Для сохранения введенной информации необходимо нажать кнопку «Записать и закрыть», либо закрыть окно крестиком для отмены ввода. Для редактирования элементов справочника появляется окно, подобное изображенному на рисунке 3.9, в котором после введенных изменений нужно будет нажать кнопку «Записать и закрыть».

Для учета выявленных неисправностей предназначен справочник «Неисправности программы», изображенный на рисунке 3.10.

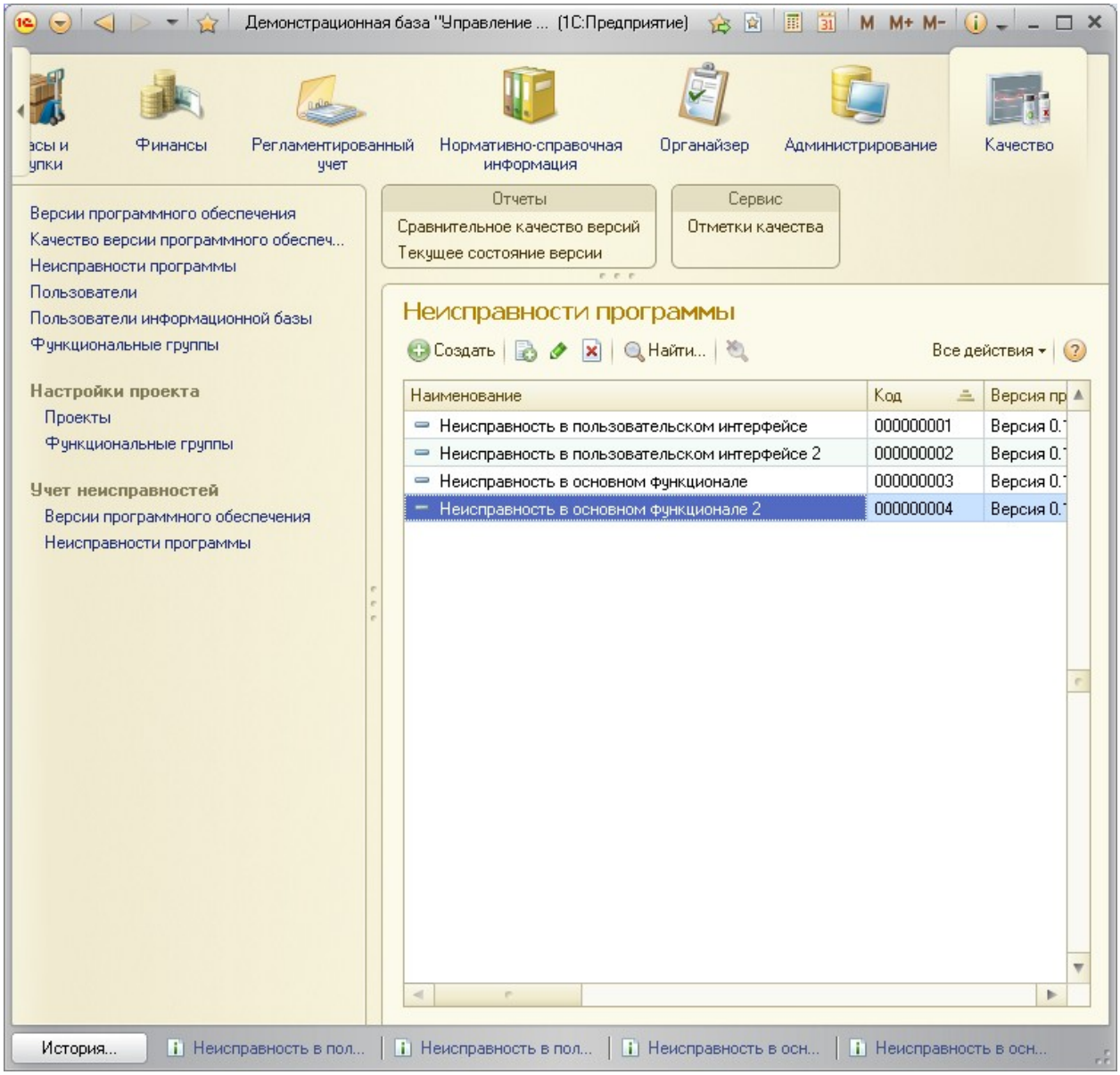

Рисунок 3.10 – Справочник «Неисправности программы» подсистемы «Качество»

Для поиска версий программ для заданного проекта можно воспользоваться стандартным поиском по справочнику, окно которого аналогично приведенному в рисунке 3.6. В нем окне необходимо заполнить поля «Что искать» и «Где искать». По нажатию на кнопку «Найти» в окне справочника «Неисправности программы» отобразятся только интересующие пользователя элементы.

Для ввода нового элемента справочника необходимо нажать на кнопку «Создать», после чего появится форма ввода нового элемента, представленная на рисунке 3.11.

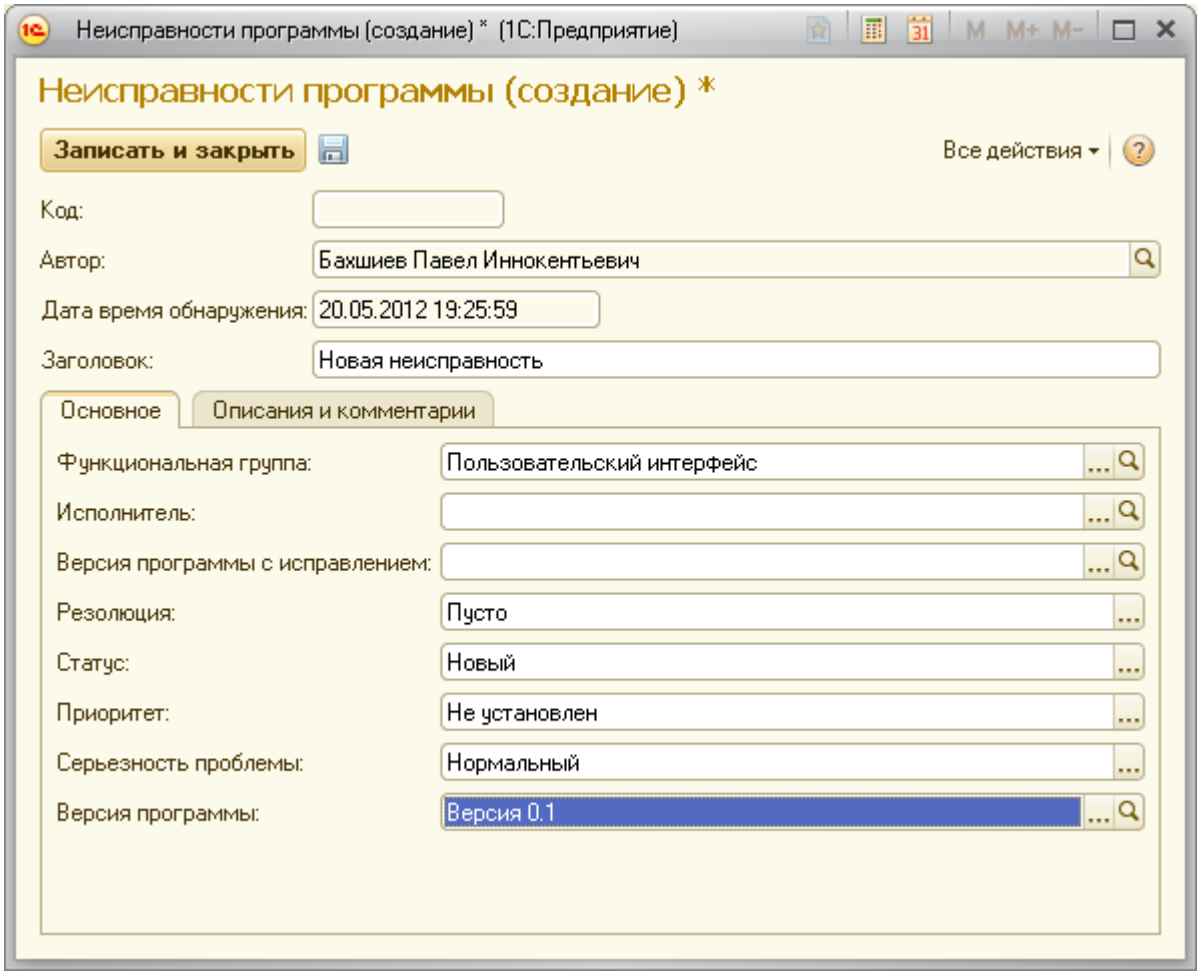

Рисунок 3.11 – Форма ввода нового элемента справочника «Неисправности программы»

Для сохранения введенной информации необходимо нажать кнопку «Записать и закрыть», либо закрыть окно крестиком для отмены ввода. Для редактирования элементов справочника появляется окно, подобное изображенному на рисунке 3.11, в котором после введенных изменений нужно будет нажать кнопку «Записать и закрыть».

