Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет**

**при Правительстве Российской Федерации»**

**(Финуниверситет)**

**Департамент анализа данных и машинного обучения**

**Факультет** **информационных технологий и анализа больших данных**

**КУРСОВАЯ РАБОТА**

**по дисциплине** «Программная инженерия»

**на тему:** «Сравнительный анализ Agile методологий Scrum и XP на примере

конкретного проекта. Предметная область: ПО для финтесс-треккера»

Направление подготовки: 09.03.03 – Прикладная информатика

Профиль: «ИТ-сервисы и технологии обработки данных в экономике и финансах»

**Выполнила студентка** 3 курса,

группы ПИ21-7,

формы обучения очной

Филиппова Е. Р.

**Руководитель:**

Журавлева М. Г.

|  |  |  |
| --- | --- | --- |
| Дата поступления работы на кафедру: | Работа допущена  к защите:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя) | Работа защищена  с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя) |
| \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2024 г. | \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2024 г. | \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Москва 2024

# **Оглавление**

Введение

1. Методология исследования
   1. Основные принципы Scrum
   2. Основные принципы Extreme Programming (XP)
   3. Преимущества и недостатки каждой из методологий
   4. Описание проекта ПО для финтесс-треккера и его особенностей
   5. Определение критериев сравнения между Scrum и XP
2. Сравнительный анализ Scrum и XP
   1. Применение Scrum в рамках проекта
      1. Описание процесса разработки по методологии Scrum
      2. Выявление преимуществ и недостатков Scrum в контексте проекта
   2. Применение XP в рамках проекта
      1. Описание процесса разработки по методологии XP
      2. Выявление преимуществ и недостатков XP в контексте проекта
   3. Сравнительный анализ результатов применения Scrum и XP
      1. Сопоставление критериев
      2. Оценка соответствия каждой из методологий специфике проекта

Заключение

Список использованных источников

Приложения

# **Введение**

В современном мире программное обеспечение играет ключевую роль в нашей повседневной жизни, оказывая влияние на различные сферы деятельности от игровой индустрии до бизнеса. С развитием технологий разработчики сталкиваются с постоянной необходимостью создания и обновления программных продуктов, что требует применения эффективных методов управления проектами. В этом контексте Agile-методологии выступают важным инструментом, обеспечивая гибкий подход к разработке и способствуя быстрой адаптации к изменениям.

Выбор подходящей Agile-методологии для конкретного проекта играет решающую роль в успешной реализации задач. В рамках данного исследования мы обращаем внимание на процесс разработки программного обеспечения для фитнес-треккера, а именно на методологии Scrum и XP. Наша **цель** заключается в изучении особенностей их применения в контексте данного проекта, а также в анализе их преимуществ и недостатков. Мы также рассматриваем их соответствие специфике требований и задач, стоящих перед разработчиками в данной области.

**Актуальность** нашего исследования обусловлена растущим интересом к Agile-методологиям в сфере разработки программного обеспечения и необходимостью выбора наиболее подходящего подхода к управлению проектами в конкретных условиях. Проекты, связанные с разработкой фитнес-треккеров, в силу своей специфики и динамичного характера, требуют особого внимания к процессу разработки, что делает результаты нашего исследования важными для разработчиков и менеджеров проектов.

**Научная проблема**, поставленная перед нами, заключается в определении оптимального подхода к управлению проектами в области разработки программного обеспечения для фитнес-треккера с учетом их специфики и требований рынка. Наша **задача** состоит в том, чтобы определить оптимальный подход к управлению проектами в области фитнес-треккеров и предоставить читателю полное представление о Scrum и XP, помогая им выбрать подход к управлению проектом. Мы планируем изучить принципы Agile методологий, проанализировать литературу, описать особенности проекта ПО для финтесс-треккера, определить критерии сравнения и проанализировать использование двух гибких методологий на его основе.

Таким образом, наша работа охватывает все аспекты сравнения Scrum и XP и их применимости в контексте разработки программного обеспечения для фитнес-треккеров.

# **Методология исследования**

## **Основные принципы Scrum**

Scrum - это методика создания программного обеспечения, которая очень популярна из-за своей гибкости и возможности быстро адаптироваться к изменениям. Ее придумали в начале 1990-х годов Кен Швабер и Джефф Сазерленд, и с тех пор она стала очень популярной.

Суть Scrum в том, что процесс разработки делится на небольшие кусочки, называемые итерациями. Каждая итерация - это как бы отдельная часть работы, в которой команда делает все, чтобы создать кусочек готового продукта. Потом они смотрят, что получилось, и получают отзывы от заказчика или пользователей. Это помогает им улучшать свою работу и адаптироваться к изменяющимся требованиям.

