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

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему Исследование методов анализа данных при управлении процессом тестирования

Обучающийся Д.С. Олейник _____
(Инициалы Фамилия) (личная подпись)

Научный руководитель к.п.н., доцент Крайнова Ольга Анатольевна
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Оглавление

Введение

Тестирование крайне важно для разработки качественного продукта, а стратегия организации процесса тестирования может сильно повлиять на скорость разработки и финальный результат [19]. Тестовая деятельность должна быть правильно организована, чтобы обеспечить высокую эффективность работы и ожидаемый уровень качества программного обеспечения. Для управления процессом тестирования используются методы анализа данных, однако на сегодняшний день общедоступные источники не представляют возможности получить четкие инструкции по проведению анализа в сфере тестирования. Для использования методов анализа данных используются знания из менеджмента, практический опыт в сфере разработки программного обеспечения, теория тестирования программного обеспечения. Таким образом, проблема исследования – отсутствие методических материалов по проведению анализа данных в сфере управления тестированием. Возможность получать готовые рекомендации, созданные специально для сферы управления тестированием, определенно повысила бы уровень качества управления тестовой деятельностью, что обуславливает актуальность темы исследования.

Анализ материалов по теме тестирования и управления тестированием показал, что основная часть теоретических знаний и рекомендаций направлена на выбор методики тестирования и на этапы управления тестированием [11], [12], [35]. Также существует достаточно источников, содержащих информацию о ролях в процессе тестирования, разделении ответственности, управлении проектом в целом, управлении коммуникациями и планировании разработки проекта [6], [9], [27], [28]. В некоторых материалах содержится рекомендательная информация о методах анализа данных, применяемых в тестировании, однако эта информация недостаточно конкретна для проведения качественного анализа [26]. Соотнеся знания из разных источников, можно создать методические рекомендательные материалы, актуальные для управления

процессом тестирования. Проблема данной научно-исследовательской работы – необходимость сбора, систематизации и анализа имеющихся теоретических данных, для использования их в качестве методических материалов в работе по управлению тестовой деятельностью.

Наибольшую ценность при решении проблемы исследования несут в себе методики управления проектами, в особенности те, которые придают большое значение применению методик анализа данных для повышения эффективности работы над проектом [4]. Процесс тестирования, как и иные процессы в разработке программного обеспечения, имеет множество способов организации, являясь гибким. Эффективность процессов тестирования зависит именно от выбора правильной стратегии организации тестовой деятельности руководителем [20]. Выбор выгодной стратегии основывается на аналитическом процессе определения целей и факторов влияния. Использование методов анализа данных в процессе управления тестированием позволяет руководителю выбирать наиболее эффективные пути управления процессами и своевременно адаптировать их в соответствии с изменениями.

Объектом исследования являются процессы управления тестированием, а предметом исследования – способы получения, анализа и использования данных для выстраивания эффективной стратегии организации процесса тестирования.

Целью данной работы является разработка рекомендаций по анализу данных для использования в процессе управления тестированием на основании исследования имеющихся теоретических материалов.

Гипотезой исследования является предположение о возможности повышения эффективности процессов тестирования с помощью применения методов анализа данных по авторским методикам.

Для достижения цели исследовательской работы поставлены следующие задачи:

– Изучение организационных процессов в разработке программного обеспечения;

– Изучение инструментария, используемого в процессах тестирования программного обеспечения;
– Обзор процессов управления в разработке программного обеспечения;
– Исследование способов получения данных в процессе тестирования программного обеспечения;
– Исследование процессов в управлении тестированием;
– Исследование методов анализа данных, применяемых в управлении тестированием;
– Разработка рекомендаций по сбору и анализу информации для применения методов анализа данных и повышения эффективности процессов тестирования;
– Разработка рекомендаций по использованию полученных результатов анализа данных для повышения эффективности процессов тестирования;
– Апробация созданных рекомендаций и методик.

Научная новизна исследования заключается в разработке рекомендательных методик, направленных на повышение эффективности рабочих процессов тестирования.

Практическая значимость работы заключается в создании инструмента (собрания методических рекомендаций) для руководителей отделов тестирования, использование которого позволит применять методы анализа данных для повышения качества и скорости процессов тестирования, что приведет к повышению качества конечного продукта и эффективности процесса реализации ИТ-проектов.

Для решения поставленных в работе задач использовались следующие методы:

– Теоретический анализ данных, полученных из источников;

– Метод наблюдения процессов разработки программного обеспечения во время прохождения практик;
– Анализ и обработка полученных в процессе наблюдения данных.

На защиту выносятся:

– Рекомендательные методики по сбору и анализу данных для выстраивания процессов тестирования программного обеспечения;
– Рекомендательные методики по применению полученных результатов анализа для повышения эффективности работы в процессе тестирования программного обеспечения.

Структура ВКР: магистерская диссертация содержит введение, четыре главы, выводы по главам, заключение, библиографический список литературы, приложение.

Первая глава посвящена различным аспектам разработки программного обеспечения, которые имеют отношение к вопросу организации процесса тестирования. Показана роль тестирования в реализации продукта, связь процессов тестирования с иными процессам в разработке программного обеспечения. Рассмотрена взаимосвязь различных отделов, ответственных за разные этапы разработки программного обеспечения и роль отдела тестирования. Проведен обзор основных инструментов, используемых в управлении разработкой и тестовой деятельностью.

Во второй главе рассмотрены основные процессы организации и управления тестовой деятельность. Показаны возможности существующих инструментов и методик для организации работы команды. Приведены способы получение данных для последующего анализа с использованием существующих инструментов и основные методы анализа данных, используемые в сфере тестирования программного обеспечения.

Третья глава содержит разработанные автором рекомендательные методики по сбору, систематизации и обработке данных, актуальных для выбора стратегии организации различных процессов в тестировании программного

обеспечения, а также рекомендации по использованию полученных результатов. Показаны способы получения данных с помощью инструментов, упомянутых во второй главе. Описан способ проведения анализа данных с помощью авторских методик. Приведены авторские рекомендации для использования в ходе проведения анализа данных для изменения стратегии процессов тестирования в сторону большей эффективности.

В четвертой главе описана апробация созданных методик в процессе прохождения практики. Показано применение авторских методик в ходе анализа данных, проводимом для оценки эффективности работы команды тестирования. Приведены результаты анализа и описана стратегия применения разработанных методических рекомендаций для повышения эффективности работы команды, проведен анализ результатов эксперимента.

Глава 1 Организация тестирования ПО

1.1 Основные понятия процесса тестирования

Рассмотрим понятия, которые будут использованы в следующих главах:

Тестирование программного обеспечения - вид деятельности в области разработки программного обеспечения. Это исследование, проводимое в отношении программного обеспечения для предоставления заинтересованным сторонам информации о его качестве.

Специалист по тестированию – сотрудник, в работу которого входят задачи по проверке качества программного обеспечения. Цель работы инженера по тестированию в повышении качества конечного продукта и предоставлению экспертного мнения о проблемах в работе продукта.

Команда – группа специалистов одного профиля, либо группа разнопрофильных специалистов, работающих над одним проектом.

Руководитель проекта (РП) / проджект менеджер (ПМ) – сотрудник, назначенный для управления командой разработки продукта. В зависимости от размера проекта руководитель проекта и проджект менеджер могут быть представлены одним специалистом, или их обязанности могут быть разделены.

Руководитель команды (team lead, тим лид) – специалист высокого уровня, назначенный для управления сотрудниками одной команды.

Заказчик – сторона, заинтересованная в получении продукта, и имеющая ожидания от его реализации. Может представлять собой компанию, отдел, человека.

Тест-кейс (test case, тестовый случай) — сценарий или описание последовательности шагов при проведении тестирования. Состоит из детального описания шагов воспроизведения и ожидаемых результатов поведения системы.

Ручное (мануальное) тестирование — проверка работы программного обеспечения тестировщиком с использованием клиентского интерфейса программы или клиентского интерфейса вспомогательных программ.

Автоматизированное тестирование — проверка работы программного обеспечения тестировщиком с использованием программных средств – автотестов.

Автотест – автоматизированный тест-кейс. Программный код, запускаемый в специальном сервисе, автоматические воспроизводящий шаги тестового сценария и проверяющий фактические результаты поведения системы.

Требования — это описание разрабатываемого программного обеспечения и его ожидаемого поведения в различных сценариях использования.

Бизнес-требования – описание свойств и функций, которые должен выполнять продукт, чтобы быть ценным для конечных пользователей.

Техническое задание – документ, содержащий подробное описание требований, являющийся основанием для разработки и тестирования. Создается командой аналитики, путем формализации требований заказчика.

Аналитик – член команды аналитики, сотрудник, находящийся в контакте с заказчиком и формализующий требования заказчика для составления ТЗ.

Ошибка (баг) — дефект программного обеспечения, отличие фактического поведения системы от ожидаемого.

Тестовое покрытие — это совокупность тестов, с помощью которых оценивают качество программного обеспечения. Чем больше проверок, тем шире тестовое покрытие.

Agile — семейство методологий разработки программного обеспечения, подразумевающая гибкость и адаптивность процессов. Включает в себя философию, систему ценностей, принципы и рекомендации по подходу к решению задач.

Kanban — методология, входящая в семейство Agile. Включает в себя набор принципов и инструментов для управления процессом разработки.

Канбан-доска – инструмент методологии канбан, подразумевающий визуальное представление рабочих задач в виде карточек на дашборде.

Scrum — методология, входящая в семейство Agile. Включает в себя типовые методы по организации процесса разработки.

Приоритет – степень важности задачи.

Баг-трекер – программное обеспечение для внесения информации о найденных ошибках. Часто работает по принципу канбан-доски.

Сервис/система управления проектами – программное обеспечение для менеджмента проекта.

Ретроспектива – метод анализа данных, направленный на оценку прошедших событий.

Предиктивный анализ – метод анализа данных, направленный на создание прогноза и выстраивание стратегии.

Данные – необработанные факты об окружающей действительности.

Анализ данных - преобразование данных в выводы, на основе которых будут приниматься решения [2].

Цель анализа данных — изучение исследуемой ситуации, выявление тенденций, прогнозирование и получение рекомендации [24].

Верификикация – проверка продукта по техническому заданию, спецификации, на соответствие точно указанным параметрам.

Валидация – проверка продукта на соответствие ожиданиям конечного пользователя.

Разработчик – член команды разработки, сотрудник, занимающийся написанием программного кода.

Метрика – качественный или количественный показатель, демонстрирующий степень эффективности выполнения процесса или уровень качества некого продукта.

Перечень сокращений, используемых в следующих главах:

ПО – программное обеспечение;

ТЗ – техническое задание;

TMS – Test management system, система управления тестированием;

ЖЦ – Жизненный цикл;

ИТ – информационные технологии;

ИС – информационная система.

1.2 Жизненный цикл ПО

В процессе реализации проекта программного обеспечения, проект переходит по разным этапам, каждый из которых характеризуется определенной деятельностью и результатом [8]. В зависимости от принятой стратегии работы для каждого проекта можно выделить жизненный цикл, по которому он реализован.

Жизненный цикл ПО – упорядоченный процесс развития программного обеспечения начиная от момента идеи и заканчивая этапом прекращения поддержки. Жизненный цикл ПО принято делить на взаимосвязанные этапы, каждый из которых дает результаты, необходимые для выполнения следующего [47]:

- инициация;
- планирование;
- разработка;
- тестирование;
- релиз;
- поддержка.

Процесс работы над каждым этапом затрагивает разные команды, участвующие в разработке ПО – команды аналитики, разработки, тестирования, дизайна, поддержки и так далее. Работой специалистов, занимающихся управлением деятельности команд, является координация действий сотрудников в соответствии с текущим этапом жизненного цикла и принятой модели ЖЦ.

Модель жизненного цикла ПО – типовая методология создания ПО, подразумевающая специфическую последовательность этапов разработки, набор

требований и готовых решений для возникающих задач [10]. Выбор модели жизненного цикла обычно продиктован следующими факторами:

- предпочтения компании;
- предпочтения заказчика;
- технические особенности разрабатываемого ПО;
- имеющиеся ресурсы для разработки;
- цели разработки.

Существуют основные модели ЖЦ, на основании которых в каждом отдельном случае строятся процессы разработки:

- каскадная модель (Waterfall Model или модель водопада);
- инкрементная модель (Incremental Model);
- итеративная или итерационная модель (Iterative Model);
- спиральная модель;
- модель «хаоса»;
- разработка через прототипирование;
- разработка через тестирование;
- модель кодирования и устранения ошибок.

Наиболее популярными для компаний на текущий день являются модели Waterfall, Incremental, Iterative. Также активно используется гибридная методология - итеративно-инкрементная модель [23].

Модель Waterfall характеризуется строгой последовательностью процессов – каждый этап начинает выполняться только после полного завершения предыдущего [32], [34]. За счет этого упрощаются процессы управления разработкой ПО и появляется возможность прогнозирования сроков и стоимости проекта. На рисунке 1 представлена модель ЖЦ «Водопад».

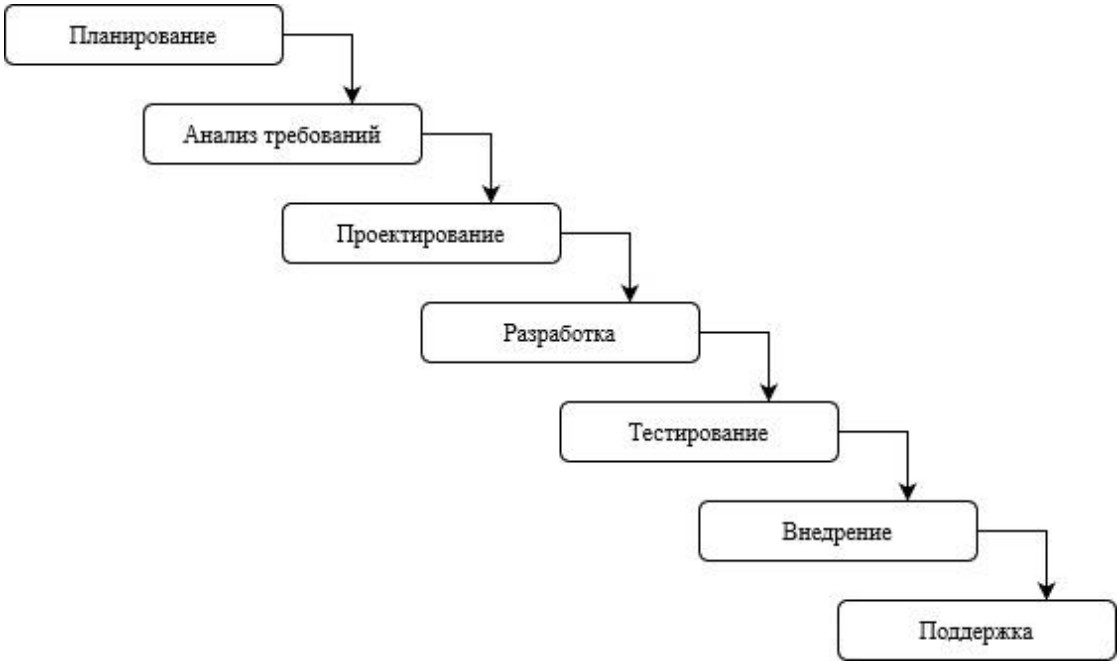

Рисунок 1 – Модель ЖЦ разработки ПО по модели водопада

Основными минусами модели является неравномерная нагрузка на команды, высокий уровень бюрократизации, высокая стоимость и сложность возвращения на предыдущий этап. Эта модель популярна для небольших проектов, в которых не будут изменяться требования, либо для проектов, где требуется высокий уровень отчетности (в основном государственные проекты) [3].

Инкрементная модель подразумевает разделение продукта разработки на инкременты – части продукта, которые будут разрабатываться последовательно [32]. Работа над каждым инкрементом представляет собой последовательное выполнение процессов. Но в отличие от модели Waterfall, так как работа ведется не над всем продуктом сразу, а лишь над его частью, не обязательно согласовывать все требования до начала разработки или полностью завершать этап, чтобы приступить к следующему – часть продукта, которая тормозит разработку, всегда можно перенести в другой инкремент. На рисунке 2 представлена инкрементная модель ЖЦ.

