**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ГРОДНЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ ЯНКИ КУПАЛЫ**

*На правах рукописи*

УДК 004:05

Залога

Анастасия Юрьевна

**Проектирование информационно-аналитической системы управления ИТ-проектами на этапе тестирования**

Магистерская диссертация на соискание степени

магистра прикладной математики и информационных технологий

по специальности

1-31 81 12 Прикладной компьютерный анализ данных

Научный руководитель

кандидат физико-математических

наук, доцент Цехан О. Б.

Допущена к защите \_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Цехан О. Б.

Гродно, 2019

**ОГЛАВЛЕНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 4](#_Toc12283040)

[ВВЕДЕНИЕ 5](#_Toc12283041)

[ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ 7](#_Toc12283042)

[ГЛАВА 1 АНАЛИЗ СОСТАВА, СТРУКТУРЫ И ОСОБЕННОСТЕЙ ФАЗЫ ТЕСТИРОВАНИЯ ПО В БИЗНЕС-ПРОЦЕССЕ РАЗРАБОТКИ ИТ-ПРОДУКТОВ 9](#_Toc12283043)

[1.1 Понятие планирования этапа тестирования 9](#_Toc12283044)

[1.2 Жизненный цикл тестирования ПО 11](#_Toc12283045)

[1.3 Основные проблемы управления проектами на стадии тестирования 15](#_Toc12283046)

[1.4 Риски в тестировании ПО 16](#_Toc12283047)

[1.5 Данные и измерения как основа выявления метрик для планирования, разработки и тестирования ИT-проектов 19](#_Toc12283048)

[1.6 Принятие решений при планировании, разработке и тестировании ИT-проектов 22](#_Toc12283049)

[1.7 Сбор данных для планирования, разработки и тестирования ИT-проектов 24](#_Toc12283050)

[Выводы по главе 1 27](#_Toc12283051)

[ГЛАВА 2 МЕТРИКИ ПО ОБЕСПЕЧЕНИЮ КАЧЕСТВА ЭТАПА ТЕСТИРОВАНИЯ И ПРОГРАММНОГО ПРОДУКТА 29](#_Toc12283052)

[2.1 Понятие качества программного продукта 29](#_Toc12283053)

[2.2 Понятие и классификация метрик 32](#_Toc12283054)

[2.2.1 Метрики по этапам тестирования 33](#_Toc12283055)

[2.2.2 Метрики по типам сущностей 37](#_Toc12283056)

[2.3 Назначение и структура метрик 42](#_Toc12283057)

[2.4 Пример вычисления значений расчетных метрик для проекта “Реализации БД “Опыт реализованных проектов” 48](#_Toc12283058)

[2.4.1 Описание проекта 48](#_Toc12283059)

[2.4.2 Выявление, расчет и применение метрик для проекта разрабатываемой БД 51](#_Toc12283060)

[Выводы по главе 2 54](#_Toc12283061)

[ГЛАВА 3 РАЗРАБОТКА ПРОЕКТА ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ УПРАВЛЕНИЯ ИТ-ПРОЕКТАМИ НА ЭТАПЕ ТЕСТИРОВАНИЯ 56](#_Toc12283062)

[3.1 Метрики в управлении этапом тестирования 56](#_Toc12283063)

[3.2 Проект подсистемы информационно-аналитической системы управления ИТ-проектами на этапе тестирования для сбора, анализа выработки и мониторинга принятых решений и показателей качества 57](#_Toc12283064)

[3.2.1 Архитектура подсистемы 57](#_Toc12283065)

[3.2.2 Проект базы данных “Опыт тестирования реализованных проектов” 60](#_Toc12283066)

[3.2.3 Проект веб-приложения 69](#_Toc12283067)

[3.2.4 Проект аналитического модуля 76](#_Toc12283068)

[3.2.5 Бизнес требования к разработке 78](#_Toc12283069)

[3.3 Место информационно-аналитической системы управления ИТ-проектами на этапе тестирования в управлении рисками в тестировании ПО. 81](#_Toc12283070)

[Выводы по главе 3 81](#_Toc12283071)

[ЗАКЛЮЧЕНИЕ 83](#_Toc12283072)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 86](#_Toc12283073)

[Список использованных источников 86](#_Toc12283074)

[Список публикаций соискателя 88](#_Toc12283075)

[ПРИЛОЖЕНИЯ 89](#_Toc12283076)

# **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

БД – База данных

ИС – Информационная система

ПО – Программное обеспечение

СУБД – Система управления базами данных

БД– Совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

ИТ – Информационные технологии

ТС – Тест кейс

# **ВВЕДЕНИЕ**

Актуальность темы достаточно очевидна, так как управление проектами является неотъемлемой частью повседневной деятельности руководителей ИТ-компаний. Выработка эффективных решений по управлению проектами требует обоснованных измерителей качества процесса и продукта, обработки большого количества разнородной информации с целью извлечения из нее знаний. Применение в процессе управления проектами обоснованных методов измерения, интеллектуального анализа данных и прикладного программного обеспечения для работы с данными позволяет рационально планировать инвестиционную деятельность, оптимизировать использование имеющихся ресурсов и избегать конфликтных ситуаций, контролировать исполнение составленного плана, анализировать фактические показатели и вносить своевременную коррекцию в ход работ, накапливать, анализировать и использовать в дальнейшем опыт реализованных проектов.

В настоящее время в существующих информационных системах управления проектами недостаточно представлена подсистема управления ИТ-проектами на этапе тестирования: нет однозначного понимания состава и принципов расчета системы показателей для оценки качества этапа тестирования, не в полной мере ведется сбор и хранение исторической информации о факторах и показателях процесса тестирования и его результатах, не в полной мере используются научно обоснованные методы анализа данных для выработки обоснованных стратегий тестирования и оперативного управления на этапе тестирования. О

Разработка и программная реализация информационно-аналитической системы управления ИТ-проектами на этапе тестирования на основе метрик позволит накапливать информацию об опыте реализации проектов в конкретной ИТ-компании, на основе которой с учетом особенностей портфеля проектов, состава команды и методов управления, реализуемых в конкретной фирме, выполнять рациональное планирование и управление этапом тестирования.

Ядром такой системы должна быть хорошо отлаженная система мониторинга, которая собирает и визуализирует данные, рассчитывает метрики и проводит анализ системы метрик, информирует участников процесса о его состоянии и предупреждает команду, реализующую проект, когда что-то работает неправильно.

# **ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ**

**Цель исследования:**разработка проекта информационно-аналитической системы управления IT-проектами на этапе тестирования для повышения эффективности управления.

**Задачи исследования:**

1. Анализ состава, структуры и особенностей фазы тестирования ПО в бизнес-процессе разработки IT-продуктов;
2. Выявление состава и метрик для факторов, влияющие на качество тестирования и показателей качества этапа тестирования;
3. Выявление информационной базы для анализа факторов и показателей качества реализации ИТ-проектов на этапе тестирования;
4. Разработка проекта базы данных "Опыт реализованных проектов";
5. Разработка и реализация средствами прикладного ПО моделей для анализа и оптимизации процесса тестирования ИТ-проектов;
6. Разработка проекта подсистемы информационно-аналитической системы управления ИТ-проектами на этапе тестирования для сбора, анализа, выработки и мониторинга принятых решений и показателей качества.

**Объектисследования:** этап тестирования в системе управления ИТ-проектами.

**Предмет исследования:** управление на этапе тестирования ИТ-проектов.

**Положения, выносимые на защиту.**

1. Классификация метрик для этапа тестирования.
2. Примеры расчета и применения метрик для управления на этапе тестирования.
3. Проект информационно-аналитической системы управления ИТ-проектами на этапе тестирования.

**Научная новизна диссертационного исследования** заключается в развитии классификации метрик программного обеспечения и определении их места в системе управления ИТ-проектом на этапе тестирования. **Практическая значимость работы** состоит в повышении качества программного обеспечения и эффективности управления разработкой ПО за счет внедрения автоматизированной информационно-аналитической системы **управления ИТ-проектами на этапе тестирования.**

**Апробация результатов работы.**

1. Международная заочная научно-практическая конференция «Тренды, риски и угрозы цифровой трансформации экономики» г. Минск., 30 апреля 2019г, г. Минск.
2. Общеуниверситетский сборник научных статей студентов и магистрантов «Наука» 2019.
3. VIII Международная научно-практическая конференция молодых ученых и студентов «Молодежь в мире современных технологий», ММСТ-2019. 6-7 июня 2019г., Херсон, Украина.
4. Акт внедрения информационно-аналитической системы управления ИТ-проектами на этапе тестирования

**Личный вклад соискателя.**

Все результаты, изложенные в работе, получены соискателем. Роль научного руководителя состояла в постановке рассматриваемых в диссертации задач и анализе полученных результатов.

**Основные результаты диссертации** опубликованы в 3 работах, 1 из них находится в печати.

**Структура и объем диссертации.**

Диссертация состоит из перечня условных обозначений, введения, общей характеристики работы, трех глав, заключения, библиографического списка и приложения. Полный объем диссертации составляет 90 страниц, где 13 рисунков, 9 таблиц, библиографический список, состоящий из 21 наименований, включая собственные публикации автора.

**Связь работы с крупными научными программами и темами.**

Диссертационное исследование проводилось в рамках темы второй половины дня «Математическое моделирование устойчивого развития социально-экономических объектов, выявление факторов повышения конкурентоспособности объектов изучения» кафедры математического и информационного обеспечения экономических систем.

# **ГЛАВА 1 АНАЛИЗ СОСТАВА, СТРУКТУРЫ И ОСОБЕННОСТЕЙ ФАЗЫ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В БИЗНЕС-ПРОЦЕССЕ РАЗРАБОТКИ ИТ-ПРОДУКТОВ**

## **1.1 Понятие планирования этапа тестирования**

Цифровая трансформация экономики предполагает интенсивное развитие ИТ-сферы. К настоящему времени технология разработки программного обеспечения (ПО) имеет обширный арсенал различных методов, средств и процедур. Вместе с тем можно выделить некоторые довольно общие стадии разработки ПО – этапов, которые проходят команды разработчиков ПО, прежде чем программа станет доступной для широко круга пользователей. Разработка ПО начинается с первоначального этапа разработки (стадия «преальфа») и продолжается стадиями, на которых продукт дорабатывается и модернизируется. Финальным этапом этого процесса становится выпуск на рынок окончательной версии программного обеспечения («общедоступного релиза»).

Таким образом, программный продукт проходит следующие стадии: анализ требований к проекту; проектирование; реализация; тестирование продукта; внедрение и поддержка.

Настоящая работа посвящена этапу тестирования ПО. **Тестирование программного обеспечения** – это проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

**Основная цель тестирования** – обеспечение должного качества продукта. **Качество программного обеспечения** (Software Quality) – это совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности [1].

Можно выделить следующие **направления, обеспечивающие достижения целей тестирования**: повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах; повысить вероятность того, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям; предоставление актуальной информации о состоянии продукта на данный момент.

**Стратегия тестирования** –это описание общего подхода к тестированию и целей тестирования, которое строится с учетом множества условий, в которых развивается программный продукт или проект. В зависимости от целей проекта и его масштаба определяются *технологии и инструменты* тестирования. Определяются *условия успешного завершения и выполнения тестов*. Устанавливаются *критерии качества* программного продукта и его основных компонентов. Предопределяются *график тестирования*. Данную совокупность мероприятий описывает стратегия тестирования или тест план.

Таким образом, под *стратегией тестирования* будем понимать совокупность следующих основных элементов: компоненты и функционал ПО, которые будут протестированы; тестовую среду; методы и типы тестирования; виды тестов, которые будут использованы (функциональные тесты, тесты под нагрузкой, тесты объема, производительности, удобства работы, распределения и т.д.); применяемые технологии тестирования (автоматическое тестирование и тестирование вручную); тестовый инструментарий.

Выбранная для конкретного проекта стратегия фиксируется в документе - тест-плане.

**Тест план** – это часть общей проектной документации, описывающий субъект, время и методы тестирования, а также занятость команды тестирования. Цель Тест плана – определить компоненты, которые будут протестированы, тестовую среду, методы и типы тестирования, виды тестов, которые будут использованы, тестовый инструментарий,ответственных лиц. Обычно выбранную стратегию документируют и утверждают на самых начальных этапах жизненного цикла программного продукта и придерживаются на всем его протяжении.

Не существует единого шаблона распределения тестов по циклам тестирования. Типы тестов зависят от количества итераций, размера итерации и типа проекта.

При тестировании центральное внимание уделяется охвату всех требований, подлежащих тестированию, то есть выполнению всех предусмотренных тестов. Это значит, что критерии завершения тестирования обычно привязаны к количественному охвату множества тестов, каждый из которых, в свою очередь, напрямую связан с каким-либо требованием. При тестировании компонентов и интеграции удобнее пользоваться критериями завершения на основе охвата кода. На разных этапах итерационного процесса разработки ПО происходит изменение критериев охвата. План тестирования должен содержать наборы критериев завершения тестирования для тестирования компонентов, интеграции и системы в целом. Для разных итераций могут быть определены разные наборы критериев завершения.

В определенных культурах тестирования план тестирования считается неофициальным нерегулярным рабочим продуктом, а в других - является в значительной степени формализован и часто требует выхода из системы. По существу, формат и содержимое этого рабочего продукта должны изменяться в соответствии с конкретными потребностями организации или проекта.

## **1.2 Жизненный цикл тестирования ПО**

Отдельно выделяют стандартные этапы тестирования, которым придерживаются большинство разрабатываемых продуктов.

***Анализ требований*** [2]***.***

Процесс статического тестирования начинается с определения и анализа требований к системе. Изучаются существующие материалы и методические вопросы (техническое задание на разработку, рабочий и технический проект, пр.), выясняются основные используемые понятия, термины и определения, ожидаемые функциональные требования к системе, такие как требования к интерфейсу (описывает входы, получаемые из внешних систем, и выходы, направляемые во внешние системы. Накладываются ли на эти интерфейсы какие-то ограничения?), данным (описывают входные и выходные данные системы, формат данных, их хранение), производительности (описывают проблемы масштабирования и синхронизации, например, сколько пользователей одновременно должна обслуживать система), требования к пользователям и человеческому фактору (кто будет работать с программным обеспечением, учет необходимого уровня удобства и простоты использования), физическим средствам тестирования (операционная система под управлением которой выполняется программный продукт, и вычислительная платформа, на которой эксплуатируется система), безопасности (доступ к программному продукту и управление данными), документации (определяется в каком виде она должна быть и должна ли быть вообще), устранению неисправностей (реакция системы на неисправности), сопровождению (определяется, как производится устранение проблем обнаруженных в системе).

***Планирование испытаний.***

Для того, чтобы тестирование было эффективным, необходимо потратить значительные средства и усилия на планирование. На этапе планирования определяется стратегия тестирования, производится оценка времени для проведения тестовых работ, определяется состав и структура испытательной системы (выявление аппаратных и программных средств тестирования), подготовка и утверждение плана проведения испытаний. План проведения испытаний дает описание подхода, который предусматривается задействовать при проведении тестирования, а также объем трудозатрат на тестирование.

***Проектирование тестов.***

На этапе проектирования и разработки тестов определяются цели тестов, спецификации для ввода каждого теста, тестовая конфигурация. Производится автоматизация часто используемых тестов, требующих больших затрат времени. Разрабатывается методика тестирования, обладающая требуемым уровнем детализации. В результате выполнения этих действий будет получен набор тестовых случаев (test cases), который может использоваться для проведения испытаний.

***Реализация и отладка тестов.***

После этапа проектирования тест необходимо проверить на наличие дефектов с целью их немедленного устранения, и затем испытать на некоторой сборке программного продукта. После прогона тестов необходимо провести анализ неудачных исходов прогона тестов, выяснить причину источника неудачи теста – программный код или сам тест.

***Системное тестирование.***

Входными данными для системного тестирования является набор отлаженных тестов. Системное тестирование проводится для удостоверения того, что программное обеспечение делает именно то, что от него ожидает пользователь. Используются два основных типа испытаний: функциональная проверка и испытания для определения рабочих характеристик. Функциональная проверка использует набор тестов, которые определяют, выполняет ли система всё то, что она должна делать с точки зрения пользователя. Производится тестирование функций программного продукта (т.е. поиск различий между разработанной программой и ее спецификацией).

Проводится многократное тестирование каждой функции системы, множества их сочетаний, имитация и проверка на наличие сбоев, применительно к каждому функциональному требованию.

После того, как тестирование подтвердило адекватность базовой функциональности системы, задачей тестирования становятся испытания для определения рабочих характеристик. В рамках данной задачи выполняется тестирование в предельных режимах, нагрузочные испытания (так например, если программа рассчитана на подключение к ней большого числа пользователей, то её необходимо тестировать при максимально возможной нагрузке.

Для программ, выполняющих операции «запись-чтение», проводится тестирование операций обработки больших массивов данных с оценкой изменения времени отклика для выявления коэффициента пропорциональности падения быстродействия с увеличением количества обрабатываемых данных. Проводятся испытания на надежность и эксплуатационную готовность. Также проводится ряд дополнительных испытаний, таких как проверка безопасности, установочная проверка, проверка на совместимость, проверка удобства и простоты использования.

***Приемочные испытания.***

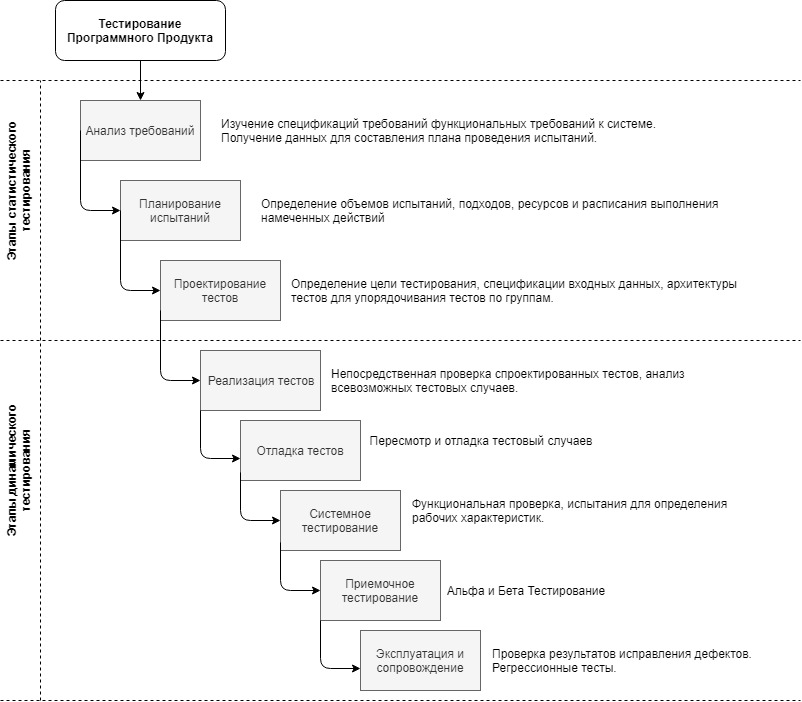
По завершении системного тестирования продукт может быть передан пользователю для проведения приемочных испытаний (acceptance testing). Производится бета-тестирование, также возможен случай, когда заказчик выполняет заранее определенный набор тестовых случаев, имитирующих типовые условия, в которых система будет работать после ввода в эксплуатацию. Заключительным видом приемочных испытаний является установочная проверка (installation testing), по условиям которой завершенная версия программного продукта устанавливается на площадках заказчика с целью получить от него подтверждение, что программный продукт соответствует всем требованиям.