Для отметок о качестве той или иной версии программы пользователю необходимо в группе «Сервис» запустить обработку «Отметки качества». В появившемся окне обработки, представленной на рисунке 3.12, нужно выбрать соответствующую версию программы.

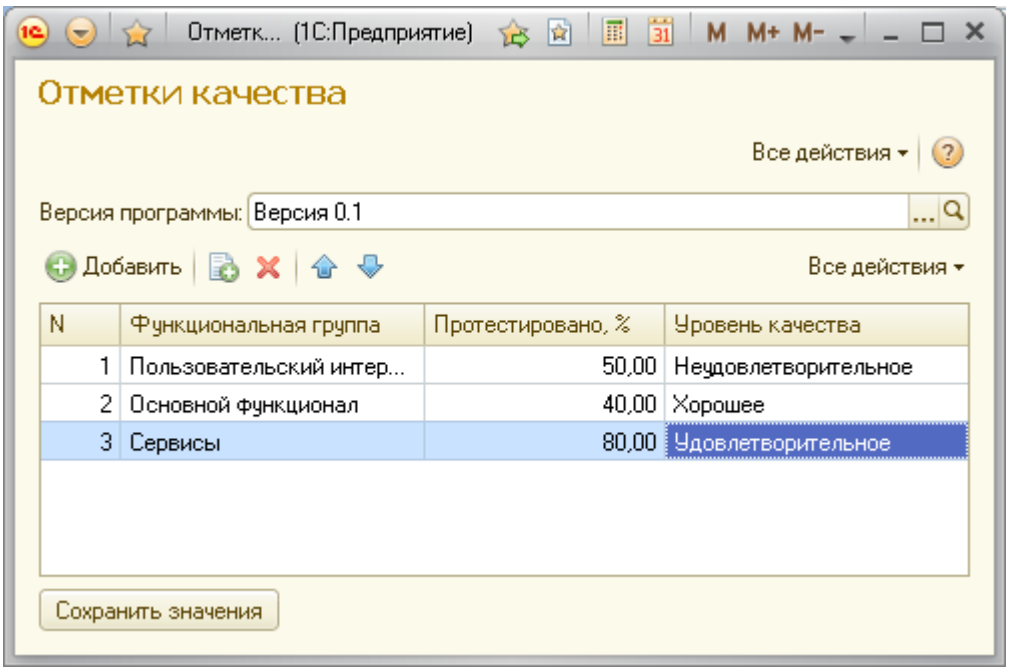

Рисунок 3.12 – Форма обработки «Отметки качества»

Для изменения значений параметров качества необходимо изменить нужные ячейки в столбцах «Протестировано, %» и «Уровень качества». Для сохранения изменений нужно нажать на кнопку «Сохранить значения».

Пользователь может при необходимости составить отчет «Сравнительное качество версий», форма которого представлена на рисунке 3.13.

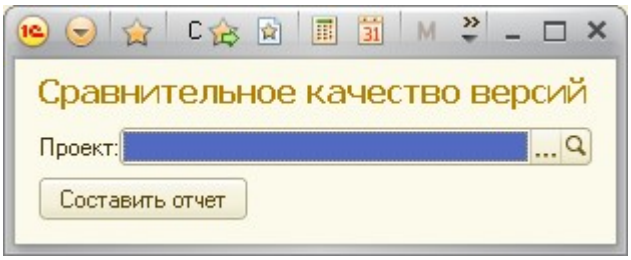

Рисунок 3.13 – Форма отчета «Сравнительное качество версий»

Для формирования отчета необходимо выбрать интересующий проект и нажать кнопку «Составить отчет».

В конфигурации также представлен отчет «Текущее состояние версии», форма которого представлена на рисунке 3.14.

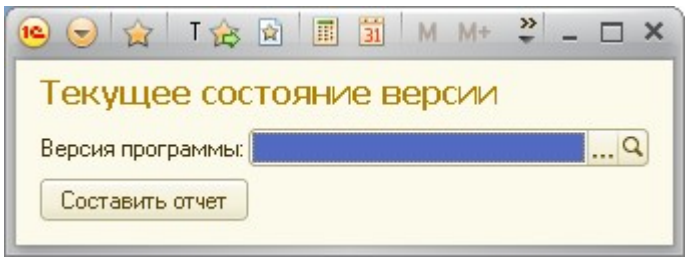

Рисунок 3.14 – Форма отчета «Сравнительное качество версий»

Для формирования отчета необходимо выбрать интересующую версию программы из справочника «Версии программного обеспечения» и нажать кнопку «Составить отчет».

4 Организационно-экономические вопросы

4.1 Основные этапы разработки

Современная инженерная деятельность предполагает не только разработку программного обеспечения, но также и концентрацию усилий специалиста, позволяющую заранее определить экономические перспективы и возможный рынок реализации разработки, оценить ожидаемую прибыль. Поэтому важной составляющей данного дипломного проекта является анализ экономических характеристик и определение экономических параметров, позволяющих сделать вывод о возможности и экономической целесообразности реализации настоящей системы контроля качества производства программного обеспечения.

Планирование содержания работ, формирование последовательности и характеристика этапов выбранных стадий разработки системы тестирования знаний представлены в таблице 4.1.

Таблица 4.1 - Стадии разработки, этапы и содержание работ

Стадии разработки	Этапы работ	Содержание работ
1. Техническое задание	Обоснование необходимости разработки программы	Постановка задачи. Сбор исходных материалов. Предварительный выбор методов решения задач.
	Разработка и утверждение технического задания	Определение требований к программе. Определение стадий, этапов и сроков разработки программы и документации на неё. Выбор языков программирования. Согласование и утверждение
2. Эскизный проект	Разработка эскизного проекта	Предварительная разработка структуры данных. Уточнение методов решения задачи. Разработка общего описания алгоритма решения задачи.
3. Технический проект	Разработка технического проекта	Разработка алгоритмов. Уточнение структуры данных. Определение структуры основных и вспомогательных модулей.

Продолжение таблицы 4.1

Стадии разработки	Этапы работ	Содержание работ
4. Рабочий проект	Разработка программы	Разработка программных модулей. Разработка интерфейса. Тестирование, отладка и исправление недочетов.
	Разработка программной документации	Разработка пояснительной записки и программных документов.
5. Внедрение	Внедрение ПО	Развертывание системы контроля качества производства программного обеспечения на сервере.

4.2 Расчет трудоемкости проекта

Расчет трудоемкости является основополагающим для определения общих затрат на реализацию проекта, так как через него, в конечном итоге, оценивается один из основных затратных показателей – совокупные затраты на оплату труда.

Общие затраты труда на разработку и внедрение проекта Q_p определяются следующим образом:

$$Q_p = \sum t_i \,, \tag{4.1}$$

где t_i – затраты труда на выполнение i-го этапа проекта.

Полный перечень работ с разделением их по этапам выполнения проекта приведен в таблице 4.2. Трудоемкость работ определяется методом прямого хронометрирования (продолжительность выполнения отдельных работ фиксируется в процессе их выполнения, т.е. постфактум).

Таблица 4.2 - Перечень работ с разделением их по этапам выполнения проекта

Этап, t_i	№ работы	Содержание работы	Трудоемкость	
			(чел/дни)	(чел/час)
1	1	Постановка задачи.	0,5	4
	2	Сбор исходных материалов.	2	16
	3	Предварительный выбор методов решения задач.	0,5	4
2	4	Определение требований к программе.	0,5	4

Продолжение таблицы 4.2

Этап, t_i		Содержание работы	Трудоемкость

	№		(чел/дни)	(чел/час)
	5	Определение стадий, этапов и сроков разработки программы и документации на неё.	1	8
	6	Выбор языков программирования.	0,5	4
3	7	Предварительная разработка структуры данных.	1	8
	8	Уточнение методов решения задачи.	2	16
	9	Разработка общего описания алгоритма решения задачи.	3	24
4	10	Разработка алгоритмов.	3	24
	11	Уточнение структуры данных.	2	16
	12	Определение структуры основных и вспомогательных модулей.	4	32
5	13	Разработка программных модулей.	10	80
	14	Разработка интерфейса.	6	48
	15	Тестирование, отладка и исправление недочетов.	18	144
6	16	Разработка пояснительной записки и программных документов.	6	48
7	17	Развертывание системы контроля качества производства программного обеспечения на сервере.	2	16

Рассчитаем по формуле (4.1) общие затраты труда на разработку и внедрение системы тестирования знаний, которые будут составлять 62 чел/дней или 496 чел/часов.