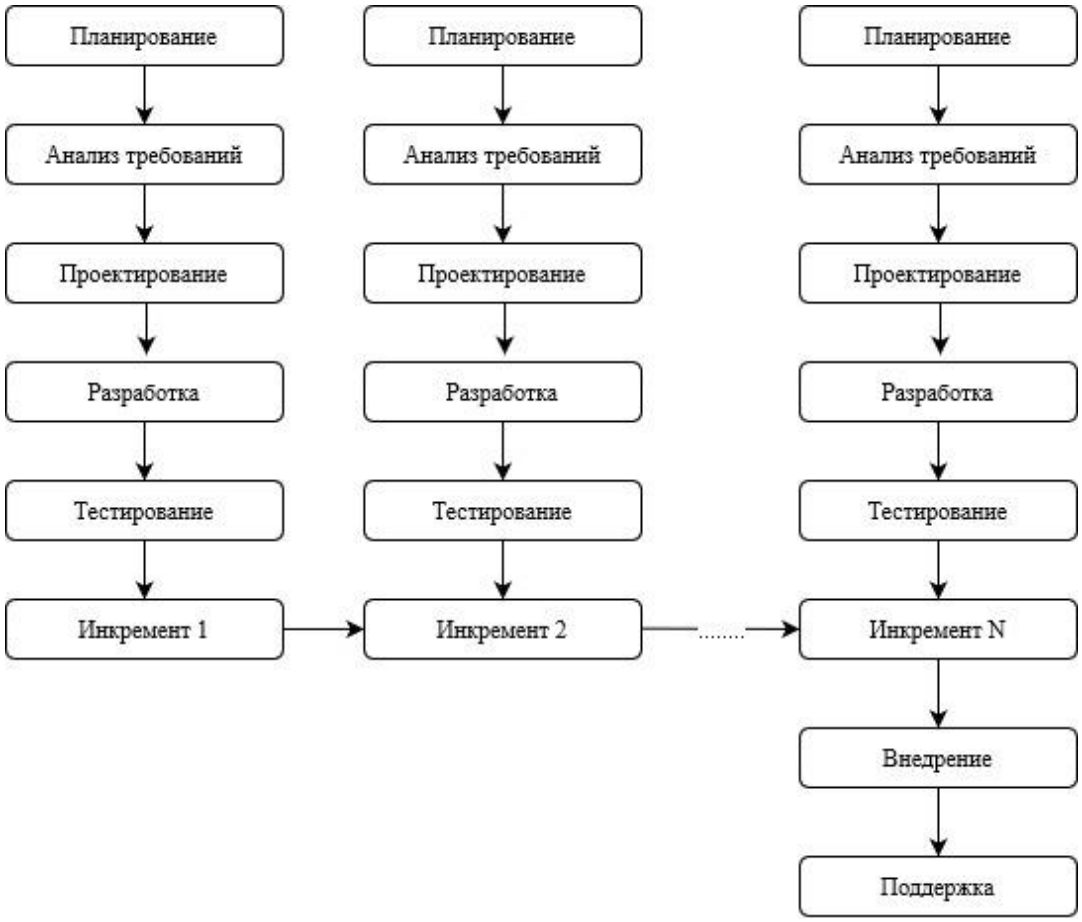

Рисунок 2 – Модель ЖЦ разработки ПО по инкрементной модели

Инкрементную модель разработки используются в случаях, когда планируемый проект удобно разделить на части – инкременты, каждый из который заказчик может получить отдельно, оценить и дать обратную связь.

Итеративная модель разработки представляет собой работу циклами – итерациями [32], [39]. В данном случаю делению на части подвергается не сам продукт, а работа по его реализации. В конце каждой итерации результатом работы будет являться новая версия продукта, в которую поэтапно будут вносить изменения. Основным плюсом итеративной модели является удобство заказчика и соответствие ценностям современного рынка – модель позволяет изменять требования к продукту уже в процессе разработки при минимальных затратах, что дает заказчику возможность изменять свой продукт согласно меняющимся

потребностям пользователей. На рисунке 3 представлена итеративная модель ЖЦ.

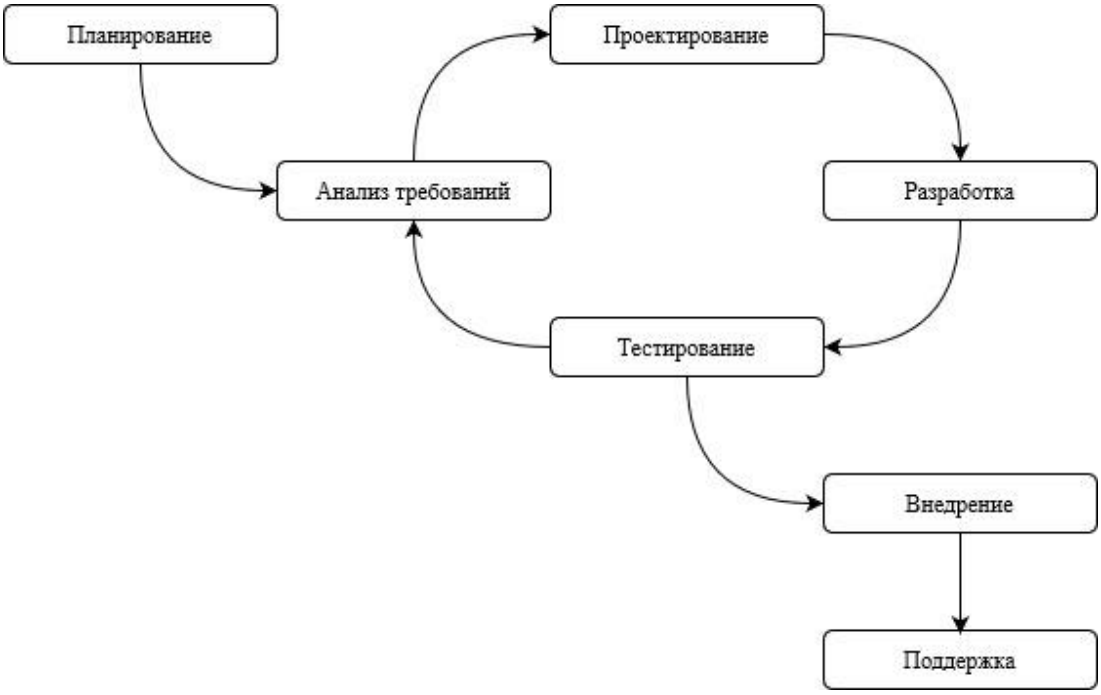

Рисунок 3 – Модель ЖЦ разработки ПО по итеративной модели

Итеративная модель на сегодняшний день используется в основном для разработки проектов пользовательского ПО средней или высокой сложности.

Инкрементный и итеративный подход к разработке могут быть похожи друг на друга, их основное различие к поставляемом заказчику продукте: при инкрементом подходе заказчик получает продукт по частям, в том время как при итеративном подходе каждая поставка является новой, улучшенной версией продукта.

1.3 Организация отделов разработки ПО

К отделам (командам), задействованным в разработке ПО обычно относят команды аналитики, дизайна, разработки, тестирования. Между сотрудниками этих команд делятся обязанности и ответственность на разных этапах разработки. Руководством, как правило, занимается проджект-менеджер, а также

тим-лиды – руководители отдельных команд. Роль тим-лида может выполнять специалист с большим опытом работы и навыками менеджмента, а также обладающий знаниями о рабочих процессах смежных команд.

Основными задачами руководителей проекта, продукта, команды и т.п. являются организация и управление работой команд, организация коммуникаций и передачи информация, мониторинг и оценка работы сотрудников [25]. Передача информации между командами проводится путем передачи документации и с помощью вербальной коммуникации – онлайн и оффлайн встреч, планерок.

Команда аналитики несет ответственность за создание и согласование требований, обсуждение концепта или идеи проекта с заказчиком. Специалисты команды создают техническое задание, которое далее используется командами разработки и тестирования. В процессе разработки ПО сотрудники команды аналитики также выполняют вспомогательную работу по поддержке других команд в ходе их работы с техническим заданием.

Команда дизайна отвечает за макеты будущего проекта. На основании ТЗ и пожеланий заказчика создаются дизайн-макеты ПО, которые используются командой разработки для создания клиентской части приложения.

Команда разработки занимается непосредственным созданием программного обеспечения – написанием кода, а также внесением корректив в написанный код по итогам тестирования или по факту получения обратной связи от заказчика.

Команда тестирования занимается обеспечением качества продукта. В обязанности специалистов по тестированию входит проверка продукта на соответствие требованиям и передача информации о состоянии и качестве продукта заказчику.

В процессе разработки команда тестирования занимает место «проверяющего» - тестированию подлежат результаты работы других команд [28]. Помимо программного кода команда тестирования проверяет требования к

продукту, техническое задание, дизайн-макеты. Смежные команды передают в команду тестирования готовый материал, а затем получают обратную связь на него.

1.4 Распределение ответственности участников проектной команды

В процессе реализации ИТ-проекта сотрудники имеют разные наборы задач и зоны ответственности. Правильная организация распределения задач и нагрузки может сильно повысить скорость и качество работы команды тестирования [14].

Для распределения рабочих задач, как правило, используются матрицы ответственности. Матрица ответственности – таблица распределения ролей, зон ответственности в проекте. Этот документ содержит полную информацию о рабочих задачах и сотрудниках, взаимосвязанных с выполнением этих задач. В случае с управлением процессом тестирования матрица ответственности может представлять из себя документ, в котором описано, за какой вид, раздел или функционал для тестирования отвечает каждый сотрудник. Такой документ позволяет сотрудникам четко определять границы своих задач и повышает эффективность коммуникации внутри команды [14]. На рисунке 4 приведен пример фрагмента матрицы ответственности с указанием ответственных за выполнение задачи, а также с комментариями к указанной информации.

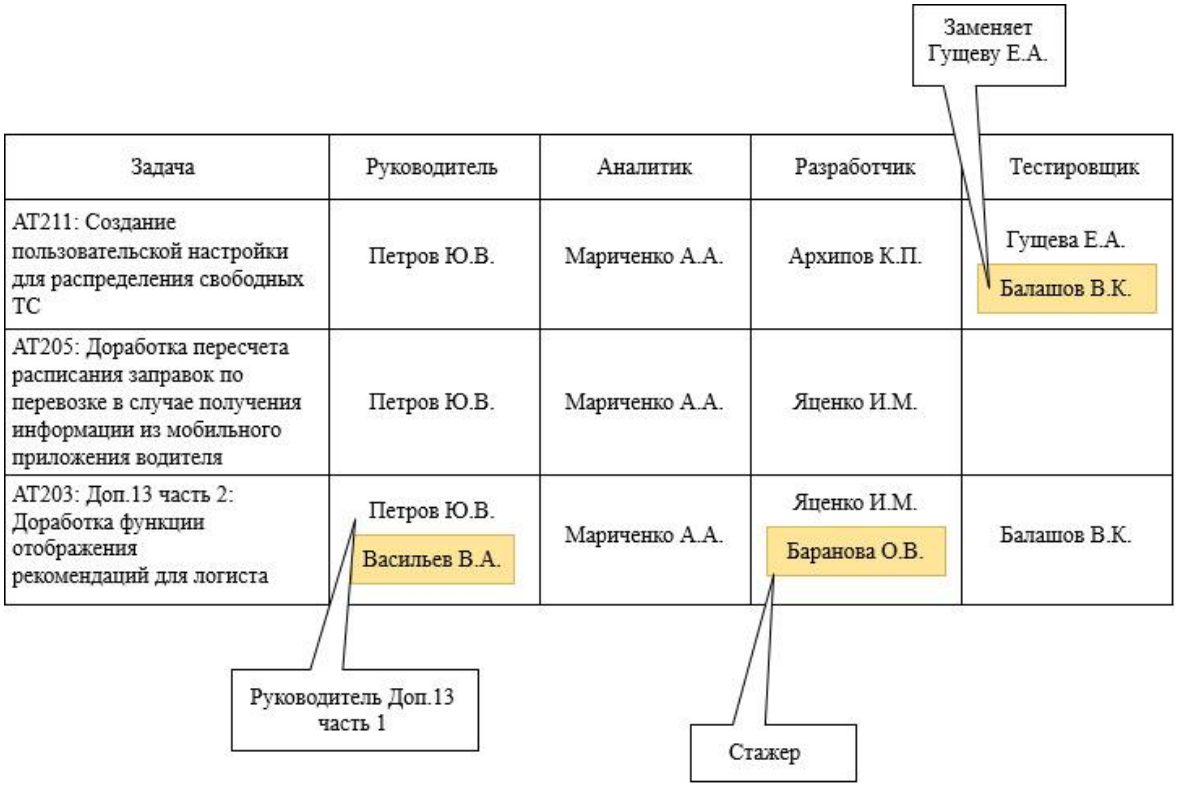

Рисунок 4 – Фрагмент матрицы с указанием ответственных

Выше представлен фрагмент документа, в котором указаны ответственные за задачи сотрудники. К примеру, используя данную матрицу, можно увидеть, что по задаче «АТ211» указаны два специалиста по тестированию и добавлен комментарий – Балашов В.К. заменяет Гущеву Е.А. Таким образом другие сотрудники смогут узнать, что тестированием по задаче занималась Гущева Е.А., а в период ее отсутствия Балашов В.К.

По задаче «АТ205» специалист по тестированию не указан. Скорее всего причина в том, что задача находится на этапе создания ТЗ, и тестировщик еще не назначен. Заполнить это пустое поле должен руководитель команды тестирования, выбрав подходящего сотрудника. Для этого руководитель может снова воспользоваться, проверив занятость сотрудников в работе по задачам, и выбрать специалиста, у которого есть ресурсы для выполнения задачи.

По задаче «АТ203: Доп.13 часть 2» указаны двое руководителей и оставлен комментарий: «Васильев В.А. – Руководитель Доп.13 часть 1» – Васильев В.А. является руководителем разработки смежной задачи. Эта информация позволит другим участникам процесса понять причину участия Васильева в решении вопросов разработки и покажет возможность обращаться к нему за информацией по смежной задаче.

Также для задачи «АТ203: Доп.13 часть 2» указаны двое разработчиков, один из которых отмечен как стажер. Этот комментарий позволит специалисту по тестированию Балашову В.К., ответственному за тестирование данный задачи, учитывать, что в программном коде, написанном Барановой О.В., вероятно, могут быть ошибки, и его нужно проверять особенно тщательно.

Матрицы ответственность часто создаются в виртуальных пространствах на платформах для управления проектами, в том числе с использованием канбан-доски. Канбан-доска – популярный элемент для управления проектами, позволяющий визуально отобразить ход работы над проектом. Под созданием виртуального пространства на платформе управления проектом понимается создание отдельной канбан-доски, на которой отображаются карточки задач команды по тестированию. Этот процесс происходит следующим образом: руководитель отдела тестирования работает с общей канбан-доской проекта, на которой отображены все задачи проекта. Руководитель проекта выделяет время на изучение существующих задач, далее отбирает те, в которых задействована работа команды тестирования и создает соответствующие карточки на доске команды. Такой метод эффективен при разработке объемного проекта, при большом количестве задач, при работе с большими командами [29].

Канбан-доска удобна в использовании для быстрого получения информации об ответственных за задачу сотрудниках. Карточка канбан-доски содержит информацию о специалистах, принимающих участие в работе над задачей, их ролями и зонами ответственности.

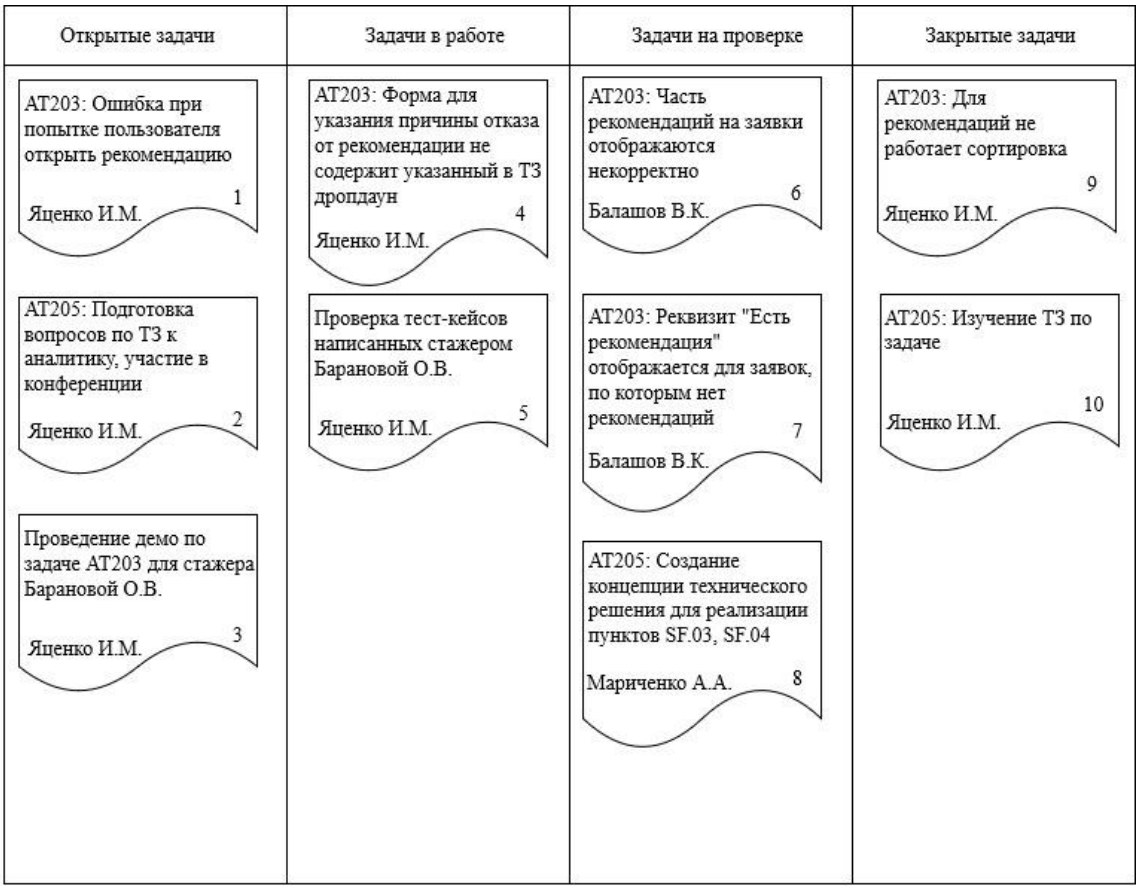

Рисунок 5 – Фрагмент канбан доски

На рисунке 5 представлен фрагмент канбан-доски специалиста команды разработки Яценко И.М. На доске отображены несколько подзадач по задаче АТ203 (номера 1; 4; 6; 7; 9). По описанию можно понять, что эти подзадачи – баги, созданные в процессе тестирования ответственным тестировщиком. К примеру, подзадачи в статусе «На проверке» (номера 6; 7) назначены на Балашова В.К. – это значит, что эти ошибки были исправлены и прогружены на окружение для тестирование, и теперь ответственный тестировщик должен проверить исправления. Ошибки в статусе «Открытые задачи», «Задачи в работе» и «Закрытые задачи» назначены на Яценко И.М., однако открыв карточку задачу можно получить информацию о том, кто автор (создатель) подзадачи.