***Сопровождение.***

Сопровождение означает проверку результатов исправления дефектов, которые были найдены заказчиком в процессе эксплуатации программного продукта, тестирование расширенных функциональных возможностей и выполнение регрессионных тестов на новых версиях программного продукта.

Цель всего этого заключается в получении подтверждения того, что ранее исправно работавшие функциональные средства не пострадали от внесения изменений в программный продукт, и новые добавленные функции также работают корректно.

Схема, описывающая жизненный цикл этапа тестирования представлена на рисунке 1.1.



**Рисунок 1.1** – **Схема жизненного цикла этапа тестирования**

**Источник: Собственная разработка автора**

## **1.3 Основные проблемы управления проектами на стадии тестирования**

Основываясь на опыте работы тестировщиков, можно выделить следующие основные проблемы, возникающие на этапе тестирования.

1. **Нечеткие требования.** Примером может стать некорректно составленные требования и разработанный по ним функционал. Данный продукт с высокой вероятностью не будет соответствовать ожиданиям заказчика.
2. **Неверные эстимации на тестирование или несоблюдение графика работ** возникают, когда разработчики не соблюдают сроки выполнения задач, не отдают на тестирование функционал в срок, и как следствие, тестировщикам остается меньше времени на тестирование.
3. **Неверное сопоставление ресурсов с задачами проекта (трудовые ресурсы-персонал).** Примером является ситуация, когда разработка функционала не требовала больших временных затрат и задачу мог реализовать один программист за 2 дня. Тестирование функционала занимала больше времени в силу своей сложности (5 дней) и требовала высоких навыков от тестировщика. Для решения поставленной задачи необходимо либо два тестировщика с небольшим опытом, либо один с высоким уровнем знаний. Но менеджер собрал команду из двух программистов и одного тестировщика. Как следствие, в условиях ограниченного по времени спринта (одна неделя), сроки сдачи функционала в релиз будут нарушены, а в релиз выйдут критичные баги.
4. **Игнорирование мелких проблем на ранних стадиях разработки проекта**, которые могут перерасти в большие проблемы на стадии тестирования.
5. **Нарушение плана разработки продукта со стороны менеджмента**. Примером может стать случай, когда менеджер не придерживается установленного списка разрабатываемого функционала в спринте, или чрезмерная внимательность к пожеланиям заказчика. Такое поведение менеджера может привести к тому, что разрабатываемый функционал будет отложен на середине, а новый передан в разработку и тестирование, потому что так захотел заказчик. Необходимо убедить его в том, что лучше закончить один спринт, а в новый спринт включить «горячие исправления», иначе это может вызвать проблемы у тестировщика, например, ему понадобится дополнительное время на переключение с одной задачи на вторую.
6. **Человеческий фактор.** Например менеджер тестирования не прислушивается к своей команде, или берет на себя слишком много задач и, естественно, не успевает все сделать сам.

Выявленные проблемы являются предпосылками для возникновения рисков в процессе тестирования

## **1.4 Риски в тестировании ПО**

**Риск –** это существующий или развивающийся фактор процесса, который обладает потенциально негативным воздействием на процесс [3]. Стоит разграничивать понятия проблемы и риска. Риск **–** это предполагаемое, возможная проблема, которая еще не случилась, а проблема **–** это то, с чем пришлось столкнуться. И риск, и проблема являются факторами, которые могут или уже затруднили работу по производству или тестированию программного продукта. Также стоит отметить, что способы работы с рисками и проблемами отличаются друг от друга. Так, проблему необходимо устранять и минимизировать последствия, а работа с рисками предполагает их минимизацию или недопущение. Работа по минимизации и избеганию рисков называется областью управления рисками.

Управление рисками можно представить в виде циклических действий, представленных на рисунке 1.2 ниже. Цикличность работы с рисками происходит в рамках каждой итерации. при этом если итерации длинные, циклов работы с рисками может быть несколько.



**Рисунок 1.2 – Схема действий по управлению рисками**

**Источник: [3]**

Основными сложностями с работой по управлению рисками могут стать внедрение таких работ в процесс планирования этапа тестирования, планирование управления рисками и анализ результатов работ, то есть анализ того, какие риски смогли избежать, какие риски оправдались и тому подобное.

Стоит отметить, что работа по управлению рисками должна следовать определенным целям. Данная работа должна приносить результаты. Следует понимать, для чего и кем будет проводиться работа по управлению рисками, кто будет участвовать в анализе результатов и в принятии решений.

***Виды рисков***, которые могут быть выявлены на этапе тестирования программного продукта представлены в таблице 1.1.

Таблица 1.1 – Виды рисков, которые могут быть выявлены на этапе тестирования ПО

|  |  |  |  |
| --- | --- | --- | --- |
| **Риск** | **Описание риска и причин появления** | **Уровень. Вероятность** | **Способы избежания или минимизации** |
| **Неверная или неполная оценка трудозатрат.** | Оценка трудозатрат лиц не участвующих в непосредственном тестировании ПО. | Проектный риск. Низкая вероятность. | Может управляться и минимизироваться группой тестирования или её менеджером, путем включения тестировщиков в процесс получения оценок по трудозатратам и ревью полученных оценок и планов проекта. |
| **Тест-план не привязан к плану проекта** | Часть работ по тестированию не будет проведена вовремя и не успеет быть реализована до дедлайна. | Процесс тестирования. Высокая вероятность появления. | План тестирования должен быть жестко связан с планом проекта по времени (на уровне одного плана работ). Либо должен быть синхронизирован на постоянной основе. |
| **Стратегия тестирования не принята заказчиком или группой разработки.** | Стратегия тестирования не будет обеспечена проектными или временными ресурсами. | Проектный риск. Низкая вероятность. | Стратегия тестирования нуждается в утверждении заказчиком и командой разработчиков. |
| **Увольнение сотрудников.** | Увольнение ключевого или не ключевого сотрудника во время разгара проекта. Дополнительные финансовые и временные затраты на введение нового сотрудника и его обучение. | Процесс тестирования. Высокая средняя появления. | Держать в “запасе” новых сотрудников (экономически не всегда возможно). Предложить больше ЗП (только к ключевым сотрудникам). |
| **Непредвиденные риски** | Изменение требований, и как следствие смещение и пересмотр планов работ и нестабильность сроков поставки версий. | Процесс тестирования. Высокая вероятность появления. | Ограничение объемов итераций по срокам, добавление всех изменений и работ с ними в следующую итерацию. |
| **Риск игнорирования рисков.** | Отсутствие принятие рисков. Нежелание принимать во внимание факт, что процесс разработки может давать сбои. И как следствие построение оптимистичных планов работ, правленные сроки сдачи работ, конфликты. | Проектный риск. Низкая вероятность. | Работа с рисками на постоянной основе. |

Источник: Собственная разработка автора

В процессе тестирования рисков немного. Гораздо больше рисков можно выявить на проектном уровне. Большинство из которых может быть решена общими усилиями команды (группы управления проектом, тестирования и разработчиков).

Для решения выявленных выше проблем следует проводить регулярный мониторинг показателей процесса тестирования. С этой целью можно использовать метрики этапа тестирования, которые позволяют измерить характеристики процесса и результатов тестирования, помогают визуализировать процесс, состояние тестов, оценить результаты тестирования и качество продукта. Также метрики помогут представить информацию о тестировании разных проектов в виде, пригодном для хранения, накопления и анализа. Это, в свою очередь, послужит инструментом планирования и управления этапом тестирования.

## **1.5 Данные и измерения как основа выявления метрик для планирования, разработки и тестирования ИT-проектов**

**Данные.**

Данные представляют собой определенным образом полученные и зафиксированные наблюдения относительно окружающей действительности. Термин данные происходит от слова data – факт. Это совокупность сведений, зафиксированных на определенном носителе в форме, пригодной для постоянного хранения, передачи и обработки. Преобразование и обработка данных позволяет получить информацию. Как указано в [4], "термин данные весьма популярен в научных исследованиях. В широком смысле он означает фактический материал, являющийся основой для обсуждения или принятия решений, а в статистике — это информация, пригодная для анализа и интерпретации."

**Информация** – это результат преобразования и анализа данных. Отличие информации от данных состоит в том, что данные – это фиксированные сведения о событиях и явлениях, которые хранятся на определенных носителях, а информация появляется в результате обработки данных при решении конкретных задач. Например, в базах данных хранятся различные данные, а по определенному запросу система управления базой данных выдает требуемую информацию.

Необходимо отметить, что универсальных определений этих понятий (данных, информации) нет, они трактуются по-разному.

Данные получаются в результате измерений исследуемых объектов. Под измерением понимают [4], "присвоение символов подопытным образцам в соответствии с некоторым правилом." Эти символы могут быть буквенными (или иными символами) или числовыми. Числовые символы также могут играть роль символов (см. ниже номинальные шкалы) или быть числами. В первом случае к ним нельзя применять правила арифметики, во втором — можно. Например, если 1 обозначает класс мужчин, а 2 – женщин, то в этом контексте 1 + 2 не имеет смысла. Однако если 2 – число выявленных за некоторый день багов, а 3 – за следующий день, то 1 + 2 = 3 имеет смысл и означает, что за два дня выявлено 5 багов.

**Виды данных.**

Исходя из того, для каких целей были собраны данные, их можно разделить на первичные и вторичные. Первичными являются такие данные, которые собраны для решения целей данного исследования. Соответственно, вторичными являются данные, собранные в рамках другого исследования, то есть для решения целей, отличных от целей данного исследования. Принято начинать с анализа вторичных данных, а уже в случае их недостаточности переходить к сбору первичных.

Для того, чтобы производить измерения, надо иметь шкалу. Шкала и единицы измерений могут быть самыми разными [4, с.15].

* Шкала наименований
* Порядковая шкала
* Интервальная
* Шкала отношений

Существует много классификаций типов данных и соответствующих шкал. Опишем основные.

**Номинальные данные.**

Термин «номинальный» имеет отношение к латинскому слову «nomen», которое означает «относящийся к именам». Этот вид данных называют номинальными, поскольку они содержат названия категорий, по которым распределяются данные. Номинальные данные по определению не упорядочены. Номинальные данные можно посчитать, можно определить процент от целого, однако нельзя вычислить среднее значение.

В случае, если доступны только две категории, данные относят к типу дихотомических. Ответы на вопросы, требующие ответа «да-нет», − это и есть дихотомические данные.

**Порядковые данные.**

Переменную можно рассматривать как порядковую, когда ее значения представляют категории с некоторым естественным для них упорядочением, например, уровни удовлетворенности обслуживанием от крайней неудовлетворенности до крайней удовлетворенности. Примеры порядковых переменных включают баллы, представляющие степень удовлетворенности или уверенности, или баллы, оценивающие предпочтение.

Так же как и номинальные данные, порядковые данные можно подсчитать и определить процент от целого, однако нет единого мнения о том, можно ли для порядковых данных рассчитать среднее значение.

**Интервальные данные.**

Переменную можно рассматривать как количественную (непрерывную), когда ее значения представляют упорядоченные категории с осмысленной метрикой, так что уместно сравнивать расстояния между значениями. Примеры количественной переменной включают возраст в годах и доход в тысячах долларов.

Интервальные данные − числовые, с ними можно производить математические операции, однако такие данные не имеют «значимой» нулевой точки − то есть при значении ноль то, что подлежит измерению, не отсутствует. Например 0:00 часов означает не отсутствие времени, а начало нового дня.

**Нормативные данные.**

Нормативные данные − числовые, и имеют много общего с интервальными данными, кроме того, что нормативные, в отличие от интервальных, имеют значимую нулевую точку. В нормативных данных ноль означает отсутствие того, что вы измеряете, − ноль минут.

Интервальные и нормативные данные могут быть либо дискретными, или непрерывными. Дискретные данные выражены ограниченным набором значений (обычно целыми числами), величины между этими значениями невозможны. Непрерывные данные могут принимать любое значение на шкале. Это не значит, что данные могут принимать все возможные числовые значения − только все значения в рамках границ шкалы.

Для простоты объяснения часто округляют непрерывные данные до определенного количества цифр. Но в таком случае эти данные остаются непрерывными, а не дискретными.

**Категориальные данные.**

Ранее мы говорилось о номинальных и порядковых данных как о способе распределить данные по категориям. Некоторые источники считают, что оба типа принадлежат к категориальным данным, где номинальные данные − неупорядоченные категориальные данные, а порядковые − упорядоченные. Другие источники относят к категориальным данным только номинальные, и считают, что понятия “номинальные данные” и “категориальные данные” − взаимозаменяемы. Эти источники относят порядковые данные к отдельной группе.

**Качественные и количественные данные.**

Качественные данные, грубо говоря, относятся к нечисловым данным, в то время как количественные данные − числовые и, соответственно, поддающиеся счету. По отношению к этим терминам существует некое общее мнение. Определенные данные всегда считаются качественными, поскольку требуют предобработки или других методов анализа, чем количественные данные. Примерами могут считаться записи прямого наблюдения либо транскрипты интервью. Подобным образом, интервальные и нормативные данные всегда считаются количественными, поскольку они всегда числовые. Однако есть некое расхождение во мнениях относительно номинальных и порядковых типов данных. Некоторые источники называют их качественными, так как их категории описательные, а не числовые. Однако, поскольку эти данные можно рассчитать и использовать для подсчета процентов, другие источники считают их количественными, поскольку они в этом смысле поддаются счету.

В соответствии с типами данных, выделяют четыре основных типа шкал, к ним относятся номинальная шкала, интервальная, порядковая и относительная.

*Номинальная*. Одина из самых очевидных и базовых шкал. По данной шкале к каждому объекту присваивается его идентификационный номер, например, номера футболистов в команде или номера телефонов.

*Порядковая.* Определяет ранг или порядок объектов. Расстояние между объектами, которые следуют друг за другом не являются равными. Нельзя сказать что расстояние межд объектом 1 и 2 будет равно расстоянию между объектом 3 и 4. Данную шкалу еще часто называют шкалой восприятия. К примеру оценка качества фильма от 1 до 5, где 5 баллов получает наиболее понравившейся фильм, а наименее 1 балл.

*Интервальная.* В данной шкале имеет значение не столько порядок следования величин, сколько величина интервала. Примером данных шкал может служить температура воздуха или возраст человека.

*Относительная или шкала отношений.* Отражает то, во сколько один показатель больше другого. В данной шкале необходим нулевой показатель, отправная точка. Например, цена на товар. На практике к данному типу шкал не часто удается привести измерения.

## **1.6 Принятие решений при планировании, разработке и тестировании ИT-проектов**

Принятия решений осуществляются на основе полученной информации и имеющихся знаний.

**Принятие решений** – это выбор наилучшего в некотором смысле варианта решения из множества допустимых на основании имеющейся информации. Взаимосвязь данных, информации и знаний в процессе принятия решений представлена на рисунке.

При рассмотрении задачи проектирования теста необходимо выделить два важных артефакта: тестовые сценарии и тестовые наборы. В отсутствие данных для тестов эти два артефакта реализовать невозможно. Они являются описаниями условий, сценариев и путей, но не предусматривают конкретных значений для реализации. Тестовые данные сами по себе не являются артефактом, но заметно влияют на успех или неудачу теста. Тестирование не может быть выполнено без данных для тестов. Они требуются для следующего:

1. Входные данные для создания условия;
2. Выходные данные для оценки требования;
3. Вспомогательные данные (как предварительное условие для теста).

Поэтому определение значений – это важная часть работы при создании тестовых наборов.

При определении фактических данных для тестов следует учитывать четыре параметра:

1. Объем данных – количество тестовых данных;

***Объем*** – это количество данных для тестирования. *Объем данных* – это важный параметр; если данных слишком мало, они могут не отразить все возможные ситуации, если же слишком много – то с ними будет трудно работать. Лучше всего, если тестирование начинается с небольшого объема данных, пригодного для критически важных (и, как правило, позитивных) тестовых наборов. По мере того как достоверность тестирования будет увеличиваться, объем данных должен расширяться, пока не охватит все существенные случаи развертывания в среде (в разумных пределах).

1. ***Полнота*** –– степень вариации тестовых данных;

Полнота данных означает степень изменения данных для тестирования. Ее можно увеличивать, создавая больше записей. Часто этого достаточно, но фактически требуется видеть изменения данных, с которыми можно встретиться в реальной ситуации. Без этого в тестировании можно упустить из виду какие-либо ошибки.

1. ***Охват*** –– применимость данных для цели тестирования;

Это применимость тестовых данных для цели теста. Она связана с объемом и полнотой данных. Большое число данных не означает, что среди них обязательно есть нужные данные. Как и в том, что касается полноты тестовых данных, мы должны убедиться, что данные отвечают цели теста, то есть приведет ли выполнение теста к ответу на поставленные вопросы.

1. Архитектура –– физическая структура тестовых данных;

Понятие физической структуры тестовых данных относится только к тестовым данным, которые применяются как вспомогательные к уже имеющимся данным, например, в базе данных или таблице правил.

Тестирование –– это не процесс, который происходит только один раз. Тестирование повторяется и в итерациях, и между ними. Для того чтобы тестирование было точным, достоверным и эффективным, тестовые данные следует вернуть в исходное состояние перед выполнением теста. Это особенно важно для автоматизированных тестов.

Точность, достоверность и эффективность тестирования достигаются тогда, когда тестовые данные не зависят от внешних факторов, и их состояние известно до, во время и после выполнения теста. Для достижения этой цели необходимо решить две проблемы:

1. *неустойчивость / изоляция* – устранение внешних влияний на данные;
2. *начальное состояние* – точное знание начального состояния данных и воспроизводимость этого состояния.

Все эти вопросы влияют на работу с базой данных для тестирования, проектирование модели тестирования и взаимодействие с другими объектами.

## **1.7 Сбор данных для планирования, разработки и тестирования ИT-проектов**

В данном разделе описаны организационные процессы по получению, обработке и анализу данных о проекте.

По способу получения, данные можно разделить на четыре типа:

***Наблюдение.*** Фиксация случаев, ситуаций или событий, известных из собственного опыта, в том числе с использованием специальных средств (например, камеры, диктофона, микроскопа и т.д.).

***Участие.*** Данные получаются благодаря опыту, который может рассматриваться как интенсивная форма наблюдения (например, опыт обучения управлению автомобилем сообщит такие нюансы относительно авто, которые невозможно получить, лишь наблюдая за ним извне).

***Измерение.*** Фиксация величины или количества какого-либо параметра (например, демографическая статистика, измерение физических величин и т.д.).