4.4 Определение численности исполнителей

Для оценки возможности выполнения проекта имеющимся в распоряжении разработчика штатным составом исполнителей рассчитывается их средняя численность, которая при реализации проекта разработки и внедрения системы тестирования знаний определяется формулой

$$N = \frac{Q_p}{F}, \tag{4.2}$$

где Q_p – затраты труда на выполнение проекта (разработка и внедрение ПО);

F – фонд рабочего времени.

Результат, полученный по формуле (4.2), округляют до большего целого, который и показывает среднее число необходимых исполнителей проекта.

Величина фонда рабочего времени определяется соотношением

$$F = T \cdot F_M , \tag{4.3}$$

где T – время выполнения проекта в месяцах;

F_M – фонд времени в текущем месяце, который рассчитывается из учета общего числа дней в году, числа выходных и праздничных дней

$$F_M = \frac{t_P \left(D_K - D_B - D_П \right)}{12}, \tag{4.4}$$

где t_P – продолжительность рабочего дня;

D_K – общее число дней в году;

D_B – число выходных дней в году;

$D_П$ – число праздничных дней в году.

Рассчитаем фонд времени для месяца 2012 года по формуле (4.4), приняв следующие исходные данные: t_p = 8 часов, D_K = 366 дней, D_B = 104 дня , $D_П$ = 12 дней. Тогда фонд времени F_M для одного месяца 2012 года будет равен 167 часам.

По формуле (4.3) определим величину фонда рабочего времени, приняв время выполнения проекта за 3 месяца. Тогда величина фонда рабочего времени F будет составлять 501 час. Согласно формуле (4.2), средняя численность исполнителей проекта N будет равна 1 человеку.

4.5 Анализ структуры затрат проекта

4.5.1 Затраты на выплату исполнителям заработной платы

Затраты на выплату исполнителям заработной платы линейно связаны с трудоемкостью и определяются соотношением

82

$$C_{зарп} = C_{з.осн} + C_{з.доп} + C_{з.от} , \tag{4.5}$$

где $C_{з.осн}$ – основная заработная плата;

$C_{з.доп}$ – дополнительная заработная плата;

$C_{з.отч}$ - отчисления с заработной платы.

Расчет основной заработной платы (оплаты труда непосредственных исполнителей) проведем по «дневной» оплате труда на основе данных по окладам

$$C_{з.осн} = T_{зан} \frac{8 \cdot O_{мес}}{F_M}, \tag{4.6}$$

где $T_{зан}$ – число дней, отработанных исполнителем проекта;

$O_{мес}$ – месячный оклад;

F_M – месячный фонд рабочего времени.

Величина месячного оклада инженера-программиста составляет 15000 руб.

Следует учесть, что должностной оклад определяет начисляемую исполнителю сумму и с нее должен быть еще удержан подоходный налог по формуле

$$O_{мес} = O\left(1 + \frac{H_{ДФЛ}}{100}\right), \tag{4.7}$$

где O – оклад;

$H_{ДФЛ}$ – налог на доходы с физических лиц.

Согласно формуле (4.7), затраты на месячный оклад исполнителей проекта будут равны 16950 руб. Приняв число дней $T_{зан}$, отработанных исполнителем проекта, за 62 дня и подоходный налог равным 13%, по формуле (4.6) рассчитаем основную заработную плату исполнителей, которая будет равна 50342,51 руб.

Расходы на дополнительную заработную плату учитывают все выплаты непосредственным исполнителям за время, не проработанное на производстве, но предусмотренное законодательством, в том числе: оплата очередных отпусков, компенсация за недоиспользованный отпуск, и др. Величина этих выплат составляет 20% от размера основной заработной платы

$$C_{з.доп} = 0{,}2 \cdot C_{з.осн},$$

(4.8)

Тогда расходы на дополнительную заработную плату составят 10068,50 руб.

Отчисления с заработной платы состоят в настоящее время в уплате единого социального налога. Налоговым кодексом РФ определяются ставки налога для отчисления в пенсионный фонд РФ, фонд социального страхования, фонды обязательного медицинского страхования (федеральный и территориальный фонды). Отчисления с заработной платы составят

$$C_{з.от} = \left(C_{з.осн} + C_{з.доп}\right) \cdot H_{соц}, \tag{4.9}$$

где $H_{СОЦ}$ – отчисления с заработной платы в виде единого социального налога (ЕСН), которые составляют 26 % от суммы основной и дополнительной заработных плат.

Таким образом, отчисления с заработной платы $C_{з.от}$ составят 15706,86 руб.

Общие затраты на выплату исполнителям заработной платы, согласно формуле 4.5, будут равны 76117,87 руб.

4.5.2 Затраты, связанные с обеспечением работ оборудованием

Как правило, оборудование, приобретенное для выполнения проекта, будет использоваться и после его завершения. Поэтому полностью относить его стоимость на один проект нельзя. Для определения доли стоимости оборудования, отнесенной к конкретному проекту, используется метод начисления амортизации.

Определяется ресурс оборудования (например, в годах работы). Общая стоимость оборудования делится на ресурс. Результат – стоимость единицы ресурса. Затем определяется объем ресурса оборудования, необходимый для реализации проекта. Доля стоимости ресурса, отнесенная на проект, равна стоимости единицы ресурса, умноженной на необходимый объем.

Средства вычислительной техники довольно быстро устаревают. Основная причина – технический прогресс, обуславливающий моральное устаревание техники. Поэтому для вычислительной техники назначаются сокращенные сроки амортизации, составляющие 2 – 3 года.

В таблице 4.3 приведен расчет стоимости оборудования, необходимого для реализации проекта.

Таблица 4.3 - Расчет стоимости оборудования, необходимого для реализации проекта

Вид оборудования	Стоимость, руб.	Период амортизации, лет	Стоимость ед. ресурса, руб.	Требуемый объем, лет	Доля, отнесенная на проект, руб.
Ноутбук ASUS Z99Le / Intel Celeron M540/DDR 1 Gb/120 Gb	18900	3	6300	0,25	1575

Таким образом, стоимость оборудования, отведенного на проект, составила 1575 руб.

4.5.3 Расчет затрат, связанных с организацией рабочих мест

Для выполнения данного проекта были использованы собственные площади, поэтому вместо арендной платы необходимо рассчитать расходы по обслуживанию помещений (плата за электричество, отопление и т.д.).

Плата за отопление помещения, вычисляется по формуле

$$C_{от} = Ц_{от} \cdot S \cdot \frac{T}{12}, \tag{4.10}$$

где $Ц_{от}$ – стоимость отопления 1 м^2 в год;

S – площадь помещения;

T – время выполнения проекта в месяцах.

Необходимая площадь помещения рассчитывает в соответствии с санитарными нормами, учитывая численность исполнителей проекта по формуле

$$S = N \cdot S_n + 5, \tag{4.11}$$

где N – средняя численность исполнителей проекта;

S_n – норма площади на одного человека.

Средняя численность исполнителей проекта была вычислена в пункте 4.4 и равна 1 человеку. Норма площади на одного человека равна 6 м^2. Тогда, согласно формуле (4.11), необходимая площадь помещения S будет равна 11 м^2.

Примем стоимость отопления 1 м^2 в год равной 126 руб. Тогда плата за отопление помещения $C_{от}$, согласно формуле (4.10) будет составлять 346,50 руб.

Стоимость расходуемой электроэнергии вычисляется по формуле

$$C_{эл} = P \cdot T \cdot F_M \cdot Ц_{эл}, \quad (4.12)$$

где $Ц_{эл}$ – стоимость 1 кВт·час;

P – суммарная мощность электроприборов;

F_M – фонд времени за месяц;

T – время выполнения проекта в месяцах.

При выполнении проекта использовался ноутбук ASUS Z99Le, мощность которого 65 Вт, и три люминесцентные лампы, мощностью 60 Вт. Таким образом, суммарная мощность электроприборов P равна 0,245 кВт. Примем стоимость 1 кВт·час равной 1 руб., тогда, согласно формуле (4.12), стоимость расходуемой электроэнергии $C_{эл}$ будет составлять 122,01 руб.