По канбан доске руководители команд могут отслеживать процессы выполнения этапов реализация проекта другими командами. Например, руководитель команды тестирования может открыть канбан доску с задачами Яценко И.М. и увидеть, что по задаче АТ205 выполнена подзадача изучения ТЗ разработчиком (номер 10), а подзадача по созданию концепции реализации части пунктов ТЗ находится в статусе «На проверке». Это может показать тим-лиду тестировщиков, что целесообразно уже сейчас назначить ответственного за тестирование по этой задаче, так как задачи, предшествующие разработке, почти выполнены.

1.5 Обзор инструментария управления процессом тестирования

Помимо использования инструментов менеджмента проектов, подходящими для управления проектом в целом, команда тестирования может использовать специальный инструментарий. Внедрение продуктов, созданных специально для организации процессов тестирования, повышает эффективность и качество работы, а также упрощает процессы коммуникации и создания отчетности, мониторинга выполнения задач [37].

TMS – Test Management System – может предложить следующий функционал:

- Создание и хранение тестовых артефактов – многие TMS имеют встроенные функции специально для создания тест-кейсов, например сохранение общих шагов, автоматическое добавление шагов в группу тестов;
- Встроенный баг-трекер – возможность создавать прогоны тестов, отмечать их выполнение;
- Автоматическое создание отчетности;
- Интеграция с автоматизированными кейсам;
- Хранение данных по изменениям/версиям тест кейсов.

Основные TMS на 2023 год и их особенности:

- TestRail – одна из самых популярных TMS. Широко используется за счет возможности создания отчетности, экспорта и импорта существующей документации, интеграции с сервисами управления проектами (Jira, Confluence) и автоматизации тестирования (GitHub, Jenkins, Selenium, Visual Studio). TestRail подходит для использования в команде, в составе которой есть сотрудники с небольшим опытом, а также при отсутствии в компании требования к интеграции всех процессов в один сервис.
- Azure DevOps – многофункциональный инструмент для управления процессами тестирования, отслеживания результатов автотестов. Сильной стороной Azure является возможность создавать пользовательские доски (dashboards), что, однако, усложняет использование данного инструмента. Azure DevOps подойдет для использования более опытными сотрудниками, а так же может использоваться как дополнительный инструмент для использования менеджерами и руководителями команд.
- Zephyr является не отдельным инструментом, а плагином для использования в тандеме с Atlassian Jira. Уступает по функционалу предыдущим инструментам, однако позволяет лучше отслеживать рабочие процессы других отделов – разработки, аналитики, и привязывать к ним задачи по тестированию. Еще один минус – отсутствие ролевой модели. Использование Zephyr лучше всего подходит командам тестирования, участники которых имеют четкое представление о рабочих процессах смежных команд.
- TestIT – российская разработка, инструмент для тестирования с широким функционалом. Обладает гибкой системой работы, возможностью кастомизации, настройки прав и создание новых ролей. Единственным минусом системы является то, что она все еще находится в стадии разработки, и существующие версии оцениваются

пользователями как «сырые», с большим количеством ошибок. TestIT на сегодняшний день используется в основном в российских стартапах.

Выбор инструментов для управления тестированием обычно продиктован политикой компании. Однако руководители команд тестирования могут принимать участие в выборе конкретной TMS, что, при правильном подходе, повышает эффективность работы команды. Выше приведены особенности популярных TMS, которые могут как повысить, так и снизить эффективность работы команды. Правильный подход в выборе инструментария заключается в рассмотрении специфики команды, индивидуальных особенностей сотрудников, принятой в компании коммуникации для сопоставления полученных данных с известными различиями разных инструментов, и выбора наиболее подходящего. Основные аспекты анализа для руководителя:

- Опыт сотрудников – насколько эффективно они смогут работать со сложной TMS.
- Компетенции сотрудников – смогут ли сотрудники работать с интеграцией задач по тестированию и задач смежных команд.
- Требования компании по интеграции задач, отчетности – насколько важен функционал TMS по интеграции и автоматизированному созданию отчетности.
- Количество сотрудников и необходимость использования ролевой модели – важна ли возможность использовать разные права доступа в TMS.

Проведя исследование по вышеуказанным пунктам, руководитель отдела тестирования может выбрать для использования TMS, которая повысит эффективность работы.

1.6 Основные виды и этапы тестирования

Процессы тестовой деятельности различаются по видам и этапам. Существует несколько подходов к разделению тестирования на виды, вот несколько основных [15], [49]:

По ширине покрытия различают:

– Регрессионное тестирование – тестирование существующий, ранее проверенной функциональности, проводимое после обновлений системы. Цель тестирования – убедиться, что модификации не создали новые дефекты в уже проверенных компонентах. Регрессионное тестирование предполагает самое широкое покрытие.
– Валидация дефектов – тестирование компонента, в котором ранее была найдена ошибка. Цель проверки – убедиться, что исправления корректны и функциональность работает согласно требованиям после внесения корректив. Ширина покрытия самая узкая и покрывает только область дефекта.
– Тестирование новые функций – тестирование, проводимое на новых компонентах. Цель – убедиться, что новая функциональность работает в соответствии с требованиями. Среднее по ширине покрытие, тесты охватывают цельный компонент, и, опционально, связи компонента [17].

По знанию кода различают:

– «Белый ящик» - тестирование с полным доступом к коду. В этом случае специалист по тестированию может изучить программный код компонента и отслеживать его выполнение в процессе проведения теста.
– «Серый ящик» - тестирование с частичным доступом к коду. В этом случае специалист по тестированию может изучить часть программного кода компонента.

– «Черный ящик» - тестирование без доступа к коду. В этом случае специалист по тестированию не имеет возможности получить информацию из программного кода.

По степени автоматизации:

– Ручное тестирование – вид тестирования, при котором все действия по проведению тестов выполняются инженерами вручную [41].
– Автоматизированное тестирование – вид тестирования, при котором инженеры используют для проведения тестов дополнительные средства автоматизации [42].
– Полуавтоматизированное тестирование – вид тестирования, при котором тесты выполняются вручную, но с использованием вспомогательных автоматизированных средств. Например, использование автоматизированных сценариев для создания необходимого окружения или подготовки тестовый данных для дальнейшей ручной работы [12].

Также в зависимости от текущей цели тестирования различают несколько этапов работы. Основные этапы тестирования [16]:

– Работа с требованиями: изучение документации, тестирование и анализ требований. Этап начинается, как правило, с получения технического задания (ТЗ) от команды аналитики. На этом этапе специалист по тестированию получает данные о требованиях и ожиданиях от работы новой функциональности [18].
– Тестирование: прохождение тест-кейсов, проверка требований по ТЗ, валидация и верификация фактических результатов поведения системы и сравнение их с ожидаемыми. На этом этапе инженеры работают с системой, подвергая ее проверкам по различным сценариям.
– Создание тестовой документации: написание тест-кейсов, отчета о тестировании, резолюции и так далее. Проводится параллельно с этапом тестирования. Стоит заметить, что этап работы с документацией

частично состоит не из профильной работы инженеров по тестированию, а из задач по соблюдению нормативов компании с точки зрения отчетных документов.

Отдельно разберем два вида проверки обнаруженного дефекта:

– Верификация – процесс проверки обнаруженной особенности поведения системы на соответствие техническому заданию;
– Валидация – процесс проверки обнаруженной особенности поведения системы на соответствие потребностям конечных пользователей;

Так как эти два вида проверок достаточно похожи друг на друга, и их часто путают, целесообразно рассмотреть верификацию и валидацию на примере:

При тестировании задачи «Отображение сведений о перевозке» обнаружено, что информация о назначенном на рейс водителе отображается не полностью:

Проведем верификацию – проверку по ТЗ. В случае, если в ТЗ указано, что сведения о перевозке должны содержать полную информацию о водителе, создается баг-репорт по найденному несоответствию ТЗ. Если в ТЗ нет четких указаний о необходимости отображать данные водителя, то противоречия с ТЗ не найдено.

Проведем валидацию – проверку по ожиданиям пользователей. Для этого воспользуемся имеющейся документацией по продукту и помощью аналитика. Если в процессе валидации выяснено, что конечным пользователям нужна полная информация о водителе, то создается баг-репорт, так как система не удовлетворяет ожиданиям пользователей, хотя и не противоречит ТЗ.

В случае, если и верификация и валидация пройдены, возможно создать запрос на мнение заказчика. Если в ТЗ нет информации о необходимости отображать полную информацию о водителе, а исследование ожиданий конечных пользователей не дало нужной информации, целесообразно попросить заказчика об уточнении требований по этому пункту.

Процесс создания запроса к заказчику стоит рассмотреть отдельно. В этом случае к анализу дефекта подключаются специалисты команды аналитики и разработки, и для достижение нужного результата важна качественная коммуникация между командами. Цель специалиста по тестированию в данном случае – описать свое видение ожидаемого результата поведения системы и показать отличия его от фактического. Сотрудник команды аналитики дополняет запрос данными известных ему бизнес-требований, валидных для данного случая. Разработчик кода должен исследовать возможности изменения кода и предложить решения. Пример:

«В ТЗ нет информации о необходимости отображать полную информацию о водителе, а исследование ожиданий конечных пользователей не дало нужной информации». В данном случае инженер по тестированию обращается к специалистам смежных команд для проведения исследования поведения системы. Специалист по аналитике предоставляет следующие данные: «По статистике, основными пользователями данной функциональности являются логисты. По известным данным о работе процессов в транспортировке грузов, логистам могут понадобиться такие данные водителя как номер телефона и номер страховки. Поэтому было бы логично включить эти данные в сведения о перевозке». Исследования специалиста разработки показывают, что с точки зрения написания кода является возможным отображение номера телефона и номера страховки на странице. После получения указанных данных специалист тестирования создает запрос заказчику, включив вышеперечисленную информацию. Заказчик оценивает переданную информацию, проводит собственную оценку и решает изменить ТЗ, включив в него указанные детали.

1.7 Компетенции инженера по тестированию

Рассмотрим основные навыки и качества специалистов по тестированию, влияющие на качество и скорость работы над задачами.

Требования к инженерам по тестированию варьируются в зависимости от конкретной позиции в компании, уровня подготовки, специфики проекта и задач, принципов и ценностей компании и так далее. Однако представляется возможным консолидировать известные нам компетенции [28], которыми может обладать инженер по тестированию, в общем виде, и показать их значимость в процессе тестирования:

- Навыки ручного тестирования – основополагающий навык для специалиста по тестированию. К навыкам ручного тестирования относятся знания видов тестирования, техник тест-дизайна, принципов работы ПО. В зависимости от уровня навыка ручного тестирования специалист зависит скорость проведения тестов и широта тестового покрытия, что напрямую влияет на эффективность тестирования.
- Навыки автоматизации – навык специалиста работы с автоматизированными тестовыми сценариями и средствами автоматизации. Способность инженера работать с автотестами повышает эффективность его работы, так как даже если его задачи не включают в себя автоматизацию сценариев, возможность использовать средства автоматизации сокращают время на такие этапы тестирования как создание тестовой среды, тестовых данных и так далее [7], [45].
- Белый / черный ящик – способность инженера тестировать только по методу черного ящика, либо по методу серого или белого ящика влияет на ширину тестового покрытия. Специалист, имеющий навык чтения и понимая программного кода, способен отслеживать выполнение кода в процессе прохождения тестовых сценариев, что дает больше информации для верификации результатов и делает тестирование более чистым, то есть качественным.
- Знание продукта – навык знания продукта заключается в понимании специалиста сути работы продукта, который он тестирует. За счет интуитивного понимания принципов работы программы и потребностей

конечных пользователей специалист быстрее видит потенциальные сценарии для тестирования и эффективнее проводит валидацию дефектов [13]. Таким образом, высокий уровень знания продукта повышает скорость тестирования и широту тестового покрытия.

– Понимание процессов разработки – навык связанный с компетенцией «белый/черный ящик», заключающийся в наличии у инженера по тестированию знаний об этапах и процессах разработки кода, который передан на тестирование. Наличие знаний о принципах построения программного кода продукта позволяет быстрее определить потенциальные места скопления дефектов и ускоряет тестирование задачи [43].
– Навыки работы с документацией – умение специалиста работать с техническим заданием, тестировать требования, читать схемы, модели и алгоритмы, приводимые в технической документации к продукту. Навык повышает эффективность тестирования за счет возможности быстрее получить и понять информацию, переданную в ТЗ.
– Самостоятельность – навык, относящийся к «soft-skills» – навыкам, не относящимся непосредственно к профессиональной области. Умение самостоятельно распределить свою нагрузку, приоритизировать свои задачи, найти нужную информацию и так далее. Навык самостоятельность сокращает количество времени, которое уходит на коммуникации с руководством и получение инструкций.
– Навыки коммуникации - следующий soft-skills навык. Коммуникация в сфере разработки ПО крайне важна, так как в подавляющем большинстве случаев предметом вопроса являются сложные концепции и комплексные алгоритмы. Способность грамотно сформулировать вопрос и активно слушать собеседника позволяет получить нужную информацию и верно ее интерпретировать, что снижает риск совершения ошибок из-за недопонимания.

Итак, в первой главе были разобраны основы процессов тестирования программного обеспечения. В ходе исследования были изучены виды жизненного цикла программного обеспечения, организация команд, задействованных в разработке ПО и зоны ответственности участников проектной команды, а также инструменты для распределения ответственности. Был рассмотрен инструментарий управления процессом тестирования, проведен обзор основных TMS, используемых на сегодняшний день. Приведены виды и этапы тестирования, описана деятельность специалистов в различных процессах тестовой деятельности и необходимые для тестирования навыки, и их влияние на эффективность тестовой деятельности.

Глава 2 Процессы в управлении тестированием

2.1 Согласование стратегии тестирования с моделью ЖЦ ПО

Модель жизненного цикла разработки ПО зависит от множества факторов, таких как: направленность компании, специфика ПО, объем имеющихся ресурсов. В подавляющем большинстве случаев команды разработки и тестирования не могут повлиять на принятую модель ЖЦ, поэтому организация процессов тестирования должна проходить с учетом модели разработки ПО, принятого для проекта [21], [46].

Самой широко используемой моделью на сегодняшний день является итеративная модель [1] для больших проектов, и каскадная модель для небольших проектов, разрабатываемых за несколько месяцев [34]. Популярным способом организации процессов разработки и тестирования ПО в компаниях является применение итеративной модель с использованием методов Agile, иногда в связке с методами Kanban [33].

Итеративная модель предполагает декомпозицию задач и непрерывный анализ результатов с корректировкой предыдущих этапов работ. При организации процесса тестирования продукта, разрабатываемого с применением итеративной модели, нужно учитывать возможное изменение требований к продукту, непоследовательность работы, необходимость возвращаться к предыдущим этапам работы. В этом случае большое значение имеет правильное распределение задач между членами команды [1] и грамотное использование практик Канбан – доски для планирования, создания карточек для задач, планирования спринтов [38].

Для каскадной модели разработки ПО организация тестирования выглядит проще, так как эта модель изначально предполагает четкую последовательность этапов - во время процесса тестирования не будут меняться требования и не придется возвращаться к пройденным этапам. Основной задачей при управлении тестовой деятельностью в каскадной модели будет контроль подготовки к этапу

тестирования – отслеживание полноты и корректности документации, согласование сроков тестирования, организация распределение зон ответственности между членами команды тестирования [34].

Нужно понимать, что для большинство компаний реальный способ разработки продукта будет представлять из себя не регламентированную модель жизненного цикла, а помесь из существующих методов. Поэтому при организации процесса тестирования не существует единого стандарта, но можно выделить основные аспекты модели разработки, которые влияют на тестовую деятельность:

- Требования. На процесс тестирования влияет то, будут ли требования меняться, возможны ли дополнения в требованиях, утверждены ли к началу разработки все документы [6].
- Инструментарий работы. Для достижения наибольшей эффективности нужно рассмотреть возможности применения инструментов по управлению проектами в процессе тестирования [27].
- Состав команды тестирования. При управлении командой инженеров по тестированию важно понимание специфики сотрудников. Знания об индивидуальных навыках сотрудниках, их уникальном опыте, коммуникациях внутри команды может оказать позитивный эффект при распределении зон ответственности и задач внутри команды [28].
- Оценка принятой в проекте коммуникации. Принятая модель коммуникации оказывает непосредственный эффект на разработку. При невозможности повлиять на принятые правила, ответственный за процесс тестирования сотрудник должен отслеживать, что принятая модель передачи информации не снижает эффективность работы, не замедляет ее, не приводит к ошибкам в работе. Эффективность работы над задачей снижается, если инженер получает информацию в неполном или недоступном для понимания виде, с задержками по времени [28].

Как правило, модель жизненного цикла разработки программного обеспечения определяется руководством компании или потребностями заказчика, поэтому задача главы отдела тестирования – выстроить процессы в соответствии с принятым ЖЦ. Самые популярные модели разработки на сегодняшний день – Agile (гибкий подход, итеративная разработка) и Waterfall (предиктивная модель со строгим порядком стадий). В ходе производственной практики были изучены основные отличия организации процесса тестирования при разработке ПО по модели Agile и по модели Waterfall.

