МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
**“САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО”**
Кафедра математической экономики

# Этапы создания мобильного приложения для бизнеса на основе технологии межличностного он-лайн общения в группе разработчиков и оптимизации функциональных блоков программного продукта

## АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
на направлению 38.03.05 – Бизнес-информатика
механико-математического факультета
**Маркова Владислава Павловича**

Научный руководитель
доцент, к. ф.-м.н., доцент _____ И.Ю. Выгодчикова

Зав. кафедрой
д. ф.-м.н., профессор _____ С.И. Дудов

Саратов 2020

**Введение.** Разработка программного обеспечения – довольно многогранная и интересная сфера, которая заключается не только в известных процессах управления, аналитики, разработки и тестирования, но также и в постановке других процессов, которые зачастую многие профессионалы считают вторичными, но на практике оказывается, что эти вторичные процессы могут сильно повлиять на основные процессы.

Так, например, небольшие изменения в общении команды могут привести к позитивному эффекту.

Каждый проект по своему уникален и на нем могут быть свои уникальные процессы, которые необходимо правильно рационализировать, при этом оставляя пространство для действий сотрудников. Многие известные исследователи в сфере программной инженерии считают, что чаще всего на проектах возникают сложности не в том, что лежит на поверхности, а несколько глубже. Дипломная работа направлена на исследование процессов общения команды и оптимизации функциональных блоков кода.

Процессы общения между разработчиками имеют высокую ценность и формально представляют собой поток обмена данными между людьми. При правильно выстроенных процессах обмена формальными данными экономятся часы разработки на проекте, программное обеспечение становится более корректным, так как информация доходит до всех участников проекта.

К сожалению, не всегда процесс обмена информацией между разработчиками в онлайн форме может происходит оптимальным образом и информация может теряться или приходить в искаженном виде. В итоге падает корректность работы ПО и оно меньше соответствует поставленным требования при разработке.

Из этого понятна важность правильных процессов общения разработчиков в команде, что будет являться одной из исследуемых тем в дипломной работе.

Другой важный показатель, от которого зависит корректность работы приложения и соответствие его требованиям – оптимальность функциональных блоков программного продукта.

Если на ранних этапах разработки команда программистов приняла неверное решение об архитектуре приложения, то это сильно замедляет разработку приложения в будущем и увеличивает затраты компании на разработку.

В связи с этим, очень важно понимать, как наиболее оптимально принимать решение по функциональным блокам программного кода и знать причинно-следственную связь с разными действиями разработчиков.

Существуют методы экстремального программирования – наиболее устоявшиеся практики, которые применяют крупные компании при коммерческой разработке. Эмпирическим путем было выяснено, что они позволяют писать наиболее эффективный код, а следовательно и повышать корректность ПО.

Также стоит упомянуть о системах контроля версий, которые позволяют сохранять абсолютно все версии приложения. От правильной организации данной системы зависит возможность доставки обновлений пользователям и правильной работы над несколькими задачами одновременно.

Стоит заметить, что стандартный цикл разработки при этом не будет видоизменяться. Различный методы хорошо применимы на разных этапах разработки приложения.

Цель данной дипломной работы – исследовать влияние процессов межличностного онлайн общения между разработчиками и оптимизации функциональных блоков программного кода на различные характеристики

ПО при применении на разных этапах, о которых будет сказано в дальнейшем.

Основные задачи данной работы:

1) Анализ потоков обмена информации и их оптимизация;
2) Исследовать методы экстремального программирования;
3) Исследовать правильную структуру организации системы контроля версии;
4) Исследовать влияние потоков обмена информации, методов экстремального программирования и организации системы контроля версий на различные характеристики программного обеспечения.

**Основная часть.** Во всей дипломной работе примеры будут строится на основе мобильного приложения на ОС iOS “Московская Электронная Школа”, входящее в ИС “Московская электронная школа”. Данный проект существует в 2013-ого года в рамках мобильного приложения для ОС iOS. Проект прошел различные стадии развития, сменилось несколько поколения разработчиков. На данный момент проект выполняет миссию создания удобного рабочего пространства для учителей, учеников и их родителей для электронного образования.