Также при расчете стоимости затрат, связанных с организацией рабочих мест стоит учесть плату за доступ в интернет, которую мы рассчитаем по следующей формуле:

$$C_{инт} = T \cdot Ц_{инт}, \quad (4.13)$$

где $Ц_{инт}$ – абонентская плата за доступ в интернет по безлимитному тарифу за месяц;

T – время выполнения проекта в месяцах.

Примем абонентскую плату за доступ в интернет за один месяц $Ц_{инт}$ равной 450 руб., тогда, согласно формуле (4.13), плата за доступ в интернет $C_{инт}$ будет составлять 1350 руб.

Затраты на канцелярские расходы $C_{канц}$ вычисляются как 5% от основной заработной платы исполнителя проекта и составляют 1192,25 руб.

Таким образом, затраты, связанные с организацией рабочих мест, для разработки системы тестирования знаний можно вычислить по формуле

$$C_{орг} = C_{от} + C_{эл} + C_{инт} + C_{канц}\text{,} \quad (4.14)$$

где $C_{от}$ – плата за отопление помещения;

$C_{эл}$ – стоимость расходуемой электроэнергии;

$C_{инт}$ – плата за доступ в интернет;

$C_{канц}$ – затраты на канцелярские расходы.

В итоге, затраты, связанные с организацией рабочих мест составят 3010,76 руб.

4.5.4 Расчет затрат на нематериальные активы

Разработка программного обеспечения связана с использованием интеллектуальных лицензионных продуктов: операционных систем, программных сред и утилит, графических изображений и шрифтов. Все они являются объектами интеллектуальной собственности, и использование их нелицензионных копий незаконно. Приобретение прав на использование таких объектов не сопровождается передачей покупателю каких-либо материальных объектов.

Расчет затрат на нематериальные активы приведен в таблице 4.4.

Таблица 4.4 - Расчет затрат на нематериальные активы

Объекты интеллектуальной собственности	Тип объекта	Стоимость лицензии, руб.	Период амортизации, лет	Стоимость ед. ресурса, руб.	Требуемый объем, лет	Доля, отнесенная на проект
Опер. система ПО Microsoft "Windows 7 Домашняя Базовая" Русская версия DVD (BOX)	коммерческое	2950	3	0	0,25	245,83
Веб-сервер Apache 2.0	бесплатное, open source	0	3	0	0,25	0
Пакет офисных программ OpenOffice 3.3	бесплатное, open source	0	3	0	0,25	0
1С:Предприятие конфигурация «1С:Управление торговлей 8.2»	коммерческое	14500	3	0	0,25	1208,33

Таким образом, проанализировав информацию об объектах интеллектуальной собственности, используемых при разработке системы тестирования знаний, делаем вывод, что затраты на нематериальные активы составят 1454,16 руб.

4.5.5 Расчет затрат на накладные расходы

Рассчитаем накладные расходы, связанные с выполнением проекта, ориентируясь на расходы по основной заработной плате. Обычно они составляют 100% расходов на основную заработную плату. Тогда накладные расходы будут равны 50342,51 руб.

4.5.6 Общие затраты на выполнение проекта

Затраты на выполнение проекта состоят из прямых затрат (на заработную плату исполнителям, затрат на закупку или аренду оборудования, затрат на организацию рабочих мест), и косвенных затрат (накладные расходы):

$$K = C_{зарп} + C_{об} + C_{орг} + C_{н.акт} + C_{накл}, \quad (4.15)$$

где $C_{зарп}$ – заработная плата исполнителей;

$C_{об}$ – затраты на обеспечение необходимым оборудованием;

$C_{орг}$ – затраты на организацию рабочих мест;

$C_{н.акт}$ – затраты на нематериальные активы;

$C_{накл}$ – накладные расходы.

Таким образом, общие затраты на выполнение проекта, согласно формуле (4.15), составят 132500,30 руб.

4.6 Выводы по организационно-экономической части

Разработанное программное обеспечение относится к индивидуальным программным продуктам, т. е. оно предназначено для решения конкретных проблем конкретных пользователей. Потребность в таких продуктах возникает при невозможности применения универсальных программ или трудоемкости их адаптации. Поэтому индивидуальные программы, как правило, сильно специализированы, максимально учитывают особенности эксплуатации и специфику предприятия, на котором они используются.

Трудоемкость операций по реализации и внедрению системы тестирования знаний в рамках данного дипломного проекта составила 62 чел/дня. Для выполнения этих операций потребуется 1 инженер-программист. Общие затраты на выполнение проекта составят 132500,30 руб.

Стоит отметить, что при разработке системы контроля качества производства программного обеспечения использовалось как коммерческое, так и свободное программное обеспечение, что привело к значительному снижению затрат на нематериальные активы.

Тем самым подтверждается экономическая целесообразность данного дипломного проекта.

5 Вопросы охраны труда

5.1 Анализ опасных факторов

В настоящее время компьютерная техника широко применяется во всех областях деятельности человека. При работе с компьютером человек подвергается воздействию ряда опасных и вредных производственных факторов: электромагнитных полей (диапазон радиочастот: ВЧ, УВЧ и СВЧ), недостаточной освещенности, шума и вибрации, статического электричества и др.

Особенности характера и режима работы, значительное умственное напряжение приводят к изменению функционального состояния центральной нервной системы, нервно-мышечного аппарата рук при работе с клавиатурой. Нерациональные конструкция и размещение элементов рабочего места вызывают необходимость поддержки неудовлетворительной рабочей позы. Длительный дискомфорт приводит к увеличению напряжения мышц и обуславливает развитие общей усталости и снижение работоспособности.

В процессе работы на ПК происходит постоянное и значительное напряжение функций зрительного аппарата, обусловленное необходимостью различения объектов (символов, знаков и т.п.), строчной структурой экрана, мерцанием изображений, недостаточной освещенностью поля экрана, контрастностью объектов различения и необходимостью постоянной переадаптации глаза к различным уровням освещенности экрана, оригинала и клавиатуры. Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

5.2 Мероприятия по защите от выявленных опасных факторов

5.2.1 Электромагнитное излучение

Допустимые значения параметров неионизирующих электромагнитных излучений от монитора компьютера представлены в таблице 5.1.

Таблица 5.1 - Допустимые значения параметров неионизирующих электромагнитных излучений (в соответствии с СанПиН 2.2.2.542-96)

Наименование параметра	Допустимые значения
Напряженность электрической составляющей электромагнитного поля на расстоянии 50 см от поверхности видеомонитора	10 В/м
Напряженность магнитной составляющей электромагнитного поля на расстоянии 50 см от поверхности видеомонитора	0,3 А/м
Напряженность электростатического поля не должна превышать: - для взрослых пользователей - для детей дошкольных учреждений и учащихся средних специальных и высших учебных заведений	20 кВ/м 15 кВ/м

Основным источником электромагнитных излучений на рабочем месте администратора или пользователя системы управления базой данных знаний является монитор компьютера, а точнее его высокочастотный трансформатор строчной развертки. Он расположен в задней части дисплейного элемента и поэтому распространение излучения направлено назад от монитора. Электропроводка, люминесцентные лампы и другие электроприборы являются источниками электромагнитных излучений. Напряженность электромагнитного поля на расстоянии 50 см по электрической составляющей в диапазоне частот 5Гц-2кГц поддерживается на уровне не более 10 В/м, в диапазоне частот 2-400 кГц – 2,5 В/м. Плотность магнитного потока в диапазоне частот 5Гц-2кГц составляет 100 нТл, в диапазоне частот 2-400 кГц – 25 нТл. Поверхностный электростатический потенциал имеет уровень не более 500 В.

Максимальный уровень рентгеновского излучения на рабочем месте оператора компьютера не превышает 10 мкбэр/ч, а интенсивность ультрафиолетового и инфракрасного излучений от экрана монитора лежит в пределах 10…100 мВт/м^2.

Для достижения безопасных условий труда используются мониторы с небольшим уровнем излучения (MPR-II, TCO-92, TCO-99), жидкокристаллические или плазменные дисплеи в соответствии с сертификацией и госстандартом. Санитарно-гигиенический надзор и контроль за электромагнитными (ЭМИ) и другими видами излучения осуществляется как на стадии выпуска ПК, так и в процессе их эксплуатации. При установке на рабочем месте ПК подключается к электропитанию с соблюдением всех норм и надежно заземляется. Для обеспечения предельно-допустимых уровней электромагнитных излучений осуществляется рациональное размещение рабочих мест, оснащенных ПЭВМ (ПК). Их месторасположение от стены помещения составляет не менее 1 метра, что контролируется аттестацией рабочего места.