Модель Agile проще с точки зрения планирования процессов, так как она является более гибкой, и позволяет перестраивать планы по рабочим задачам в процессе итераций [1]. Например, если задача оказалась сложнее ожидаемого, ее можно декомпозировать на две подзадачи, выполнение одной из которых перенести в следующую итерацию. Для модели Waterfall планирование процессов – как раз одна из самых сложных задач в управлении. Требования и результаты каждого этапа работы не должны изменяться в процессе, поэтому в процессе организации тестирования крайне важно качественно проанализировать задачи и ресурсы для их выполнения, чтобы дать оценку необходимому для выполнения этапа времени и составить план работы над задачами для команды [34].

На стадии организации работы команды в процессе выполнения задач модель Waterfall становиться более удобной. Детальное планирование делает процесс работы более строгим и четким, что позволяет избежать недопонимания между коллегами, путаницы в последовательности выполнения задач. В случае применения модели Waterfall работа специалиста, занимающегося организацией процесса тестирования, в основном состоит из мониторинга результатов тестирования и оценки качества выполнения задач. При работе по модели Agile этот этап становиться более сложным: требования к итерации могут меняться, объем работы по задачам может оказаться выше прогнозируемого, а распределением задач по времени и сотрудникам руководитель команды

занимается уже в процессе этапа [31]. Всё это обеспечивает гибкость данной модели, но усложняет задачу по организации работы команды.

Третье основное отличие заключается в подходе к контролю качества тестирования. Если в разработке по Agile можно допустить недочеты, которые могут быть исправлены в следующей итерации, то в Waterfall качество разработки оценивается строго по окончанию этапа. Это значит, что работы по мониторингу выполнения задач и качеству тестирования должны выполняться более углубленно.

В таблице 1 приведены основные аспекты организации процесса тестирования для моделей Agile и Waterfall.

Таблица 1 – Организационные процессы в тестировании

Этап / Модель	Agile	Waterfall
Планирование	Требуется приблизительная оценка; процессы не детализируются.	Требуются детальные планы на все рабочие задачи, планы по распределению работы, оценка нужного времени.
Тестовая деятельность	Необходимо постоянное отслеживание процессов и обновления данных по итерации.	Проводится мониторинг выполнения задач сотрудниками.
Контроль качества	Проводится регрессионное тестирование при наличие возможности.	Регрессионное тестирование необходимо, в команде может быть организована перекрестная проверка сотрудниками работы коллег.

Различия в подходе к управлению процессами в случаях использования моделей Agile и Waterfall показывают важность анализа жизненного цикла разработки ПО, принятого для проекта, для эффективной организации процессов тестирования. При планировании работы команды специалистов по тестированию, руководитель в первую очередь должен проанализировать используемую в компании модель разработки и спланировать процессы в

соответствии с особенностями этой модели. Стоит учитывать, что компании могут использовать как классические модели, так и смешанные – в случае, когда для повышения эффективности разработки модели комбинируются между собой. Поэтому при анализе стоит применять технику рассмотрения всех аспектов итоговой модели по-отдельности, как сделано в таблице выше.

2.2 Организация коммуникаций в команде

В задачи руководителя отдела тестирования входит организация процессов по передачи информации внутри команды и между командами – например, с командой разработки. Основной формат коммуникации – собрания или онлайн-встречи, на которых присутствующие обсуждают текущие задачи.

На этапе организации коммуникаций руководитель производит анализ процессов, состава команды и рабочих задач для определения формата, регулярности и продолжительности встреч [4]. Так же он должен оценить установленный формат коммуникации со смежными командами и работать над повышением эффективности обмена данными по задачам.

Тим-лид команды тестирования единолично организует и управляет коммуникациями внутри команды. Речь об онлайн или оффлайн встречах, на которых присутствуют специалисты по тестированию и обсуждают рабочие задачи своей зоны ответственности. Основные цели внутрикомандных коммуникаций:

- Обмен опытом: специалисты обмениваются информацией о найденных ошибках в системе, узких местах, путях решения различных проблем. Таким образом распространяется ценная информация, повышающая скорость и качество тестирования.
- Синхронизация работа: задачи по тестированию могут быть взаимосвязаны, либо, наоборот, тестирование одной функциональности может блокировать тестирование другой. В таком случае важно, чтобы

специалисты имели возможности обсудить возможность работы сообща, не препятствуя тестированию задач, проводимому коллегами.

– Командная помощь: в случае, когда специалист сталкивается со сложностью в работе, высокий уровень коммуникации в команде может позволить ему быстро получить информационную поддержку и преодолеть появившиеся барьеры в выполнении задач.

Таким образом, качественная коммуникация в команде способна повысить скорость работы и качество итогового продукта, что напрямую влияет на эффективность работы.

Что нужно учитывать при организации коммуникации:

– Опыт сотрудников в команде – начинающим (junior) специалистам требуется больше онлайн или оффлайн встреч для качественного выполнения задач, в то время как с опытными (senior) тестировщиками иногда можно ограничиться перепиской в рабочем мессенджере.

– Распределение задач влияет на необходимость регулярных встреч команды. Четкое закрепление задач за сотрудником снижает количество необходимых коммуникаций, а разделение задач на части между сотрудниками, или работа с взаимосвязанными задачами повышает это количество [12].

По результатам анализа рабочих условий и процессов руководитель отдела тестирования может выстроить эффективную систему коммуникаций, что будет повышать качество процессов тестирования [50]. Приведенные выше аспекты являются первостепенными для исследования и сильнее всего влияют на итоговый план коммуникаций.

2.3 Организация коммуникации между командами

Помимо встреч команды тестирования, специалисты по тестированию также участвуют во встречах с сотрудниками смежных команд (аналитики, разработки и так далее). Основные виды встреч:

- Общая планерка – встреча всех специалистов, занятых на проекте. Обсуждение хода разработки, проблем, представление результатов.
- Консультация – встреча специалистов смежных команд, в ходе которой специалисты одной команды запрашивают некую информацию по подготовленным вопросам, а специалисты второй команды предоставляют запрошенную информацию.
- Демонстрация – встреча специалистов смежных команд, в ходе часть сотрудников демонстрируют остальным некий функционал и объясняют принципы его работы. Например, сотрудник команды аналитики, сотрудник команды разработки и сотрудник команды тестирования показывают своим коллегам готовую задачу, над которой они работали. Результатом такой встречи будет повышение уровня знаний о продукте у сотрудников, не задействованных в работе над разработанным функционалом.

Основные цели коммуникаций между командами:

- Демонстрация хода работы над задачами: команды разработки и аналитики на общих встречах дают специалистам по тестированию информацию о своей работе над задачами. Эта информация может быть полезна и повысить эффективность будущего тестирования. К примеру, информация о том, на основе какой части существующего программного кода пишется новая функциональность, дает инженеру по тестированию информацию о том, где могут появиться ошибки после интеграции новой функциональности.

– Передача знаний: получение знаний о работе новой функциональности от разработчиков или аналитиков. К примеру, специалист команды аналитики может устроить демонстрацию использования новой функциональности, по написанным им же сценариям. Это ускорит процесс изучения требований тестировщиком и таким образом повысит эффективность его работы.
– Демонстрация работы кода: в случае, когда при работе над задачей требуется проверки вида «серый ящик» или «белый ящик», целесообразно запросить демонстрацию от разработчика, работавшего с кодом, для экономии времени. Таким образом специалист по тестированию потратит меньше времени на поиск нужного фрагмента кода и задействованных методов и процедур.

Тим-лид команды тестирования не может единолично организовывать коммуникации с другими командами, но работает над этим вместе с тим-лидами смежных команд и с проджект-менеджером. При организации коммуникаций ему важно понимать степень необходимости в подобных коммуникациях для своей команды [15]. Для анализа этой потребности главным критерием является наличие у членов команды тестирования понимания смежных процессов – разработки, аналитики. В случае, когда члены команды тестирования имеют плохое представление о процессах смежных команд, имеет большое значение организация общих встреч с командами разработки и аналитики [6].

2.4 Методы мониторинга работы команды

Для понимания эффективности работы и оценки рисков руководителю отдела тестирования нужно проводить мониторинг процессов. Это позволяет оценить, насколько качественно выполняются задачи, соблюдаются ли сроки, справляется ли команда с нагрузкой. Качественный анализ выполнения задач может позволить руководителю быстро адаптировать стратегию работы команды для повышения эффективности работы [14].

Для оценки хода работы руководитель может использовать функционал TMS, систему баг-трекинга или ПО для управления проектами с функцией отслеживания рабочего времени (например Atlassian Jira) [48]. Подробнее о каждом способе:

– Использование TMS для контроля тестовой деятельности является самым удобным способом, так как инструменты, как правило, имеют функционал предоставления отчетов по выбранным данным. Таким образом руководитель команды имеет возможность мгновенно получить отчет с результатами работы, в которых можно увидеть данные по пройденным тест кейсам, изменениям в кейсах, созданным тестовым прогонам.
– Отслеживание работы в баг-трекинговой системе нужно в первую очередь для контроля корректности заведения багов сотрудниками, а также для оценки хода работы над задачей и прогнозирования времени, необходимого на повторную проверку.
– В некоторых системах управления проектам, например в Atlassian Jira, есть функция списывания рабочего времени. Сотрудник может указать, сколько времени у него заняла та или иная работа и оставить комментарий. Мониторинг этих данных позволяет руководителю оценивать работу каждого сотрудника по отдельности.

Выбор средств мониторинга и контроля выполнения задач руководитель производит за счет исследования рабочих процессов и данных о команде. Как видно из описания способов оценки работы, каждый из них имеет свои особенности. Анализ рабочих процессов дает руководителю возможность увидеть, какой функционал будет наиболее актуален для поддержания высокой эффективности процессов. Например:

– В случае работы с сотрудниками с небольшим опытом работы важно иметь возможность детально отслеживать работу каждого из них, что

делает актуальным мониторинг рабочего времени (например в Atlassian Jira) [48].

– В случае необходимости контролировать выполнение большого количества задач эффективных решением будет использование TMS с автоматизированной системой отчетности (например, TestRail).
– При работе над задачей с критичным уровнем важности логично осуществлять контроль в баг-трекере, для получения наиболее точной оценки выполнения задачи.

Проведение регулярного анализа рабочих процессов дает руководителю возможность адаптировать процессы мониторинга и контроля выполнения задач к изменяющимся условиям работы и в зависимости от состава команды.

2.5 Приоритизация и распределение задач

Каждая задача в проекте имеет множество реквизитов, среди основных указываются приоритет и исполнитель. Приоритет – степень важности задачи, исполнитель – сотрудник, выполняющий работу над задачей. Приоритет и исполнителя задач, входящих в тестовую деятельность, определяет руководитель команды. Распределение и приоритизация задач тим лидом влияет на эффективность работы, поэтому для руководителя целесообразно иметь стратегию анализа задачи для принятия этих решений [28].

Приоритет задачи зависит от ее сложности и срочности, но не существует единой формулы для определения приоритета, что усложняет оценивание задачи руководителем. Качество оценки приоритета влияет на последовательность выполнения задач сотрудниками и играет большую роль в вопросе сроков выполнения этапов проекта.

Способ определения приоритета различается в разных компаниях и определяют политикой компании. Далее приведены правила приоритизации в компании НВП Софтвэр СПб:

– Задачи по основной функциональности, имеющие связь с другими задачами, на работу по разработке которых требуется много времени получают приоритет «1» – наивысший приоритет. Эти задачи проверяются в первую очередь и распределяются равномерно по членам команды для экономии времени на их проверку.
– Приоритет «2» получаются задачи со сложной функциональностью, без связки с другими задачами. То есть ошибка в такой задаче не блокирует смежную функциональность.
– Задачи со средним уровнем сложности относятся к приоритету «3» или «4» в зависимости от объема работы по разработке.
– К приоритету «5» относятся самые простые, тривиальные задачи – изменения в пользовательском интерфейсе, текстовых шаблонах сообщений.

Правила приоритизации задач устанавливаются руководством компании, но, как правило, они строятся на срочности и сложности задачи.

Вторым важным решением руководителя является назначение задаче исполнителя. В рамках проекта этот процесс представляет из себя работу с набором задач, которые нужно распределить между сотрудниками, с учетом их компетенций, загруженности, личных навыков. Грамотное назначение задач позволяет равномерно и эффективно распределить рабочую нагрузку, что повышает эффективность и скорость работы.

Если в рамках приоритизации основную роль играет оценки самой задачи, то в рамках распределения задач руководителю также важно проанализировать имеющиеся ресурсы – компетенции сотрудников, их занятость. Рассмотрим важность понимания компетенций сотрудников для распределения задач с точки зрения видов тестирования:

– Доступ к коду: задачи различаются по необходимости работы с кодом в процессе тестирования. При проверке части задач инженеру может понадобиться отследить исполняемый код, найти экспортный сценарий,

пройти часть кода по шагам. Для этого специалисту важно наличие компетенции работы с кодом. Таким образом, руководителю важно иметь информацию о навыках сотрудниках работы с программным кодом, для возможности грамотного распределения задач, при тестировании которых требуется использование «серого» или «белого» ящика.

– Навыки автоматизации: при работе с некоторыми задачами инженерам важны умения работы со средствами автоматизации. Начиная от задач, для тестирования которых требуется большое количество тестовых данных, заканчивая задачами, включающими в себя масштабные тест-кейсы, которые целесообразно автоматизировать. При распределении таких задач руководитель основывается на информации о навыках сотрудников команды в сфере автоматизации.
– Знание продукта при регрессионном тестировании. В зависимости от того, насколько тот или иной сотрудник знает тестируемый продукт, на выполнение задачи по регрессионному тестированию потребуется разное время. Регрессионное тестирование предполагает проверку существующей функциональности, причем, как правило, включает в себя большое количество тестов. Руководителю целесообразно распределять основную нагрузку по регрессионному тестированию между сотрудниками, которые имеют высокий уровень знания продукта, так как в противном случае у специалистов уйдет большое количество времени на изучение существующей функциональности.

Таким образом, для эффективного планирования нагрузки, руководителю важно иметь подробную информацию о навыках и компетенциях своих сотрудников.

2.6 Решение инцидентов в процессе тестирования

В процессе тестирования у специалистов могут возникать проблемы и сложности. Это ситуации, в которых скорость и качество работы над задачей может быть снижено, из-за возникших у сотрудника препятствий [36]. Предупреждение возникновения таких проблем и их решение – прямая задача руководителя, реализация которой может положительно сказаться на эффективности процесса тестирования. Рассмотрим основные виды инцидентов в процессе тестирования [28]:

- Проблемы с получением информации для проведения тестирования – к таким проблемам относится неполное ТЗ, отсутствие ответов на вопросы к команде аналитики и разработки, проблемы с предоставлением необходимой документации по продукту, и так далее. Например, в ТЗ не дана информация по ожидаемому поведению системы в некотором случае, при этом специалист команды аналитики не может дать ответ на вопрос по ожидаемому поведению системы. Подобные инциденты снижают скорость работы из-за необходимости выделять время на поиск нужной информации.
- Недостаток личных навыков и компетенций – проблема возникающая в случае назначения сотруднику задачи, для решения которой ему не хватает технических навыков. Например, для тестирования задачи необходима проверка заполнения базы данных, но сотрудник не владеет навыком работы с базами данных. Подобный инцидент снизит темп работы сотрудника из-за необходимости запрашивать помощь, а также может снизить качество тестирования.
- Недостатки организации. К примеру, отсутствие организованных планерок, слишком долгая выдача доступов к системам, завышенные требования к документации, медленное решение организационных проблем компанией. Подобные инциденты вызывают простои в работе специалистов и снижают скорость работы над задачами.

Повышение эффективности процессов тестирования нацелено в том числе на решение подобных проблем специалистов. При работе руководителя команды над рабочими инцидентами, клиентом, заинтересованном в результате, является в первую очередь специалист по тестированию, чьи условия работы будут улучшены. На рисунке 6 показана карта пути клиента, когда клиент – специалист команды тестирования.

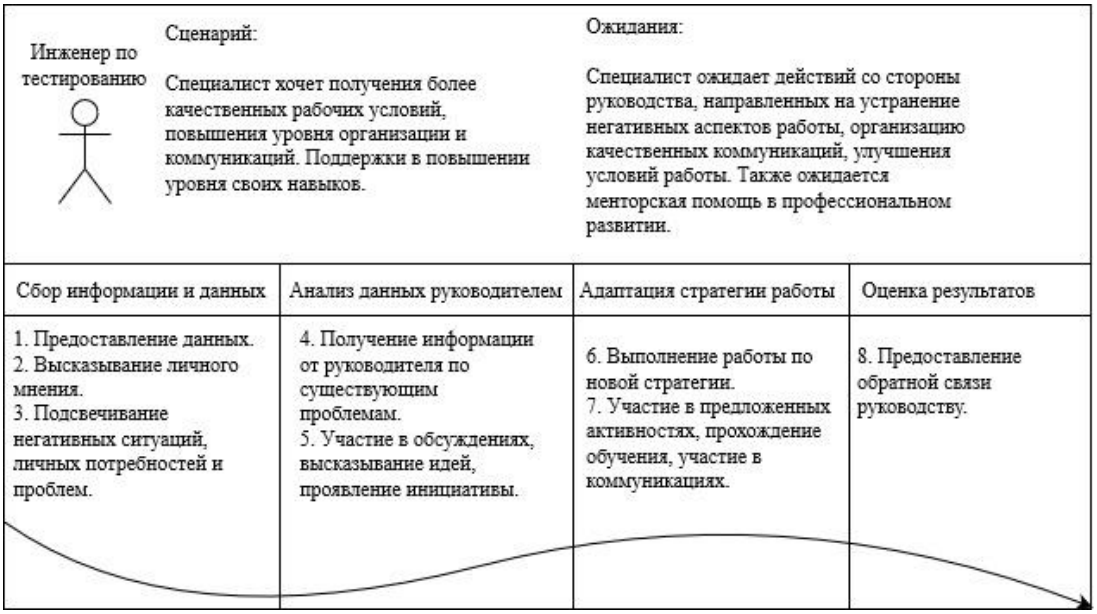

Рисунок 6 – Карта пути клиента – члена команды тестирования

На схеме продемонстрированы действия руководителя команды тестирования по предупреждению и решению возникающих в процессе тестирования проблем. Любые сложности, возникающие в процессе тестирования, снижают скорость тестирования и делают его менее качественным, что отражается на общей эффективности процесса разработки. В задачи руководства входит решение подобных проблем, то есть улучшение условий работы специалистов, для повышения эффективности процесса тестирования.