***Интерегация.*** Данные получаются посредством вопросов к людям (информация относительно убеждений людей, их мотивации и т.д.).

Главной задачей тестирования является оценка соответствия конечного результата требованиям. Проектные требования стоят в основе описания необходимых для разработки продукта временных, денежных, трудовых ресурсов; необходимых инструментов и данных.

**Источники требований:**

Технические задания (ТЗ и ТУ) – документ в котором требования обычно изложены конкретно, однозначно и организованно. С такими требованиями работать проще всего. Иногда такие требования нуждаются в уточнении. К примеру, из требования «Передаваемые по сети данные должны шифроваться» непонятно, шифровать только значимые данные, или и служебные тоже.

Требования могут быть высказаны заказчиком в ходе обсуждения проекта. Заказчик в большинстве случаев сам не знает, что конкретно ему нужно. Часто высказывает противоречивые или плохо сформулированные требования. У заказчика, еще на самых начальных этапах разработки полезно поинтересоваться, насколько критично то, или иное требование. Это поможет спланировать реализацию функций разрабатываемого продукта от важных к менее важным.

Требования могут быть обусловлены характером изделия. Например, 24 кадра в секунду минимум для видеоплеера. Такой вид требований не обязательно документировать.

В общем случае невозможно определить все требования с самого начала. Перечисленные источники – это лишь то, к чему можно обратиться на начальных стадиях разработки. Источником новых требований может стать любой документ в том числе и новые желания заказчика. По завершении некоторых задач, полезно просматривать результаты работ на предмет новых требований.

Еще примеры источников требований: модель доступа, предполагаемая архитектура, обсуждения проекта с заказчиком, use-case`ы, некоторые стандарты и ГОСТы.

После того, как требования собраны, они должны быть задокументированы. Все требования желательно хранить централизованно.

Выделяются следующие группы требований:

***Функциональные требования*** – функциональность, свойства, возможности, безопасность, и т.д.

***Требования надежности*** – частота сбоев, возможность восстановления, предсказуемость поведения.

***Требования производительности*** – время отклика, точность, доступность, использование ресурсов.

***Возможность поддержки и конфигурирования*** – адаптивность, соответствие стандартам, возможность конфигурирования.

***Требования к реализации*** – требования к ресурсам, языки и средства разработки, аппаратное обеспечение.

***Требования к интерфейсам*** – ограничения, накладываемые необходимостью взаимодействия с внешними системами.

***Управление системой*** – требования к инструментам управления системой, к интерфейсам управления и пользовательскому интерфейсу.

*Т****ребования удобства и эргономики*** – человеческий фактор, справочная система, документация.

***Юридические вопросы*** – авторское право, использование компонентов сторонних разработчиков и пр.

Возможные атрибуты требований:

***Уникальный идентификатор****,* состоящий из первой буквы названия класса требований (Ф – функциональные, Н – надежность) и целого числа. Нумерация списков в Word может сместиться в случае перемещения требования или добавлении нового, а такой идентификатор (записанный простым текстом) никуда не денется. При ссылке на требование из другого документа достаточно указать этот идентификатор.

***Актуальность*** («Действительно», «Снято», «Под вопросом») – часть требований в процессе разработки снимается, но не следует сразу удалять их. Возможно, кто-то ссылался на них из других документов.

***Источник требования*** – документ, либо лицо, выдвинувшее данное требование. Если требование сформулирован плохо, или непонятно откуда взялось, можно будет уточнить, обратившись к источнику.

***Дата введения требовани****я*.

***Проектирование*** – когда и кем требование было учтено при проектировании, и в каком проектном документе можно найти решение, подтверждающее, что требование учтено.

***Реализация*** – когда и кем требование было реализовано, и в каком модуле/файле/документе можно найти реализацию этого требования.

***Тестирование*** – когда и кем была проверена и подтверждена корректность реализации требования. Также может быть указан использовавшийся при проверке Test-Case.

Существуют также и организационные требования, которые обычно утверждаются менеджерами проекта на самых первых этапах общения с заказчиком:

***Временные затраты*** (протяженность проекта);

***Денежные затраты*** (стоимость проекта);

***Трудовые затраты*** (команда: менеджеры, разработчики, тестировщики, и тд);

***Организационные процессы*** (полномочия и ответственность членов команды, процессы выхода новых разработок в продакшн) и т.д.

Помимо требований существуют иные источники информации, которые также необходимы при разработке и тестировании приложения. Например для решения задачи эстимации можно применить опыт прошлых проектов. К даным источником можно отнести:

***Экспертное мнение*;**

***Исторические примеры*** (например опыт прошлых подобных проектов, знания);

***Нормативные источники*** (нормы качества; ISO) и т.д.

**Знания** – это зафиксированная и проверенная практикой обработанная информация, которая использовалась и может многократно использоваться для принятия решений.

Знания – это вид информации, которая хранится в базе знаний и отображает знания специалиста в конкретной предметной области. Знания – это интеллектуальный капитал. Формальные знания могут быть в виде документов (стандартов, нормативов), регламентирующих принятие решений или  учебников, инструкций с описанием решения задач. Неформальные знания – это знания и опыт специалистов в определенной предметной области.

Основными исходными данными, которые необходимо собрать о разрабатываемом проекте, являются проектные данные. Исходные данные могут быть применены в расчетных метриках, необходимых для выбора наиболее подходящей стратегии тестирования, либо сами являются метриками. Отсутствие единой базы систематизации накопленных знаний и результатов применения метрик является проблемой анализа и планирования этапа тестирования и ИТ-проекта в целом. Что, в свою очередь, затрудняет понимание и назначение тех или иных метрик.

## **Выводы по главе 1**

Тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов. Основными этапами тестирования являются анализ требований, планирование испытаний, проектирование тестов, реализация и отладка тестов, системное тестирование, приемочные испытания, сопровождение.

В процессе тестирования могут возникать следующие проблемы: нечеткие требования, неверные эстимации на тестирование или несоблюдение графика работ, неверное сопоставление ресурсов с задачами проекта (трудовые ресурсы-персонал), игнорирование мелких проблем на ранних стадиях разработки проекта, нарушение плана разработки продукта со стороны менеджмента, человеческий фактор. Проблемы являются предпосылками для возникновения рисков. На этапе тестирования программного продукта могут быть выявлены следующие виды рисков: неверная или неполная оценка трудозатрат, тест-план не привязан к плану проекта, стратегия тестирования не принята заказчиком или группой разработки, увольнение сотрудников, непредвиденные риски, риск игнорирования рисков.

Для решения выявленных выше проблем следует проводить регулярный мониторинг показателей процесса тестирования. С этой целью можно использовать метрики этапа тестирования, которые позволяют измерить характеристики процесса и результатов тестирования, помогают визуализировать процесс, состояние тестов, оценить результаты тестирования и качество продукта. Также метрики помогут представить информацию о тестировании разных проектов в виде, пригодном для хранения, накопления и анализа. Для формирования системы метрик используются различные типы данных и различные шкалы.

Принятия решений осуществляются на основе полученной информации и имеющихся знаний. В результате преобразования и анализа данных получают информацию, которая может использоваться для принятия решений при планировании, разработке и тестировании ИT-проектов.

Основными исходными данными, которые необходимо собрать о разрабатываемом проекте, являются проектные данные. Исходные данные могут быть применены в расчетных метриках, необходимых для выбора наиболее подходящей стратегии тестирования, либо сами являются метриками. Отсутствие единой базы систематизации накопленных знаний и результатов применения метрик является проблемой анализа и планирования этапа тестирования и ИТ-проекта в целом. Что, в свою очередь, затрудняет понимание и назначение тех или иных метрик.

# **ГЛАВА 2 МЕТРИКИ ПО ОБЕСПЕЧЕНИЮ КАЧЕСТВА ЭТАПА ТЕСТИРОВАНИЯ И ПРОГРАММНОГО ПРОДУКТА**

## **2.1 Понятие качества программного продукта**

**Качество программного обеспечения** — способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям [5].

Другие определения из стандартов:

Весь объем признаков и характеристик программ, который относится к их способности удовлетворять установленным или предполагаемым потребностям [6].

Степень, в которой система, компонент или процесс удовлетворяют потребностям или ожиданиям заказчика или пользователя [7].

**Модели качества.**

Стандарт ISO/IEC 25010:2011 (ГОСТ Р ИСО/МЭК 25010-2015) определяет модель качества продукта, которая включает восемь характеристик верхнего уровня [8]:

1. Функциональная пригодность;
2. Уровень производительности;
3. Совместимость;
4. Удобство использования (юзабилити);
5. Надежность;
6. Защищённость;
7. Сопровождаемость;
8. Переносимость (мобильность).

В этом стандарте модель качества продукта (англ. software product quality model) рассматривается отдельно от субъективного качества в использовании, которое может сильно отличаться для различных стейкхолдеров. Модель качества в использовании (англ. quality in use model) включает следующие характеристики верхнего уровня[8]:

1. Результативность;
2. Производительность;
3. Удовлетворенность;
4. Свобода от риска;
5. Покрытие контекста.

Роберт Гласс в известной книге «Факты и заблуждения профессионального программирования» утверждает, что большинство профессиональных разработчиков согласны с выделением семи показателей качества как основных[9]:

1. Переносимость;
2. Надежность;
3. Эффективность;
4. Удобство использования (юзабилити);
5. Тестируемость;
6. Понятность;
7. Модифицируемость.

Главной задачей процесса тестирования является подтверждение того, что продукт является качественным. Основной проблемой является выявление показателей которые бы четко отразили качество продукта. Данных показателей не существует, так как качество разработанного продукта оценивают разные люди (заказчик, разработчик, тестировщик, пользователь) для которых характеристики качества разные.

Такая путаница может объясняться несколькими причинами.

1. Качество это не отдельно взятая идея или понятие, скорее многомерная и разноплановая концепция;
2. Для любого понятия и определения существует несколько уровней абстракции, например, когда люди говорят о качестве, одна часть понимает под этим слишком широкий и размытый смысл, в то время как другая может ссылаться на конкретное определение и значение;
3. Термин качество является неотъемлемой частью нашего повседневного общения, однако общепринятое и профессиональное использование может быть весьма сильно отличаться.

В качестве инструмента управления качества могут служить выявляемые или рассчитанные метрики. Они способны отразить и визуализировать процесс тестирования на основе которого можно судить о качестве или непригодности программного продукта к использованию. Так например метрики оценки качества требований способны на ранних этапах указать заказчику о неправильности составления требований и предотвратить разработку неверного кода и будущих багов. Что, в свою очередь, несет снижение рисков, а также временных и материальных затрат.

С точки зрения тестировщика **качество** (quality) – показатель степени соответствия продукта предъявляемым к нему требованиям. Следует помнить, что качество продукта определяется качеством процесса его разработки.

Ниже представлены некоторые общепринятые рассуждения о качестве:

1. Если заказчик доволен продуктом – продукт качественный;
2. Если продукт соответствует требованиям – продукт качественный;
3. У качественного продукта всегда есть какие-то преимущества и нет серьёзных недостатков;

Ниже приведены лишь некоторые примеры того, как могут формулироваться критерии качества.

1. Покрытие требований тестами – не менее 80%;
2. Закрыто 100% известных критических дефектов, 90% дефектов средней критичности, 50% остальных дефектов;
3. Общий показатель прохождения тестов – не менее некоторого значения: X = (Passed/Executed)\*100%.

**Соответствие требованиям** предполагает, что требования должны быть настолько четко определены, что они не могут быть поняты и интерпретированы некорректно. Позже, на этапе разработки, производятся регулярные измерения разработанного продукта, для определения соответствия требованиям. Любые несоответствия должны рассматриваются как дефекты – отсутствие качества. Например, спецификация на определенную модель радиостанции может требовать возможности принимать определенную частоту радиоволн на расстоянии более чем 30 километров от источника вещания. В случае, если радиостанция не способна выполнить данное требование, она не удовлетворяет требования к качеству и должна быть признана негодной и некачественной. Исходя из тех же принципов, если Кадиллак соответствует всем требованиям к машинам Кадиллак, значит это качественная машина. Если Шевроле соответствует всем требованиям к машинам Шевроле, следовательно, это тоже качественная машина. Эти две машины могут быть совершенно разными по стилю, скорости и экономичности, но если обе измерять по стандартным для них наборам, тогда они обе будут являться качественными машинами.

**Пригодность к использованию** определение  принимает во внимание требования и ожидания конечных пользователей продукта, которые ожидают, что продукт или предоставляемый сервис будет удобным для их нужд. Однако разные пользователи могут использовать продукт по-разному, это означает, что продукт должен обладать максимально разнообразными вариантами использования. Согласно определению Juran каждый вариант использования это характеристика качества и все они могут быть классифицированы по категориям в качестве параметров пригодности к использованию.

Эти два определения качества («соответствие требованиям» и «пригодность к использованию») по существу одинаковы. Разница в том, что вариант «пригодность к использованию» указывает на важную роль требований и ожиданий заказчика. Роль заказчика, связанная с качеством, никогда не может быть переоценена. С точки зрения заказчика, качество продукта, который он приобрел, состоит из множества различных факторов, таких как: цена, производительность, надежность и т.д.

Выводы о качестве программного продукта является его соответствие с реальными требованиями, явным и неявным. Очень часто неявные требования настолько очевидны для заказчика или пользователя, что он даже не предполагает, что они неизвестны разработчикам. Для примера вернемся к нашим автомобилям — заказчик может детально описать требования к дизайну, параметрам двигателя, оформлению салона, цвету кузова, но нигде не указать, что шины должны быть круглыми, а лобовое стекло – прозрачным.

Заказчик будет удовлетворен только тогда, когда купленный продукт будет полностью удовлетворять его реальным и жизненным требованиям, как специфицированным, так и нет.

## **2.2 Понятие и классификация метрик**

Согласно международному стандарту [ISO 14598](http://www.protesting.ru/documentation/info_isoiec14598_1_ed1_0_en.pdf) [10]: **Метрика**– это количественный масштаб и метод, который может использоваться для измерения.

Введение и использование метрик необходимо для улучшения контроля этапа тестирования, что позволяет повлиять на принятие своевременных управленческих решений. Метрики могут стать хорошим инструментом визуализации критериев качества.

Цель контроля тестирования состоит в получении обратной связи и визуализации процесса тестирования. Необходимую для контроля информацию собирают (как вручную, так и автоматически) и используют для оценки состояния и принятия решений, таких как покрытие требований или кода тестами, или критерии окончания тестирования. Метрики также могут быть использованы для оценки прогресса выполнения запланированных работ и освоения бюджета.

Метрики, применяемые на этапе тестирования ПО, можно классифицировать по различным признакам.

Рассмотрим возможные классификации метрик по следующим признакам:

1. По этапам тестирования;
2. По сущностям.

По способу получения значений метрики можно разделить на:

1. Расчетные метрики, которые вычисляются по определенным правилам из некоторых первичных показателей, измеренных в заданных шкалах;
2. Измеряемые, которые сами являются первичной информацией.

### 2.2.1 Метрики по этапам тестирования

Метрики, которые могут быть применены на разных **этапах тестирования,** приведены в таблице 2.1.

Таблица 2.1 – Метрики, применяемые на разных этапах тестирования

|  |  |
| --- | --- |
| **Этап тестирования** | **Метрики применяемые на этапах тестирования** |
| ***Анализ требований*** | Качество требований; Сложность требований. |
| ***Тест дизайн*** | Скорость разработки ТС (ТС – тест кейсы); Сложность ТС. Количество шагов; Пригодность теста к smoke тестам; |
| ***Функциональное тестирование*** | Плотность дефектов; Коэффициент ретеста; Коэффициент регрессии; Происхождение дефектов |
| ***Автотестирование*** | Скорость разработки автотестов; Длительность выполнения автотестов; Дефекты в автотестах; Время жизни дефекта; Статус тестового набора. |
| ***Внедрение и сопровождение*** | Количество пропущенных дефектов в PROD; Статистика использования. |

Источник: Собственная разработка автора

Метрики **анализа требований** основаны на качественно-количественной оценке. Можно измерить как характеристики каждого конкретного требования, так и всего набора требований в целом. Также можно выделить оценки (или веса) каждой характеристики требований. Это могут быть веса от 1 до 5. К характеристикам требований можно отнести:

***Полноту*** – свойство, означающее, что совокупность артефактов, описывающих требования, исчерпывающим образом описывает всё то, что требуется от разрабатываемой системы);

***Понятность***–лингвистическая корректность. Требования должны быть лексически и синтаксически верны, написаны в корректном стиле на основе определенных соглашений принятых в предметной области. Данная характеристика предполагает что слова «можно, вкоэфиуиозможно» будут переведены в слова «должен, обязан»;

***Осуществимость*** – физическая возможность выполнить требования;

***Проверяемость*** – есть физическая возможность проверить выполнение требования;

Можно определить различные варианты расчета метрик оценки *качества требований*. Применение разных вариантов может быть полезно для разных наборов требований. Например, метрики расчета оценки качества отдельного требования (R1) можно рассчитать следуя примеру ниже:

***Вариант 1.*** *R1 = (Полнота + Понятность) \* Осуществимость \* Проверяемость;*

***Вариант 2*.** *R2 = (Полнота + Понятность) \* Осуществимость + Проверяемость, где* ***R***–оценка качества требований*.*

Пример расчета оценки качества требований по вышеуказанным формулам представлен в таблице 2.2.

Таблица 2.2 – Пример расчета оценки качества требований к программному обеспечению

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Требования** | **Полнота** | **Понятность** | **Осуществимость** | **Проверяемость** | **R1** | **R2** |
| **Req1** | 1 | 3 | 1 | 2 | **8** | **6** |
| **Req2** | 0 | 2 | 3 | 1 | **6** | **7** |
| **Всего** | ***0,5*** | ***2,5*** | ***2*** | ***1,5*** | ***7*** | ***6,5*** |

Источник: Собственная разработка автора

Метрики анализа требований могут быть полезны при оценки качества требований и выявление проблем с требованиями на самых ранних этапах разработки программного продукта, что позволит избежать рисков денежных затрат на исправление требований и функционала на поздних этапах разработки ПО.

Для **тест дизайна** можно выделить оценки такие, как:

***Количество шагов***– количество шагов необходимых для того, чтобы выполнить тест кейс.

***Пригодность теста к smoke тестам***– количество тестов включенных в самый необходимый набор тестов;

***Сложность или размер*** – это оценочное соотношение самого простого теста ко всем существующим (например, к самому простому тест кейсу применятся оценка 1, далее относительно этого кейса оцениваются остальные).

Сложив сложность всех тест кейсов можно понять сложность тестового набора в целом. С оценкой тестового набора, например в часах можно вычислить время его выполнения тестировщиком, т.е. решить задачу эстимации выхода продукта в релиз. Для этого можно присвоить коэффициенты тестировщикам с разным уровням мастерства (например, junior=4; midel=8 и т.д.). Пример расчета сложности прохождения тестового набора представлен в таблице 2.3.