Для того, чтобы верно проанализировать передачу информацию между сотрудниками, эффективнее всего использовать DFD-диаграммы, то есть диаграмму поток данных. Как мы видим, слабыми местом DFD-диаграммы на текущий момент является отсутствие связи команды разработчиков и заказчиков.  Изобразить обмен информации на данный момент можно следующим образом:

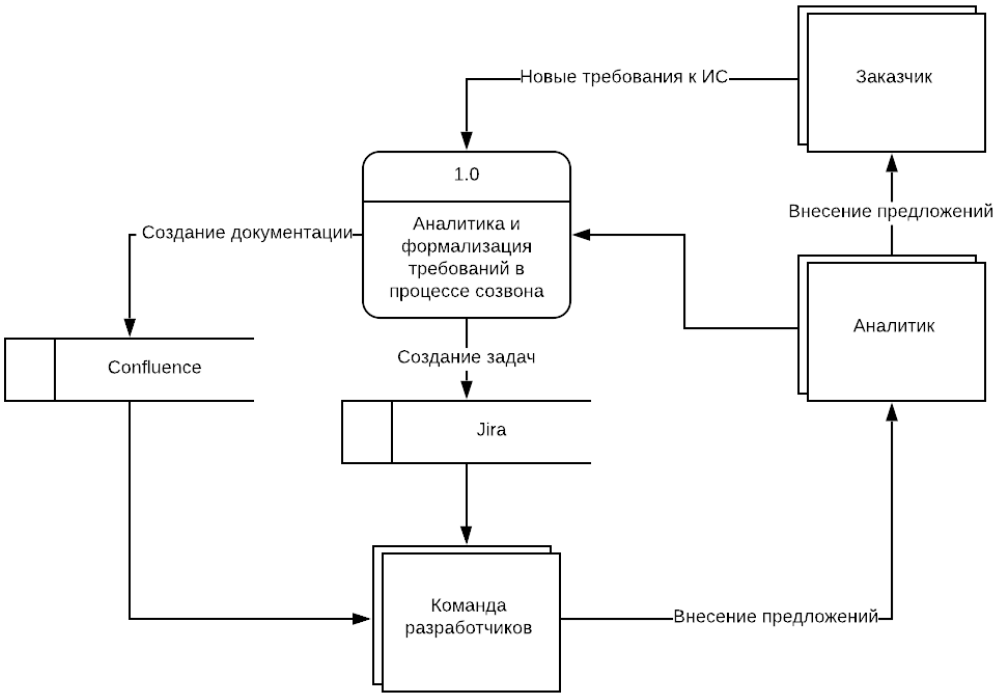


Рисунок 1 – диаграмма AS IS потоков обмена информации

Опишем диаграмму, которая будет более оптимальным образом производить обмен информации между сотрудниками. Ключевое различие в том, что обмен информации происходит не централизованно на аналитике, а всей командой, что уменьшает получение обратной связи от команды разработчиков.

На основе нее можно сделать следующий вывод, который позволяет оптимизировать обмен информации между командой на других проектах: если представить, что DFD-диаграмма представляет собой ориентированный граф, то задачу оптимизации онлайн-общения команды можно свести к тому, чтобы построить наиболее короткий цикл в графе. Его полное отсутствие невозможно, т.к. будет отсутствовать связь с внешней средой и развитие требований.

Оптимизации потоков информации хорошо срабатывают на всех этапах разработки ПО, но как уже понятно из контекста получения информации, больше эффективности будет показано на этапе аналитики.