2.7 Метрики для оценки результатов тестирования

Для отслеживания эффективности работы существуют метрики тестирования [22]. Это количественные показатели, помогающие руководителям получить понимание об успехе работы и уровне достижения поставленных задач.

Основные используемые метрики:

– Пройденные/проваленные тесты – успешность прохождения тестов (насколько много ошибок было найдено). Метрика имеет процентное выражение и показывает итоговую успешность этапа тестирования. Может использоваться для оценки времени, необходимого на следующие этапы.

– Непройденные тест-кейсы – количество тестов, которые не удалось пройти – количественная метрика. Используется для выявления причин блокировки тестирования. Можно вычислять метрику как в абсолютном количестве заблокированных тест-кейсов, так и в процентом соотношении к общему количеству тест-кейсов, заявленных для прохождения. В таком случае метрика более показательна, позволяет акцентировать внимание на факторах, препятствующих проведению тестирования.

– Открытые/закрытые баги – показывает скорость устранения багов. Может использоваться для оценки скорости работы команды разработки или отдельных сотрудников.

– Переоткрытые/закрытые баги – метрика показывает успешность исправления багов разработчиками. Говорит о качестве работы команды разработки или отдельных сотрудников.

– Количество багов с указанием приоритета/серьезности – позволяет оценить общий объем и сложность кода, переданного на тестирование. Большое количество багов с высоким приоритетом показывает позволяет определить превышение оптимального объема кода,

находящегося в тестировании одновременно. Возможно, будет принято разделить задачу по тестированию на части.

- Эффективность тест-кейсов: «количество найденных дефектов / количество ТК» - показывает эффективность существующих тест-кейсов. Эта метрика связана с принципом «парадокс пестицида» из теории тестирования. Если тест-кейсы не обновляются, то их эффективность для нахождения ошибок постепенно снижается. Таким образом эта метрика позволяет увидеть низкую эффективность тест-кейсов и понять, что необходимо обновлять тестовые сценарии. Для тим-лида команды тестирования это так же может дать информации о компетентности сотрудников.
- Утечка дефектов – количество багов, найденных после релиза. Показывает, в первую очередь, компетентность тестировщика. «Исчерпывающее тестирование невозможно» - еще один принцип теории тестирования. Однако при сравнении разных сотрудников по метрике утечки дефектов можно увидеть, что для более опытных сотрудников эта метрика, как правило, будет ниже.
- Тестовое покрытие требований – оценка того, на сколько процентов написанные тест-кейсы покрывают техническое задание. Исчисляется в процентах на основании субъективной оценки. Метрика ценна для оценки качества написанной документации и компетентности сотрудника, создававшего её. Оценить данную метрику может как специалист по тестированию, так и по аналитике.

Указанные метрики могут помочь руководителю команды провести анализ факторов влияния: компетенций сотрудников, эффективности работы смежных команд, условий работы (например, метрики 2 и 5).

2.8 Управление эффективностью процессов тестирования

Исходя из материалов, представленных в предыдущих параграфах, можно сделать следующие выводы:

- Процесс тестирования является неотъемлемой частью процесса разработки ПО;
- Процесс тестирования менее автономен, чем процессы разработки и аналитики, и напрямую зависит от этих процессов;
- Процесс тестирования состоит из различных этапов и видов, включает в себя вспомогательные задачи (работа с документацией, коммуникация);
- Процесс тестирования может быть организован различными способами, руководитель команды тестирования имеет возможность по-разному распределять свои рабочие ресурсы и ресурсы команды;
- В зависимости от варианта принятой руководителем стратегии организации процесс тестирования может быть более или менее эффективно построен, то есть количество возникающих в ходе работы над задачами проблем может быть снижено за счет грамотной организации, темп работы увеличен, качество итогового продукта повышено.
- Основным фактором, влияющим на важные для заказчика показатели – скорость и качество разработки – является время, уделяемое инженерами работе над задачами непосредственно. Чем большее количество рабочего времени инженеры проводят за тестированием, тем быстрее будет темп реализации и тем выше будет качество продукта, получаемого заказчиком [28].

На рисунке 7 показаны 3 основных процесса, входящие в разработку ПО: аналитика, разработка, тестирование. Исходя из исследования, проведенного в предыдущих главах, выделены основные аспекты процесса тестирования, работа с которыми способна повысить эффективность тестовой деятельности, что

приведет к повышению эффективности разработки продукта в целом и соответствует бизнес-целям компании. Повышение эффективности процессов тестирования соответствует бизнес-целям ИТ-компаний, так как напрямую влияет на качество и скорость разработки продукта, являясь одним из трех основных процессов, входящих в разработку ПО [40].

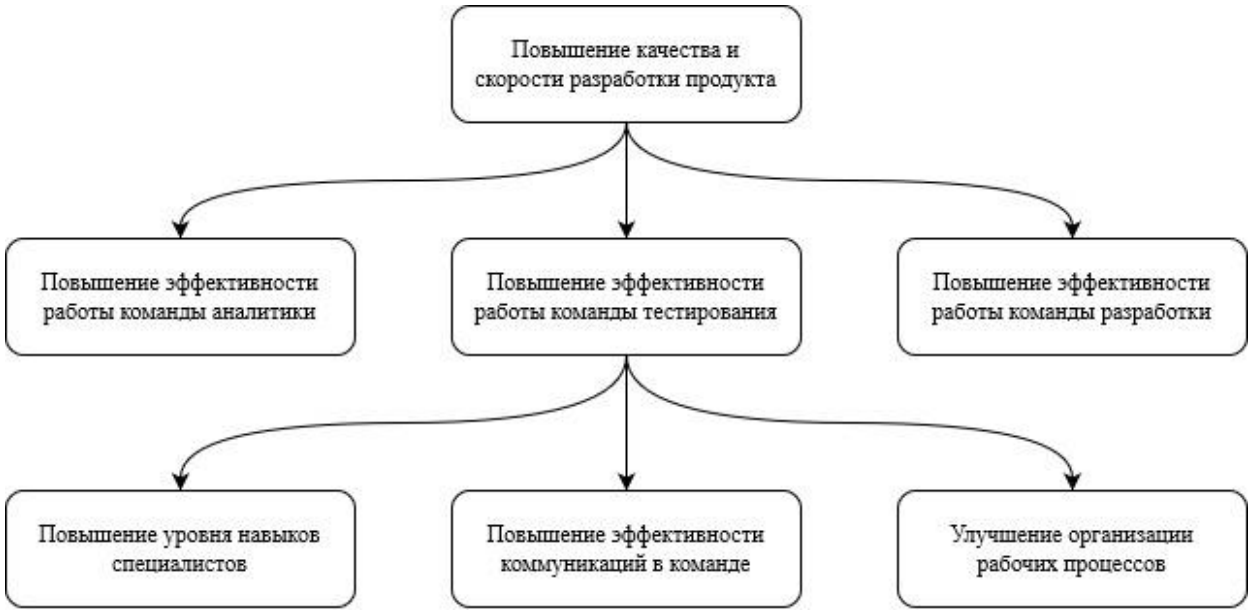

Рисунок 7 – Схема реализации бизнес-целей компании

В сфере IT-разработки существуют два основных процесса по повышению эффективности: ретроспектива и предиктивный анализ. Это методы анализа данных, используемые для достижения наибольшей эффективности процессов и выбору оптимальной стратегии работы [26]. Как правило, методом ретроспективы и предиктивного анализа (прогноза) пользуются в конце итерации, спринта или проекта. Результатом проведения ретроспективы является оценка работы команды за прошедший период, разбор сложностей, выделение существующих проблем и причин их возникновения. Далее, с помощью предиктивного анализа, определяется стратегия работы на следующий период, таким образом, чтобы достичь максимально возможной эффективности.

Ретроспективный анализ содержит следующие этапы:

– отбор и подготовка данных;
– анализ выбранных данных;
– интерпретация полученных результатов;
– подготовка выводов по результатам анализа.

Этапы предиктивного анализа:

– получение данных;
– подготовка и рассмотрение решений по влиянию на эффективность;
– выбор стратегии и создание плана действий;
– создание прогноза о результате действия выбранной стратегии.

Таким образом, использование методики предиктивного анализа данных применяется для поиска и выбора оптимальной стратегии действия в будущем, используя данные, полученных путем проведения ретроспективного анализа прошедшего периода.

Эти два метода предполагают сбор, обработку и анализ данных для выстраивания наиболее выгодной стратегии работы. Имеющиеся методики проведения ретроспективы и создания прогноза содержат информацию о необходимости сбора и обработки данных. Однако существующие источники не дают ответа на вопрос о том, откуда и какие именно данные брать для анализа, как проводить обработку данных, и какие решения рассматривать при создании прогноза. На сегодняшний день, как правило, эти методы анализа данных применяются с использованием личных компетенций и навыков руководителя команды [26].

Во второй главе было проведено исследование процессов, входящих в организацию и управление тестированием, которое показало важность применения методов анализа данных для принятия эффективных решений. Каждый рассмотренный процесс имеет множество вариантов реализации и для выбора наиболее выгодного из них необходима оценка разного рода информации: жизненного цикла разработки ПО, опыт работы и компетенции

сотрудников, количество задач. Были рассмотрены общие методы решения организационных задач, таких как организация коммуникаций и приоритизация задач, рассмотрены основные системы управления тестированием для управления командой и приведены метрики, позволяющие руководителю получить данные, актуальные для анализа и оценки. Была показана важность повышения эффективности процессов тестирования для достижения бизнес-целей компании и для разрешения и предупреждения проблем в работе сотрудников. Исследование методов анализа данных, используемых в управлении процессами разработки ПО, в том числе процессом тестирования, показало необходимость создания методик по сбору и обработке данных, используемых в анализе.

Глава 3 Разработка методических рекомендаций по управлению процессом тестирования

3.1 Определение факторов влияния

Для повышения эффективности процессов тестирования с помощью проведения ретроспективы необходима оценка факторов влияния. Фактор влияния – условие, параметр, характеристика, оказывающая влияние на процесс, и, соответственно, воздействующая на эффективность. Для определения потенциальных факторов влияния процесса тестирования была создана схема (приложение А - «Схема тестирования задачи», нотация ARIS). Схема представляет собой визуальное отображение стратегии работы над тестированием задачи, переданной специалисту. На схеме показаны события, процессы и исполнители, задействованные в процессе тестирования задачи.

Работа над задачей начинается с события «Получена задача на тестирование». Как правило, в момент получения задачи на тестирование специалист получает ТЗ – «Техническое задание», которое изучает в процессе «Изучение ТЗ, составление тест-плана». Первый и второй логические операторы XOR (1) и XOR (2) приводят нас к получению консультаций и демонстраций по задаче. В этих процессах принимают участие «Аналитик» и «Разработчик», и наступают они в случае наличия у специалиста по тестированию вопросов по ТЗ. Это может произойти из-за неполноты ТЗ, недостатка информации, противоречивых данных или недоступности ТЗ для понимания (например, при использовании аналитиком при написании ТЗ специфичных терминов, отсутствии нужных ссылок и разъяснений, отсутствии схем, диаграмм, референсов и так далее). Таким образом выделим первые факторы влияния:

- качество ТЗ;
- доступность ТЗ;
- коммуникация с командой аналитики;
- коммуникация с командой разработки.

После события «Составлен тест-план, создание тест-кейсов» (приложение А) мы попадаем в процесс «Тестирование задачи». Этот процесс цикличен: оператор XOR (3) приводит нас либо к найденному дефекту – «Обнаружены дефекты», либо к событию «Сборка протестирована» при отсутствии дефектов. То есть мы будем возвращаться в процесс «Тестирование задачи» до тех пор, пока в сборке будут находиться дефекты. Это влияет на время тестирования, поэтому выделим следующий фактор влияния:

– качество передаваемого кода.

В случае обнаружения дефектов мы оказываемся в процессе «Анализ дефектов». Анализ дефектов представляет собой исследование найденной особенности поведения системы. Процесс «Анализ дефектов» подробно показан на схеме (приложение Б - «Схема анализа дефектов», нотация ARIS). На схеме продемонстрированы основные процессы анализа дефектов: верификация (проверка на противоречие техническому заданию), валидация (проверка на соответствие потребностей пользователей, в случае, когда противоречия ТЗ не найдено) и создание запроса к заказчику (в случае, когда и верификация и валидация пройдены, но поведение системы не сочтено целесообразным и выгодным). На данной схеме также продемонстрировано участие специалистов команды аналитики и разработки в процессе тестирования, а также выделен аспект важности использования «Бизнес-требований» - общей документации по продукту, используемой в процессе валидации дефектов. Выделим следующий фактор влияния:

– наличие и полнота документации по продукту.

Следующий оператор XOR (4) представляет собой разветвление стратегии на два случая – а) дефект не прошел валидацию и/или верификацию и на него заведен баг-репорт, б) дефект прошел валидацию и верификацию, однако было решено задать вопрос заказчику для уточнения ожидаемого поведения системы. Сценарий б) приводит нас к процессу «Получение обратной связи от заказчика», что определяет следующий фактор влияния:

– получение обратной связи от заказчика.

К сожалению, не всегда заказчик реагирует на вопросы оперативно и предоставляет качественные, подробные ответы. Этот фактор влияния имеет потенциальное воздействие на скорость тестирования.

Оператор XOR (5) показывает разветвление сценария после ответа заказчика: найденный дефект может быть сочтен как ожидаемым поведением системы, так и противоречащим ожиданиям.

Начиная с события «Создан баг-репорт» можно увидеть участие исполнителя «Разработчик» в процессах «Работа с дефектом» и (опционально, после XOR (6)) «Участие в консультации». Событие «Участие в консультации» наступает в случае отсутствия у тестировщика возможности самостоятельной проверки исправлений. К примеру, если был налажен механизм формирования пакетов данных для отправки в мобильное приложение, которое еще не передано в тестирование, то специалисту по тестированию может понадобиться консультация разработчика по вопросу проверку содержания пакета данных. Необходимость технической консультации в первую очередь зависит от технических компетенций самого специалиста по тестированию. Также отметим, что XOR (7) возвращает нас к событию «Создан баг-репорт» в случае, когда исправления при проверке показали некорректное поведение системы, то есть когда исправления были не качественными, что говорит о низком качестве исправлений багов разработчиками и снижает скорость тестирования. Выделим следующие факторы влияния:

– технические компетенции инженера по тестированию;

– исправление багов разработчиками.

После события «Дефект устранен» и возвращения в процесс «Тестирование задачи», при отсутствии новых дефектов, мы попадаем в событие «Сборка протестирована» и процесс «Написание отчета о тестировании», который приводит нас к конечному событию «Написан отчет о тестировании».

Итак, найденные факторы влияния представлены в таблице 2.

Таблица 2 – Факторы влияния

Качество ТЗ
Доступность ТЗ
Коммуникация с командой аналитики
Коммуникация с командой разработки
Качество передаваемого кода
Наличие и полнота документации по продукту
Получение обратной связи от заказчика
Технические компетенции инженера по тестированию
Исправление багов разработчиками

В таблице приведены факторы влияния, определенные с помощью детализации процесса тестирования задачи. Именно эти факторы влияния нужно оценивать, чтобы получить данные для проведения ретроспективы и создания предиктивного прогноза.

3.2 Методика оценки факторов влияния

В качестве рекомендательной методики для руководителей команд тестирования создана методика оценки факторов влияния. Инструмент не требует специальных навыков оценивания и вычислений и предполагает субъективную оценку, либо оценку на основе существующих метрик тестирования, имеющихся результатов работы и так далее. Предложенная методика разработана в виде анкеты, которая содержит список факторов влияния, определенных в предыдущем параграфе. Оценивание предлагается проводить путем независимого анкетирования членов команды для оценки каждой из задач, над которой работал специалист, используя предложенную анкету оценки, представленную в таблице 3.

Таблица 3 – Анкета для оценки факторов влияния

Фактор влияния:	Оценка 1 - 5
Качество ТЗ	-
Доступность ТЗ	-
Коммуникация с командой аналитики	-
Коммуникация с командой разработки	-
Качество передаваемого кода	-
Наличие и полнота документации по продукту	-
Получение обратной связи от заказчика	-
Технические компетенции инженера по тестированию	-
Исправление багов разработчиками	-

Для удобства рекомендована следующая расшифровка оценок:

Оценка «5» (отлично) выбирается, если уровень реализации фактора оказался выше ожиданий специалиста (выше уровня принятых требований, стандартов). Например, если ТЗ было передано раньше срока.