Таблица 2.3 – Пример расчета сложности прохождения тестового набора

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест кейс** | **Шагов** | **Smoke** | **Сложность (размер)** |
| ТС 1 | 10 | Yes | **3,5** |
| ТС 2 | 3 | No | **1** |
| **Итого:** |  |  | **Общая сложность тестового набора 4,5** |

Источник: Собственная разработка автора

***Покрытие требований тест кейсами***

Еще одной большой задачей является оценка покрытия. Некоторые считает покрытие по написанным вручную кейсам. Главный минус, это человеческий фактор. Некоторые считает покрытие кода. Но здесь не учитывается с какими параметрами тот или иной метод был вызван. Существует вариант подсчета покрытие популярных пользовательских юзкейсов. Причем рассчитывается состояния системы и атомарные переходы между ними. Главное отличие от первого варианта расчета — отсутствие человеческого фактора.

***Принцип:***

1. Вычисление топ N популярных состояний тестируемой системы;
2. Подсчет парных переходов между состояниями системы;
3. Сортировка кейсов по числу переходов;
4. Проход кейсов сверху вниз;

*Тпок = (Lпок/Lвсего)\*100%, где Тпок* – тестовое покрытие, *Lпок* –количество требований проверяемых ТК, *Lвсего* – общее количество требований.

Метрики применяемые на стадии тестового дизайна полезны в решении задачи эстимации временных затрат на этапе тестирования.

**Функциональное тестирование**.

***Происхождение дефектов*** – на каком этапе тестирования был обнаружен дефект;

***Время жизни дефекта***– как долго дефект исправлялся, можно применить только для самых критичных и важных багов;

***Статус тестового набора***– отражает статус прохождения тест кейса), время прохождения ТС;

*Плотность дефекта = количество дефектов на единицу размера: на весь продукт, спринт, функцию, объем исходного кода и т.п.*

Коэффициент ретеста – показывает какую часть в тестировании занимает перепроверка дефектов

*DrTI1=Nclose/Kopen*, где *Nclose* – количество скрытых дефектов, а *Kopen* – количество новых дефектов.

*DrTI2=Tretest/Ttest*, где *Tretest* – затраченное время на ретест, а *Kopen* – затраченное время на тестирование нового функционала.

***Регрессия.*** Наиболее важный этап тестирования ПО.

(2.1),

где, Ndefr – к-во дефектов в старом функционале, а Kdefn – количество дефектов в новом функционале.

(2.2),

где, *TCrf* – количество *ТС* в статусе *Fail*, *NCrt* – всего *ТС* в регрессионном наборе.

***Индикатор регресси*:**

*Kregr*1>1, при создании нового функционала, был сломан старый. наиболее плохой показатель так как, багов больше в старой версии приложения.

*Kregr*1<1, новый функционал также повлиял на старый, но в меньшей степени. Средний показатель при наличии которого продукт нельзя считать качественным.

*Kregr*1>0.1, на каждые 10 дефектов в старом функционале найдено не более 1 дефекта в новом.

Метрики применяемые на этапе функционального тестирования полезны для оценки качества разработанного функционала.

**Внедрение и сопровождение.**

***Эффективность устранения дефектов*** – соотношение количества устраненных дефектов к общему числу найденных дефектов.

(2.3),

где, *Dfix* – количество устраненных дефектов, *Dflater* –количество дефектов, найденных заказчиком (на PROD, UAT).

***Накладные расходы***– Это работы которые необходимы в процессе разработки, но не участвуют напрямую в тестировании. К ним можно отнести митинги, настройка окружения, ожидание починки (функционала.

*Кпр = (Тм+Те+..+Тф)/Тобщ*  (2.4),

где *Тобщ* – общее время работы (например 60 часов).

Метрики применяемые на этапе функционального тестирования полезны для оценки качества тестирования разработанного функционала.

Цель контроля тестирования состоит в получении обратной связи и визуализации процесса тестирования. Необходимую для контроля информацию собирают (как вручную, так и автоматически) и используют для оценки состояния и принятия решений, таких как покрытие (например, покрытие требований или кода тестами) или критерии выхода (например, критерии окончания тестирования). Метрики, также могут быть использованы для оценки прогресса выполнения запланированных работ и освоения бюджета.

### 2.2.2 Метрики по типам сущностей

Также можно сгруппировать метрики по ***типам сущностей***, участвующих в обеспечении качества и тестировании программного обеспечения, а именно:

1. Метрики по [тест кейсам (Test Cases)](http://www.protesting.ru/testing/testcase.html). Таблица 2.4;
2. Метрики по багам / дефектам. Таблица 2.5;
3. Метрики по задачам. Таблица 2.6;
4. Метрики автотестов.

Таблица 2.4 – Метрики по тест кейсам

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Passed/Failed Test Cases | Метрика показывает результаты прохождения тест кейсов, а именно отношение количества удачно пройденных к завершившимся с ошибками. В идеале, к концу проекта, количество провальных тестов стремиться к нулю |
| Not Run Test Cases | Метрика показывает количество тест кейсов, которые еще необходимо выполнить в данной фазе тестирования. Имея данную информацию, мы можем проанализировать и выявить причины, по которым тест не были проведены |

Источник: собственная разработка автора

Таблица 2.5 – Метрики по багам

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Open/Closed Bugs | Метрика показывает отношение количества открытых багов к закрытым (исправленным и перепроверенным) |
| Reopened/Closed Bugs | Метрика показывает отношение количества переоткрытых багов к закрытым (исправленным и перепроверенным) |
| Rejected/Opened Bugs | Метрика показывает отношение количества отклоненных багов к открытым |
| Bugs by Severity | Количество багов по серьезности |
| Bugs by Priority | Количество багов по приоритету |

Источник: собственная разработка автора

Метрики "Open/Closed Bugs", "Bugs by Severity" и "Bugs by Priority" хорошо визуализируют степень приближения продукта к достижению критериев качества по багам. Имея требования к количеству открытых багов, после каждой итерации тестирования сравнивают их с реальными данными, тем самым видя места, где нам нужно прибавить, для скорейшего достижения цели.

Метрики "Reopened/Closed Bugs" и "Rejected/Opened Bugs" направлены на отслеживание работы отдельных участников групп разработки и тестирования.

Допустим, имеются ситуацию, когда количество переоткрываемых после починки багов не уменьшается или даже растет. Это является сигналом к тому, что необходимо провести анализ причин, т.к. подобная ситуация может показать, что:

1. Требования к функции можно трактовать по разному;
2. Тестировщик не точно описал проблему;
3. Некачественное поверхностное решение проблемы (фикс бага).

Для примера необходимости метрики "Rejected/Opened Bugs" можно рассмотреть следующий случай:   
 Наблюдают, что процент отклоненных (Rejected) багов очень большой. Это может значить:

1. Требования к функции можно трактовать по разному;
2. Тестировщик не точно описал проблему;
3. Разработчик не желает исправлять допущенную им ошибку или не считает, что это на самом деле ошибка. (Эта проблема является прямым следствием 2-ой, возникшей из-за неточного описания).

Все эти проблемы заметно дестабилизируют обстановку на проекте. Поэтому, при их возникновении, рекомендуется провести короткую беседу с руководителями проектных групп, чтобы впоследствии уменьшить количество переоткрытых и отклоненных дефектов.

Таблица 2.6 – Метрики по задачам

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Deployment tasks | Метрика показывает количество и результаты установок приложения. Процедура установки приложения была описана в статье [Процедура проведения установки новой версии ПО (Deployment WorkFlow)](http://alexeybulat.blogspot.com/2007/12/deployment-workflow.html). В случае, если количество отклоненных командой тестирования версий будет критически высоким, рекомендуется срочно проанализировать и выявить причины, а также в кратчайшие сроки решить имеющуюся проблему. |
| Still Opened Tasks | Метрика показывает количество все еще открытых задач. К окончанию проекта все задачи должны быть закрыты. Под задачами понимаем следующие виды работ: написание документации (архитектура, требования, планы), имплементация новых модули или изменение существующих по запросам на изменения, работы по настройке стендов, различные исследования и многое другое |

Источник: собственная разработка автора

**Метрики автотестов**

Собираясь вводить метрики в автотесты, необходимо понимать, для чего это нужно.

Специфика автотестов:

1. Наличие негативных пользовательских сценариев;
2. Интеграционные проверки с тестовыми стендами сторонних сервисов;
3. Кроссбраузерность;
4. Конфигурирование CI и тестовых виртуалок;
5. Предоставление наших продуктовых АТ другим сервисным командам.

Необходимо обращать внимание не только на стабильность теста и время прогона, но и на:

1. Время красного прогона. Отображает как много времени уходит с момента остановки теста до выявления и отображения ошибки;
2. Эффективность теста по времени. Отображает как много времени уходит на зеленые и красные прогоны, и подсчета соотношение. Тесты с низкой эффективностью стоят первыми в очереди на рефакторинг;
3. Относительное потерянное время. Эта метрика показывает сколько минут теряется на каждый прогон этого теста из-за нестабильности.

**Метрики релизного цикла**

Еще одним местом применения метрик может быть релизный цикл.

Для этого можно подсчитать сколько времени задача находится на каждой стадии релизного цикла.

Из всего этого можно:

1. посчитать сколько тестировщиков должны трудиться над выпуском релизов, а сколько может переключиться на другие задачи, такие как рефакторинг, допиливание инфраструктуры, инструментов и т.д;
2. понять оптимальное соотношение ролей и, как следствие, прогнозировать количество вакансий;
3. определить точки приложения силы для такой внутри командной цели, как технический релиз за час.

Ниже перечислены метрики в зависимости от сущностей применимых в тестировании.

**Расчетные метрики**

**Итерация** – абстрактная единица, под которой понимается некоторый отрезок времени, в течение которого выполнялся один раунд тестирования. (Более точного определения пока не сформулировали.)

**Метрика: Доля повторно открытых дефектов.**

*Коэффициент ретеста = заново открытые / открытые в первый раз* (на одной итерации тестирования).

*Ограничение:* Не учитываются дефекты, критичность которых ниже чем незначительные.

*Обоснование:* Повторно открытые дефекты это дополнительно затраченное время на перепроверку исправления дефекта, а также потенциально опасное место.

**Метрика: Убойность тестов**

*Убойность тестов = число дефектов\*100 / число тестов, где*  *Убойность тестов* – это количество дефектов на 100 тестов. *Число дефектов* – все дефекты, которые были найдены по шагам из тестов на данной итерации тестирования. *Число тестов* – все тесты, которые были пройдены за итерацию.

*Ограничение*: Не учитываются дефекты критичность которых ниже чем незначительные.

*Обоснование*: Метрика косвенно отражает качество работы тестировщика, или работы программистов.

**Метрика: Число дефектов найденных вне теста.**

*Число дефектов найденных вне теста = число дефектов, найденных вне тестов\*100 / число тестов*

*Правило:* «Нельзя приписывать тесту найденный дефект, если шаги для воспроизведения этого дефекта отличаются больше чем на 30% от шагов, описанных в тесте».

*Обоснование:* Метрика напрямую отображает недостатки тестов. Идеальным сценарием считался бы случай, когда все баги были бы найдены при прохождении тестовых шагов.

**Метрика: Эффективность тестирования**

*Эффективность тестирования = число дефектов/трудозатраты (чел/ч), где Эффективность тестирования* – это число дефектов, найденных за один час одним человеком.

*Обоснование:* Метрика отражает эффективность использования ресурсов.

**Метрики: Статуса продукта**

*Статус продукта = % реализованных требований,*

**Метрики: Статуса тестирования**

*Статус тестового набора = Отношение количества удачно пройденных к завершившимся с ошибками (желательно с учётом приоритетов).*

*Статус тестирования = % покрытия функционала тестами и/или автотестами;*

*Средний срок жизни дефекта.* Полезен при анализе затраченного времени на исправления дефекта. Отражает как много времени уходит с момента нахождения бага до его закрытия;

Метрики статуса продукта и статуса тестирования являются конечными показателями. Все остальные вышеперечисленные являются косвенными и помогают влиять на результат, корректировать действия. Чтобы понять косвенные показатели, нужно знать конечные.

**Метрики ради метрик**

Типичным анти-паттерном являются «метрики ради метрик». Как уже было отмечено выше, метрики — это лишь инструмент, который помогает количественно оценить, насколько хорошо команда достигаете целей. Допустим, такая метрика как количество тестов не является показательной. Так как количество тестов не гарантирует их качество.

## **2.3 Назначение и структура метрик**

Для удобства и наглядности возможных вариантов расчета метрик, которые могут быть применены к данным этапа тестирования можно воспользоваться толкованием метрик из таблицы 2.7.

Таблица 2.7 – Толкование метрик в разрезе их назначения и способов расчета

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Метрика** | **Тип/ Назначение** | **Компоненты/шкала** | **Формула** | **Возможные значения** | **Толкование** |
| **Анализ требований** | | | | | |
| Качество требований | *Расчетная*/ Помогает оценить качество требований и отправить требования на доработку еще до начала работы над ПО | Полнота, Понятность, Осуществляемость, Проверяемость/ | *Вариант 1.*  R1 = (Полнота + Понятность) \* Осуществимость \* Проверяемость;  *Вариант 2.*  R2 = (Полнота + Понятность) \* Осуществимость + Проверяемость | Если все характеристики принимают значение в интервале [0, n], то метрика будет принимать значения:  R1 = [0; 2n3];  R2= [0; 4n2] | Чем ближе значение метрики к правой границе интервала ее значений, тем качественней требование. |
| Сложность требований | *Расчетная*/ Инструмент решения задачи эстимации сроков тестирования и учета возможных рисков. | Количество требований и количество связей с каждым требованием./ | Среднее количество связей каждого требования с остальными требованиями [11]. | Значения от 0 до 1.  1 будет означать, что каждое требование связано с каждым, а 0 о том что связей нет. | Степень взаимосвязанности требований. Степень связанности является приемлемой если не превышает 0,2-0,3. В противном случае доработка в рамках одного из требований будет вести к цепочке изменений, а значит и возможных ошибок, в значительной части продукта. |
| **Тест дизайн** | | | | | |
| Скорость разработки ТС (ТС – тест кейсы) | *Измеряемая/* Служит инструментом эстимации планирования тест-дизайна. ожет послужить в отчетный период. |  | Среднее время написание тест кейсов за единицу времени. | минуты, часы, дни, недели |  |
| Сложность ТС | *Расчетная*/ Инструмент оценки сложности тестового набора и решения задачи эстимации сроков тестирования. | Количество шагов необходимых для того, чтобы выполнить тест кейс/ | Оценочное соотношение самого простого теста ко всем существующим. | Оценки или веса. Например, к самому простому тест кейсу применятся оценка 1, далее относительно этого кейса оцениваются остальные. | Наиболее затратным тест кейсром будет считаться кейс с наибольшим количеством шагов. Есть смысл применять данную метрику с отсылкой на коэффициенты тестировщиков с разными уровням мастерства/знанием проекта (например, junior=4; midel=8 и т.д.). Так например, наиболее сложные задачи поручать наиболее опытным/знающим проект тестировщикам, что позволит сократить временные затраты. |
| Пригодность теста к smoke тестам | *Измеряемая/* тестовый набор, предназначенный для проверки основного функционала | /Номинальная | Количество тест кейсов, пригодных к smoke тестам | Yes (пригоден для smok тестирования)/ No (непригоден) | Можно считать минимальным тестовым набором |
| Покрытие требований тест кейсами | *Расчетная/* позволяет оценить и выявить слабые места в покрытии тестового набора. | Количество тестов, количество требований/ | Количество тестов приходящихся на 1 требование. Общее количество тестов разделенное на общее количество требований [11]. | [0. общее количество тестов] | Если требования атомарные, метрика покажет отклонение покрытия каждого требования от среднего уровня. Чем больше значение отличается от 1, тем меньше\больше тестов написано для одного требования, чем обычно. Если  коэффициент будет равен или близок к 0. Для них нужно рассмотреть возможность добавления тестов.  Если требования не атомарные, то данная метрика позволит убедиться только в том, что для каждого требования есть хотя бы 1 тест. Для этого коэффициент всегда должен быть больше 1. |
| **Функциональное тестирование** | | | | | |
| Плотность дефектов | *Расчетная*/Помогает обнаружить чать или функционал ПО, который является наиболее проблемным. Данная информация полезна при планировании работ с данным функционалом и при анализе рисков. | Количество дефектов в отдельном модуле, количество дефектов в продукте | Доля дефектов, приходящаяся на отдельный модуль в течение итерации или релиза. |  | Коэффициент больший 0,3 говорит о большом количестве найденных дефектов. |
| Коэффициент ретеста | *Расчетная*/ Помогает проанализировать затраченное командой время на доработку неисправного функционала. | Количество дефектов в функционале до и после исправления ошибок. | Доля повторно обнаруженных дефектов на общее количество ошибок, включая ранее исправленные и новые. |  | Чем ближе значение к 0, тем меньше было внесено ошибок в существующий функционал. Если значение больше 0,5 то можно сделать вывод о том, что более половины времени уходит на восстановление работавших ранее функций ПО. |
| Происхождение дефектов | *Измеряемая/*Помогает выявить на каком из этапов тестирования было найдено наибольшее количество багов. Следует также применять в градации критичности бага. | Баг репорт | К описанию бага следует добавлять этап на котором он был найден. | Этап тестировани документации, этам тест-дизайна, этам функционального тестирования, этап регресси, этап внедрения, этап сопровождения | Баги найденные на этапе внедрения и сопровождения следует считать наиболее экономически затратными. |
| Срок жизни дефекта | *Измеряемая/*Полезен при анализе затраченного времени на исправления дефекта. | Баг репорт | Отражает как много времени уходит с момента нахождения бага до его закрытия. | минуты, часы, дни, недели |  |
| Убойность тестов | *Расчетная*/Метрика косвенно отражает качество работы тестировщика, или работы программистов. | Количество дефектов, количество тестов | Количество дефектов умноженное на сто разделить на число тестов. |  | Убойность тестов – это количество дефектов на 100 тестов.  Не учитываются дефекты критичность которых ниже чем незначительные. |
| Эффективность тестирования | *Расчетная*/Метрика отражает эффективность использования ресурсов | Количество дефектов, количество тестов | Количество дефектов разделенное на трудозатраты (чел/ч) | Количество дефектов, найденных за один час одним человеком. |  |
| Число дефектов найденных вне теста | *Расчетная*/Метрика помогает отражать недостатки разработанных тестов. | Количество дефектов, количество тестов | Количество дефектов, найденных вне тестов умноженное на сто разделить на число тестов |  | Нельзя приписывать тесту найденный дефект, если шаги для воспроизведения этого дефекта отличаются больше чем на 30% от шагов, описанных в тесте. Идеальным сценарием считался бы случай, когда все баги были бы найдены при прохождении тестовых шагов. |
| **Статус продукта** | | | | | |
| Статус продукта | *Измеряемая/*Метрика помогает определить количество оставшихся работ по разработке ПО в рамках итераций. | Количество требований, количество решенных задач | Отношение реализованных требований к запланированным | Процент выполненных задач | Продукт может считаться готовым к приемке заказчиком, когда основной функционал был реализован на 100%. А не не основной функционал (например дизайн), на 80-90%. |
| **Автотестирование** | | | | | |
| Скорость разработки автотестов | *Измеряемая/*Время затраченное на написание и разработка автотестов. Служит инструментом эстимации и планирования автоматизации тестирования. | Автотест | Среднее время написание автотестов за единицу времени. | минуты, часы, дни, недели |  |
| Длительность выполнения автотестов | *Измеряемая/*Помогает выявить автотесты с низкой проходимостью и отправить их на доработку. |  | Отображает как много времени уходит на зеленые и красные прогоны, и подсчета соотношение. | секунды, минуты |  |
| Дефекты в автотестах/ нестабильность автотеста. | *Расчетная*/Эта метрика показывает сколько времени затрачивается на каждый прогон тест из-за нестабильности. Полезен в принятии решения об исключении или доработке автотеста. |  | Подсчет среднего времени на прогон автотеста до зеленого прохода. Например автотест был запущен 10 раз из которых последний оказался зеленым. Следует подсчитать время на прогон всех десяти тестов. | секунды, минуты |  |
| Статус тестового набора | *Измеряемая/*Метрика показывает результаты прохождения тест кейсов  Применяется для анализа и выявления причин, по которым тесты не были проведены. | Количество удачно/не удачно пройденных тестов, количество не запущеных тестов | Отношение количества удачно пройденных к завершившимся с ошибками. | Passed/Failed | Чем ближе значение удачно пройденных тестов к общему количеству тестов, тем ближе этап тестирования к окончанию. Чем выше отношение удачно пройденных тестов к завершившимся с ошибками, тем лучше качество разрабатываемого функционала. |
| Количество тест кейсов, которые еще необходимо выполнить в данной фазе тестирования. | Not Run |
| **Внедрение и сопровождение** | | | | | |
| Количество пропущенных дефектов в PROD | *Измеряемая/*Данная метрика оценивает качество выпущенного в релиз функционала. | Количество багов в пользовательской версии продукта. | Колличество багов, которые нашел пользователь либо заказчик. |  | Баги найденные в продакшене обычно считаются наиболее дорогими в исправлении. |
| Удовлетворенность пользователей продуктом | *Измеряемая/* Полезна в оценке правильности разграничения командой важности функционала и вариантов решений. | Рейтинг продукта/качество продукта. | Опрос пользователей, рейтинг на площадках продаж и т.д. | Оценка от 1 до 5 звезд в App Store, количество лайков в соц сетях и т.д. | Метрика определяет удовлетворенность пользователя продуктом, соответствует его ожиданиям. |