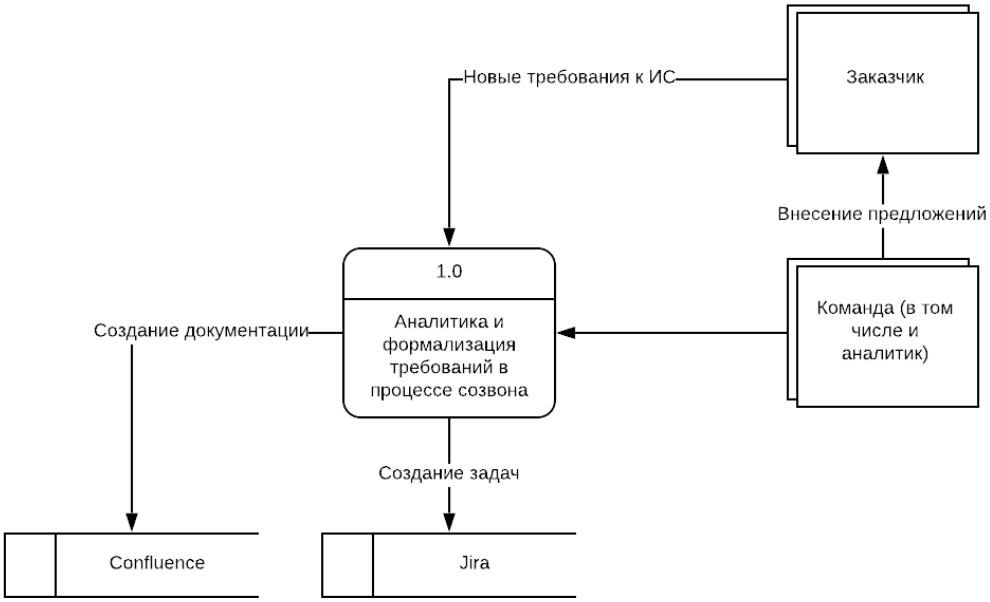


Рисунок 2 – диаграмма AS TO BE обмена информации

Однако, не стоит забывать, что человеческие ресурсы – самый важный элемент в ИТ-команда. Процессы не должны подавлять возможность сотрудников проявлять свою индивидуальность. На основе этого можно сформулировать следующий принцип:

*Регламентация процессов должна подходить под команду, а не наоборот. В таком случае сохраняется индивидуальность подхода каждого члена команды и при этом есть общее понимание движения команды. В ином же случае человеческий фактор игнорируется и не позволяет каждому участнику полноценно применить свой опыт.*

Экстремальное программирование – набор практик и подходов к процессу разработки программного обеспечения со стороны технических команд. Философия экстремального программирования основана на том, чтобы достичь “экстремума” продуктивности команды при неизменяемости ресурсов, предоставленных на проект. Основные методы, на которые стоит обратить внимание.

Разработка через тестирование постулирует, что сначала необходимо написать тесты для кода, а уже затем код. Это позволяет

мыслить в обратную сторону, изначально обеспечивать работоспособность приложения. Кроме того, достигается модульность приложения. Улучшает эффективность на этапах разработки и тестирования.

Непрерывная интеграция – постоянная доставка обновлений программного обеспечения заказчику продукта (конечному пользователю). Во втором десятилетии 21 века получила сильное распространение за счет удобного связывания с системами контроля версий. Уменьшает время, за которое тратиться на переход от разработки к тестированию, и от тестирования к выпуску приложения.

Введение стандартов оформления упростит ведение разработки и разработчики будут меньше времени тратить на то, чтобы понять код своих коллег. Как уже понятно, уменьшает время на разработку, т.к. меньше времени надо тратить на чтение.

Простота проектирование позволит не создавать сложных конструкций, а действовать исходя из необходимости. Кроме того, простая архитектура, которая соответствует текущей необходимости сокращает временные затраты.

Проведение рефакторинга кода позволит поддерживать в актуальном состоянии, уменьшая количество устаревших модулей, которую мешают простому изменению остального кода.