Оценка «4» (хорошо) выбирается, если уровень реализации находится на уровне ожиданий. Например, если ТЗ было передано в согласованный срок и в полном объеме;

Оценка «3» (удовлетворительно) выбирается, если уровень реализации находится ниже ожиданий, но основные требования не нарушены (сроки работы, объем информации и так далее). К примеру, если ТЗ было передано вовремя, но часть информации была добавлена позднее, однако это не привело к опозданию.

Оценка «2» (неудовлетворительно) выбирается, когда реализация находится ниже ожиданий, нарушены основные требования (произошло опоздание, из-за неполноты информации была допущена ошибка, обеспечения не хватает для решения запланированной задачи). Например в случае, когда ТЗ было передано с опозданием, из-за чего были нарушены сроки тестирования.

Оценка «1» (плохо) выбирается, если произошло критическое нарушение реализации, такое как острый конфликт в команде, саботаж рабочих процессов,

не получение ответов на запросы и так далее. К примеру, в случае конфликта с аналитиков, разрабатывающим ТЗ, и его отказа вносить необходимые коррективы.

Шкала оценки не является фиксированной и может быть изменена, однако важно, чтобы метод оценивания был единым для всех опрашиваемых сотрудников и заранее оговорен, то бишь представлен команде тестирования перед анкетированием.

В таблице 4 представлен пример заполнения таблиц тремя сотрудниками.

Таблица 4 – Результаты оценивания факторов влияния сотрудниками

Фактор влияния:	Оценка 1 - 5					
Сотрудник, задача:	1	2.1	2.2	3.1	3.2	3.3
Качество ТЗ	4	4	4	4	4	5
Доступность ТЗ	3	4	3	3	4	4
Коммуникация с командой аналитики	5	4	4	4	4	5
Коммуникация с командой разработки	5	4	4	4	4	4
Качество передаваемого кода	4	5	4	4	4	3
Наличие и полнота документации по продукту	4	3	3	3	4	4
Получение обратной связи от заказчика	-	2	-	3	-	4
Технические компетенции	3	4	4	5	5	5
Исправление багов разработчиками	4	5	4	4	4	4

Расшифровка примера:

Сотрудник №1 – младший инженер по тестированию – в течение последней итерации работал над одной задачей и столкнулся со сложностями в понимании ТЗ и пробелами в своих навыках. При этом коммуникация с командами аналитики и разработки по вопросам, возникающим в процессе тестирования, была на отличном уровне. Коммуникации с заказчиком у сотрудника не было.

Сотрудник №2 – инженер по тестированию – в течение последней итерации работал с двумя задачами. Первая полученная задача была передана с

качественным ТЗ и отличным кодом с минимум дефектов. Однако в процессе тестирования, при анализе найденных особенностей поведения системы, появилась необходимость в получении дополнительных данных по продукту. Сотрудник остался недоволен документацией по продукту, так как в ней не содержалось нужной ему информации. Обратная связь от заказчика также оказалась оценена низко, так как после создания запроса на получение дополнительной информации заказчик сильно задержал ответ, из-за чего произошло опоздание по срокам выполнения задачи. В процессе работы над второй задачей сотрудник встретился со сложностями в понимании ТЗ, которое содержало неизвестную ему терминологию. При попытке обратиться к документации по продукту, указанные термины не были найдены.

Сотрудник №3 – старший инженер по тестированию - в течение последней итерации тестировал 3 задачи. При изучении ТЗ по первой из них возникла сложность с исследованием задействованных бизнес-процессов, так как в ТЗ не были включены схематичные описания. Документация по продукту также не содержала их. Для получения необходимой информации был составлен вопрос к заказчику, однако ответ пришел недостаточно быстро. В процессе работы с третьей задачей специалист столкнулся с получением на тестирование кода низкого качества. В самом начале тестирования были обнаружены блокирующие работу баги, функциональность была частично не доделана. Фактор «Исправление багов разработчиками» также был оценен низко, так как большая часть багов после исправлений все еще показывала некорректную работу, а исправления вносились медленно.

После получения результатов анкетирования от членов команды по их задачам руководитель команды тестирования собирает результаты оценок и вычисляет усредненные результаты, путем нахождения среднего арифметического. Итоговые результаты оценки показаны в таблице 5.

Таблица 5 – Усредненные оценки факторов влияния

Фактор влияния:	Оценка
Качество ТЗ	4,2
Доступность ТЗ	3,5
Коммуникация с командой аналитики	4,3
Коммуникация с командой разработки	4,2
Качество передаваемого кода	4
Наличие и полнота документации по продукту	3,5
Получение обратной связи от заказчика	3
Технические компетенции инженера по тестированию	4,3
Исправление багов разработчиками	4,2

Анализ результатов показал что основные зоны повышения эффективности, это факторы влияния «Доступность ТЗ», «Наличие и полнота документации по продукту» и «Получение обратной связи от заказчика». С учетом известных нам деталей проблем, возникших у специалистов в процессе работы над задачами, можно порекомендовать следующие методы реагирования:

Для повышение показателя фактора «Доступность ТЗ» возможным представляется составить внешний запрос к руководителю проекта для установки диалога с командой аналитики и обсуждения изменений в шаблонной форме ТЗ. Для повышение показателей факторов «Наличие и полнота документации по продукту» и «Получение обратной связи от заказчика» рекомендуется составить внешний запрос к руководству компании.

3.3 Методика оценки компетенций сотрудников

В контексте проведения ретроспективы и для создания качественного прогноза руководителю команды важно понимать возможности своей команды. С целью повышения эффективности работы важно также отслеживать

профессиональное развитие и рост членов команды, выявлять их сильные и слабые стороны в работе.

В компаниях часто используется система грейдинга сотрудников – распределения специалистов по рангам, должностям. Также существует популярная система оценки ИТ-специалистов по уровню подготовленности: Junior (начинающий, стажер), Middle (специалист со средним уровнем опыта) и Senior (опытный специалист). Для детальной оценки компетенций удобно использовать матрицы – перечни навыков с методикой оценивания [28].

В рамках создания рекомендаций по анализу данных для руководителей команд тестирования разработана матричная система грейдинга специалистов по тестированию [38]. Предложенная система (Приложение В) предполагает кастомизацию в зависимости от специфики работы компании и специфики продукта. В таблице показан способ оценки специалиста по компетенциям направления тестирования, а также дана рекомендация по добавлению специфических показателей в матрицу.

Как можно заметить, для оценки приведенных факторов предлагаются 3 основных способа: по итогам работы, в результате интервью, основываясь на отзывах коллег. В большинстве компаний есть принятая система оценки персонала, в рамках которой установлены методы оценки сотрудников. Вышеперечисленные методы являются именно теми, которые используют для оценки эффективности специалистов, занятых в командах по разработке ИТ-продуктов [14]. Руководителю команды тестирования в масштабах команды предлагается использовать эту схему, так как она успела хорошо себя зарекомендовать [25]. Проводить оценку сотрудников можно как в рамках централизованной оценки персонала в компании, так и для личного использования в ходе планирования работы команды. Классический пример проведения оценки:

– Консолидация данных о задачах сотрудника. В рамках данного этапа удобно использовать таск-трекер, например Jira. Руководитель

обозревает задачи сотрудника за оцениваемый период, получает подробную детализацию работы над задачей с помощью используемой в компании TMS (Test Management System).

- Оценка выполнения задач руководителем. Оценка технической сложности, сроков выполнения, тестового покрытия, возникших в ходе работы проблем, пропущенных ошибок. Сюда так же можно отнести качество написанных автотестов, ручных тестов. Получив вышеперечисленные данные из TMS, руководитель может дать оценку эффективности работы сотрудника по различным параметрам.
- Запрос на обратную связь от коллег из смежных команд: получение устного или письменного отзыва на работу сотрудника от специалистов по разработке, аналитике, дизайну. Руководитель получает информацию о коммуникациях специалиста. К примеру о том, как часто сотрудник запрашивает технические консультации, насколько подробно описывает найденные ошибки в баг-трекере, быстро ли откликается на запросы, предоставляет ли запрашиваемую информацию в полном объеме.
- Проведение интервью с сотрудником. Этап аналогичен собеседованию: руководитель получает обратную связь лично от сотрудника, об успехах и сложностях в работе, обсуждает с ним результаты своей личной оценки задач сотрудника и отзывы от коллег. Рассматривает выявленные на этапе 2 слабые места сотрудника и получает информацию о причинах проблем в работе над задачами.
- На основании пройденных этапов заполняется матрица компетенций и руководитель принимает решение о позиции сотрудника, а также, совместно с сотрудником, строит планы по индивидуальному развитию, на основании интересов сотрудника и его слабых мест.

Рекомендации по добавлению показателей в матрицу компетенций: в процессе работы над задачами и мониторинга выполнения работы сотрудниками руководителю предлагается отслеживать сложности, возникающие в работе

специалистов, увеличивающие затраты времени на задачу, требующие помощи или консультации, снижающие качество выполнения задачи. На основании собранных данных выделять общие паттерны – шаблоны ситуаций, влияющих на эффективность работы сотрудника.

Пример: при анализе проблемных задач руководитель замечается повторяющийся паттерн – в числе задач с низким качеством выполнения есть несколько, для работы над которыми требовались знания HTML/CSS. Это будет поводом внести в матрицу компетенций этот навык.

Разработанная матрица удобна в использовании для оценивания компетенций сотрудников и отслеживания их развития. Информация, полученная с помощью использования предложенной матрицы компетенций, позволит руководителю команды тестирования получить данные, необходимые для ретроспективного и предиктивного анализа.

3.4 Использование метрик для проведения ретроспективы

Для оценки результатов работы команды тестирования существует множество метрик. В процессе планирования процессов метрики нужны для оценки используемой стратегии и её изменения с целью повышения эффективности. Показатели метрик могут быть полезны для использования в ходе проведения ретроспективы и для создания будущей стратегии работы в ходе проведения предиктивного анализа. На основании существующих метрик [30], [44] разработаны авторские метрики для оценки эффективности используемой стратегии тестирования:

– Метрика «Время коммуникаций». Метрика, показывающая, какое количество времени у специалистов ушло на коммуникации. Для формирования метрики нужно разделить время, потраченное специалистами по тестированию на получение консультаций и обсуждение задач с коллегами, на полное время их работы. Метрику удобно выражать в процентах. К примеру, если метрика равна 20%, это

значит что из всего времени работы 20% времени ушло на обсуждение задач с аналитиками, разработчиками, дизайнерами и так далее. Метрика может показать как внешние, так и внутренние проблемы. К примеру, если показатель метрики 50%, то есть половину времени работы над задачами сотрудники потратили на обсуждение рабочих вопросов, это может быть поводом для обращения к руководству (например, если проблема в некачественном ТЗ) или для введения общих консультаций с командой аналитики.

– Метрика «Сложность задач» - процент задач, по которым специалистам была необходима техническая консультация. Это отношение количества задач, по которым сотруднику понадобилась техническая помощь, к общему количеству задач сотрудника в периоде. Метрика может помочь руководителю выделить слабые зоны: при применение метрики для всей команды есть возможность оценить общий уровень технической компетенции в контексте поставляемых задач, и, возможно, подать запрос руководству на расширение штата и найм нового сотрудника. При применения метрики для отдельных сотрудников она может указать на технические пробелы специалиста, в случае чего руководитель может согласовать дополнительное обучение и временно скорректировать перечень задач сотрудника.
– Метрика «Время тестирования». Метрика показывает, какой процент времени работы над задачей специалист потратил непосредственно на тестирование функционала – проведение тестов, обнаружение дефектов, проверку исправлений ошибок. Для расчета этой метрики нужно разделить время, потраченное на проведение тестов, на общее время работы над задачей. Результат расчета отображает соотношение затрат времени на работу над различными вспомогательными этапами задачи, такими как: изучение ТЗ, написание тест-кейсов, создание баг-репортов, написание отчета о тестировании, к, непосредственно, тестированию.

Использование этой метрики может указать на утечки времени сотрудника в работе и на слабые места в бизнес-процессах компании. К примеру, на слишком высокие требования к документации, что приводит к трате большого количества времени на оформление документов.

Ключевым элементом метрик являются временные затраты [5]. Наиболее эффективное тестирование достигается, когда максимально возможное количество рабочего времени уходит именно на тестовую деятельность, а не на вспомогательные процессы, такие как получение консультаций, коммуникации, работы с документацией [28]. Таким образом, с использованием предложенных метрик в процессе ретроспективного анализа возможно оценить, насколько эффективной была работа над задачами в оцениваемом периоде.

Для расчета метрик удобно использовать инструменты менеджмента проектов, заточенные под задачи тестирования. Самым удобными вариантом для получения данных для расчетов можно назвать комбинацию инструментов Jira + TestRail. Эти инструменты являются популярными среди компаний в том числе за возможность предоставления статистических данных [28]. Использование Jira позволяет отслеживать временные затраты по разным задачам и временные затраты на различные этапы работы внутри одной задачи [48], в то время как с помощью TestRail можно оценить время, затраченное на само тестирование с помощью таймера прохождения тест-кейсов.

Рисунок 8 показывает пример подсчета времени с помощью журнала работ в системе Attlassian Jira.

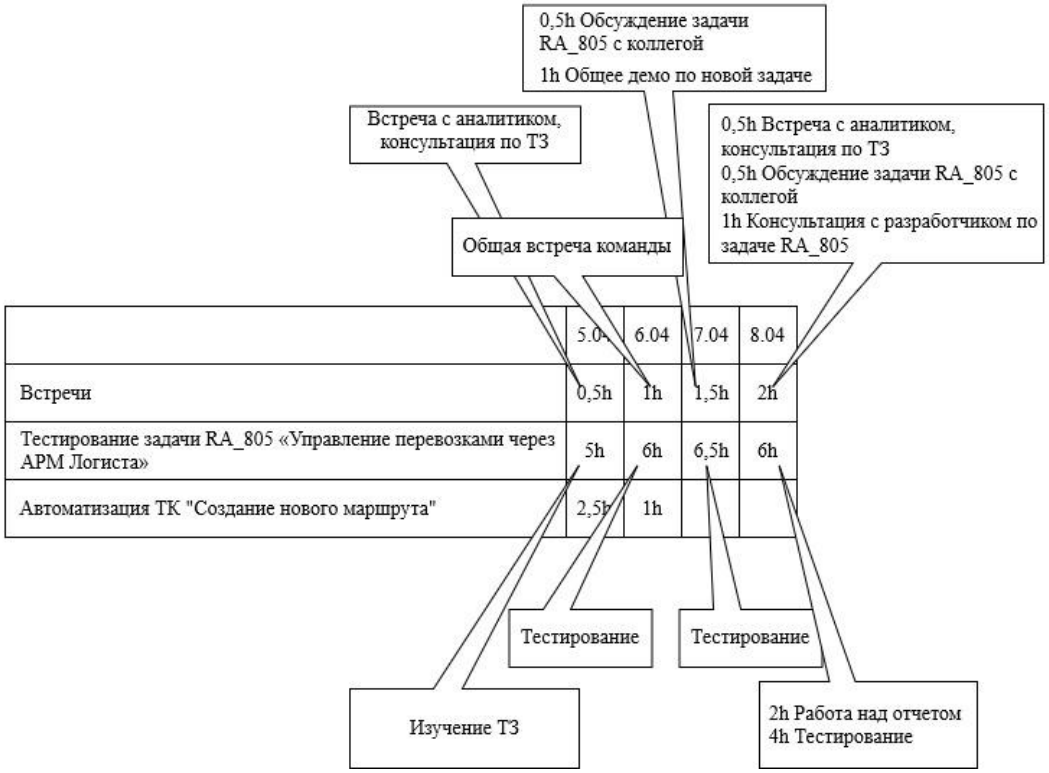

Рисунок 8 – Фрагмент журнала работ в сервисе Attlassian Jira

Как видно из рисунка, журнал работ в Jira позволяет сотрудникам разделять временные затраты по рабочим задачам, а также оставлять пометки о конкретной активности. Например, руководитель, используя общий журнал работ команды, может увидеть, что 8 апреля сотрудник потратил 2 часа на коммуникации и 6 часов на работу над задачей, из которых 4 часа заняло само тестирование, и еще 2 часа – работа над отчетом. Эти данные позволят руководителю сделать расчеты и получить показатели метрик за выбранный период.

Перечисленные метрики целесообразно использовать при регулярном мониторинге работы команды тестирования в процессе проведения ретроспективы. Анализ показателей эффективности работы команды, полученных путем формирования метрик, позволит руководителю провести

качественный предиктивный анализ и адаптировать принятую стратегию тестирования для более эффективной работы.

Исходя из данных исследования процессов тестирования и процессов в управлении тестированием собраны основные рекомендации, которые целесообразно рассмотреть в ходе проведения предиктивного анализа для улучшений показателей метрик:

Для улучшения показателя метрики «Время коммуникаций» предлагается рассмотреть следующие решения:

- сокращение количества личных консультаций путем организации общих встреч;
- повышение уровня компетенции сотрудников в смежных процессах разработки и аналитики путем обучения;
- работа над повышением качества передаваемой документации.

При работе над показателем метрики «Сложность задач» возможны следующие решения:

- повышение уровня технических компетенций и навыков сотрудников путем обучения;
- найм новых сотрудников;
- пересмотр системы распределения задач для более корректного распределения нагрузки, пересмотр системы приоритизации задач;
- проведение организованных технических демонстраций для повышения темпа ознакомления специалистов с новым функционалом.