5.2.2 Освещение

В качестве источников света применяются люминесцентные лампы типа ЛБ и ЛТБ, мощностью 60 Вт. Для освещения рабочих мест администраторов и пользователей системы управления базой данных применяются светильники серии ЛПО36 с зеркалированными решетками, укомплектованные высокочастотными пускорегулирующими аппаратами (ВЧ ПРА).

На рабочих местах устроено боковое естественное освещение (КЕО = 1,2 %). Освещенность на поверхности стола в зоне размещения рабочего документа равна 300–500 лк.

Отраженная блесткость на рабочих поверхностях (экран, стол, клавиатура) ограничивается за счет правильного расположения рабочих мест по отношению к источникам естественного и искусственного освещения.

Местное освещение на рабочем месте администратора системы обеспечивается светильником, установленном непосредственно на рабочем месте. Светильник располагается ниже или на уровне линии зрения администратора системы управления базой данных так, чтобы избежать ослепления оператора и не создавать бликов отражения на экране дисплея.

5.2.3 Микроклимат

Работа, связанная с разрабатываемой в данном дипломном проекте системой управления базой данных, относится к категории I а. К категории I а относятся работы, производимые сидя и не требующие физического напряжения, при которых энергозатраты составляют до 120 ккал/ч. При выполнении таких работ, температура воздуха должна быть в холодный период года не более 22÷24°С, в теплый период года не более 23÷25°С. Относительная влажность на рабочих местах должна быть 40÷60%, а скорость движения воздуха - не более 0,1м/с.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении.

Объем помещений, в которых размещены компьютеры, не должен быть меньше 19,5 $м^3$/человека с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в таблице 5.2.

Таблица 5.2 - Нормы подачи свежего воздуха в помещения, где расположены компьютеры

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, $м^3$ /на одного человека в час
Объем до 20 $м^3$ на человека 20…40 $м^3$ на человека Более 40 $м^3$ на человека	Не менее 30 Не менее 20 Естественная вентиляция

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха).

5.2.4 Шум

Источниками шума на рабочих местах с ПК являются шумные агрегаты вычислительных машин (вентилятор, жесткий диск и т.п.), установки кондиционирования воздуха и другое оборудование.

В таблице 5.3 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 5.3 - Предельные уровни звука, дБ, на рабочих местах.

Категория напряженности труда	Категория тяжести труда			
	I. Легкая	II. Средняя	III. Тяжелая	IV. Очень тяжелая
I. Мало напряженный	80	80	75	75
II. Умеренно напряженный	70	70	65	65
III. Напряженный	60	60	-	-
IV. Очень напряженный	50	50	-	-

Труд администраторов и пользователей при работе с разработанной в данном дипломном проекте системе управления базой данных с точки зрения напряженности относится к категории IV – очень напряженный, – а с точки зрения тяжести труда – к категории I – легкая. Таким образом, уровень шума на рабочих местах администраторов и пользователей системы управления базой данных, согласно предельным уровням звука, не превышает 50 дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, облицованы звукопоглощающими материалами.

5.2.5 Пожарная безопасность

Помещения с ЭВМ по взрыво-пожароопасности относятся к категории В – пожароопасные с наличием твердых горючих материалов. Источником возникновения пожара могут быть проводники и детали, находящиеся под напряжением. Высокая концентрация токопроводящих элементов создает опасность их нагрева с нарушением изоляции, часто изготовленной из горючих материалов, и воспламенения в результате короткого замыкания. Под действием электрических искр изоляция проводов может загореться. При ремонтных работах применяют электроаппаратуру, паяльники и сварочные агрегаты. Появляются дополнительные источники зажигания, что создает повышенную опасность возникновения пожара. Поэтому при таких работах строго соблюдаются правила пожарной безопасности.

Учитывая высокую стоимость современных видеотерминальных устройств и компьютеров, горючесть материалов, помещения, в которых должны располагаться ПЭВМ проектируют I или II степени огнестойкости.

Для предотвращения возникновения пожара используют негорючие изоляционные материалы и различные средства оповещения огня. Курение в помещениях строго запрещено. В рабочих кабинетах имеются углекислотные огнетушители ОУ-5. Углекислый газ обладает диэлектрическими свойствами и обеспечивает сохранность электрооборудования при пожаротушении. Оператор оборудования обладает навыками применения средств борьбы с огнем. В случае возникновения пожара обесточивают помещение, оповещают пожарную охрану и руководство, принимают меры по тушению огня. Не дожидаясь прибытия пожарного подразделения, приступают к ликвидации пожара имеющимися в наличии средствами тушения пожара. Если очаг пожара находится под напряжением, применяются углекислотные огнетушители. В любом случае оборудование отключается от электропитания.

5.2.6 Электробезопасность

Персональные ЭВМ относятся к электроустановкам напряжением до 1000 В. В качестве основной меры защиты от поражения электротоком в помещениях, где работают ПЭВМ, применяется защитное заземление либо зануление, в зависимости от вида электрической сети.

Для компьютерной сети наиболее приемлемый вариант – защитное заземление. Имеются специальные клеммы для подключения заземления ($R_З = 40$ Ом).

Все токоведущие ПК закрыты корпусом. Снимать корпус при включенном оборудовании строго запрещается. Корпус ПК снабжен блокировкой, которая не допускает снятие корпуса при включенном оборудовании.

Администраторы системы управления базой данных имеют первую квалификационную группу по электробезопасности. Они также обладают навыками оказания первой доврачебной помощи пострадавшим от электрического тока.

5.3 Эргономические требования к рабочему месту

При организации рабочего места администраторов и пользователей системы управления базой данных следует обеспечить соответствие конструкции всех элементов рабочего места и их взаимного положения эргономическим требованиям.

Конструкция рабочего стола обеспечивает оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей характера работы.

Высота рабочей поверхности стола для взрослых пользователей регулируется в пределах 680-800 мм, а при отсутствии такой возможности высота рабочей поверхности стола составляет 725 мм. Рабочий стол имеет пространство для ног высотой не менее 600 мм, шириной не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650мм.

Рабочие поверхности столов администраторов и пользователей системы управления базой данных не имеют острых углов и краев. Конструкция рабочих стульев обеспечивает поддержание рациональной рабочей позы при работе на ПК и позволяет изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

Конструкция обеспечивает:

- ширину и глубину поверхности сиденья не менее 400 мм;
- поверхность сиденья с закругленным передним краем;
- регулировку высоты поверхности сиденья в пределах 400-550 мм и углам наклона вперед до 15° и назад 5°;
- высоту опорной поверхности спинки 300±20 мм, ширину - не менее 380 мм;
- радиус кривизны горизонтальной плоскости - 400 мм;
- угол наклона спинки в вертикальной плоскости в пределах 0±30°;
- регулировку расстояния спинки от переднего края сиденья в пределах 260-400 мм;
- стационарные или съемные подлокотники длиной не менее 250 мм и шириной 50-70 мм;
- регулировку подлокотников по высоте над сиденьем в пределах 230±30 мм

внутреннего расстояния между подлокотниками в пределах 350-500 мм.

Экран монитора находится на оптимальном от глаз пользователя расстоянии в 600-700 мм.

По высоте экран устанавливается на столе или подставке так, так, чтобы угол между нормалью к центру экрана и горизонтальной линией взора составлял 20 градусов.

Угол наблюдения экрана, а также других средств отображения в горизонтальной плоскости (угол разворота монитора относительно оператора) в общем случае не превышает 60 градусов.

Клавиатура располагается на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Рабочие места с ПК по отношению к световым проемам располагаются так, что естественный свет падает сбоку, преимущественно слева.

5.4 Режим труда и отдыха

В таблице 5.4 представлены сведения о регламентированных перерывах, которые необходимо делать при работе на компьютере, в зависимости от продолжительности рабочей смены, видов и категорий трудовой деятельности с ВДТ (видеодисплейный терминал) и ПЭВМ[11].

Таблица 5.4 - Время регламентированных перерывов при работе на компьютере

Категория работы с ВДТ или ПЭВМ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных перерывов, мин	
	Группа А, количество знаков	Группа Б, количество знаков	Группа В, часов	При 8-часовой смене	При 12-часовой смене
I	до 20 000	до 15 000	до 2,0	30	70
II	до 40 000	до 30 000	до 4,0	50	90
III	до 60 000	до 40 000	до 6,0	70	120

Примечание. Время перерывов дано при соблюдении указанных Санитарных правил и норм. При несоответствии фактических условий труда требованиям

Санитарных правил и норм время регламентированных перерывов следует увеличить на 30%.