Также, если команда будет при этом соответствовать, то некоторые принципы экстремального программирования покажут более высокую эффективность. Если затем посмотреть статистику после применения этих методов, то можно увидеть значительный количественный рост числа выполняемых задач:

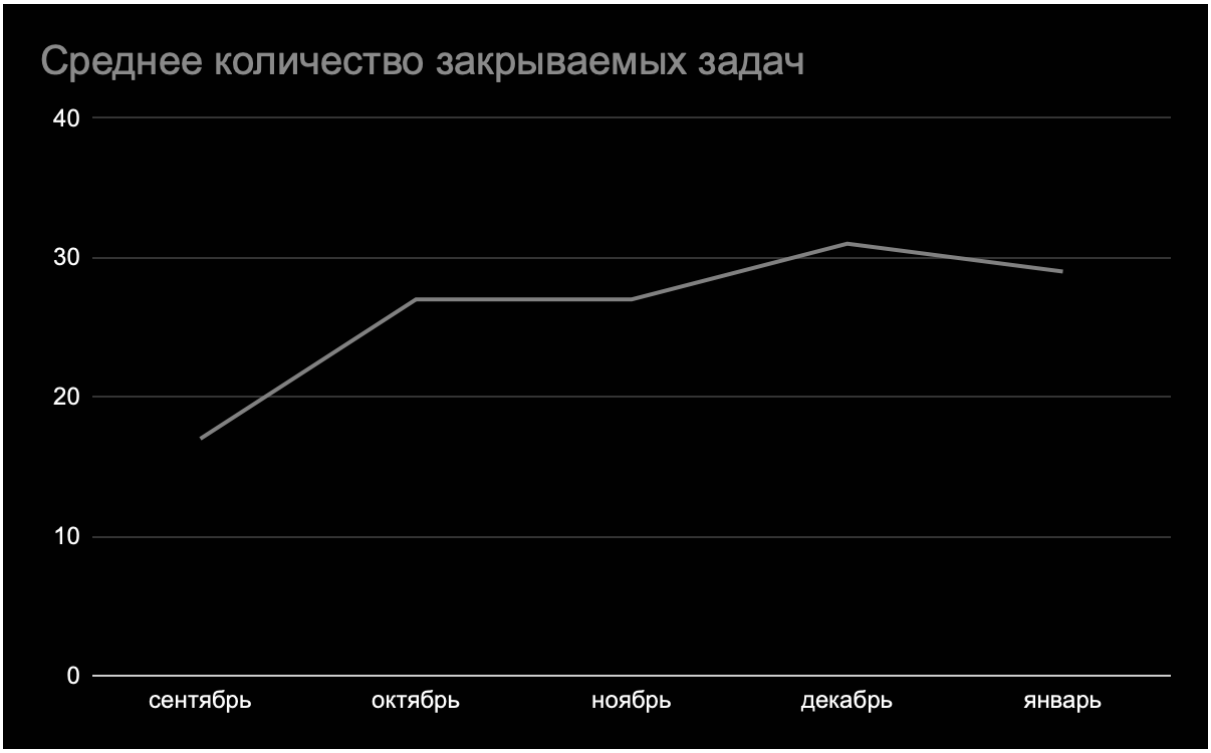


Рисунок 3 - рост числа выполняемых задач

Для облегчения процесса разработки на проекте используются системы контроля версий, хранящие в виде графа в каждой вершине версию приложения. Это значительно облегчает процессы разработки, тестирования и выпуска приложения. Кроме того, применение непрерывной интеграции вместе с системой контроля версий становится значительно проще.

Кроме того, в системах трекинга задач наподобие Jira можно указывать ветки версий, которые связаны между собой. Благодаря этому отслеживание становится намного проще.

Существует стандартный шаблон ведение системы контроля версий Gitflow, который позволяет легко управлять релизами, тестированием и разработкой и включает в себя ряд веток, изображенных на рисунке ниже.