Улучшение показателя метрики «Время тестирования» возможно с помощью принятия следующих мер:

- снижение временных затрат на работу с документацией за счет реорганизации системы документооборота и отчетности;
- снижение временных затрат на работу с ТЗ путем пересмотра шаблонной формы ТЗ и диалога с командой аналитики для

выявления причин возникновения сложностей у команды тестирования.

В результате исследования и анализа факторов влияния, методов их оценки и способов использования результатов анализа для повышения эффективности рабочих процессов были разработаны рекомендации для руководителей. Проведен обзор основных факторов влияния в процессе тестирования ПО и предложены инструменты для их оценивания. Для получения численных показателей эффективности процессов были разработаны метрики и рекомендации по использованию показателей метрик. Разработанный авторские рекомендации предполагают использование в ходе ретроспективного и предиктивного анализа данных, проводимого для повышения эффективности работы.

Глава 4 Апробация предложенных методик

4.1 Проведение ретроспективы

В рамках научного исследования была проведена апробация предложенных методик. Разработанные рекомендации были применены во время работы в команде тестирования, в ходе проведения регулярной ретроспективы и для работы над предиктивным анализом. После периода апробации был проведен ретроспективный анализ эффективности работы команды за весь период применения методик.

Для проведения эксперимента был выбран период с 1 июня до 1 октября 2024 года (4 месяца). За это время продуктовая команда провела 4 релиза, каждый из которых содержал достаточное для оценивания количество задач.

Эксперимент предполагает, что результаты оценивания эффективности работы команды в конце периода апробации покажут целесообразность предложенных автором методик.

Для получения данных, нужных для проведения ретроспективы были использованы разработанные таблицы факторов влияния и метрики. В начале периода апробации, а также после каждого из релизов проводилось анкетирование команды по методу оценивания факторов влияния (заполнение таблиц) и расчет метрик тим-лидом команды тестирования.

Анкетирование проводилось путем независимого заполнения анкеты «Факторы влияния» членами команды тестирования. После получения результатов руководитель рассчитывал среднюю оценку по каждому оцененному фактору. Пример расчета оценки приведен в параграфе 3.2, в процессе апробации применялся описанный метод для всех членов команды (каждый сотрудник заполняет анкету на каждую из своих задач в итерации, после этого результаты усредняются путем вычисления среднего арифметического).

Периоды проведения ретроспективы:

– до начала апробации – анализ данных проведен к 1 июня;

– период с 1 июня по 5 июля – «Июнь»;

– период с 6 июля по 2 августа – «Июль»;

– период с 3 августа по 30 августа – «Август»;

– период с 31 августа по 1 октября – «Сентябрь».

После проведения анкетирования в июне были выделены усредненные результаты, представленные в таблице 6.

Таблица 6 – Результаты оценок факторов влияния до начала апробации

Фактор влияния:	Оценка 1 - 5
Качество ТЗ	4,5
Доступность ТЗ	3
Коммуникация с командой аналитики	3,5
Коммуникация с командой разработки	4
Качество передаваемого кода	4,5
Наличие и полнота документации по продукту	4
Получение обратной связи от заказчика	3
Технические компетенции	3,5
Исправление багов разработчиками	4,5

Расчет метрик за две прошедшие итерации (апрель и май) в июне показал следующие результаты:

– «Время коммуникаций» – 20%.

– «Сложность задач» – 13 из 22 задач (59%).

– «Время тестирования» - 50%.

Из результатов анализа факторов влияния и показателей метрик можно сделать вывод, что текущая стратегия работы имеет низкую эффективность в аспектах коммуникации с командой аналитики, доступности ТЗ, количества времени, уходящего вспомогательные процессы, технических компетенций членов команды тестирования.

4.2 Предиктивный анализ

В ходе предиктивного анализа были рассмотрены авторские рекомендации по улучшению показателей метрик и оценок факторов влияния. Для повышения эффективности стратегии были рассмотрены следующие решения:

- Организация общих консультаций по новому функционалу с участием команды аналитики – данная мера должна снизить необходимость технических консультаций, количество времени, уходящее на изучение ТЗ, так как изучение ТЗ является вспомогательным процессом. Эта меры должна улучшить показатель оценки фактора влияния «Коммуникация с командой аналитики» и показатель мертики «Время тестирования».
- Изменение шаблонной формы отчетности – предполагается упрощение процесса работы над отчетностью для снижения времени на создание отчетной документации, так как этот процесс является вспомогательным. Эта мера должна улучшить показатель метрики «Время тестирования».
- Согласование обучения для сотрудников – в процессе анализа рабочих задач сотрудников были выявлены технические пробелы, из-за которых специалистам требовались консультации. Данная мера предполагает кратковременное обучение специалистов для закрытия технических пробелов. Эта мера должна улучшить показатели метрики «Сложность задач» и фактора влияния «Технические компетенции».

Помимо этого были выбраны решения по реагированию на негативные факторы влияния:

- Тим-лидом команды был составлен запрос к руководству по проблеме с получением обратной связи от заказчика.
- Была поставлена задача по реорганизации шаблона технического задания для повышения уровня доступности ТЗ.

Таким образом был проведен предиктивный анализ, результатом которого является детализация изменений стратегии работы на будущий период и предположение о положительной тенденции в эффективности работы команды. Предполагается, что выбранные решения повысят показатели оценки и приведут к повышению общей эффективности процессов.

4.3 Адаптация стратегии

Для отслеживания эффективности принятых мер была проведена оценка промежуточных результатов. Первые промежуточные результаты, представленные в таблице 7, были получены в июле за период «Июнь».

Таблица 7 – Результаты оценок факторов влияния за период «Июнь»

Фактор влияния:	Оценка 1 – 5
Качество ТЗ	4,5
Доступность ТЗ	3
Коммуникация с командой аналитики	4
Коммуникация с командой разработки	4
Качество передаваемого кода	4,5
Наличие и полнота документации по продукту	4
Получение обратной связи от заказчика	3
Технические компетенции	3,5
Исправление багов разработчиками	4,5

Результат расчета метрик за период «Июнь»:

– «Время коммуникаций» – 17%.

– «Сложность задач» – 5 из 12 задач (42%).

– «Время тестирования» – 60%.

Как видно из результатов анкетирования, после введения мер по адаптации стратегии, изменилась оценка фактора влияния «Коммуникация с командой аналитики», что связано с быстрым внедрением решения по организации общих встреч для обсуждения нового функционала. Результат показывает эффективность данной меры. Однако остальные факторы влияния на текущем этапе не изменили своей оценки, в первую очередь из-за невысокого темпа принятия организационных решений руководством.

Показатели метрик также демонстрируют позитивные результаты введения практики встреч командой аналитики – снизились показатели времени, затрачиваемого на вспомогательные процессы и коммуникацию, и также количество задач, по которым специалисты консультировались.

В августе были собраны и проанализированы показатели за период «Июль», результаты представлены в таблице 8.

Таблица 8 – Результаты оценок факторов влияния за период «Июль»

Фактор влияния:	Оценка 1 – 5
Качество ТЗ	4,5
Доступность ТЗ	3
Коммуникация с командой аналитики	4
Коммуникация с командой разработки	4
Качество передаваемого кода	4,5
Наличие и полнота документации по продукту	4
Получение обратной связи от заказчика	4,5
Технические компетенции	3,5
Исправление багов разработчиками	4,5

Результат расчета метрик за период «Июль»:

– «Время коммуникаций» – 15%.

– «Сложность задач» – 4 из 11 задач (36%).

– «Время тестирования» – 60%.

В июне и июле были проведены встречи с заказчиком, на которых обсуждались проблемы коммуникации. Организация обратной связи была пересмотрена в сторону большей эффективности, что сразу отразилось на результатах анкетирования сотрудников.

Показатели метрик также демонстрируют позитивные результаты введения практики встреч со смежными командами – снизился показатель времени, затрачиваемого на вспомогательные процессы, а также количество задач, по которым специалисты консультировались.

В таблице 9 представлены показатели за период «Август».

Таблица 9 – Результаты оценок факторов влияния за период «Август»

Фактор влияния:	Оценка 1 – 5
Качество ТЗ	4,5
Доступность ТЗ	4
Коммуникация с командой аналитики	4,5
Коммуникация с командой разработки	4
Качество передаваемого кода	4,5
Наличие и полнота документации по продукту	4
Получение обратной связи от заказчика	4,5
Технические компетенции	4
Исправление багов разработчиками	4,5

Результат расчета метрик за период «Август»:

– «Время коммуникаций» – 20%.

– «Сложность задач» – 3 из 15 задач (20%).

– «Время тестирования» – 70%.

В июле совместно с командой аналитики была изменена шаблонная форма ТЗ, благодаря чему оно стало доступнее для понимания специалистов, что демонстрирует улучшенная оценка фактора влияния «Доступность ТЗ». Также можно выделить улучшения показателя фактора влияния «Коммуникация с командой аналитики», достигнутые за счет организованных встреч.

Изменения также отразились на метрике «Время тестирования» – благодаря реорганизации процесса создания ТЗ снизилось время на изучение документации, за счет чего количество времени, затрачиваемое на тестирование непосредственно, увеличилось.

В начале октября были собраны и проанализированы финальные результаты адаптации стратегии за экспериментальный период. Результаты представлены в таблице 10.

Таблица 10 – Результаты оценок факторов влияния за период «Сентябрь»

Фактор влияния:	Оценка 1 – 5
Качество ТЗ	4,5
Доступность ТЗ	4
Коммуникация с командой аналитики	4,5
Коммуникация с командой разработки	4
Качество передаваемого кода	4,5
Наличие и полнота документации по продукту	4
Получение обратной связи от заказчика	4,5
Технические компетенции	4,5
Исправление багов разработчиками	4,5

Расчет метрик за период «Сентябрь» показал следующие результаты:

– «Время коммуникаций» – 10%.
– «Сложность задач» – 2 из 12 задач (17%).
– «Время тестирования» – 80%.

Улучшение показателя фактора влияния «Технические компетенции» за последние две итерации было достигнуто за счет организации обучения для сотрудников.

Как видно из показателей метрик, улучшилась метрика «Время тестирования» за счет снижения количества времени, уходящего на создание отчетности и изучение ТЗ. Это следствие реорганизации шаблона технического задания, которое было запланировано в июне, и работы с командой аналитики по повышению уровня доступности ТЗ.

4.4 Анализ результатов

Оценка успешности адаптации стратегии проводилась с использованием предложенных метрик. После начала реализации выбранных решений ретроспектива проходила четыре раза: в начале июля, в начале и в конце августа, в начале сентября и октября. Адаптация стратегии проходила постепенно и по полученным промежуточным результатам можно отследить эффективность принятых мер и решений. В таблице 11 рассмотрены окончательные результаты с описанием реализованных за период решений.

Таблица 11 – Начальные и окончательные оценки факторов влияния

Фактор влияния:	Оценка 1 - 5	
	Июнь	Октябрь
Качество ТЗ	4,5	4,5
Доступность ТЗ	3	4
Коммуникация с командой аналитики	3,5	4,5
Коммуникация с командой разработки	4	4
Качество передаваемого кода	4,5	4,5
Наличие и полнота документации по продукту	4	4
Получение обратной связи от заказчика	3	4,5
Технические компетенции	3,5	4,5
Исправление багов разработчиками	4,5	4,5

Также в процессе адаптации стратегии были улучшены метрики эффективности работы, представленные в таблице 12.

Таблица 12 – Начальные и окончательные показатели метрик

Метрика:	Начальная оценка	Финальная оценка
«Время коммуникаций»	20%	10%
«Сложность задач»	42%	17%
«Время тестирования»	50%	80%

Как было упомянуто в параграфе 3.5, временные затраты являются ключевым элементом эффективности стратегии тестирования. Если в начале периода апробации из 100% времени работы над задачей специалисты занимались непосредственно тестированием лишь половину этого времени, то к концу периода этот показатель достиг 80% времени, что говорит о возможности более тщательного и качественного тестирования, повышения темпа работы в целом. Помимо этого, можно увидеть, что снизилось количество рабочего времени, уходящего на задачи по коммуникации, такие как коммуникации со смежными командами и участие в консультациях, что говорит о повышении количества времени работы над основными задачами, а также показывает положительное влияние и результативность мер, принятых в рамках адаптации стратегии тестирования.

Улучшение показателя метрики «Сложность задач» демонстрирует повышение технических возможностей специалистов по тестированию, достигнутое путем повышения технических навыков и улучшения коммуникации со смежными командами.

Принятые за период адаптации решения и результаты их реализации:

– Организация общих консультаций по новому функционалу с участием команды аналитики – решение было реализовано в первую итерацию периода апробации, первые промежуточные результаты показали

действенность принятой меры – показатель фактора влияния «Коммуникация с командой аналитики» был улучшен с 3,5 до 4 баллов.

- Изменение шаблонной формы отчетности – показатель метрики «Время тестирования» был улучшен с 50% до 80%.
- Согласование обучения для сотрудников – показатель фактора «Технические компетенции» был повышен с 3.5 до 4.5 баллов, что можно отследить по промежуточным результатам. Именно в этот период было проведено корпоративное обучение для сотрудников.
- Создание запроса к руководству по проблеме с получением обратной связи от заказчика – оценка фактора «Получение обратной связи от заказчика» была улучшена с 3 до 4.5 баллов за счет решения проблемы с медленным реагированием на запросы со стороны заказчика.
- Реорганизация шаблона технического задания для повышения уровня доступности ТЗ – промежуточные результаты показателей за «Август» показали действенность меры, показатель фактора влияния «Доступность ТЗ» был повышен с 3 до 4 баллов.

В параграфе была рассмотрена проведенная апробация. Описаны основные этапы: ретроспективный анализ, предиктивный анализ, адаптация стратегии по авторским рекомендация. Проведен анализ результатов апробации, путем сравнения результатов проведения ретроспективы. Таким образом, исходя из сравнения результатов до и после использования авторских методик по сбору и анализу данных можно сделать вывод о целесообразности их использования и практической пользе. Промежуточные результаты показали действенность принятых мер и корректность прогнозов, созданных в ходе предиктивного анализа.

Заключение

В результате проведения исследования и написания выпускной квалификационной работы был проведен анализ существующих методик и инструментов, использующихся в управлении процессами тестирования. Были выполнены задачи:

- Изучение организационных процессов в разработке программного обеспечения.
- Изучение инструментария, используемого в процессах тестирования программного обеспечения.
- Обзор процессов управления в разработке программного обеспечения.
- Исследование способов получения данных в процессе тестирования программного обеспечения.
- Исследование процессов в управлении тестированием.
- Исследование методов анализа данных, применяемых в управлении тестированием.
- Разработка рекомендаций по сбору и анализу информации для применения методов анализа данных и повышения эффективности процессов тестирования.
- Разработка рекомендаций по использованию полученных результатов анализа данных для повышения эффективности процессов тестирования.
- Апробация созданных рекомендаций и методик.

В первых двух главах был проведен обзор существующих теоретических источников, которое позволило выделить основные процессы тестирования ПО и их влияние на эффективность работы. Также были продемонстрированы основные инструменты и методы, используемые в процессе управления тестовой деятельностью.

Третья глава содержит авторские методики, разработанные на основе первых двух глав. Разработанные методики представлены в виде анкет и метрик, что делает их удобными для использования. С помощью этих методик руководители команд могут проводить ретроспективный и предиктивный анализ работы команд тестирования, а также улучшать систему планирования работы, получать более широкие и детальные данные о процессах, выявлять сильные и слабые места в работе команды. Использование созданных рекомендации позволит специалистам, управляющим процессами тестирования добиться повышения эффективности работы и, как следствие, качества продукта и скорости работы.

Целесообразность применения созданных методик доказана в ходе апробации, которая описана в четвертой главе. Полученные результаты демонстрируют корректность полученных в ходе анализа результатов и действенность решений, принятых для повышения эффективности работы.

Полученные методики предлагается внедрять в учебный материал в рамках корпоративного обучения сотрудников. Наряду с повышением уровня навыков руководства, для специалистов, занимающихся управлением процессами тестирования, целесообразно организовывать курс по методам сбора и анализа данных для проведения ретроспективы и предиктивного анализа, чтобы дать специалистам навыки, актуальные именно для сферы их деятельности.

Список используемых источников

1. Александрова, Т.В. Повышение эффективности проектного управления в организации на основе гибкой методологии Agile // Экономика и бизнес: теория и практика. 2019. №9. С. 11–15.

2. Андерсон, К. Аналитическая культура. От сбора данных до бизнес-результатов. Изд. Манн, Иванов и Фербер, 2017 г. — 149 с.

3. Брауде, Э. Технология разработки программного обеспечения / Э. Брауде. — Санкт-Петербург: Питер, 2004. — 469 с.

4. Вахрушина М. А. Управленческий анализ: учеб. пособие – Москва: Омега-Л, 2005. — 157 с.

5. Вечканов, Г. Экономическая теория / Г. Вечканов, Г. Вечканова. – Москва: Эксмо, 2007. – 137 с.

6. Вигерс, К. Разработка требований к программному обеспечению / пер. с англ.. – Москва: Русская Редакция, 2004. – 334 с.

7. Волков В.Г., Автоматизированная система контроля и обеспечения надежности программных средств // http://www.unn.ru/pages/issues/vestnik/99999999_West_2009_5/27.pdf (дата обращения: 10.09.2024).

