БУ ВО «СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
КАФЕДРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
УПРАВЛЕНИЕ КАЧЕСТВОМ. КЕЙС

Выполнил: студент группы № 606-12,
Речук Дмитрий Максимович
Дата сдачи работы:
Принял: ст. преподаватель кафедры АиКС,
Гребенюк Елена Владимировна
Дата проверки работы:
Оценка:

Сургут 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является подбор оптимальных методов управления качеством, применимых к рассматриваемому кейсу. Для достижения этой цели ставятся следующие задачи: рассмотреть представленную ситуацию и разработанные мероприятия, направленные на устранение проблем в процессе разработки; проанализировать возможности с точки зрения реальности выполнения всех намеченных мероприятий; разработать собственные варианты решения выявленных проблем и обосновать выбор оптимального из них.

ОПИСАНИЕ КЕЙСА

Борис занимает должность ведущего IT-менеджера в компании, специализирующейся на разработке программного обеспечения. Вот его история: «Когда-то я начинал как тестировщик. В те времена меня часто раздражали программисты с их вечными оправданиями: то «это не баг, а фича», то «да, баг, но мелкий — пусть остается». Как это «пусть остается», если из-за этого вся система зависает!? Потом я сам стал программистом. И тут всё перевернулось: меня начали выводить из себя бесконечные возвраты задач на доработку. То тестировщикам что-то не нравится, то что-то не работает. Зачем вообще вызывать контекстное меню в этом окне и вставлять туда непонятные символы? Кто до такого додумался? Это же абсурд, в реальной жизни ни один пользователь так не поступит! Не буду исправлять, оставлю как есть! В общем, типичная история — противостояние программистов и тестировщиков. Но затем я стал IT-менеджером и понял, что эта вражда разрушает общее дело. К счастью, опыт работы и в роли тестировщика, и в роли программиста помог мне увидеть корень проблемы и найти способ её решить».

Пять лет назад процесс разработки в компании был прост и понятен: Задачи → Программирование ↔ Тестирование → Релиз. Однако тестировщики узнавали о задачах только на этапе передачи их на тестирование, не получая информации о начале разработки. Программисты же, не дожидаясь результатов тестирования, сразу брались за новые задачи. Этот подход считался образцом идеальной «инкапсуляции»: программисты получали новые задачи и возвраты от тестировщиков, а затем передавали готовое в тестирование; тестировщики принимали задачи от программистов, возвращали их на доработку или выпускали версию. Каждый был погружен в свой мир и занимался своим делом. Казалось бы, всё логично.

Но у процесса была конечная цель — выпуск качественного ПО с нужным функционалом в установленные сроки. И тут начинались сложности. Менеджер приходил к программистам с вопросом «Когда будет готово?», а те

отвечали: «Мы всё сделали, спросите у тестировщиков». Менеджер шел к тестировщикам, а те говорили: «Нам только утром передали сборку, мы только начали тестировать. Уже нашли кучу багов, будет возврат. Когда будет релиз? Спросите у программистов». Менеджер оказывался в замкнутом круге, а релиз всё откладывался.

В итоге менеджер собирал программистов и тестировщиков вместе, чтобы найти выход. Решение сводилось к выработке правил взаимодействия между ними, а затем менеджер следил за их соблюдением. Спустя несколько месяцев напряженной работы сформировались следующие правила:

- Совместное планирование. Версии планировались заранее с участием и программистов, и тестировщиков. Это позволяло всем заранее понимать, что предстоит делать: например, тестировщики могли подготовить тестовые планы и окружение.
- Маленькие версии. Сокращение объема версий уменьшало затраты на обновление кода и повторное тестирование, позволяя работать в основной ветке.
- Ничегонеделанье. Если тестирование отставало, а задач накапливалось слишком много, программистам разрешалось приостановить работу, чтобы не увеличивать разрыв.
- Раннее информирование. Тестировщик, найдя дефект в большой задаче, сразу сообщал об этом программисту, не дожидаясь полного завершения тестирования. Или программист заранее делился с тестировщиком деталями реализации, чтобы тот подготовил тесты.

Эти правила улучшили ситуацию, но не решили проблему полностью. Они словно латали мелкие дыры, но фундаментальная неисправность оставалась. Стоило менеджеру ослабить контроль, как все договоренности забывались, и процесс возвращался к исходной точке. Вражда между программистами и тестировщиками никуда не делась.

Борис, как IT-менеджер, долго анализировал причины, размышлял о мотивах, эмоциях и потребностях людей. И вдруг его осенило: конфликт заложен в самой структуре подхода «одни пишут код, другие тестируют». У программистов цель — «разработать и передать», у тестировщиков — «протестировать», а цель «выпустить качественный релиз» есть только у менеджера. Пока он прилагает усилия, цель достигается, но у команд нет общего пути — их интересы расходятся.