В соответствии с [11] все виды трудовой деятельности, связанные с использованием компьютера, разделяются на три группы:

- *Группа А:* работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;
- *Группа Б:* работа по вводу информации;
- *Группа В:* творческая работа в режиме диалога с ЭВМ.

Для пользователей работа с разработанной в данном дипломном проекте системой управления базой данных относится к группе В - творческая работа в режиме диалога с ЭВМ III категории тяжести. Для администраторов системы управления базой данных работа относится к группе Б - работа по вводу информации II категории тяжести.

Заключение

В настоящем дипломном проекте разработана конфигурация информационной системы контроля качества производства ПО для платформы «1С:Предприятие 8.2» на базе стандартной конфигурации «1С:Управление торговлей 8.2», определена ее архитектура, реализованы логическая и физическая модели структуры базы данных, разработаны классы, модули и функции подсистемы. Также описан процесс ее установки, эксплуатации и технические требования.

Система контроля качества построена на платформе «1С:Предприятие 8.2», которая способна работать в режиме толстого клиента, тонкого клиента и веб-клиента. Для работы с приложением на клиентской стороне в режиме веб-клиента используется веб-браузер. Для работы в режиме тонкого и толстого клиента используется платформа «1С:Предприятие 8.2».

Выполнено технико-экономическое обоснование проекта. Разработка и использование информационной системы контроля качества производства программного обеспечения проанализировано с точки зрения охраны труда и безопасности жизнедеятельности пользователей.

Список использованных источников

1Селищев, Н. 1С: Предприятие 8.2. Управление торговлей / Н. Селищев. - СПб.: Питер, 2011. – 400 с.

2Реализация прикладных задач в системе 1С Предприятие 8.2. / А. П. Габец [и др.]. - М.: 1С-Паблишинг, 2010. - 714 с.

3ЗАО «1С» Официальный сайт фирмы-разработчика [Электронный ресурс] – Режим доступа: http://1c.ru/, свободный. – Загл. с экрана.

4Wikipedia [Электронный ресурс] - Режим доступа: http://ru.wikipedia.org, свободный. – Загл. с экрана.

5 Разработка управляемого интерфейса / В. А. Ажеронок [и др.]. - М.: 1С-Паблишинг, 2010. - 514 с. - ISBN: 978-5-9677-1148-0.

6Кулямин, В. В. Методы верификации программного обеспечения [Электронный ресурс] - Режим доступа: http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf, свободный.

7Хрусталева, Е. Ю. Разработка сложных отчетов в 1С:Предприятии 8. Система компоновки данных (+CD) / Е. Ю. Хрусталева. - М.: 1С-Паблишинг, 2008. - 514 с. - ISBN 978-5-9677-0963-0.

8Купить 1С 8.2. Как перйти с версии 1С 8.1 на 8.2. Как обменять 1С 8.1 на 8.2 http://www.1sshop.ru/index.php3?id=131 [Электронный ресурс] - Режим доступа свободный.

9 Каталог товаров. Самара. Сеть супермаркетов цифровой техники DNS [Электронный ресурс] – Режим доступа: http://samara.dns-shop.ru/catalog/, свободный. – Загл. с экрана.

10 Каталог товаров. Самара. Сеть супермаркетов цифровой техники DNS [Электронный ресурс] – Режим доступа: http://samara.dns-shop.ru/catalog/, свободный. – Загл. с экрана.

11 СанПин 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организации работ [Электронный ресурс] – Режим доступа:

http://www.med-pravo.ru/PRICMZ/SanRules/2003/2.2.2-2.4.1340-03.htm, свободный. – Загл. с экрана.

12 Инструкция по установке (КБУ) [Электронный ресурс] – Режим доступа: http://wiki.kint.ru/index.php/Инструкция_по_установке_(КБУ), свободный. – Загл. с экрана.

13 Справочники Учебник по 1С mista.ru [Электронный ресурс] – Режим доступа: http://www.mista.ru/tutor_1c/sprav.htm, свободный. – Загл. с экрана.

14 Калбертсон, Р. Быстрое тестирование / Р. Калбертсон, К. Браун, Г. Кобб. – Вильямс, 2002. - 384 с. - ISBN 5-8459-0336-X.

15 ISO 9126 - Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/ISO_9126, свободный. – Загл. с экрана.

16 Home :: Bugzilla :: bugzilla.org [Электронный ресурс] – Режим доступа: http://www.bugzilla.org/, свободный. – Загл. с экрана.

17 Bugzilla - Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/Bugzilla, свободный. – Загл. с экрана.

18 JIRA- Track bugs, tasks, and projects for software development | Atlassian [Электронный ресурс] – Режим доступа: http://www.atlassian.com/software/jira/overview, свободный. – Загл. с экрана.

19 Atlassian JIRA - Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/Atlassian_JIRA, свободный. – Загл. с экрана.

20 Overview - Redmine [Электронный ресурс] – Режим доступа: http://www.redmine.org/, свободный. – Загл. с экрана.

21 Redmine - Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/Redmine, свободный. – Загл. с экрана.

22 Bug & Issue Tracking Software, Online Project Management, Subversion and GIT Repository Hosting [Электронный ресурс] – Режим доступа: http://www.bontq.com/, свободный. – Загл. с экрана.

23 Bontq - Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/Bontq, свободный. – Загл. с экрана.

24 DevProject Manager - Project management application for software authors [Электронный ресурс] – Режим доступа: http://www.gaijin.at/en/dldevproject.php, свободный. – Загл. с экрана.

25 BugTracker.NET Home - Free Bug Tracking [Электронный ресурс] – Режим доступа: http://ifdefined.com/bugtrackernet.html, свободный. – Загл. с экрана.

26 Профессиональная разработка в системе 1С:Предприятие 8 / А. П. Габец [и др.]. - М.: 1С-Паблишинг, 2006. - 808 с. - ISBN: 5-9677-0268-7.

27 Справка 1С:Предприятие 8 [Электронный ресурс] : [интерактив. учеб.]. –М.: 1С, 2010. – Системные требования: ПК от 486 DX 66 МГц ; RAM 16 Мб ; Windows XP. – Загл. с экрана.

28 1С:Предприятие - Каталог статей - GeterX Corp [Электронный ресурс] – Режим доступа: http://geterx.3dn.ru/publ/1s_predprijatie/3, свободный. – Загл. с экрана.

Приложение 1

Листинг основных модулей конфигурации

Листинг модуля формы «ФормаОтчета» отчета «Сравнительное качество версий»

```
&НаКлиенте
Процедура Отчет(Команда)
	Если НЕ Отчет.Проект.Пустая() тогда
		ТабДок = ПолучитьОтчетНаСервере();
		Если НЕ (ТабДок = неопределено) тогда
			ТабДок.Показать();
		КонецЕсли;
	Иначе
		соб = новый СообщениеПользователю;
	соб.Текст = "Выберите проект.";
	соб.Сообщить();
	КонецЕсли;
КонецПроцедуры

&НаСервере
Функция ПолучитьОтчетНаСервере()
	//Инициализируем список уровней серьезности неисправностей, которые будут участвовать в выборке
	СписокСерьезностей = Новый СписокЗначений;

СписокСерьезностей.Добавить(Перечисления.УровниСерьезностиНеисправностей.ПроблемаНайденнаяЗаказчиком);

СписокСерьезностей.Добавить(Перечисления.УровниСерьезностиНеисправностей.Блокирующий);

СписокСерьезностей.Добавить(Перечисления.УровниСерьезностиНеисправностей.Критичный);

СписокСерьезностей.Добавить(Перечисления.УровниСерьезностиНеисправностей.Мажорный);
```

```
СписокСерьезностей.Добавить(Перечисления.УровниСерьезностиНеисправностей.Норм
альный);
      ТабДок = Новый ТабличныйДокумент;
    Макет = Отчеты.СравнительноеКачествоВерсий.ПолучитьМакет("Макет");
      ОбластьШапка = Макет.ПолучитьОбласть("Шапка");
      ОбластьШапка.Параметры.ДатаОтчета = Формат(ТекущаяДата(),"ДЛФ=Д");
```

