Колледж Автономной некоммерческой образовательной организации высшего образования

«Научно-технологический университет «Сириус»

**Практическая работа по предмету Введение в специальность (Информационные технологии)**

**«Agile-методологии»**

Работу подготовил:

Студент группы К0709-24/1

Лагутин М.Д.

Проверила:

преподаватель Яковлева С.В.

Оглавление

[**Введение** 3](#_Toc192622193)

[**1 Манифест Agile** 4](#_Toc192622194)

[**2 Ключевые принципы Agile** 6](#_Toc192622195)

[**3 Популярные Agile-методологии** 7](#_Toc192622196)

[**3.1 Scrum** 7](#_Toc192622197)

[**3.2 Kanban** 7](#_Toc192622198)

[**3.3 Extreme Programming (XP)** 7](#_Toc192622199)

[**Выводы** 9](#_Toc192622200)

[**Список литературы** 10](#_Toc192622201)

**Введение**

Что такое Agile? Agile — это набор принципов и практик, направленных на повышение эффективности работы команд через постоянное взаимодействие, адаптацию к изменениям и быстрое получение результатов. Эта методология управления проектами появилась как альтернатива традиционным каскадным моделям разработки, таким как Waterfall, которые предполагают строго последовательный процесс планирования, реализации и тестирования продукта. Гибкость подхода заключается в возможности корректировки целей проекта на любом этапе жизненного цикла, что позволяет быстро реагировать на изменения требований заказчика или рынка. Именно благодаря своей способности адаптироваться к изменениям требований и обеспечивать высокую скорость разработки этот подход и стал особенно популярным.

В IT-сфере Agile стал незаменимым инструментом для управления сложными проектами, обеспечивая высокую скорость разработки и быструю обратную связь от пользователей. Это помогает минимизировать риски, связанные с длительным циклом разработки, когда продукт создается на основе устаревших требований и потребностей клиентов.

В данной работе я подробно остановлюсь на применении Agile в IT-сфере, где этот подход зарекомендовал себя как один из наиболее эффективных инструментов для управления сложными проектами. Я рассмотрю различные методологии Agile, Манифест Agile и его ценности, ключевые принципы и преимущества.

**1 Манифест Agile**

Созданный группой разработчиков, манифест провозглашает четыре ценности:

* Люди и взаимодействие важнее процессов и инструментов — Agile ставит людей в центр процесса. Командная работа, коммуникация и доверие важнее строгого следования инструментам или бюрократическим процедурам. Процессы могут устаревать, а инструменты — тормозить работу. Люди, напротив, генерируют идеи, адаптируются к изменениям и решают задачи творчески.
* Рабочий продукт важнее исчерпывающей документации — Agile фокусируется на создании продукта, который приносит реальную ценность, а не на подготовке идеальной документации. Клиенты и пользователи оценивают результат, а не бумаги. Чрезмерная документация часто устаревает еще до завершения проекта.
* Сотрудничество с заказчиком важнее условий контракта — Agile делает ставку на партнерство с заказчиком, а не на строгое выполнение условий договора. Требования часто уточняются в процессе работы. Гибкость позволяет создать продукт, который действительно нужен, а не тот, который был задуман изначально.
* Готовность к изменениям важнее изначального плана — Agile принимает изменения как неотъемлемую часть разработки, даже на поздних этапах. Планы важны, но они должны быть адаптивными. Рынок, технологии и потребности пользователей постоянно меняются. Agile позволяет оставаться конкурентоспособным, быстро реагируя на эти изменения.

Стоит отметить, что эти ценности не отвергают процессы, документацию, контракты или планирование. Они лишь ставят приоритет на гибкость вместо жесткости, практическую ценность вместо формализма, доверие и взаимодействие вместо иерархии. Эти принципы особенно важны в условиях неопределенности, где традиционные подходы (например, Waterfall) могут привести к провалу из-за неспособности быстро меняться.

# **2 Ключевые принципы Agile**

* Итеративная разработка - вместо того чтобы строить продукт "от начала до конца" за один этап (как в Waterfall), Agile разбивает работу на короткие циклы (итерации или спринты). Каждая итерация добавляет законченную часть функционала.
* Самоорганизация команд - Agile-команды сами решают, **как** выполнить задачу, вместо того чтобы ждать указаний от руководителя. Это повышает вовлеченность и скорость принятия решений. Не нужно ждать одобрения менеджера для каждого шага, а также участники чувствуют себя владельцами продукта, а не "исполнителями приказов".
* Регулярная обратная связь от заказчика - заказчик (или его представитель) постоянно участвует в проекте: проверяет промежуточные результаты, дает фидбек и корректирует требования. Если заказчик недоволен, это выявляется на ранних этапах, а не перед сдачей проекта.
* Готовность менять требования на любом этапе - Agile принимает, что требования могут меняться даже на поздних стадиях. Это не "проблема", а возможность создать лучший продукт. Благодаря этому продукт соответствует актуальным потребностям рынка. Тем более, лучше изменить направление на ранней стадии, чем выпустить устаревший продукт.

**3 Популярные Agile-методологии**

**3.1 Scrum**

Scrum — это популярная методология управления проектами, основанная на принципах Agile, которая ориентирована на разработку продуктов короткими итерациями, называемыми спринтами. Основной акцент делается на регулярную поставку рабочей версии продукта, самоорганизацию команды и адаптацию к изменениям.