Инкрементальный подход в Scrum означает, что команда добавляет новые функции и возможности к продукту на каждом этапе работы. Вместо того чтобы сразу делать полностью готовый продукт, они делают сначала самые необходимые функции, а потом добавляют к ним новые. Это помогает быстрее предложить пользователям рабочий продукт и узнать, что им нужно улучшить.

Во время работы над проектом, его разбивают на короткие отрезки времени, называемые спринтами. Каждый спринт обычно длится от одной до четырех недель. В конце каждого спринта команда показывает результат своей работы, что позволяет заказчику видеть прогресс и вносить коррективы, если что-то не так.

Основные принципы Scrum включают несколько важных аспектов, которые помогают команде успешно выполнять проект. Например, команда должна быть самоорганизованной и иметь ясные цели, чтобы знать, что они должны достичь.

Вот краткое описание каждого из них:

1. Scrum определяет три основные роли: Product Owner, Scrum Master и Development Team. Каждая из трех ключевых ролей играет важную роль в обеспечении успешной реализации проекта.
   1. Product Owner (Владелец продукта): Он представляет интересы заказчика и пользователей продукта. Основная задача Владельца продукта - определить, что конкретно должен включать в себя продукт, и убедиться, что команда разработки понимает эти требования. Он отвечает за создание и обновление Product Backlog - списка задач и требований, необходимых для достижения целей проекта. Также определяет приоритеты в Product Backlog, учитывая бизнес-потребности и ожидания заказчика.
   2. Scrum Master (Мастер Scrum): Это специалист, который следит за правильным применением методологии Scrum в проекте. Он помогает команде разработки использовать Scrum наилучшим образом и устранять препятствия, которые могут возникнуть. Мастер Scrum проводит обучение и коучинг членов команды по методологии Scrum, помогает им применять принципы и практики Scrum.
   3. Development Team (Команда разработки): Это группа специалистов, ответственных за создание продукта. Команда разработки сама принимает решения о том, как выполнять работу и как достигать поставленных целей. У нее должны быть все необходимые навыки и компетенции для выполнения задач из Product Backlog. Они работают коллективно, чтобы создавать новую функциональность на протяжении каждого спринта.
2. В Scrum используются различные артефакты, которые играют важную роль в организации и управлении процессом разработки. Эти артефакты помогают команде разработки и заинтересованным сторонам понять текущее состояние проекта, определить приоритеты и принимать информированные решения.
   1. Бэклог продукта (Product Backlog): Это список всех задач, требований и идей, которые могут быть реализованы в рамках проекта. Product Backlog поддерживается и обновляется Владельцем продукта и содержит задачи, необходимые для достижения целей проекта. Задачи описываются в виде User Stories или других формализованных элементов и часто сопровождаются приоритетами и оценками сложности. Он служит источником задач для планирования спринтов и определения направления развития продукта.
   2. Бэклог спринта (Sprint Backlog): Это список задач, выбранных командой разработки для выполнения в рамках текущего спринта. Формируется на основе элементов из Product Backlog, выбранных для реализации в конкретном спринте. Sprint Backlog содержит подробное описание каждой задачи, сроки выполнения и ответственных исполнителей. Может подвергаться изменениям в процессе выполнения спринта.
   3. Инкремент (Increment): Это работающая версия продукта, которая достигается в конце каждого спринта. Инкремент должен быть полностью завершенным и работоспособным, содержать новую функциональность или улучшения, которые могут быть протестированы и использованы. Каждый новый Increment представляет собой шаг вперед в развитии продукта и демонстрирует прогресс команды разработки.
3. Существуют определенные события в Scrum, которые играют важную роль в управлении процессом разработки и обеспечивают команде возможность оценивать свой прогресс и адаптировать свою работу.
   1. Планирование спринта (Sprint Planning): Это событие происходит в начале каждого спринта и направлено на определение целей и плана работы на следующий период времени. На этом событии команда разработки совместно с Владельцем продукта обсуждает задачи, которые будут выполнены в следующем спринте, и устанавливает цели, которые они планируют достигнуть к его завершению.
   2. Ежедневный Scrum (Daily Scrum): Это короткое событие, которое проводится ежедневно в течение спринта. Каждый член команды отвечает на три вопроса: о проделанной работе, планах на день и препятствиях. Это позволяет команде быть в курсе текущего состояния проекта и выявлять любые проблемы или препятствия.
   3. Обзор спринта (Sprint Review): Это событие происходит в конце каждого спринта и направлено на представление выполненной работы заказчику или заинтересованным сторонам. На обзоре команда разработки демонстрирует продукт и обсуждает его с заказчиком, получая обратную связь для дальнейшего улучшения.
   4. Ретроспектива спринта (Sprint Retrospective): Это событие происходит после завершения каждого спринта и направлено на анализ работы команды и выявление возможностей для улучшения. Члены команды обсуждают, что прошло хорошо и что можно улучшить, чтобы повысить эффективность работы в следующем спринте.