Источник: собственная разработка автора на основе [11]

## **2.4** **Пример вычисления значений расчетных метрик для проекта “Реализации БД “Опыт реализованных проектов”**

В данном разделе продемонстрируем расчет метрик для условного проекта “Реализация БД “Опыт реализованных проектов”.

## **2.4.1 Описание проекта**

**Этап начальной разработки БД:**

1. анализ деятельности компании-заказчик информационной системы;
2. анализ структуры компании;
3. спецификация требований;
4. определение целей;
5. сферы применения;
6. границы возможностей.

**Концептуальное проектирование базы данных**

1. Анализ требований к базе данных, выявление представлений конечных пользователей и требований к обработке транзакций;
2. Определение сущностей, атрибутов и связей;
3. Разработка ER-диаграмм (от ER - Entity-Relationship - Сущность-Связь);
4. Нормализация;
5. Проверка модели данных, выявление основных процессов (правила ввода, обновления и удаления данных);
6. Проверка отчетов, запросов, представлений, целостности, совместного использования и безопасности.

**Анализ предметной области.**

Компания-заказчик информационной системы, использующей базу данных – ИТ-компания, осуществляет аутсорсинговые услуги по разработке и тестированию программного обеспечения, сервисной поддержки программных продуктов, модернизации и интеграции бизнес-приложений.

Компания владеет несколькими офисами, которые подключены в единую информационную сеть. Компания ведет различные проекты по разработке программных решений для разных заказчиков.

В любом из офисов сети каждый член компании должен иметь возможность доступа к БД. Пользователями системы являются член компании или команды, который должен иметь возможность внесения информации о текущем проекте в любое время. Также он должен иметь возможность получения информации о уже внесенных проектах. Подключение к БД должно осуществляться с помощью логина и пароля, доступ к БД должен быть только у сотрудников компании.

К разработанной БД должна быть возможность подключения систем, направленных на управление процессам тестирования

Требуемыми функциями системы, использующей базу данных, являются:

1. регистрация пользователя в системе;
2. выбор характеристик разрабатываемого программного продукта;
3. постройка рекомендаций по стратегии тестирования, основанного на уже реализованных проектов (данные из БД);
4. статистический и сравнительный анализ данных о реализованных проектах.

**Требования к БД.**

БД “Опыт реализованных проектов” должна обладать минимальным набором сущностей, атрибутов и связей, описанных в пункте 2.3. Также в базе данных создаваемой информационной системы все таблицы должны находиться в третьей нормальной форме.

БД должна позволять работать со следующими запросами и отчетами**:**

1. *Количество разработанных программных продуктов* в зависимости от вида приложения, устройства и его операционной системы;
2. *Количество модулей* по видам разработанных приложений, устройств и операционной систем;
3. *Сложность модулей* по видам разработанных приложений, устройств и операционной систем;
4. Затраченное командой *время на разработку модулей* по видам разработанных приложений, устройств и операционной систем, в зависимости от опыта и количества членов команды (программистов и менеджеров);
5. Затраченное командой *время на тестирование модулей* по видам разработанных приложений, устройств и операционной систем, в зависимости от опыта и количества членов команды (тестировщиков и менеджеров);
6. *Виды тестовой документации*, которые были или не были применены к покрытию модулей по видам разработанных приложений, устройств и операционной систем;
7. *Индикатор покрытия* тестовой документацией модулей по видам разработанных приложений, устройств и операционной систем (метрика Покрытие требований тест кейсами из таблицы 2.7);
8. *Индикатор покрытия* автотестами модулей по видам разработанных приложений, устройств и операционной систем (метрика Покрытие требований тест кейсами из таблицы 2.7);
9. *Виды* багов, которые были найдены при тестировании модулей по видам разработанных приложений, устройств и операционной систем;
10. *Количество* багов, которые были найдены при тестировании модулей по видам разработанных приложений, устройств и операционной систем;
11. *Этап* тестирования на котором был найден баг в модуле по видам разработанных приложений, устройств и операционной систем.

## **2.4.2 Выявление, расчет и применение метрик для проекта разрабатываемой БД**

Пример расчета метрики оценки качества требований представлен в таблице 2.8. Допустим при анализе запросов и отчетов БД было определено 11 требований. К ним были применены оценки качества из раздела 2.2. А именно, в качестве весов была выбрана линейная система от 0 до 3, где 0 это полное несоответствие критерия качеству, а 3 удовлетворение критерию качества:

Таблица 2.8 – Расчет метрики оценки качества требований к запросам и отчетам к БД “Опыт реализованных проектов”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Требования**  **Пользователь должен иметь возможность внести/получить информацию о** | **Полнота** | **Понятность** | **Осуществимость** | **Проверяемость** | **R1** | **R2** |
| 1 | Количестве разработанных программных продуктов в зависимости от вида приложения, устройства и его операционной системы | 3 | 1 | 3 | 2 | **24** | **14** |
| 2 | Количестве модулей по видам разработанных приложений, устройств и операционной систем | 3 | 2 | 3 | 1 | **15** | **16** |
| 3 | Сложности модулей по видам разработанных приложений, устройств и операционной систем | 3 | 2 | 3 | 2 | **30** | **17** |
| 4 | Затраченном командой времени на разработку модулей по видам разработанных приложений, устройств и операционной систем, в зависимости от опыта и количества членов команды (программистов и менеджеров) | 3 | 2 | 3 | 2 | **30** | **17** |
| 5 | Затраченном командой времени на тестирование модулей по видам разработанных приложений, устройств и операционной систем, в зависимости от опыта и количества членов команды (тестировщиков и менеджеров) | 3 | 2 | 3 | 2 | **30** | **17** |
| 6 | Видах тестовой документации, которые были или не были применены к покрытию модулей по видам разработанных приложений, устройств и операционной систем | 3 | 0 | 3 | 2 | **18** | **11** |
| 7 | Покрытии тестовой документацией модулей по видам разработанных приложений, устройств и операционной систем | 3 | 2 | 3 | 3 | **45** | **18** |
| 8 | Покрытии автотестами модулей по видам разработанных приложений, устройств и операционной систем | 0 | 2 | 3 | 1 | **6** | **7** |
| 9 | Видах багов, которые были найдены при тестировании модулей по видам разработанных приложений, устройств и операционной систем | 3 | 2 | 3 | 2 | **30** | **17** |
| 10 | Количестве багов, которые были найдены при тестировании модулей по видам разработанных приложений, устройств и операционной систем | 3 | 0 | 3 | 1 | **9** | **10** |
| 11 | Этапе тестирования на котором был найден баг в модуле по видам разработанных приложений, устройств и операционной систем | 3 | 2 | 3 | 3 | **45** | **18** |
| НАБОР ТРЕБОВАНИЙ**:** | | **2,73** | **1,55** | **3,00** | **1,91** | **25,64** | **14,73** |

Источник: собственная разработка

Исходя из выбранной линейной системы от 0 до 3, можно рассчитать, что:

1. Значение R1 в диапазоне от 30 до 54 говорит о том, что требования разработаны качественно; а значение R1 в диапазоне от 2 до 35 говорит о том, что требования разработаны некачественно;
2. Значение R2 в диапазоне от 17 до 21 говорит о том, что требования разработаны качественно; а значение R2 в диапазоне от 3 до 17 говорит о том, что требования разработаны некачественно.

Анализируя оценочную таблицу требований можно сделать вывод о том, что требования с порядковым номером 3, 4, 5, 7, 9, 11 разработаны качественно и подходят для дальнейшей работы с ними. Остальные же, а именно 1, 2, 6, 8, 10, требуют доработки. Доработка может включать в себя уточнение у заказчика информационных упущений и тому подобное. Что касается анализа набора требований, то из-за половины недоработанных отдельных требований весь набор следует считать некачественно разработанным.

***Пример расчета метрики планирования тест-дизайна.***

Прежде всего выявим показатели, необходимые для расчета метрик, описанных в разделе 2.2.1.

*Минимальное количество тестов / пригодность теста к smoke тестам.* Исходя из требований, представленных в таблице 2.8, можно подсчитать минимальный набор необходимых тестов. Всего представлено 11 технических требований. Для расчета минимального набора тестов учтем, что необходимо проверить вывод и ввод информации в БД. Поэтому минимальное количество тестов будет равно 11 умножить на 2, то есть 22.

*Максимальное количество тестов.* Минимальный набор требований можно оформить в виде чек-листа. По опыту, в среднем, на одно требование приходится 3-5 тест-кейса. Итого, если требования запроса/ответа покрывать чек-листами то их количество будет составлять 22 теста, а если тест-кейсами то 22\*5=110 тестов. В идеале, тест-кейсы должны покрывать все возможные варианты срабатывания системы. Но на практике применение этого метода не представляется возможным из-за огромного количества входных значений. Поэтому говоря о наборе тест-кейсов, подразумевают всевозможные тесты, которые был способен предусмотреть тестировщик. Необходимо иметь в виду и включать в расчеты человеческий фактор, как было указано в пункте 2.2.1 в описании метрики “Покрытие требований тест кейсами”. Поэтому к 110 тест-кейсам нужно прибавить те, которые были упущены тестировщиком. Допустим, что 30% от всех возможных проверок было упущено, тогда 157 - это число всех возможных проверок.

Исходя из этого можно рассчитать ***метрику покрытия требований тестовыми случаями,*** применяя формулу:

*Тпок = (Lпок/Lвсего)\*100%, где Тпок* – тестовое покрытие, *Lпок* –количество требований проверяемых ТК, *Lвсего* – общее количество требований.

Исходя из вышеизложенного для нашего примера при использовании чек-листа имеем:

*Lпок=22, Lвсего=157,* откуда получаем

***Тпок*** *= (22/157)\*100%=14%.*

При использовании тест-кейсом имеем:

*Lпок=110, Lвсего=157,* откуда получаем

***Тпок*** *= (110/157)\*100%=70%.*

Для решения задачи прогнозирования рассчитаем ***метрику трудозатрат*** на написание и прохождения тестовых случаев. Для этого необходимо прибегнуть к средним показателям. Так, например, *производительность*:

1. Тест-проектирования (написание тест-кейсов) занимает 30 тестов/день;
2. Выполнения тестов занимает 50 тестов/день [15].

Произведя расчет можно предполагать, что тест-проектирование займет:

1. 22 / 30 = 0,73 *чел-дней*, если использовать чек-листы;
2. 110 / 30 = 3,6 *чел-дней,* если использовать тест-кейсы.

Подобным образом можно произвести расчет затраченного времени на проведения тестов в два раунда (первый проход тестов и затем регрессионный):

1. 22 / 50 \* 2 = 0,88 *чел-дней*, если использовать чек-листы;
2. 110 / 50 \* 2 = 4,4 *чел-дней,* если использовать тест-кейсы.

Из полученных данных можно также произвести расчет ***метрики числа предполагаемо найденных багов***, исходя из вида тестовой документации, по которой будут производиться тесты.

Исходя из экспертных оценок на один тест приходится в среднем 0,775 бага [15]. А это значит что:

1. 22 \* 0,775 = 17,05 бага будет найдено если использовать чек-лист;
2. 110 \* 0,775 = 88,25 бага будет найдено если использовать тест-кейсы

## **Выводы по главе 2**

Качество программного обеспечения – это способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям. Главной задачей процесса тестирования является подтверждение того, что продукт является качественным. Стандарт ISO/IEC 25010:2011 определяет модель качества продукта, которая включает восемь характеристик верхнего уровня: функциональная пригодность; уровень производительности; совместимость; удобство использования (юзабилити); надежность; защищённость; сопровождаемость; переносимость (мобильность). С точки зрения тестировщика качество – показатель степени соответствия продукта предъявляемым к нему требованиям.

Введение и использование метрик необходимо для улучшения контроля этапа тестирования, что позволяет повлиять на принятие своевременных управленческих решений. Метрика – это количественный масштаб и метод, который может использоваться для измерения. Метрики могут стать хорошим инструментом визуализации критериев качества.

Цель контроля тестирования состоит в получении обратной связи и визуализации процесса тестирования. Необходимую для контроля информацию собирают (как вручную, так и автоматически) и используют для оценки состояния и принятия решений, таких как покрытие требований или кода тестами, или критерии окончания тестирования. Метрики также могут быть использованы для оценки прогресса выполнения запланированных работ и освоения бюджета.

Метрики, применяемые на этапе тестирования ПО, можно классифицировать по следующим признакам: по этапам тестирования; по сущностям. По способу получения значений метрики можно разделить на расчетные метрики, которые вычисляются по определенным правилам из некоторых первичных показателей, измеренных в заданных шкалах; на измеряемые, которые сами являются первичной информацией.

# **ГЛАВА 3 РАЗРАБОТКА ПРОЕКТА ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ УПРАВЛЕНИЯ ИТ-ПРОЕКТАМИ НА ЭТАПЕ ТЕСТИРОВАНИЯ**

## **3.1 Метрики в управлении этапом тестирования**

В соответствии с целью настоящей работы определим классы метрик в соответствии с их применением на различных этапах управления тестированием ПО.

Расчетные метрики, применяемые на этапе анализа документации и тест-дизайна (таблица 2.1), необходимы для этапа планирования стратегии тестирования. Их можно отнести к **классу метрик планирования.**

Метрики функционального тестирования, автотестирования, внедрения и сопровождения из таблицы 2.1 невозможно применить по отношению к требованиям. Такие метрики можно отнести к **классу результативности.**

К любым метрикам стоит подходить опираясь на задачи проекта разработки ПО. Каждый проект по разработке ПО уникален. Это делает один вид метрик необходимым для одного проекта, и совершенно неподходящим для другого. Зачастую тестировщикам приходится разрабатывать несколько планов дальнейших действий по тестирования и обосновывать заказчику риски применения каждого сценария. С этой точки зрения метрики служат инструментом планирования. Метрики *класса планирования* способны предложить вариант быстрой разработки и обосновать низкое качество ПО, либо объяснить заказчику причину долгой разработки.

Так, например, анализируя результаты расчетных метрик и использование их в планировании применения различного вида тестовой документации, можно сделать вывод о том, что использование чек-листов в качестве тестовой документации, позволяет значительно сократить время на их написание и прохождения. Но при этом процент найденных багов будет сильно отличаться от случая, когда использовались тест-кейсы.

Применение различного вида тестовой документации зависит от типа проекта, от времени, отведенного для реализации проекта. Обычно тест кейсами покрывают модули со сложным функционалом или высокой связанностью с другими модулями. Это делается для того, чтоб минимизировать упущенные баги. Остальные модули, более легкие в реализации или менее важные с точки зрения функционала, покрывают чек-листами или не применяют какую-либо тестовую документацию вообще.

Также метрики класса планирования способны решать задачи эстимации проведения тестирования.

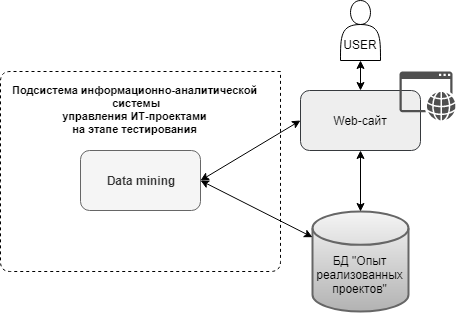
Метрики *класса результативности* рассчитываются только в процессе разработки и тестирования программного продукта и после его реализации. Они отражают в динамике процесс тестирования программного продукта позволяют подготавливать отчетность перед заказчиком. Также на основе данных метрик можно подвергать анализу достигнутые результаты, допущенные ошибки или, наоборот, успехи.

Значения данных метрик могут быть внесены в БД опыта использования метрик только в процессе реализации реального проекта. В отличие от метрик планирования, которые могут быть применены в случае отсутствия в БД информации о завершенных проектах, результативные метрики смогут быть полезны для планирования только после накопления результатов нескольких реально реализованных и выпущенных программных продуктов.