Решение оказалось простым, но глубоким: дать программистам и тестировщикам единую цель — выпуск качественного ПО в срок. Ведь без этой цели их локальные задачи теряют смысл. Изменить мышление было непросто, но Борис, убежденный в своей правоте, был готов преодолеть любое сопротивление. Вот что он сделал:

1. Реорганизация структуры. Вместо отдельных отделов разработки и тестирования создали команды, объединяющие программистов и тестировщиков.
2. Переезд. Каждой команде выделили свою комнату, чтобы устранить изоляцию программистов и тестировщиков.
3. Пропаганда. Борис произнес немало вдохновляющих речей, объясняя цель реорганизации и донося до всех общую миссию.
4. Увольнения. Те, кто не принял перемены, покинули компанию, уступив место более подходящим сотрудникам.

Результат превзошел ожидания. Напряжение между программистами и тестировщиками исчезло. Появились ощутимые улучшения:

- Взаимная поддержка повысила качество продуктов.
- Программисты стали подсказывать тестировщикам слабые места для проверки.
- Отношение к дефектам изменилось: никто не ищет виноватых, разработчики благодарят тестировщиков за помощь в улучшении продукта.

- Договоренности о взаимодействии начали соблюдаться естественно, без принуждения.
- Процесс стал работать как часы: регулярные релизы, качественное ПО.

За полгода компания преобразилась, доказав, что общая цель способна объединить даже самых непримиримых соперников.

АНАЛИЗ ПРЕДЛОЖЕННЫХ МЕРОПРИЯТИЙ

Изменение организационной структуры отдела. Переход к командной структуре, объединяющей программистов и тестировщиков, требует времени. Формирование новых команд предполагает разработку новых правил взаимодействия и рабочих процессов. Кроме того, межличностные отношения внутри команд не выстраиваются мгновенно — на их развитие уходит определённый период, что может замедлить адаптацию к изменениям.

Переезд подразделений. Учитывая уровень доходов IT-компаний и средние зарплаты программистов и тестировщиков, аренда и обустройство помещений для новых команд не представляют финансовой проблемы. Организовать такое мероприятие относительно просто и быстро, что делает его одним из наиболее легко реализуемых шагов.

Пропаганда. Этот процесс включает две задачи: повышение мотивации сотрудников и донесение общей цели. Вторая часть вполне достижима — общая цель (выпуск качественного ПО в срок) имеет конкретные и понятные последствия, которые легко объяснить. Однако с мотивацией сложнее: даже самые вдохновляющие речи не гарантируют, что сотрудники проникнутся корпоративным духом, так как это зависит от индивидуальных факторов.

Увольнение. Проведение увольнений — технически несложная задача, однако поиск новых сотрудников может стать вызовом. Рынок программистов и тестировщиков уже давно характеризуется высоким спросом, и конкуренция за квалифицированные кадры велика. Компаниям с ограниченным бюджетом будет сложно привлечь подходящих специалистов, что может затянуть процесс обновления штата.

СОБСТВЕННЫЕ РЕШЕНИЯ

Премии. Введение системы вознаграждений за высокую производительность способно мотивировать как отдельных сотрудников, так и целые команды. Это может ускорить адаптацию к новой структуре и повысить вовлечённость в достижение общей цели.

Повышение квалификации. Финансирование курсов по тимбилдингу и коммуникациям поможет улучшить взаимодействие внутри новых команд. Такие навыки особенно полезны на этапе формирования коллективов, где важно наладить доверие и сотрудничество между программистами и тестировщиками.

Оптимизация бизнес-процессов. Внедрение подходов, устраняющих неэффективности (например, автоматизация рутинных задач или пересмотр этапов разработки), может повысить скорость и качество работы команд, минимизируя прежние конфликты между ролями.

Изменение планирования. Перенос сроков промежуточных и финальных результатов может снизить давление на сотрудников, сделав работу более комфортной. Однако это сопряжено с риском финансовых потерь из-за возможного увеличения сроков выпуска продукта, что требует тщательного баланса.

ЗАКЛЮЧЕНИЕ

В итоге цель была достигнута, а поставленные задачи выполнены. Анализ предложенных мероприятий показывает, что каждое из них при грамотной реализации способно принести положительные результаты. Изменение структуры и подходов к работе позволило устранить ключевые проблемы, улучшить взаимодействие и обеспечить стабильный выпуск качественного ПО. Предложенные дополнительные решения усиливают эффект, создавая условия для долгосрочного успеха.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Отчет о научно-исследовательской работе. М., 2017. 26 с. — URL: https://cs.msu.ru/sites/cmc/files/docs/2021-11gost_7.32-2017.pdf (дата обращения 09.04.2025).