На основе вышеизложенного, Scrum становится мощным инструментом управления проектами, обеспечивая командам способность быстро реагировать на изменения в требованиях и на рынке. Его методология широко применяется в различных отраслях и проектах, что делает его одним из наиболее востребованных подходов к разработке программного обеспечения в настоящее время. В результате Scrum становится неотъемлемым инструментом для успешного управления проектами в современном бизнесе.

## **Основные принципы Extreme Programming (XP)**

Extreme Programming (XP) - это такая методология разработки программного обеспечения, которая встраивается в подход Agile. Её главная цель - улучшить качество программ и ускорить процесс их доставки заказчику. Принципы XP нацелены на создание гибкой и адаптивной среды разработки. Давайте теперь вместе рассмотрим эти принципы подробнее:

1. Обратная связь (Feedback) играет ключевую роль в методологии Extreme Programming (XP), где она рассматривается как один из основных элементов успешной разработки программного обеспечения. Обратная связь в XP включает в себя два основных аспекта: обратную связь от заказчика и внутреннюю обратную связь в команде разработки.
   1. XP активно поддерживает взаимодействие с заказчиком на протяжении всего процесса разработки. Регулярные сеансы обратной связи позволяют заказчику выражать свои требования, предпочтения и ожидания относительно продукта. Это помогает команде понять и уточнить требования, а также оперативно реагировать на изменения и корректировать планы разработки. Благодаря обратной связи от заказчика, команда может быть уверена, что создаваемый продукт соответствует его ожиданиям и требованиям.
   2. Внутренняя обратная связь в команде разработки включает обмен знаниями, опытом и мнениями между участниками команды. Это может включать обсуждение различных подходов к решению проблем, обмен идеями, а также предоставление конструктивной критики и обратной связи по работе других членов команды. Благодаря этому процессу команда может постоянно улучшать свою работу, обмениваясь опытом и находя наилучшие решения для достижения поставленных целей. Внутренняя обратная связь также помогает команде быстро реагировать на возникающие проблемы и адаптироваться к изменениям в проекте.

Быстрая и эффективная обратная связь в XP способствует улучшению коммуникации внутри команды и с заказчиком, обеспечивает прозрачность процесса разработки и позволяет команде успешно адаптироваться к изменениям, что в конечном итоге приводит к достижению успешных и качественных результатов.

1. Принцип простоты в методологии Extreme Programming (XP) представляет собой стремление к созданию программного кода и архитектуры продукта, которые максимально просты и понятны. Цель этого принципа - упростить процесс разработки и поддержки программного обеспечения.
   1. Создание простого и понятного кода. В XP простота не только в реализации, но и в понимании. Код должен быть легким для чтения и понимания другими членами команды. Это облегчает обслуживание и развитие программы, так как разработчики могут быстро понять существующий код и вносить изменения.
   2. Избегание излишней сложности. Команда XP старается избегать излишней сложности в проекте. Сложные алгоритмы или структуры данных могут усложнить код и увеличить риск ошибок. Вместо этого предпочтение отдаётся простым и эффективным решениям.
   3. Упрощение поддержки и развития. Принцип простоты также способствует упрощению поддержки и развития продукта. Простой код упрощает поиск ошибок, добавление новых функций и изменения в существующем функционале. Это позволяет команде быстро реагировать на изменения требований заказчика и адаптироваться к изменяющимся условиям.

Принцип простоты в Extreme Programming способствует созданию более надежного, управляемого и эффективного программного обеспечения. Следуя этому принципу, команда разработки может достичь более высокого уровня качества продукта и обеспечить его успешное развитие и поддержку на протяжении всего жизненного цикла.

1. Итерации в методологии Extreme Programming (XP) - это ключевой элемент, который играет важную роль в разработке программного обеспечения. Они предполагают итеративный подход, что означает, что продукт создается и поставляется порциями. Это помогает команде быстро получать обратную связь от заказчика и гибко адаптироваться к изменениям в требованиях.