## **3.2 Проект подсистемы информационно-аналитической системы управления ИТ-проектами на этапе тестирования** [**для сбора, анализа выработки и мониторинга принятых решений и показателей качества**](#_4jpj0b3)

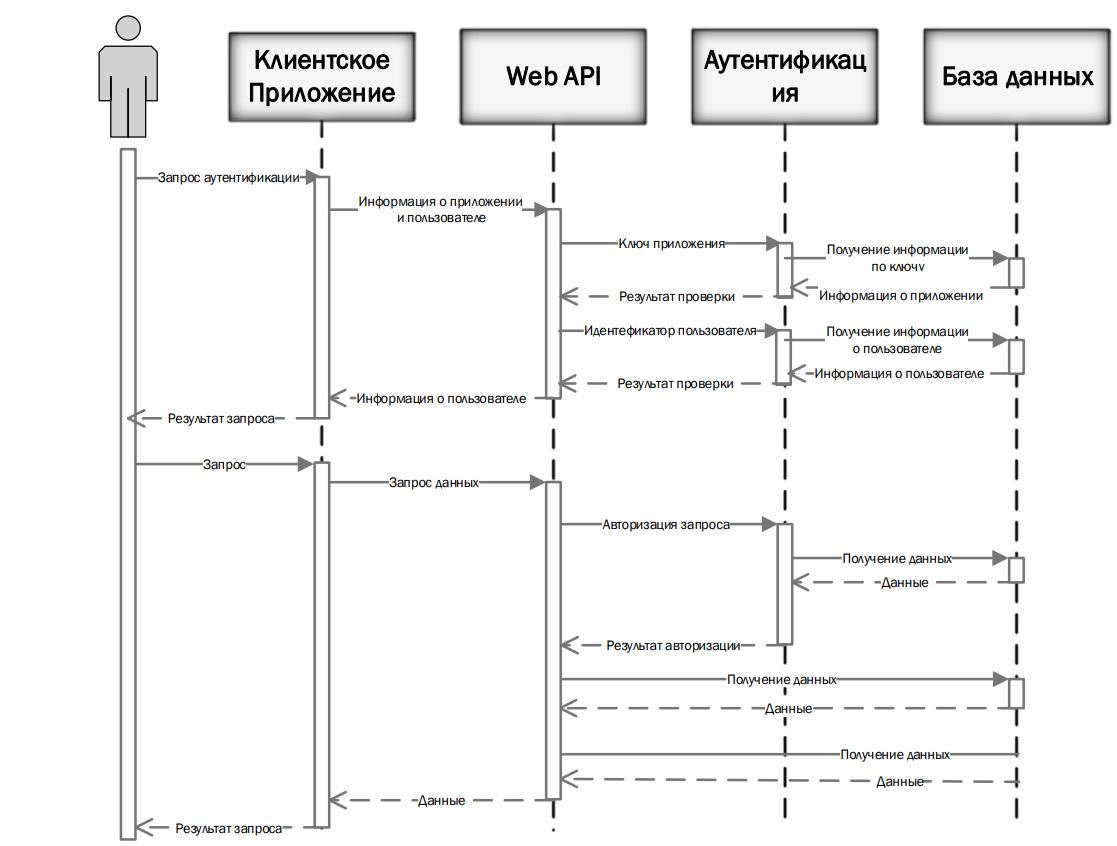
### 3.2.1 Архитектура подсистемы

Подсистема состоит из трех основных компонентов: веб-приложение, база данных “опыт реализованных проектов”, аналитический модуль обработки данных. Пользователь сможет работать напрямую только с веб-приложением, которое будет взаимодействовать с базой данных и аналитическим модулем незаметно для пользователя. На рисунке 3.1 представлена схема взаимодействия основных компонентов подсистемы.



**Рисунок 3.1 – Схема действий по управлению рисками**

**Источник: собственная разработка автора**

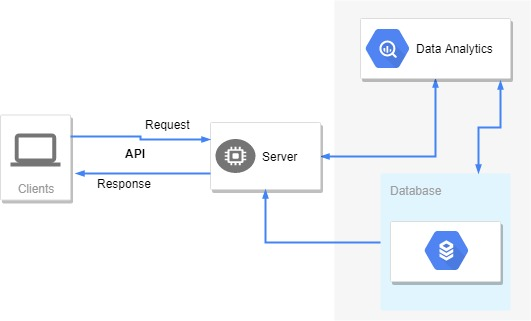
Диаграмма последовательности передачи пользовательских данных представлена на рисунке 3.2. 

**Рисунок 3.2 – Диаграмма последовательности получения данных с сервера БД**

**Источник: собственная разработка автора**

Архитектура клиент-серверного взаимодействия для данного приложения выглядит следующим образом: клиент посылает HTTP запросы на сервер, сервер обрабатывает их и возвращает результат в текстовом JSON-формате. «Общение» происходит согласно заранее составленному API.

Задача клиентской-части состоит во взаимодействии с пользователем, передаче пользовательского запроса серверу, получение запроса от серверной части и представление его в удобном для пользователя виде. Программа-сервер обрабатывает запросы клиента и выдает ответы. Схема организации клиент-серверного взаимодействия представлена на рисунке 3.3



**Рисунок 3.3 – Схема организации клиент-серверного взаимодействия**

**Источник: собственная разработка автора**

### 3.2.2 Проект базы данных “Опыт тестирования реализованных проектов”

В литературе предлагается множество вариаций определения понятия «база данных», отражающих скорее общее понимание тех или иных авторов, однако общепризнанная единая формулировка отсутствует.

Определения из международных стандартов и национальных стандартов, разработанных на основе международных:

**База данных**,сокр. **БД** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных [13].

**База данных** — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживает одну или более областей применения [14].

**Система управления базами данных**, сокр. **СУБД** (*англ. Database Management System, сокр. DBMS*) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [15].

**СУБД** — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а так­же предоставляет средства для администрирования БД [16].

Основная цель проектирования БД – это обеспечение хранение в ней всей необходимой информации, а также сокращение избыточности данных. Проектирование БД позволяет уменьшить затраты на многократные операции обновления избыточных копий, а также разумно использовать вычислительные ресурсы и память.

Проектирование устраняет возможность возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте.

Проектирование должно минимизировать дублирование данных, наладить удобство их обновления и обработки. Для удовлетворения этих требований необходимо определить, из каких отношений должна состоять БД, какие атрибуты должны входить в эти отношения [17].

База данных “Опыт реализованных проектов” позволит хранить, накапливать, визуализировать опыт этапа тестирования реализованных проектов. В свою очередь, спроектированная БД может послужить инструментом для получения информации, на основе которой можно более эффективно решать задачи планирования как этапа тестирования, так и самих будущих проектов.

**Сущность** – это некоторый объект реального мира, которая может существовать независимо. На ER-диаграмме сущность изображается прямоугольником, в котором указывается ее имя. Сущность имеет экземпляры, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию. **Экземпляр сущности** – конкретный объект, характеризующийся набором значений атрибутов сущности. Предлагается выделить следующие сущности:

**1)** **Пользователь (User)**

1. Идентификатор пользователя;

2. Имя;

3. Фамилия;

4. Дата Рождения;

5. Пароль;

6. Логин;

7. Время добавления;

8. Время удаления;

9. Идентификатор должности;

10. Идентификатор опыта.

**2)** **Должность**

1. Идентификатор должности;

2. Должность.

**3)** **Опыт**

1. Идентификатор опыта;

2. Опыт.

**4)** **Проект (Project)**

1. Идентификатор проекта;

2. Идентификатор пользователя;

3. Название;

4. Описание;

5. Дата начала;

6. Дата окончания;

7. Время создания;

8. Время удаления;

9. Идентификатор описания разрабатываемого продукта.

**5)** **Роль пользователя (UserRole)**

1. Идентификатор роли;

2. Идентификатор пользователя;

3. Название;

4. Описание;

5. Время создания;

6. Время обновления.

**6)** **Описание разрабатываемого продукта (SoftwareDevelopmentScope)**

1. Идентификатор Описания разрабатываемого продукта;

2. Идентификатор вида устройства;

3. Идентификатор операционной системы;

4. Идентификатор типа приложения;

5. Идентификатор сферы деятельности.

**7)** **Тип приложения**

1. Идентификатор устройства;

2. Описание типа приложения.

**8)** **Вид устройства**

1. Идентификатор вида устройства;

2. Описание вида устройства.

**9)** **Операционная система**

1. Идентификатор операционной системы;

2. Название ОС.

**10) Сферы деятельности**

1. Идентификатор сферы деятельности;

2. Название сферы деятельности.

**11)** **Команда**

1. Идентификатор команды;

2. Идентификатор пользователя;

3. Идентификатор требования;

4. Название команды;

5. Количество участников.

**12)** **Требования**

1. Идентификатор требования;

2. Идентификатор проекта;

3. Название;

4. Описание;

5. Дата начала разработки;

6. Дата окончания разработки;

7. Дата начала тестирования;

8. Дата окончания тестирования;

9. Время создания требования;

10. Время обновления требования.

**13)**  **Дефекты**

1. Идентификатор дефекта;

2. Идентификатор требования;

3. Название;

4. Описание;

5. Серьезность бага;

6. Приоритет бага

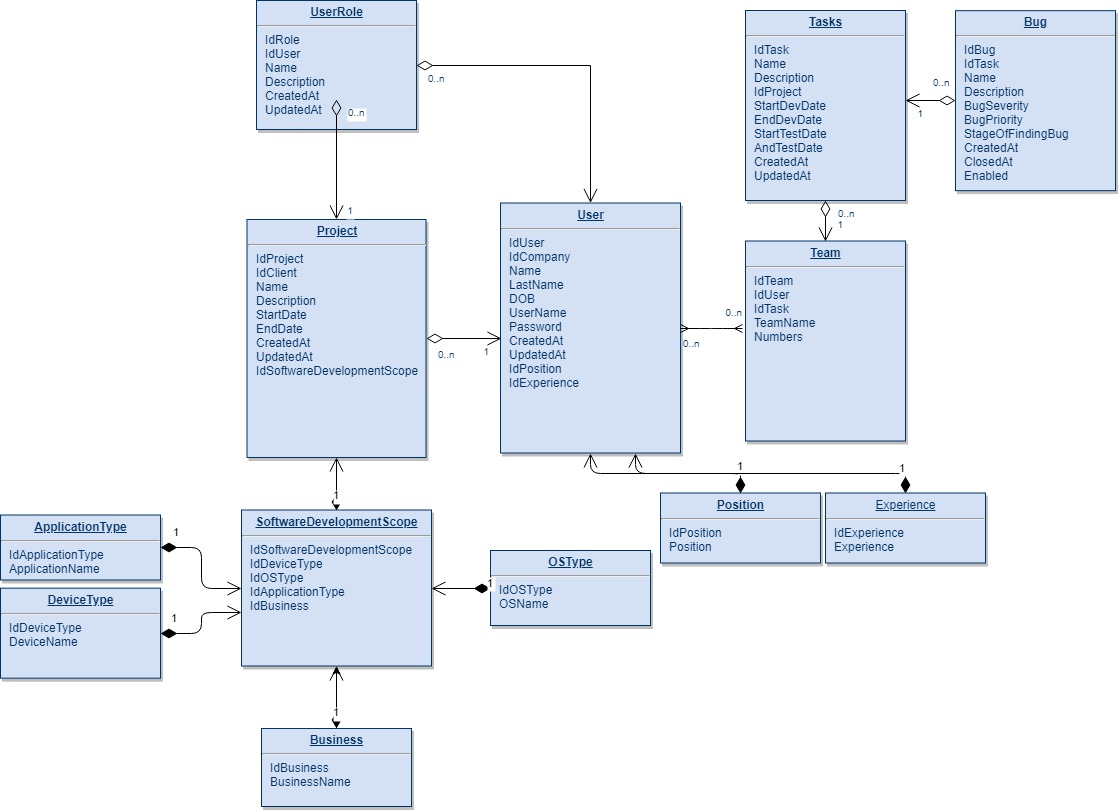
7. Этап тестирования в котором был найден баг

8. Время создания

9. Время закрытия

10. Количество шагов

Схема концептуальной модельи базы данных представлена на рисунке 3.4.



**Рисунок 3.4** **– Концептуальная модель базы данных**

**Источник: собственная разработка автора**

Предложения по заполнению таблиц представлены ниже:

1. **Тип устройства** (*Device type*):
   1. Мобильное;
   2. Десктопное;
2. **Тип операционной системы** (OS type):
   1. Microsoft Windows;
   2. Apple Mac Os X;
   3. Linux;
   4. iOS;
   5. Google Android;
   6. Windows Phone;
3. **Тип приложения** (Application type):
   1. Веб-приложение
   2. Нативное (Native)
   3. Гибридное приложение
4. **Бизнес** (Business)*:*
   1. финансовая деятельность;
   2. образование;
   3. охрана здоровья;
   4. предоставление услуг;
   5. торговля;
   6. деятельность отелей и ресторанов;
   7. производство машин и оборудования;
   8. производство и перераспределение электроэнергии;
   9. строительство;
   10. научная деятельность;
   11. игры;
5. **Должность разработки** (Team):
   1. разработчик-программист;
   2. тестировщик;
   3. бизнес-аналитик;
   4. проектный менеджер;
   5. архитектор;
   6. владелец проектом;
6. **Количество членов команды** (Number of team members). Обычно количество команды составляет от 5 до 9 человек.
7. **Опыт членов команды** (Position of team members):
   1. Junior;
   2. Middle;
   3. Senior;
8. **Модуль** (Module). Единица или группировка бизнес-функционала, которую необходимо разработать, протестировать и выпустить в производство. Для удобства, обычно все задачи продукта разбиваются на группы, а затем каждая группа на отдельные модули. Обычно модели имеют следующую структуру: «Будучи пользователем <*тип пользователя*> я хочу сделать <*действие*>, чтобы получить <*результат*>». Такая структура удобна тем, что понятна как разработчикам, так и заказчикам.
9. **Сложность модуля** (Module complexity). Для оценки сложности можно использовать различные метрики. Так, например, зачастую прибегают к методу стори поинтов (Story Points). Оценивая сложность модуля, ему выставляется оценка с использованием цифр *ряда Фибоначчи (1,2,3,5,8,13,21,34,55)*. Также в БД можно использовать любые другие метрики оценки сложности, которые будут удобны для проекта и команды. Например *линейную шкалу (1,2,3,4 … n)* или *степень двойки (1,2,4,8 … 2n*); *размеры одежды (XS, S, M, L, XL)* и т.д.
10. **Сроки разработки модуля** (Module development). Предполагается что в данную таблицу будут вноситься временные сроки затраченные на разработку отдельного модуля. Временные сроки можно указывать в минутах, часах, неделях или месяцев.
11. **Сроки тестирования модуля** (Module testing). Предполагается что в данную таблицу будут вноситься временные сроки затраченные на разработку отдельного модуля. Временные сроки можно указывать в минутах, часах, неделях или месяцев.
12. **Тип тестовой документации** (Test documentation type)
    1. *Тест план;*
    2. *Чек-лист* – это документ, описывающий что должно быть протестировано. При этом чек-лист обычно является низкого уровня детализации. В нем указывают саму идею теста. Например, если необходимо проверить поле ввода, то чек лист будет написан в виде “Проверить поле ввода <*где*>”, в документе не будут указаны варианты ввода (язык, буквы или цифры и т.д.);
    3. *Тест сценарии / Тест кейсы.* Документ, описывающий манипуляции с приложением и ожидаемый результат. Например, если необходимо проверить поле ввода, то тест-кейс будет написан в виде “Проверить поле ввода <*где*>, <*при каких условиях*>, <*варианты данных для ввода*>”. Тест-кейс отличается своей детализированностью.
    4. *Баг репорт.* Документ, описывающий место нахождения бугу, условия его нахождения, шаги которые привели к его обнаружению, его важность и приоритетность;
13. **Процент покрытия модуля тестовой документацией** (Module coverage by test documentation). Предполагается что в данную таблицу будет вноситься количественная мера, а именно процентное соотношение всех возможных требований к модулю и написанных тестовых сценариев к ним.
14. **Процент покрытия модуля автотестами** (Module coverage by autotests). Предполагается что в данную таблицу будет процентное соотношение всех возможных требований к модулю и написанных автотестов к ним.
15. **Количество багов найденных в модуле** (Bugs number). Количественная мера найденных багов в модуле. Рекомендуется использовать значения от 0 до 500.
16. **Серьезность бага** (Bug Severity). Это атрибут, характеризующий влияние дефекта на работоспособность приложения.
    1. *S1 Блокирующая (Blocker);*
    2. *S2 Критическая (Critical);*
    3. *S3 Значительная (Major);*
    4. *S4 Незначительная (Minor);*
17. **Приоритет бага** (Bug Severity). Это атрибут, указывающий на очередность выполнения задачи или устранения дефекта. Можно сказать, что это инструмент менеджера по планированию работ. Чем выше приоритет, тем быстрее нужно исправить баг.
    1. *P1 Высокий (High);*
    2. *P2 Средний (Medium);*
    3. *P3 Низкий (Low)* [18];
18. **Этап нахождения бага** (Stage of finding the bug). В данную таблицу необходимо вносить этап в котором баг был обнаружен и задокументирован. Этап нахождения бага может указать на стоимость его исправления. Так, например, исправление бага, который был найден на этапе анализа требований потребует наименьшего времени а значит и стоимости, в сравнении с тем багом, который был найдет пользователем или заказчиком.
    1. *анализ документации;*
    2. *смоук тестирование;*
    3. *функциональное тестирование;*
    4. *регрессионное тестирование;*
    5. *Е2Е тестирование;*
    6. *заказчиком;*
    7. *после релиза.*

Инструментом реализации данной базы данных может послужить Microsoft SQL Server.

**Microsoft SQL Server** — система управления реляционными базами данных, разработанная корпорацией Microsoft. Важнейшие характеристики данной СУБД - это:

1. простота администрирования;
2. возможность подключения к Web;
3. быстродействие и функциональные возможности механизма сервера СУБД;
4. наличие средств удаленного доступа.

В комплект средств административного управления данной СУБД входит целый набор специальных мастеров и средств автоматической настройки параметров конфигурации. Также данная БД оснащена замечательными средствами тиражирования, позволяющими синхронизировать данные ПК с информацией БД и наоборот. Входящий в комплект поставки сервер OLAP дает возможность сохранять и анализировать все имеющиеся у пользователя данные. Данная СУБД представляет собой современную полнофункциональную базу данных, которая идеально подходит для малых и средних организаций. Стоит также заметить, что общемировой тенденцией в XXI веке стал практически повсеместный переход на платформу LINUX, а SQL Server функционирует только в среде Windows. Поэтому использование SQL Server целесообразно, только если для доступа к содержимому БД используется исключительно стандарт ODBC, в противном случае лучше использовать другие СУБД.

В качестве вывода можно сделать заключение о том, что проект БД “Опыт реализованных проектов” позволит хранить, накапливать, визуализировать данные о этапе тестирования реализованных проектов. Предполагается что разработанная БД будет внедрена в компанию, которая разрабатывает программные продукты. Будет заполняться и пополняться членами команды на различных проектах и разных этапах разработки программного продукта и его тестирования. Описанные сущности БД являются основой и могут быть изменены и расширены на усмотрение компании, в которую данная БД будет внедрена. Целесообразность использования данной БД можно объяснить полезностью в решении задачи планирования, как этапа тестирования, так и самих будущих проектов.