Продолжение приложения 1

```
      ОбластьШапка.Параметры.Проект = Отчет.Проект;
      ОбластьШапка.Параметры.Описание = Отчет.Проект.Комментарий;
      ОбластьШапка.Параметры.Завершен = Отчет.Проект.Завершен;
      ТабДок.Вывести(ОбластьШапка);
      //Найдем все функциональные группы для данного проекта
      Запрос = Новый Запрос;
      Запрос.Текст = "Выбрать
      |СПР.Ссылка КАК Ссылка
      |
      |Из Справочник.ФункциональныеГруппы как СПР
      |
      |Где (СПР.Владелец = &Проект)
      |
      |";
      Запрос.УстановитьПараметр("Проект", Отчет.Проект);
      ТаблицаЗначенийКачества = Новый ТаблицаЗначений;
      ТаблицаЗначенийКачества = Запрос.Выполнить().Выгрузить();
      Если ТаблицаЗначенийКачества.Количество() = 0 тогда
            Сообщить("По данному проекту не найдено ни одной функциональной
группы для оценки качества");
            Возврат неопределено;
      КонецЕсли;
      //Найдем все релизы по данному проекту
      Запрос = Новый Запрос;
      Запрос.Текст = "Выбрать
      |СПР.Ссылка КАК Ссылка
      |
      |Из Справочник.Релизы как СПР
      |
      |Где СПР.Владелец = &Проект
      |";
```

```
Запрос.УстановитьПараметр("Проект",   Отчет.Проект);
Результат = Запрос.Выполнить();
ОбластьКачество = Макет.ПолучитьОбласть("Качество");
//Если есть такие релизы, то выводим информацию
Если НЕ Результат.Пустой() Тогда
      ОбластьВерсияПрограммы = Макет.ПолучитьОбласть("ВерсияПрограммы");
```

Продолжение приложения 1

```
      Выбор = Результат.Выбрать();
      Пока Выбор.Следующий() Цикл
            ОбластьВерсияПрограммы.Параметры.ВерсияПрограммы =
Выбор.Ссылка;
            ТабДок.Вывести(ОбластьВерсияПрограммы);
            //Заполняем характеристики качества версии программы
            Для Каждого ТекСтрока Из ТаблицаЗначенийКачества Цикл
                  //Делаем запрос к регистру сведений для получения
характеристик
                  //для каждой функциональной группы
                  ОбластьКачество.Параметры.ФункциональнаяГруппа =
ТекСтрока.Ссылка;
                  Отбор = Новый Структура("Релиз, ФункциональнаяГруппа",
Выбор.Ссылка, ТекСтрока.Ссылка);
                  ЗначениеРесурсов =
РегистрыСведений.КачествоРелиза.Получить(Отбор);
                  ОбластьКачество.Параметры.УровеньКачества =
ЗначениеРесурсов.УровеньКачества;
                  ОбластьКачество.Параметры.ПроцентПротестированности=
ЗначениеРесурсов.ПроцентПротестированности;
                  //Ищем количество серьезных неисправностей, найденных в
данной версии программы
                  Запрос = Новый Запрос;
                  Запрос.Текст = "Выбрать
                  |СПР.Ссылка КАК Ссылка
                  |
                  |Из Справочник.Баги как СПР
                  |
                  |Где (СПР.Релиз = &Релиз)
                  |И (СПР.Серьезность В (&Серьезность) И
                  |( СПР.ФункциональнаяГруппа = &ФункциональнаяГруппа ))
```

```
                        |
                        |";
                        Запрос.УстановитьПараметр("Релиз",   Выбор.Ссылка);
                        Запрос.УстановитьПараметр("Серьезность",
СписокСерьезностей);
```

Продолжение приложения 1

```
                        Запрос.УстановитьПараметр("ФункциональнаяГруппа",
ТекСтрока.Ссылка);
                        Результат = Запрос.Выполнить();
                        КоличествоСерьезныхНеисправностей =
Результат.Выбрать().Количество();
                        ОбластьКачество.Параметры.КоличествоСерьезныхБагов=
КоличествоСерьезныхНеисправностей;
                        ТабДок.Вывести(ОбластьКачество);
                КонецЦикла;
            КонецЦикла;
      КонецЕсли;
    Возврат ТабДок;
КонецФункции
```

Листинг модуля формы «ФормаОтчета» отчета «Текущее состояние версии»

```
&НаКлиенте
Процедура ПолучитьОтчет(Команда)
      Если НЕ Отчет.ВерсияПрограммы.Пустая() тогда
            ТабДок = ПолучитьОтчетНаСервере();
            ТабДок.Показать();
      Иначе
            соб = новый СообщениеПользователю;
      соб.Текст = "Введите значение версии программы.";
      соб.Сообщить();
      КонецЕсли;
КонецПроцедуры

&НаСервере
Функция ПолучитьОтчетНаСервере()
```

```
    ТабДок = Новый ТабличныйДокумент;
   Макет = Отчеты.ТекущееСостояниеВерсииПрограммы.ПолучитьМакет("Макет");
    ОбластьШапка = Макет.ПолучитьОбласть("Шапка");
    ОбластьШапка.Параметры.ДатаОтчета = Формат(ТекущаяДата(),"ДЛФ=Д");
    ОбластьШапка.Параметры.Проект = Отчет.ВерсияПрограммы.Владелец;
    ОбластьШапка.Параметры.Релиз = Отчет.ВерсияПрограммы.Наименование;
```

Продолжение приложения 1

```
    ОбластьШапка.Параметры.Описание = Отчет.ВерсияПрограммы.Описание;
    ОбластьШапка.Параметры.ПутьКФайлу = Отчет.ВерсияПрограммы.ПутьКФайлу;
    ОбластьШапка.Параметры.Статус = Отчет.ВерсияПрограммы.Статус;
    ОбластьШапка.Параметры.ДатаВерсии =
Отчет.ВерсияПрограммы.ДатаВремяВерсии;
    ТабДок.Вывести(ОбластьШапка);
    ОбластьБаг = Макет.ПолучитьОбласть("Баг");

    //Поиск неисправностей, которые были найдены в данной версии программы
    Запрос = Новый Запрос;
    Запрос.Текст = "Выбрать
    |СПР.Серьезность КАК Серьезность,
    |СПР.Приоритет КАК Приоритет,
    |СПР.Статус КАК Статус,
    |СПР.Резолюция КАК Резолюция,
    |СПР.Описание КАК Описание,
    |СПР.ФункциональнаяГруппа КАК ФункциональнаяГруппа,
    |СПР.ВерсияПрограммыСИсправлением КАК ВерсияПрограммыСИсправлением
    |Из Справочник.Баги как СПР
    |Где СПР.Релиз = &Релиз
    |";
    Запрос.УстановитьПараметр("Релиз",   Отчет.ВерсияПрограммы.Ссылка);
    Результат = Запрос.Выполнить();

    //Если есть такие неисправности, то выводим информацию
    Если НЕ Результат.Пустой() Тогда
        ОбластьШапкаНайденныеБаги =
Макет.ПолучитьОбласть("ШапкаНайденныеБаги");
        ТабДок.Вывести(ОбластьШапкаНайденныеБаги);
        НайденныеБагиВТекущейВерсии = Результат.Выбрать();
        Пока НайденныеБагиВТекущейВерсии.Следующий() Цикл
```

```
            ОбластьБаг.Параметры.Заполнить(НайденныеБагиВТекущейВерсии);
            ТабДок.Вывести( ОбластьБаг);
        КонецЦикла;
    КонецЕсли;

    //Поиск неисправностей, которые были испарвлены разработчиками
```

Продолжение приложения 1

```
    //в данной версии программы
    Запрос = Новый Запрос;
    Запрос.Текст = "Выбрать
    |СПР.Серьезность КАК Серьезность,
    |СПР.Приоритет КАК Приоритет,
    |СПР.Статус КАК Статус,
    |СПР.Резолюция КАК Резолюция,
    |СПР.Описание КАК Описание,
    |СПР.ФункциональнаяГруппа КАК ФункциональнаяГруппа,
    |СПР.ВерсияПрограммыСИсправлением КАК ВерсияПрограммыСИсправлением
    |Из Справочник.Баги как СПР
    |Где СПР.ВерсияПрограммыСИсправлением = &ЦелевойРелиз
    |";
    Запрос.УстановитьПараметр("ЦелевойРелиз",
Отчет.ВерсияПрограммы.Ссылка);
    Результат = Запрос.Выполнить();

    //Если есть такие неисправности, то выводим информацию
    Если НЕ Результат.Пустой() Тогда
        ОбластьШапкаИсправленныеБаги =
Макет.ПолучитьОбласть("ШапкаИсправленныеБаги");
        ТабДок.Вывести(ОбластьШапкаИсправленныеБаги);
        ИсправленныеБагиВТекущейВерсии = Результат.Выбрать();
        Пока ИсправленныеБагиВТекущейВерсии.Следующий() Цикл

    ОбластьБаг.Параметры.Заполнить(ИсправленныеБагиВТекущейВерсии);
            ТабДок.Вывести( ОбластьБаг);
        КонецЦикла;
    КонецЕсли;

    //Поиск неисправностей, которые были испарвлены разработчиками
```

```
      //в данной версии программы и были переоткрыты тестировщиками
      //для повторного исправления
      Запрос = Новый Запрос;
      Запрос.Текст = "Выбрать
      |СПР.Серьезность КАК Серьезность,
      |СПР.Приоритет КАК Приоритет,
```