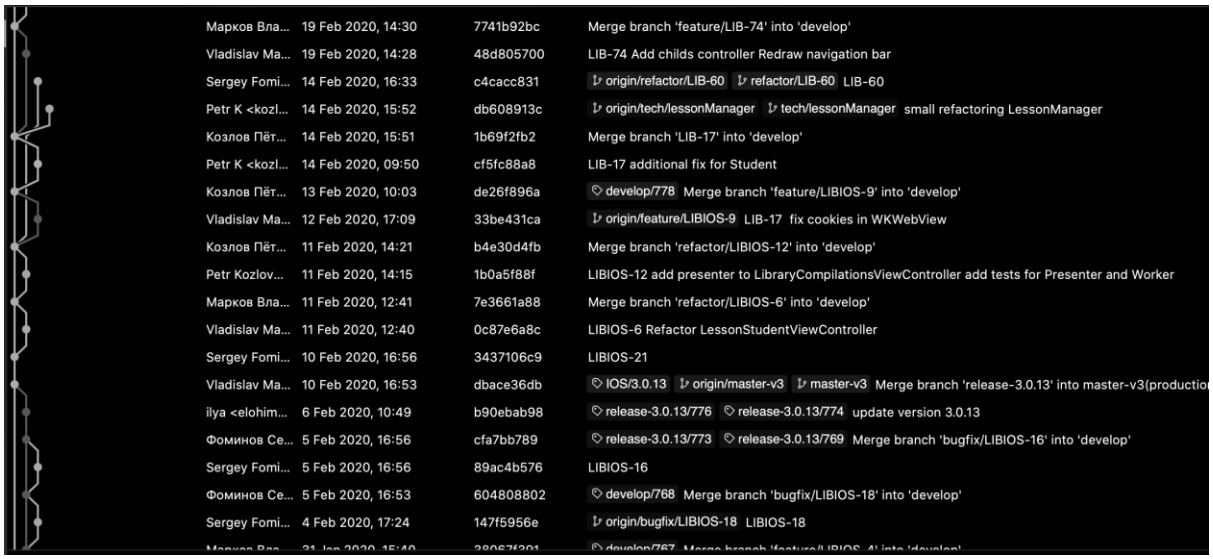


Рисунок 4 – Gitflow проекта

Если оценивать улучшение проектных показателей с помощью оптимизации потоков информации, то удалось оптимизировать временные затраты, а также сделать рабочее пространство команды значительно проще.

Хранение документов стало значительно проще, а цикл обработки информации значительно сократился. Процесс веденных ретроспектив помог значительно упростить технические и управленческое развитие процессов на проекте.

Введение методов экстремального программирования хорошо работает на проекте, так же можно довольно легко отследить эти изменения при помощи количественных показателей на проекте. Так, например, статистика сбоев в приложении значительно уменьшилась:


Рисунок 5 – рост показателей статистики работы приложения без сбоев

Также стоит обратить внимание на то, что задачи категории “баги” заметно уменьшились, то есть больше времени получается уделяться разработке нового функционала.