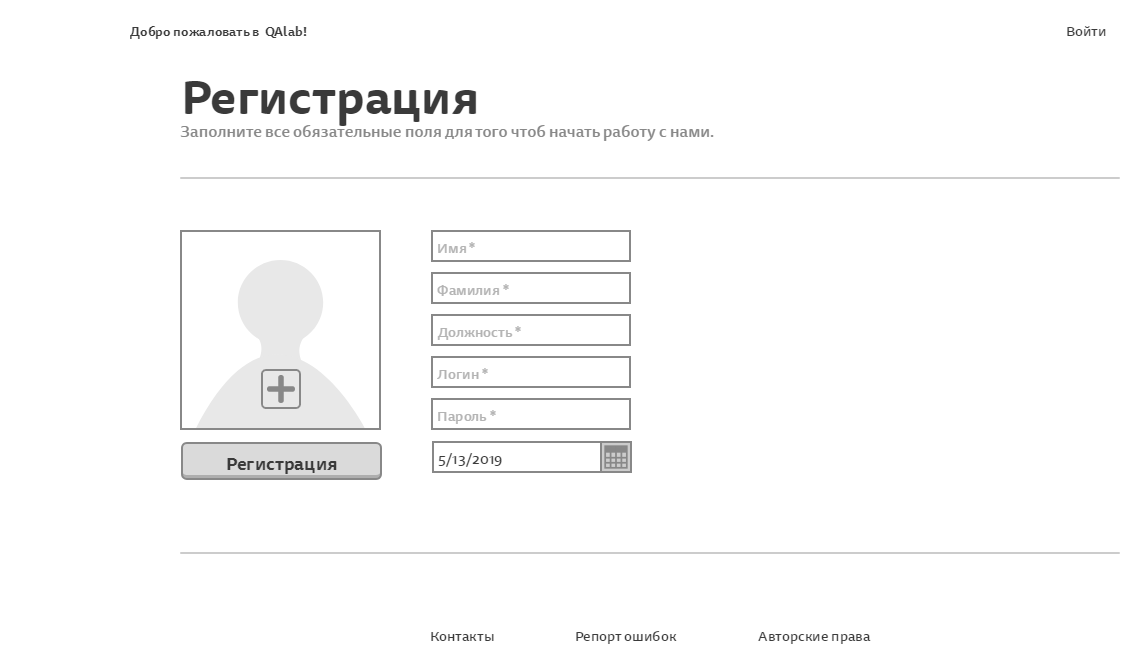
### 3.2.3 Проект веб-приложения

Подсистема информационно-аналитической системы управления ИТ-проектами на этапе тестирования для сбора, анализа выработки и мониторинга принятых решений и показателей качества будет выполнена в качестве Веб-приложения. Данное приложение может быть быть внедрено в ИТ-компанию, предоставляющая услуги разработки, сопровождения и тестирования программных продуктов в разных отраслях.

Веб-приложение имеет доступ к БД “Опыт реализованных проектов”, на основе информации из которой можно разрабатывать различные варианты стратегий этапа тестирования.

**Проектирование интерфейса приложения.**

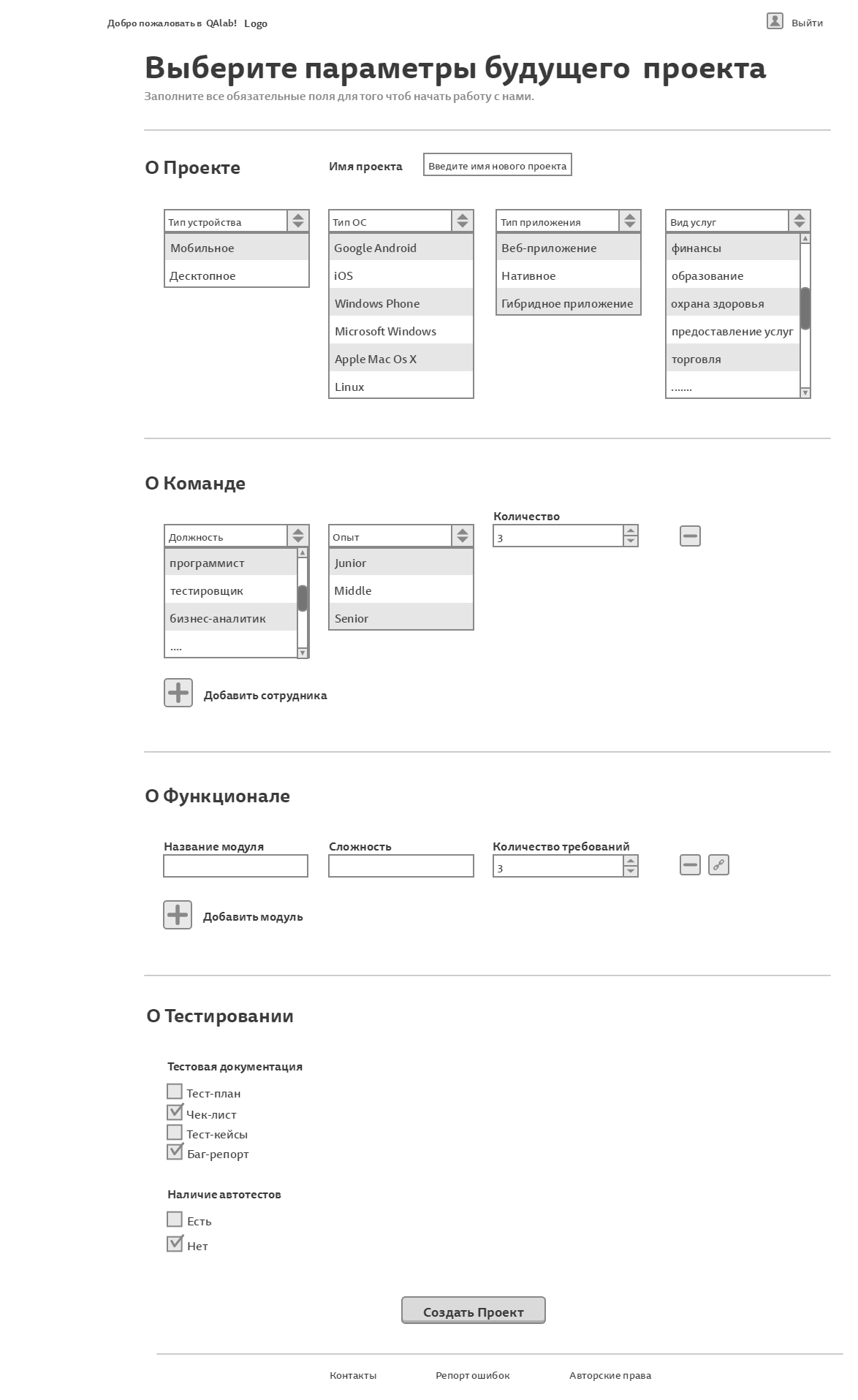
При регистрации пользователя нужно заполнить 5 полей личных данных, таких как: имя пользователя, должность, фамилия, имя, пароль. После заполнения полей регистрации, данные записываются в базу данных. Аккаунту присваивается роль пользователя. Происходит перенаправление на главную страницу. Макет интерфейса регистрации представлен на рисунке 3.5.



**Рисунок 3.5 – Макет интерфейса регистрации пользователя**

**Источник: собственная разработка автора**

После регистрации пользователя в системе, ему открывается главная страница приложения. На главной странице расположены поля для выбора данных о проекте. Главная страница разделена на четыри секции: о проекте, о команде, о функционале, ор тестировании. Ниже, на рисунке 3.6 представлен макет страницы выбора параметров будущего проекта.

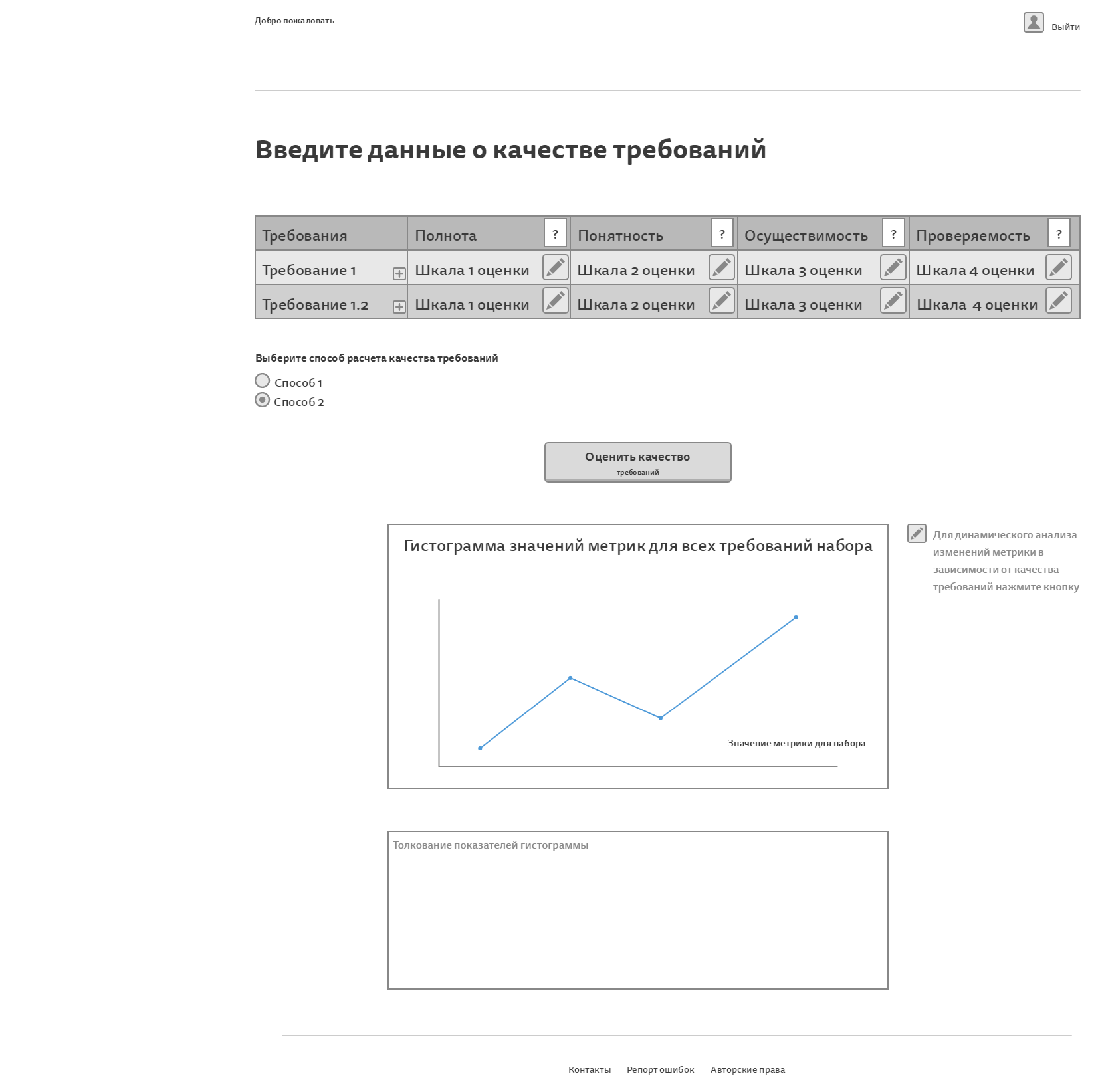


**Рисунок 3.6 – Макет интерфейса страницы ввода данных о будущем проекте**

**Источник: собственная разработка автора**

Также подсистема имеет страницу ввода данных о проекте для дополнения БД. Каждый пользователь имеет возможность вести и быть включенным в несколько проектов одновременно. Пользователь может вносить данные как во время реализации проекта, так и после его закрытия, если он был его участником.

Пользователь имеет возможность расчета, визуализации, интерактивного моделирования значений метрик в зависимости от значений различных показателей, через которые они рассчитываются. Макет пользовательской страницы для ввода первичной информации, визуализации и интерактивного моделирования представлен на рисунке 3.7.



**Рисунок 3.7 – Макет пользовательской страницы для ввода первичной информации, визуализации и интерактивного моделирования**

**Источник: собственная разработка автора**

Доступ к веб-приложению должен быть ограничен только для сотрудников компании.

**Диаграмма вариантов использования**

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц,

участвующими в процессе. Учитывая требования, предъявляемые к системе, были выделены следующие роли, рисунок 3.8; рисунок 3.9:

**Гость:**

* регистрация/вход;

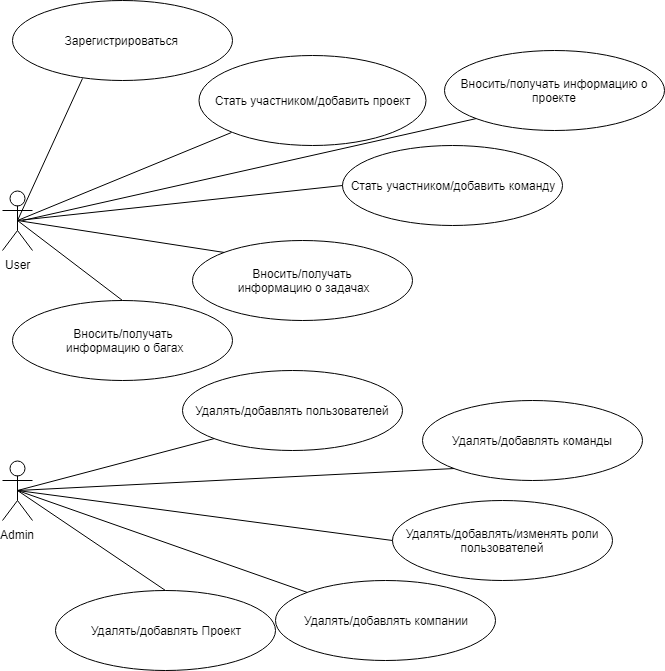
**Пользователь:**

* Просмотр всей информации на сайте;
* Возможность вносить данные (создавать) о новом проекте, участником которого он является;
* Возможность вносить данные о проекте, участником которого он является;
* Возможность вносить данные о будущем проекте, на основе которых будет строиться стратегия тестирования;
* Стать участником команды;
* Создать команду.

**Администратор:**

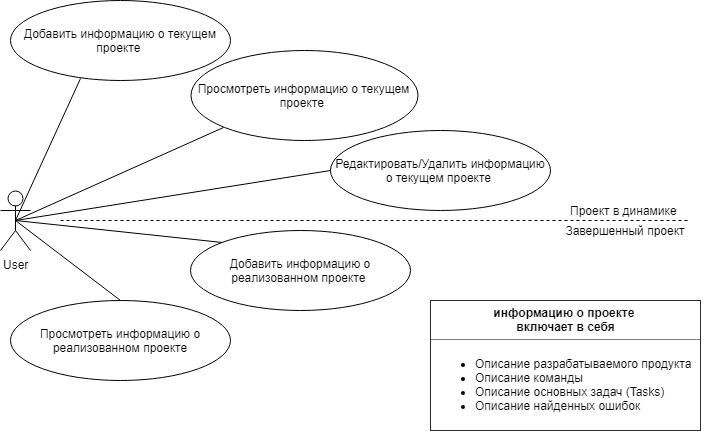
* Просмотр всей информации на сайте;
* Управление контентом на сайте;
* Управление пользователями.

Более подробно варианты использования системы представлены на рисунках 3.8 и 3.3.



**Рисунок 3.8 – Диаграмма вариантов использования «Работа пользователей с сайтом и их роли»**

**Источник: собственная разработка автора**



**Рисунок 3.9 – Диаграмма вариантов использования «Работа пользователей с сайтом и их роли»**

**Источник: собственная разработка автора**

### 3.2.4 Проект аналитического модуля

Аналитический модуль предназначен для обработки данных, хранящихся в бд аналитической системы с целью извлечения знаний, которые затем могут использоваться при управлении этапом тестирования. Аналитический модуль взаимодействует с базой данных по нескольким направлениям. Модуль может подвергать анализу первичные данные. Также модуль способен использовать первичные данные в для вычисления расчетных метрик.

Основное назначение аналитического модуля **–** применение методов Data Mining для извлечения знаний из системы метрик этапа тестирования.

Примеры запросов к базе данных, применяемых для задач планирования и управления этапом тестирования:

1. На каком этапе тестирования было выявлено наибольшее количество дефектов;
2. На каком этапе тестирования было выявлено наибольшее количество дефектов с серьезностью критические и выше;
3. Количество дефектов в разрезе критичности и этапов;
4. На каком этапе тестирования было выявлено наибольшее количество дефектов с приоритетностью средний и выше;
5. Количество багов в разрезе этапов тестирования;
6. Количество багов найденных заказчиком;
7. Максимальное количество багов с высоким приоритетом;
8. Количество сложных багов (в разрезе шагов) найденных до выпуска продукта в релиз (найденных заказчиком/пользователем).

В результате взаимодействия с базой данных (см. таблицу А1 в приложении А) будут получены следующие ответы на запросы:

1. Наибольшее количество дефектов было выявлено на этапах: функционального тестирования (6 из 20);
2. На этапе функционального (2 из 20), регрессионного (2 из 20), и Е2Е (2 из 20) тестирования было выявлено наибольшее количество дефектов с серьезностью критические и выше;
3. На этапе функционального тестирования было выявлено наибольшее количество (4 из 20) дефектов с приоритетностью средний и выше;
4. Количество багов в разрезе этапов тестирования (рисунок 3.10.);



**Рисунок 3.10 – Вывод по запросу “Количество багов в разрезе этапов тестирования”**

**Источник: собственная разработка автора**

1. Количество багов, найденных заказчиком равно 1 из 20;

### 3.2.5 Бизнес требования к разработке

Для разработки спроектированной информационно-аналитической системы потребуется:

1. **Тимлид**/бизнес-аналитик (Junior/Middle, 1 человек);
2. **Разработчик БД** (Junior/Middle, 1 человек);
3. **Dot.Net разработчик** (Junior/Middle,1 человек);
4. **Тестировщик** (Junior/Middle,1 человек).

Программное обеспечение проекта информационно-аналитической системы будет разрабатываться в рамках аутсорсинговой ИТ-компании. Компания будет выступать в роли заказчика и исполнителя. Спроектированная информационно-аналитическая система предлагается как разработка внутренней системы для повышения эффективности этапа тестирования. В качестве исполнителей предлагается выбрать внутренних сотрудников, которые будут реализовывать проект как дополнительный развивающий навыки проект (например, в качестве выполнений целий по самосовершенсвованию).

**Роли исполнителей.**

*Тимлид/бизнес-аналитик.*

1. Обоснование и определение бизнес-потребности проекта, целей и задач;
2. Описание будущих положительных итогов (выгода, которую получит компания внедрив проект);
3. Выявление и формулировка бизнес-требований, пользовательских требований, функциональных требований;
4. Планирование сроков реализации ПО.