Продолжение приложения 1

```
      |СПР.Статус КАК Статус,
      |СПР.Резолюция КАК Резолюция,
      |СПР.Описание КАК Описание,
      |СПР.ФункциональнаяГруппа КАК ФункциональнаяГруппа,
      |СПР.ВерсияПрограммыСИсправлением КАК ВерсияПрограммыСИсправлением
      |Из Справочник.Баги как СПР
      |Где (СПР.ВерсияПрограммыСИсправлением = &ЦелевойРелиз)
      |И (СПР.Статус = &Статус)
      |";

      Запрос.УстановитьПараметр("ЦелевойРелиз",
Отчет.ВерсияПрограммы.Ссылка);
      Запрос.УстановитьПараметр("Статус",
Перечисления.СтатусыНеисправностей.Переоткрытый);
      Результат = Запрос.Выполнить();

      //Если есть такие неисправности, то выводим информацию
      Если НЕ Результат.Пустой() Тогда
            ОбластьШапкаПереоткрытыеБаги =
Макет.ПолучитьОбласть("ШапкаПереоткрытыеБаги");
            ТабДок.Вывести(ОбластьШапкаПереоткрытыеБаги);
            ПереоткрытыеБагиВТекущейВерсии = Результат.Выбрать();
            Пока ПереоткрытыеБагиВТекущейВерсии.Следующий() Цикл

      ОбластьБаг.Параметры.Заполнить(ПереоткрытыеБагиВТекущейВерсии);
                  ТабДок.Вывести( ОбластьБаг);
            КонецЦикла;
      КонецЕсли;

      //Поиск неисправностей, которые были испарвлены разработчиками
      //в данной версии программы и были проверены тестировщиками
```

```
Запрос = Новый Запрос;
Запрос.Текст = "Выбрать
|СПР.Серьезность КАК Серьезность,
|СПР.Приоритет КАК Приоритет,
|СПР.Статус КАК Статус,
|СПР.Резолюция КАК Резолюция,
```

Продолжение приложения 1

```
|СПР.Описание КАК Описание,
|СПР.ФункциональнаяГруппа КАК ФункциональнаяГруппа,
|СПР.ВерсияПрограммыСИсправлением КАК ВерсияПрограммыСИсправлением
|Из Справочник.Баги как СПР
|Где (СПР.ВерсияПрограммыСИсправлением = &ЦелевойРелиз)
|И (СПР.Статус В (&Статус))
|";
СписокСтатусов = Новый СписокЗначений;
СписокСтатусов.Добавить(Перечисления.СтатусыНеисправностей.Закрытый);

СписокСтатусов.Добавить(Перечисления.СтатусыНеисправностей.Проверенный);
Запрос.УстановитьПараметр("ЦелевойРелиз",
Отчет.ВерсияПрограммы.Ссылка);
Запрос.УстановитьПараметр("Статус",  СписокСтатусов);
Результат = Запрос.Выполнить();

//Если есть такие неисправности, то выводим информацию
Если НЕ Результат.Пустой() Тогда
    ОбластьШапкаПроверенныеБаги =
Макет.ПолучитьОбласть("ШапкаПроверенныеБаги");
    ТабДок.Вывести(ОбластьШапкаПроверенныеБаги);
    Выбор = Результат.Выбрать();
    Пока Выбор.Следующий() Цикл
        ОбластьБаг.Параметры.Заполнить(Выбор);
        ТабДок.Вывести( ОбластьБаг);
    КонецЦикла;
КонецЕсли;

//Ищем все функциональные группы, доступные для данного проекта
Запрос = Новый Запрос;
Запрос.Текст = "Выбрать
```

```
      |СПР.Ссылка КАК Ссылка
      |Из Справочник.ФункциональныеГруппы как СПР
      |Где (СПР.Владелец = &Проект)
      |";
      Запрос.УстановитьПараметр("Проект", Отчет.ВерсияПрограммы.Владелец);
      Результат = Запрос.Выполнить();
```

Продолжение приложения 1

```
      ОбластьКачество = Макет.ПолучитьОбласть("Качество");

      //Если есть такие группы есть, то выводим информацию
      Если НЕ Результат.Пустой() Тогда
            ОбластьШапкаКачество = Макет.ПолучитьОбласть("ШапкаКачество");
            ТабДок.Вывести(ОбластьШапкаКачество);
            ФункциональныеГруппы = Результат.Выбрать();
            Пока ФункциональныеГруппы.Следующий() Цикл
                  ОбластьКачество.Параметры.ФункциональнаяГруппа =
ФункциональныеГруппы.Ссылка;

                  //запрос к регистру для получения текущих данных
                  Отбор = Новый Структура("Релиз, ФункциональнаяГруппа",
Отчет.ВерсияПрограммы, ФункциональныеГруппы.Ссылка);
                  ЗначениеРесурсов =
РегистрыСведений.КачествоРелиза.Получить(Отбор);
                  ОбластьКачество.Параметры.УровеньКачества =
ЗначениеРесурсов.УровеньКачества;
                  ОбластьКачество.Параметры.ПроцентПротестированности=
ЗначениеРесурсов.ПроцентПротестированности;
                  ТабДок.Вывести(ОбластьКачество);
            КонецЦикла;
      КонецЕсли;
    Возврат ТабДок;
КонецФункции
```

Листинг модуля формы «Форма» обработки «Отметки качества»

```
&НаСервере
Процедура ОбновитьЗначения()
      //Очищаем таблицу перед отображением новых данных
```

```
	Объект.Качество.Очистить();
	//Ищем все функциональные группы, доступные для данного проекта
	Запрос = Новый Запрос;
	Запрос.Текст = "Выбрать
	|СПР.Ссылка КАК Ссылка
```

Продолжение приложения 1

```
	|Из Справочник.ФункциональныеГруппы как СПР
	|Где (СПР.Владелец = &Проект)
	|";
	Запрос.УстановитьПараметр("Проект", Объект.Релиз.Владелец);
	Результат = Запрос.Выполнить();
	//Если есть такие группы есть, то выводим информацию
	Если НЕ Результат.Пустой() Тогда
		Выбор = Результат.Выбрать();
		Пока Выбор.Следующий() Цикл
			Строка = Объект.Качество.Добавить();
			Строка.ФункциональнаяГруппа = Выбор.Ссылка;
			//запрос к регистру для получения текущих данных
			Отбор = Новый Структура("Релиз, ФункциональнаяГруппа",
Объект.Релиз, Выбор.Ссылка);
			//Отбор.Вставить("ФункциональнаяГруппа", Выбор.Ссылка);
			ЗначениеРесурсов =
РегистрыСведений.КачествоРелиза.Получить(Отбор);
			Строка.УровеньКачества = ЗначениеРесурсов.УровеньКачества;
			Строка.ПроцентПротестированности=
ЗначениеРесурсов.ПроцентПротестированности;
		КонецЦикла;
	КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура Команда1(Команда)
	 СохранитьЗначенияНаСервере();
КонецПроцедуры

&НаСервере
Процедура СохранитьЗначенияНаСервере()
	Для каждого ТекСтрока из Объект.Качество Цикл
```

```
            НаборЗаписей =
РегистрыСведений.КачествоРелиза.СоздатьНаборЗаписей();
            НаборЗаписей.Отбор.Релиз.Установить(Объект.Релиз);

НаборЗаписей.Отбор.ФункциональнаяГруппа.Установить(ТекСтрока.ФункциональнаяГр
уппа);
```

Продолжение приложения 1

```
            НоваяЗапись = НаборЗаписей.Добавить();
            НоваяЗапись.Релиз = Объект.Релиз;
            НоваяЗапись.ФункциональнаяГруппа = ТекСтрока.ФункциональнаяГруппа;
            НоваяЗапись.УровеньКачества = ТекСтрока.УровеньКачества;
            НоваяЗапись.ПроцентПротестированности =
ТекСтрока.ПроцентПротестированности;
            НаборЗаписей.Записать();
      КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура РелизПриИзменении(Элемент)
      ОбновитьЗначения();
КонецПроцедуры
```