Каждая итерация в XP представляет собой своего рода цикл разработки, который включает несколько этапов:

* 1. Анализ: Здесь команда определяет, какие функции или изменения будут включены в текущую итерацию и проводит анализ требований.
  2. Проектирование: Определяется архитектура и дизайн новых функций, а также планируются необходимые изменения в существующем коде.
  3. Реализация: Программисты приступают к написанию кода для воплощения запланированных изменений и функциональности.
  4. Тестирование: После завершения разработки проводится тестирование, включая как автоматизированные, так и ручные тесты, для проверки корректности работы новой функциональности и выявления возможных ошибок.

1. Принцип маленьких релизов в методологии Extreme Programming (XP) - это стратегия, которая стимулирует команду к частым и небольшим выпускам программного продукта. Этот подход снижает риски, повышает качество и увеличивает ценность продукта для заказчика на каждом этапе разработки.
   1. В XP предпочтение отдается частым выпускам продукта, чтобы заказчик мог получать обратную связь по ходу проекта. Чем чаще команда выпускает новые версии продукта, тем быстрее можно понять, что работает хорошо, а что требует улучшений. Это позволяет команде гибко адаптировать планы и изменять приоритеты в соответствии с реальными потребностями заказчика.
   2. Предпочтение маленьким релизам в XP помогает снизить риски разработки. Небольшие изменения проще контролировать и управлять, что уменьшает вероятность серьезных ошибок или непредвиденных проблем. Более того, маленькие релизы позволяют быстрее исправлять обнаруженные ошибки и улучшать функционал на основе обратной связи.
   3. Частые и небольшие релизы также увеличивают ценность продукта на каждом этапе разработки. Вместо ожидания завершения всего проекта для предоставления заказчику функционала, команда может постепенно внедрять новые возможности и улучшения, что позволяет заказчику быстрее начать использовать и получать пользу от продукта.
2. В Extreme Programming (XP) особое внимание уделяется проверке продукта перед его выпуском. Это важно для того, чтобы убедиться, что программа работает правильно и соответствует требованиям заказчика. Есть два основных способа проверки:
   1. Ручное тестирование: Люди из команды тестируют программу, проверяя, как она работает. Они могут делать разные виды тестов, чтобы найти ошибки и улучшить программу. Например, они могут проверить, что все кнопки на экране работают правильно или что программа работает быстро.
   2. Автоматическое тестирование: Это когда специальные программы тестируют программу автоматически. Они проверяют, что все работает правильно, и находят ошибки. Это помогает быстрее найти и исправить проблемы.
3. Парное программирование (Pair Programming) - это одна из ключевых практик, предложенных методологией Extreme Programming (XP). В рамках этой практики два программиста работают вместе за одним компьютером. Они обсуждают, планируют и пишут код вместе. Это помогает делать программы лучше и быстрее.
   1. Когда программисты работают в паре, они делятся знаниями и опытом друг с другом. Это помогает каждому из них стать лучше в своей работе и лучше понять проект.
   2. Также парное программирование помогает улучшить качество кода. Вместе программисты могут находить ошибки и улучшать структуру кода, делая его более понятным и надежным.
   3. Благодаря постоянному обсуждению кода, вероятность обнаружения ошибок увеличивается. Это позволяет сделать программу более стабильной и надежной.
   4. Все участники команды в парном программировании принимают участие в принятии решений. Это позволяет избежать ошибок и лучше понимать проект.
4. Коллективное владение кодом (Collective Code Ownership) - это один из фундаментальных принципов Extreme Programming (XP), который предполагает, что все члены команды могут вносить изменения и улучшения в код. Этот принцип помогает распределить знания и ответственность в команде, а также улучшает качество и стабильность кода.
   1. Один из плюсов такого подхода - это то, что все члены команды могут знать о проекте как можно больше. Все имеют доступ к коду и могут его менять, что улучшает общение в команде, делает обмен опытом проще и позволяет быстрее реагировать на изменения и проблемы.
   2. Кроме того, такой подход помогает улучшить качество кода. Поскольку каждый член команды может предлагать свои идеи и исправления, ошибки легче находить и устранять, а код становится более понятным и читабельным.Коллективное владение кодом также способствует п
   3. Еще одно преимущество - это устойчивость процесса разработки. Если кто-то из команды временно отсутствует или уходит из проекта, другие могут легко взять на себя управление над его кодом и продолжить работу. Это делает команду менее зависимой от отдельных членов и обеспечивает непрерывность работы.
   4. Коллективное владение кодом также способствует повышению мотивации и ответственности. Каждый знает, что его вклад важен, и чувствует себя более вовлеченным и ответственным за результат.

Итак, принципы Extreme Programming обеспечивают эффективную работу команды разработки, способствуют достижению высокого качества продукта и ускоряют процесс доставки ценности продукта заказчику.