*Разработчик БД*

1. Проектирование БД “Опыт реализованных проектов”;
2. Разработка БД “Опыт реализованных проектов”;
3. Исправление ошибок на этапе разработки;
4. Внедрение и сопровождение БД “Опыт реализованных проектов”.

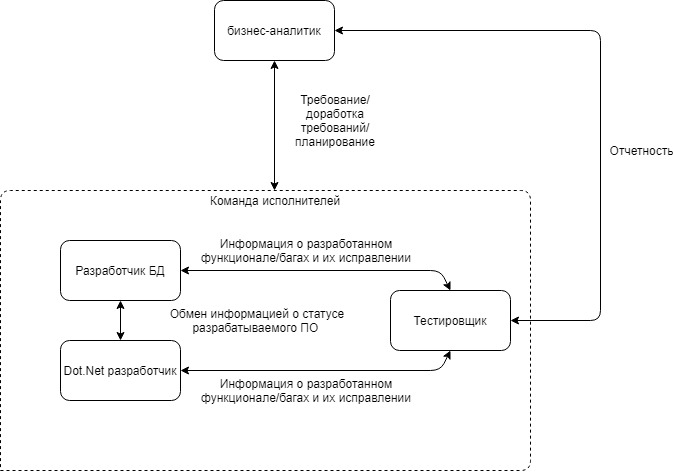
*Dot.Net разработчик*

1. Проектирование архитектуры Веб-приложения;
2. Разработка приложения;
3. Разработка клиент-серверной стороны (API);
4. Исправление ошибок на этапе разработки;
5. Внедрение и сопровождение.

*Тестировщик*

1. Тестирование основного функционала (Все виды тестирования для веб приложений);
2. Ведение ошибок разработки и их документация;
3. Наблюдение и контроль процессов разработки и тестирование;
4. Ведение отчетности по итерациям перед Тимлидом/бизнес-аналитиком.

Поток информации между командой разработки представлен на рисунке 3.11.



**Рисунок 3.11 – Схема “Поток информации между командой разработки”**

**Источник: собственная разработка автора**

Типы потоков информации между бизнес-аналитиком и командой разработчиков:

1. Цель и задачи проекта (документация)
2. Требования и их доработка (документация)
3. План итераций и временные эстимации (документация)

Между разработчиками и тестировщиками

1. Требования на улучшение ПО
2. Тест-план
3. Баг репорты

## **3.3 Место информационно-аналитической системы управления ИТ-проектами на этапе тестирования в управлении рисками в тестировании ПО.**

Анализируя схему действий по управлению рисками, представленную на рисунке 1.2, можно утверждать, что подсистема способна стать инструментом работы по управлению рисками при планировании работ этапа тестирования, мониторинга и отчетности результатов, в анализе ошибок и извлечению уроков.

Для проведения работ по планированию можно использовать аналитический модуль подсистемы, который будет работать с метриками “этапа планирования”. Использование графических редакторов и инструментов визуализации веб-приложения подсистемы решит задачу отчетности и мониторинга результатов. Работа БД “Опыт тестирования реализованных проектов” по накоплению, хранению и выводу информации о реализованных проектах способна стать полезной в анализе ошибок и извлечению уроков.

## **Выводы по главе 3**

В соответствии с применением метрик на различных этапах управления тестированием ПО выделены метрики класса планирования и класса результативности. Метрики класса планирования позволяют предложить вариант быстрой разработки и обосновать низкое качество ПО, объяснить заказчику причину долгой разработки. Метрики класса результативности отражают в динамике процесс тестирования программного продукта, позволяют подготавливать отчетность перед заказчиком. На основе данных метрик можно подвергать анализу достигнутые результаты, допущенные ошибки или, наоборот, успехи.

Подсистема информационно-аналитической системы управления ИТ-проектами на этапе тестирования для сбора, анализа выработки и мониторинга принятых решений и показателей качества реализуется в виде веб-приложения, которое может быть внедрено в ИТ-компанию, предоставляющую, услуги разработки, сопровождения и тестирования программных продуктов в разных отраслях. Веб-приложение имеет доступ к БД “Опыт реализованных проектов”, на основе информации из которой можно разрабатывать и оценивать разные варианты стратегий этапа тестирования. Подсистема способна стать инструментом работы по управлению рисками при планировани работ этапа тестирования, мониторинга и отчетности результатов, в анализе ошибок и извлечению уроков.

# **ЗАКЛЮЧЕНИЕ**

Качество программного обеспечения – это способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям. Тестирование – одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов. Главной задачей процесса тестирования является подтверждение того, что продукт является качественным. Стандарт ISO/IEC 25010:2011 определяет модель качества продукта, которая включает восемь характеристик верхнего уровня: функциональная пригодность; уровень производительности; совместимость; удобство использования (юзабилити); надежность; защищенность; сопровождаемость; переносимость (мобильность). С точки зрения тестировщика качество – показатель степени соответствия продукта предъявляемым к нему требованиям.

В процессе тестирования могут возникать различные проблемы (нечеткие требования, неверные эстимации на тестирование или несоблюдение графика работ, неверное сопоставление ресурсов с задачами проекта (трудовые ресурсы-персонал), игнорирование мелких проблем на ранних стадиях разработки проекта, нарушение плана разработки продукта со стороны менеджмента, человеческий фактор), которые являются предпосылками для возникновения рисков.

Для недопущения и решения проблем следует проводить регулярный мониторинг показателей процесса тестирования. С этой целью можно использовать метрики этапа тестирования, которые позволяют измерить характеристики процесса и результатов тестирования, помогают визуализировать процесс, состояние тестов, оценить результаты тестирования и качество продукта и могут стать хорошим инструментом визуализации критериев качества. Метрики помогут представить информацию о тестировании разных проектов в виде, пригодном для хранения, накопления и анализа.

Введение и использование метрик необходимо для улучшения контроля этапа тестирования, что позволяет повлиять на принятие своевременных управленческих решений. Цель контроля тестирования состоит в получении обратной связи и визуализации процесса тестирования. Необходимую для контроля информацию собирают (как вручную, так и автоматически) и используют для оценки состояния и принятия решений, таких как покрытие требований или кода тестами, или критерии окончания тестирования. Метрики также могут быть использованы для оценки прогресса выполнения запланированных работ и освоения бюджета.

Для формирования системы метрик используются различные типы данных и различные шкалы. Информацию, которая используется для принятия решений при планировании, разработке и тестировании ИT-проектов, можно получать в результате преобразования и анализа данных и имеющихся знаний. Основными исходными данными, которые необходимо собрать о разрабатываемом проекте, являются проектные данные. Исходные данные могут быть применены в расчетных метриках, необходимых для выбора наиболее подходящей стратегии тестирования, либо сами являются метриками. Отсутствие единой базы для формирования и расчета метрик, систематизации накопленных знаний и результатов применения метрик является проблемой анализа и планирования этапа тестирования и ИТ-проекта в целом.

Метрики, применяемые на этапе тестирования ПО, можно классифицировать по следующим признакам: по этапам тестирования; по сущностям. По способу получения значений метрики можно разделить на расчетные метрики, которые вычисляются по определенным правилам из некоторых первичных показателей, измеренных в заданных шкалах; на измеряемые, которые сами являются первичной информацией.

В соответствии с применением метрик на различных этапах управления тестированием ПО выделены классы метрик: метрики класса планирования и класса результативности. Метрики класса планирования можно использовать для выработки предложений вариантов быстрой разработки, с их помощью можно обосновать низкое качество ПО, либо объяснить заказчику причину долгой разработки. Метрики класса результативности отражают в динамике процесс тестирования программного продукта, позволяют подготавливать отчетность перед заказчиком. Также на основе данных метрик можно подвергать анализу достигнутые результаты, допущенные ошибки или, наоборот, успехи.

С целью упрощения формирования системы метрик, увеличения информативности их применения, обеспечения регулярного мониторинга показателей процесса и результатов тестирования в работе предложена подсистема информационно-аналитической системы управления ИТ-проектами на этапе тестирования для сбора, анализа выработки и мониторинга принятых решений и показателей качества.

Подсистема реализуется в виде веб-приложения. Веб-приложение имеет доступ к БД “Опыт реализованных проектов”, на основе информации из которой можно предлагать и оценивать различные варианты стратегий этапа тестирования. Подсистема способна стать инструментом работы по управлению рисками при планировании работ этапа тестирования, мониторинга и отчетности результатов, в анализе ошибок и извлечении уроков.

Приложение может быть внедрено в ИТ-компанию, предоставляющую, услуги разработки, сопровождения и тестирования программных продуктов в разных отраслях. Внедрение спроектированной информационно-аналитической системы **управления ИТ-проектами на этапе тестирования** позволит оптимизировать этап тестирования, повысить качество разрабатываемого программного обеспечения и эффективность управления разработкой ПО.

# **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

## **Список использованных источников**

1. ИСО 8402-94 Управление качеством и обеспечение качества [Электронный ресурс] // stroysvoimirukami.ru – Электронный журнал. – Режим доступа: http://stroysvoimirukami.ru/iso-8402-94/. – Дата доступа: 01.03.2018.
2. Курс «Тестирование ПО» Тема 4 «Разработка тестов» ЦОТ [Электронный ресурс] // ppt-online.org – ЦОТ "Белхард" – Режим доступа:https://ppt-online.org/17885. – Дата доступа: 01.03.2018.
3. Риски в тестировании ПО [Электронный ресурс] // software-testing.ru – Портал тестирование и качество ПО. – 1. Режим доступа:http://software-testing.ru/library/testing/general-testing/335-risk-management-in-software-testing/ . – Дата доступа: 01.03.2018.
4. Афифи, А. Статистический анализ: Подход с использованием ЭВМ. / С. Эйзен . Пер. с англ Мир,. – 1982. – C. 488.
5. ISO/IEC 25000:2014 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE [Электронный ресурс] // www.sis.se – Режим доступа:https://www.sis.se/api/document/preview/917156/. – Дата доступа: 20.03.2018.
6. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению [Электронный ресурс] // http://docs.cntd.ru – Электронный фонд правовой и нормативно-технической документации – Режим доступа: http://docs.cntd.ru/document/gost-r-iso-mek-9126-93/ . – Дата доступа: 20.03.2018.
7. C/S2ESC - Software & Systems Engineering Standards Committee. [Электронный ресурс] // http://www.mit.jyu.fi – Режим доступа: http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE\_SoftwareEngGlossary.pdf. – Дата доступа: 20.03.2018.
8. ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов [Электронный ресурс] // http://docs.cntd.ru – Электронный фонд правовой и нормативно-технической документации – Режим доступа: http://docs.cntd.ru/document/1200121069/ . – Дата доступа: 20.03.2018.
9. Гласс, Р. Факты и заблуждения профессионального программирования. Facts and Fallacies of Software Engineering. / Пер. с англ. – СПб.: Символ-Плюс, 2007. – 240. c., ил.
10. ISO/IEC 14598-6 Software engineering – Product evaluation – Part 6: Documentation of evaluation modules [Электронный ресурс] // http://bcc.portal.gov.bd – Режим доступа: http://bcc.portal.gov.bd/sites/default/files/files/bcc.portal.gov.bd/page/adeaf3e5\_cc55\_4222\_8767\_f26bcaec3f70/ISO\_IEC\_14598-6.pdf – Дата доступа: 01.04.2018.
11. Самые важные метрики QA [Электронный ресурс] // doitsmartly.ru Тестирование ПО. – Режим доступа: https://doitsmartly.ru/all-articles/sw-testing/133-the-most-important-metrics-in-qa.htm/ – Дата доступа: 15.04.2018.
12. Оценка проектов тестирования [Электронный ресурс] // www.slideshare.net – Доклад Григория Сенина на встрече Московского клуба тестировщиков в офисе Билай. Режим доступа: https://www.slideshare.net/rinauzhevko/ss-46816672 – Дата доступа: 01.05.2018.
13. ГОСТ Р ИСО МЭК ТО 10032-2007: Эталонная модель управления данными. Идентичен ISO/IEC TR 10032:2003 Information technology — Reference model of data management [Электронный ресурс] // http://docs.cntd.ru – Электронный фонд правовой и нормативно-технической документации – Режим доступа: http://docs.cntd.ru/document/gost-r-iso-mek-to-10032-2007/ – Дата доступа: 09.07.2018.
14. ГОСТ 33707-2016 (ISO/IEC 2382:2015) Информационные технологии (ИТ). Словарь [Электронный ресурс] // http://docs.cntd.ru – Электронный фонд правовой и нормативно-технической документации – Режим доступа: https://webstore.iec.ch/publication/22380 / – Дата доступа: 09.07.2018.
15. ГОСТ Р ИСО/МЭК ТО 10032-2007 Эталонная модель управления данными. Идентичен ISO/IEC TR 10032:2003 Information technology – Reference model of data management [Электронный ресурс] // http://www.gostrf.com – Информационно-справочная система – Режим доступа: http://www.belstat.gov.by/ofitsialnaya-statistika/solialnaya-sfera/uslugi/publikatsii\_11/index\_289/ . – Дата доступа: 09.07.2018.
16. Кузнецов, С.Д. Основы баз данных. / С.Д. Кузнецов // 2-е изд. – М.: Интернет-университет информационных технологий БИНОМ. Лаборатория знаний, 2007. – 484 с.
17. Проектирование реляционных баз данных [Электронный ресурс] // http://www.protesting.ru – сайт Мои Лекции.ру – Режим доступа: https://mylektsii.ru/5-100107.html . – Дата доступа: 09.07.2018.
18. Серьезность и Приоритет Дефекта [Электронный ресурс] // mylektsii.ru – сайт Про Тестинг – Тестирование Программного Обеспечения – Режим доступа: http://www.protesting.ru/testing/bugpriority.html. – Дата доступа: 10.09.2018.

## **Список публикаций соискателя**

1–А. Праулова, Н.С. Динамика численности населения Гродненской области / Н.С. Праулова // Устойчивое развитие экономики: международные и национальные аспекты: материалы III Междунар. науч.-практ. конф. (Полоцк, 18-19 апр. 2019 г.) – (в печати);

2–А. Праулова, Н.С. Зеленая экономика Гродненской области / Н.С. Праулова // Проблемы прогнозирования и государственного регулирования социально-экономического развития: материалы XIX Междунар. науч. конф. (Минск, 18-19 окт. 2018 г.). В 3 т. Т. 3 / редкол.: В.В. Пинигин [и др.]. – Минск: НИЭИ М-ва экономики Респ. Беларусь, 2018. – 228с. – С.127-129.

3–А. Праулова, Н.С. Прогнозирование демографической ситуации в Гродненской области / Н.С. Праулова // Молодежь и научно-технический прогресс: материалы ХII Междунар. науч.-практ. конф. (Губкин, 18 апр. 2019 г.). В 3 т. Т. 3 / Сост.: Е. Н. Иванцова, В. М. Уваров [и др.]. – Губкин ; Старый Оскол : ООО «Ассистент плюс», 2019. – 453с. – С.84-87.

# **ПРИЛОЖЕНИЯ**

**ПРИЛОЖЕНИЕ А**

Таблица А1. – Пример данных по найденным багам, хранящихся в БД

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Шаги** | **Приоритет** | **Серьезность** | **Этап нахождения** | **Даты** | |
| **Открыт** | **Закрыт** |
| 1 | Название1 | 6 | P2 Средний (Medium) | S1 Блокирующая (Blocker) | Е2Е тестирование | 01.01.2019 10:00:00 | 01.02.2019 18:00:00 |
| 2 | Название2 | 13 | P3 Низкий (Low) | S3 Значительная (Major) | анализ документации | 17.01.2019 0:00:00 | 21.02.2019 0:00:00 |
| 3 | Название3 | 3 | P3 Низкий (Low) | S4 Незначительная (Minor) | функциональное тестирование | 18.01.2019 0:00:00 | 22.02.2019 0:00:00 |
| 4 | Название4 | 15 | P1 Высокий (High) | S2 Критическая (Critical) | функциональное тестирование | 28.01.2019 0:00:00 | 17.02.2019 0:00:00 |
| 5 | Название5 | 15 | P2 Средний (Medium) | S2 Критическая (Critical) | смоук тестирование | 16.01.2019 0:00:00 | 16.02.2019 0:00:00 |
| 6 | Название6 | 11 | P2 Средний (Medium) | S4 Незначительная (Minor) | анализ документации | 29.01.2019 0:00:00 | 05.02.2019 0:00:00 |
| 7 | Название7 | 2 | P3 Низкий (Low) | S4 Незначительная (Minor) | регрессионное тестирование | 21.01.2019 0:00:00 | 05.02.2019 0:00:00 |
| 8 | Название8 | 4 | P2 Средний (Medium) | S3 Значительная (Major) | смоук тестирование | 01.02.2019 0:00:00 | 24.02.2019 0:00:00 |
| 9 | Название9 | 2 | P1 Высокий (High) | S4 Незначительная (Minor) | после релиза | 13.01.2019 0:00:00 | 16.02.2019 0:00:00 |
| 10 | Название10 | 15 | P2 Средний (Medium) | S4 Незначительная (Minor) | функциональное тестирование | 14.01.2019 0:00:00 | 17.02.2019 0:00:00 |
| 11 | Название11 | 5 | P1 Высокий (High) | S1 Блокирующая (Blocker) | регрессионное тестирование | 04.01.2019 0:00:00 | 12.02.2019 0:00:00 |
| 12 | Название12 | 8 | P3 Низкий (Low) | S4 Незначительная (Minor) | Е2Е тестирование | 25.01.2019 0:00:00 | 06.02.2019 0:00:00 |
| 13 | Название13 | 13 | P2 Средний (Medium) | S3 Значительная (Major) | заказчиком | 04.01.2019 0:00:00 | 15.02.2019 0:00:00 |
| 14 | Название14 | 5 | P1 Высокий (High) | S1 Блокирующая (Blocker) | Е2Е тестирование | 25.01.2019 0:00:00 | 18.02.2019 0:00:00 |
| 15 | Название15 | 9 | P2 Средний (Medium) | S2 Критическая (Critical) | регрессионное тестирование | 02.01.2019 0:00:00 | 27.02.2019 0:00:00 |
| 16 | Название16 | 3 | P3 Низкий (Low) | S3 Значительная (Major) | функциональное тестирование | 27.01.2019 0:00:00 | 10.02.2019 0:00:00 |
| 17 | Название17 | 1 | P2 Средний (Medium) | S1 Блокирующая (Blocker) | функциональное тестирование | 29.01.2019 0:00:00 | 13.02.2019 0:00:00 |
| 18 | Название18 | 2 | P2 Средний (Medium) | S3 Значительная (Major) | функциональное тестирование | 04.01.2019 0:00:00 | 18.02.2019 0:00:00 |
| 19 | Название19 | 15 | P3 Низкий (Low) | S4 Незначительная (Minor) | регрессионное тестирование | 15.01.2019 0:00:00 | 04.02.2019 0:00:00 |
| 20 | Название20 | 14 | P2 Средний (Medium) | S1 Блокирующая (Blocker) | анализ документации | 09.01.2019 0:00:00 | 08.02.2019 0:00:00 |