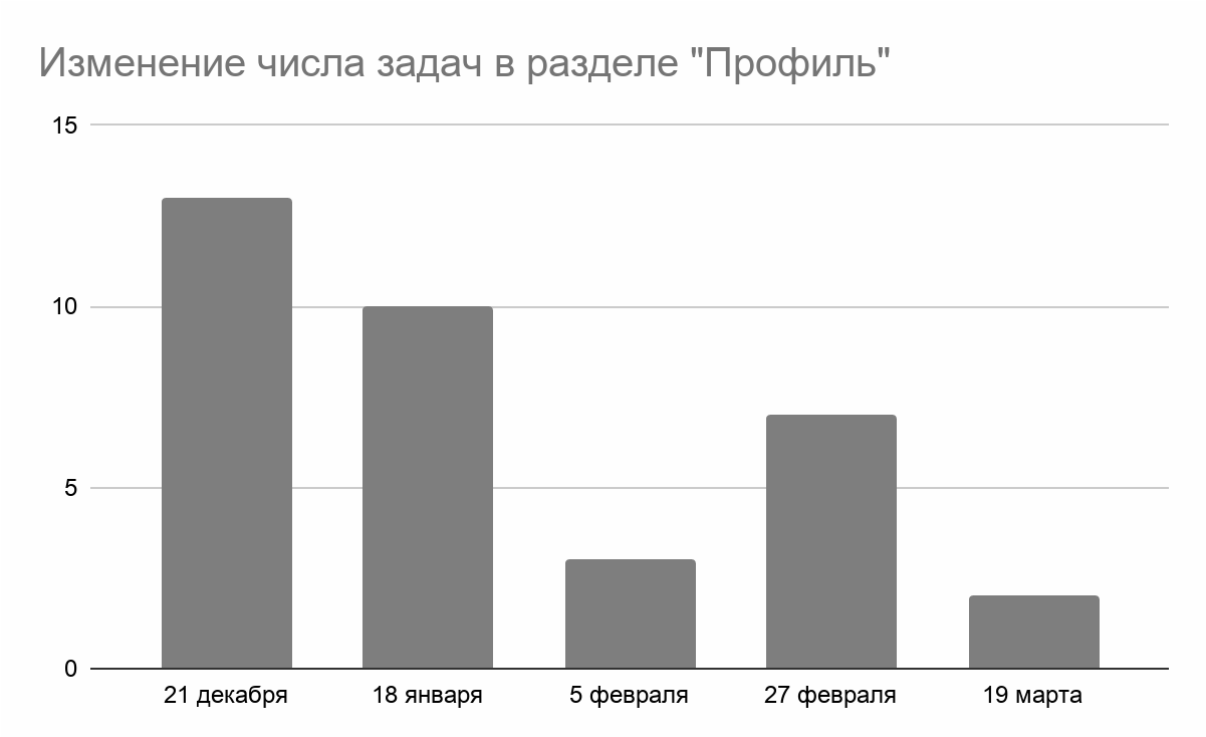


Рисунок 6 - изменение числа задач в разделе “Профиль”

Также можно заметить, что был небольшой скачок в феврале – в это время был введен новый функционал, что всегда приводит к появлению новых багов. Однако, методы экстремального программирования помогают быстро нивелировать проблемы при введении нового функционала.

Также можно заметить, что при команду за время внедрения экстремального программирования удалось уменьшить в два раза. Высвободившиеся человеко-часы были направлены на другие проекты, что позволяет увеличить прибыльность компании. Таким образом, подбор подходящих методов разработки на проект в ИТ-компании позволяет повысить прибыльность предприятия.

Дополняя это структурированной работой с системой контроля версий, то можно иметь доступ к каждой нужной версии приложения спустя множество лет работы. Это позволяет структурировано обработать историю приложения и обращаться через Jira к нужным версиям приложения.

**Заключение.** При проведении работы были закреплены и получены многие необходимые профессиональные и учебные навыки. Главная цели работы – исследовать влияние процессов межличностного онлайн общения между разработчиками и оптимизации функциональных блоков программного кода на различные характеристики ПО был достигнута.

Рассмотрено влияние описанных выше методик на:

1) Качественные и количественные характеристики ПО;
2) Временные трудозатраты при разработке ПО;

Таким образом, подводя итоге можно сказать следующее: использование технологий межличного онлайн общения и оптимизация функциональных блоков программного кода приводит как и у к уменьшению издержек производства ИТ компании, так и увеличении качества выпускаемой продукции.

Указанные выше задачи были также достигнуты:

1) Анализ потоков обмена информации и их оптимизация производились на системах таск-трекинга и ведения документации. Показана эффективность при структуризации данных компонентов ИС на временных издержках;
2) Исследованы конкретные методы экстремального программирования (непрерывная интеграция, модульное тестирование и введение стандартов оформления кода) на разных этапах разработки приложения;
3) Описана модель GitFlow для наиболее оптимальной организации системы контроля версий. Показана эффективность для выпусков обновления приложений, структуризации работы, а также упрощения непрерывной интеграции)

Данные методики описаны и показана их эффективность в рабочей практике. Многие из методов экстремального программирования были внедрены в рабочую деятельность организации.

Все поставленные задачи и цели выпускной работы выполнены. В дополнение к этому повышена производительность и качество работы на предприятии.