8. ГОСТ Р ИСО/МЭК 12207-99. Информационная технология. Процессы жизненного цикла программных средств. Дата введения: 2000-07-01. – М.: Издательство стандартов, 2000. – 24 с.

9. ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. Дата введения: 1994-07-01 – М.: Издательство стандартов, 2004. – 5 с.

10. ГОСТ Р 53622-2009 Информационные технологии. Информационно-вычислительные системы. Стадии и этапы жизненного цикла, виды и комплектность документов. Дата введения: 2009-12-15 – М.: Стандартинформ, 2019. – 4 с.

11. ГОСТ Р 56922-2016. Системная и программная инженерия. Тестирование программного обеспечения. Дата введения: 2016-05-18 – М.: Стандартинформ, 2019. – 16 с.

12. Дастин Э. Тестирование программного обеспечения. Внедрение, управление и автоматизация / Э. Дастин, Д. Рэшка, Д. Пол; Пер. с англ. М. Павлов. - М.: Лори, 2013. - 463 с.

13. Калбертсон, Р. Быстрое тестирование / Р. Калбертсон, К. Браун, Г. Кобб. — Москва: Вильямс, 2002. — 201 с.

14. Канер Сэм, Фолк Джек, Нгуен Кек Енг Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / С. Канер. — Калининград: ДиаСофт, 2001. — 428 с.

15. Котляров В.П. Основы современного тестирования программного обеспечения: учебное пособие/В.П.Котляров, Т.В.Коликова – СПб.: Питер, 2004. — 162 с.

16. Куликов С. С. Тестирование программного обеспечения. Базовый курс: практ. пособие. – Минск: Четыре четверти, 2015. – 203 с.

17. Липаев В. В. Тестирование компонентов и комплексов програм [Электронный ресурс] : учебник / В. В. Липаев. — М. : СИНТЕГ, 2010. — 203 с. — Режим доступа: http://www.iprbookshop.ru/27301.html (дата обращения: 10.09.2024).

18. Липаев В.В. Тестирование крупных комплексов программ на соответствие требованиям: учебник/В.В.Липаев – М.: ИПЦ «Глобус», 2008. – 308 с.

19. Липаев В.В., Позин Б.А., Блау С.А., Анализ стратегий тестирования логики программ, Кибернетика, 1982, "2, с 45-50.

20. Макаридина О.П. Управленческие решения. - Екатеринбург: Издательство УрГУПС, 2011. – 113 с.

21. Мартюков А. С. О необходимости разработки гибкого процесса тестирования интернет-приложений // Новые информационные технологии в автоматизированных системах. 2011. С. 22–27.

22. Метрики тестирования [Электронный ресурс]. - Режим доступа: https://testengineer.ru/software-testing-metrics/ (дата обращения: 10.09.2024).

23. Орлов, С.А. Технологии разработки программного обеспечения – Санкт-Петербург: ПИТЕР, 2002. – 489 с.

24. Основные задачи анализа данных и последовательность их решения. Анализ данных в электронных таблицах [Электронный ресурс]. - Режим доступа: https://oblakoz.ru/conspect/528101 (дата обращения: 10.09.2024).

25. Планирование человеческих ресурсов проекта [Электронный ресурс]. - Режим доступа: https://intuit.ru/studies/courses/646/502/lecture/11398 (дата обращения: 10.09.2024).

26. Ретроспектива и предиктивная аналитика [Электронный ресурс]. - Режим доступа: https://digital-academy.ru/blog/retrospektiva-prediktivnaya-analitika?ysclid=m2ygol72gp896044634 (дата обращения: 10.09.2024).

27. Савастюк С. Методологии тестирования ПО. Какую выбрать? [Электронный ресурс] — Режим доступа: https://xbsoftware.ru/blog/metodologii-testirovaniya-po-kakuyu-vybrat/ (дата обращения: 10.09.2024).

28. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах. – М.: Дело, 2007.

29. Савкин В. Принципы управления качеством программ [Электронный ресурс]. Электрон. текстовые дан. – М.: Открытые системы, 2008-2018. – Режим доступа: http://www.osp.ru/os/2008/06/5344965 (дата обращения: 10.09.2024).

30. Скрипкин, К. Г. Экономическая эффективность информационных систем / 2002. – 162 с.

31. Соловьев С.В. Технология разработки прикладного программного обеспечения – М.: Академия Естествознания, 2011. — Режим доступа: https://www.monographies.ru/ru/book/view?id=141 (дата обращения: 10.09.2024).

32. Соммервилл, И. Инженерия программного обеспечения – Москва: ВИЛЬЯМС, 2002. – 103 с.

33. Тестирование и качество ПО. [Электронный ресурс] — Режим доступа: http://software-testing.ru/ (дата обращения: 10.09.2024).

34. Технология каскадного тестирования программного обеспечения [Электронный ресурс]. - Режим доступа: http://software-testing.ru/about/trainers/94-rukol (дата обращения: 10.09.2024).

35. Уровни тестирования программного обеспечения [Электронный ресурс]. - Режим доступа: - http://www.protesting.ru/testing/testlevels.html (дата обращения: 12.09.2024).

36. Хатько Е. Е. Москва, Долгопрудный: МФТИ, 2009. Один из подходов к анализу системы тестирования сложных программных комплексов. Современные проблемы фундаментальных и прикладных наук.

37. Хатько Е. Е., Филиппов, В. А. Проблемы качества тестирования программного обеспечения для мультизадачных пользовательских комплексов. Качество. Инновации. Образование. 3 2011.Т. 3, с. 32-35.

38. Шмеилин Б. 3. Современные технологии тестирования WEB приложений. Системы и средства информатики. 2009 г., с. 141.

39. Barry W. Boehm, A Spiral Model of Software Development and Enhancement, TRW Defense System Group, 1988.

40. Carapola A. Lord of the Infrastructure: A Roadmap for IT Infrastructure Managers, 2016.

41. Changyou Xing, Guomin Zhang, Ming Chen. Research on universal network performance testing model/ International Symposium on Communications and Information Technologies, 2007.

42. Functional Testing: A Complete Guide with Types and Example [Электронный ресурс]. — Режим доступа: https://www.softwaretestinghelp.com/guide-to-functional-testing/ (дата обращения: 08.09.2024).

43. Heiskanen H., Maunumaa M., Katara, M. Test Process Improvement for Automated Test Generation. Tampere: Tampere University of Technology, Department of Software Systems, 2010.

44. Jie M., Honlin Zh., Wenbo X., Jin L., Reliability Testing Methods for Critical Information System based on State Random [Электронный ресурс]. - Режим доступа: http://www.ipcsit.com/vol16/6-ICICM2011M009.pdf (дата обращения: 12.09.2024).

45. Khatko E, Fillipov V. Mobile applications testing processes metrics and optimization criteria. Software Engineering. 5, 2012 г.

46. Kim G.-B. F method of generating massive virtual clients and model-based performance test/ Fifth International Conference on Quality Software, 2005.

47. Kostogryzov A., V.Panov, B.Pozin, V.Sablin. Mathematical modeling of processes in systems life cycles in compliance with standarts requirements of ISO/IEC 15288 and ISO/IEC 12207, Spincose, Montreal, Canada, 2003.

48. Makinen M. Model Based Approach to Software. Helsinki : Helsinki University of Technology, 2007.

49. Product Guides & Tutorials // Atlassian [Электронный ресурс]. — Режим доступа: https://www.atlassian.com/software/jira (дата обращения: 12.09.2024).

50. Software Testing Fundamentals [Электронный ресурс]. — Режим доступа: http://softwaretestingfundamentals.com/ (дата обращения: 08.09.2024).

51. Yamamoto S. A Continuous Approach to Improve IT Management. Procedia Computer Science. 2017.

Приложение А

Схема тестирования задачи

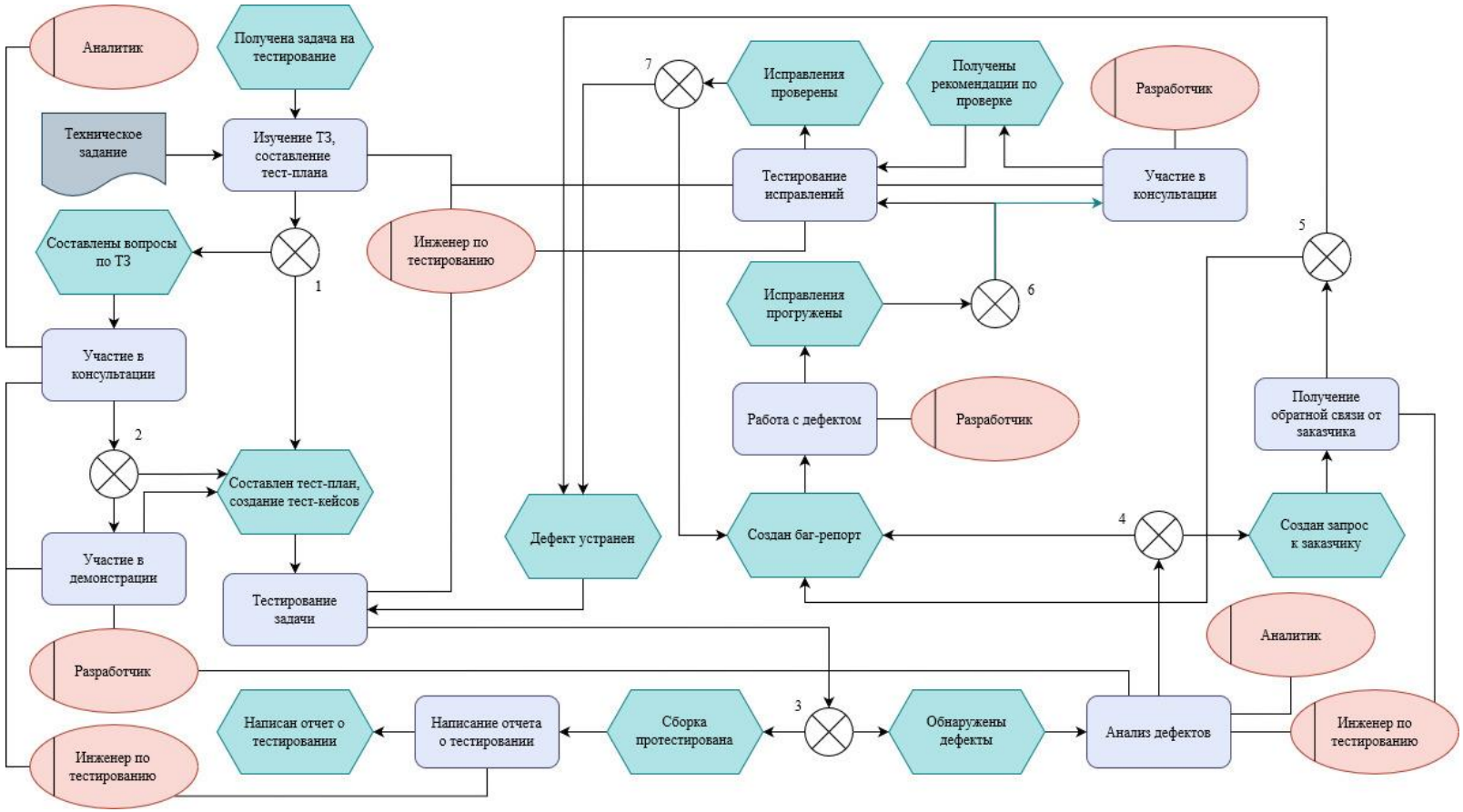

Рисунок А.1 – Схема тестирования задачи

Приложение Б

Схема анализа дефектов

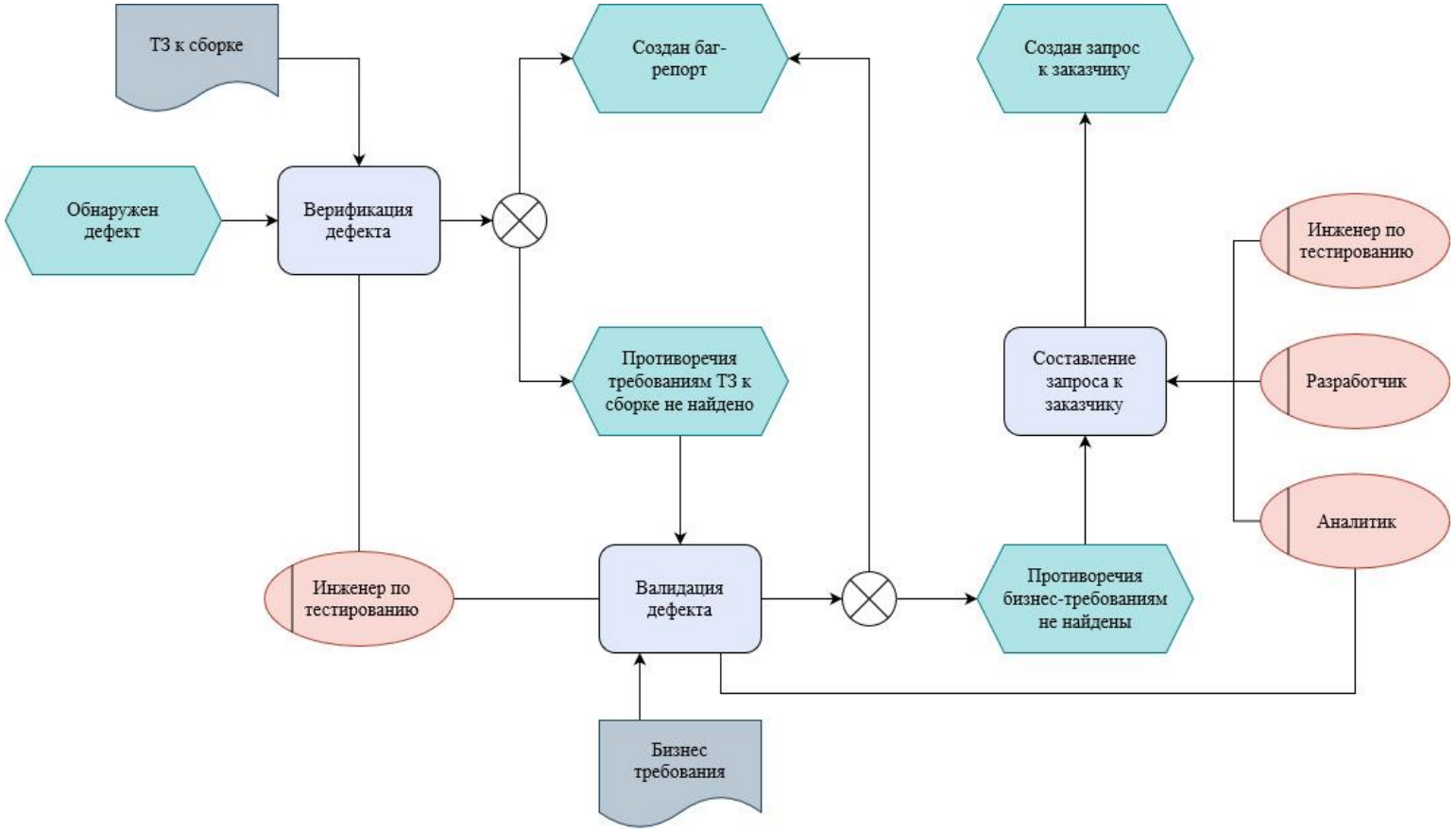

Рисунок Б.1 – Схема анализа дефектов

Приложение В

Матрица компетенций

Таблица В.1 – Матрица оценки компетенций сотрудников

	Младший специалист по тестированию	Специалист по тестированию	Старший специалист по тестированию
Навыки ручного тестирования	Понимание теории тестирования, знание основных техник тест-дизайна.	Умение применять основные техники тест-дизайна, понимание принципов работы приложений для тестирования серверной части.	Уверенное использование техник тест-дизайна, тестирование серверной части, способность консультировать по техникам тестирования.
Навыки автоматизации	Умение запускать автотесты, отслеживать их выполнение.	Умение самостоятельно написать короткий / несложный автотест.	Уверенное использование автоматизации в работе, написание сложных автотестов.
Белый / черный ящик (доступ к коду)	Тестирование по принципу черного ящика (без доступа к коду).	Тестирование по принципу черного ящика с пониманием части кода ПО.	Тестирование по принципу белого ящика с полным доступом к коду.
Знание продукта	Изучение продукта, получение консультаций. Тестирование по готовым тест-кейсам.	Понимание продукта, самостоятельное изучение незнакомых частей по ТЗ.	Полное знание всех частей ПО, способность консультировать по продукту.

Продолжение таблицы В.1

Понимание процессов разработки	Изучение процессов работы команды разработки, получение консультаций.	Понимание основных процессов разработки, понимание информации, полученной от разработчиков.	Общение с разработчиками, выстраивание своей работы на основе хода разработки.
Навыки работы с документацией	Изучение ТЗ, составление тест-кейсов и баг-репортов.	Составление чек-листов, отчетов о тестировании.	Создание тест-плана, оценка полноты документации по задачам, проверка чужой документации.
Самостоятельность	Выполнение задач согласно постановке руководителя.	Способность самостоятельно приоритизировать свои задачи.	Ведение нескольких задач одновременно, уверенный тайм-менеджмент.
Навыки коммуникации	Участие в обсуждениях в роли слушателя.	Способность принимать участие в обсуждении задач, высказывание своего мнения.	Инициация обсуждений, предложение своих идей.
Роль в команде	Является непосредственно обучаемым, нуждается в координации.	Способен полностью самостоятельно организовать свою работу.	Способен координировать младших сотрудников.