* Проект делится на небольшие временные интервалы, называемые спринтами. Обычно один спринт длится от 2 до 4 недель. В конце каждого спринта команда должна представить готовый к использованию продукт или его часть (инкремент).
* Роли: Scrum-мастер (Фасилитатор, убирающий препятствия и обеспечивающий соблюдение процесса), продукт-овнер (Определяет цели продукта, управляет бэклогом, приоритезирует задачи), команда разработки.
* Артефакты: бэклог продукта (список всех требований и функций, которые нужно реализовать для достижения целей проекта), спринт-бэклог (план работы на конкретный спринт, включающий задачи, выбранные командой из бэклога продукта), инкремент (готовая часть продукта)
* Встречи: планирование (встреча перед началом нового спринта, где команда выбирает задачи из бэклога продукта и планирует свою работу на предстоящие несколько недель), дейли-митинги (ежедневные 15-минутные встречи, на которых каждый член команды рассказывает о проделанной работе, планах на день и возможных препятствиях), ретроспективы (обсуждение прошедшего спринта с целью анализа эффективности работы команды и поиска возможностей для улучшения процессов).

**3.2 Kanban**

Kanban — это методология управления проектами, которая акцентирует внимание на непрерывной работе над задачами и стремится минимизировать количество незавершенных дел (Work In Progress). Этот подход отличается от Scrum тем, что не привязан к жестким временным итерациям. Kanban лучше всего подходит для проектов с постоянно меняющимися приоритетами и гибкими процессами. Основные принципы Kanban включают:

* Визуализация процессов через доску: для наглядного представления текущего состояния проекта используется доска Kanban, которая делится на несколько колонок, отражающих этапы работы. Например, такие столбцы, как "Запланировано", "В работе", "Тестирование" и "Готово". Это позволяет всей команде видеть прогресс по каждому этапу разработки.
* Ограничение работы в процессе (WIP): один из ключевых аспектов Kanban заключается в установлении лимита на количество задач, одновременно находящихся в процессе выполнения. Это помогает избежать перегрузки команды и способствует повышению производительности. Как только одна задача выполнена, можно приступать к следующей, соблюдая установленные ограничения.
* Непрерывный поток задач без фиксированных спринтов: в отличие от Scrum, где работа ведется в рамках временных спринтов, Kanban предполагает отсутствие фиксированного цикла планирования. Команда непрерывно работает над новыми задачами, когда завершаются предыдущие, что обеспечивает высокую адаптивность процесса.

**3.3 Extreme Programming (XP)**

Extreme Programming (XP) — это методология разработки программного обеспечения, которая помогает командам создавать качественный код, быстро адаптироваться к изменениям и минимизировать количество ошибок. XP фокусируется на повышении эффективности и гибкости разработки, а также на улучшении взаимодействия между разработчиками и заказчиками. Основные принципы XP включают:

* Парное программирование: два программиста работают за одним компьютером: один пишет код, другой проверяет каждую строчку. Это позволяет значительно снизить количество ошибок и улучшить качество кода. Проверяющий программист не только следит за правильностью кода, но и предлагает улучшения, а также помогает избежать ошибок, которые могут возникнуть из-за усталости или невнимательности. Парное программирование также способствует обмену знаниями и опытом между членами команды.
* TDD (Test-Driven Development): перед написанием кода разработчики создают автоматические тесты, которые проверяют, как должна работать функция. Это позволяет убедиться, что код работает правильно с самого начала. Тесты пишутся до того, как будет написан сам код, что помогает разработчикам сосредоточиться на требованиях и функциональности. Если код не проходит тесты, он дорабатывается до тех пор, пока не начнет соответствовать ожиданиям. Это также помогает быстро выявлять и исправлять ошибки при внесении изменений в код.
* Рефакторинг: постоянное улучшение, упрощение и оптимизация кода без изменения его функционала. Рефакторинг помогает поддерживать код в чистоте и легкости для понимания, что делает его более устойчивым к изменениям и новым требованиям. Чем чище и проще код, тем легче его поддерживать и вносить в него изменения. Это также снижает вероятность появления ошибок и упрощает процесс разработки.
* Короткие релизы: новые версии продукта выпускаются каждые 1–2 недели. Это позволяет клиентам быстрее получать обновления и видеть результаты работы команды. Короткие циклы разработки помогают команде быстрее получать обратную связь от пользователей и адаптироваться к их потребностям. Это также снижает риск того, что команда будет работать над функциями, которые в итоге окажутся ненужными.
* Присутствие клиента: заказчик или его представитель активно участвует в процессе разработки, отвечает на вопросы команды и тестирует промежуточные результаты. Это позволяет команде быстро реагировать на изменения требований и подстраиваться под новые идеи. Постоянное взаимодействие с клиентом помогает избежать недопонимания и обеспечивает, что продукт соответствует ожиданиям и потребностям конечных пользователей.

**Выводы**

Таким образом, Agile-методологии предоставляют компаниям эффективный инструмент для организации работы, способствующий значительному улучшению взаимодействия внутри команды, постоянному повышению качества конечного продукта и существенному сокращению сроков его выхода на рынок. Это становится критически важным преимуществом для организаций, стремящихся сохранить и укрепить свои позиции в условиях стремительно развивающегося и постоянно меняющегося технологического ландшафта. Гибкость и адаптивность, заложенные в основу Agile-подходов, позволяют компаниям оперативно реагировать на внешние вызовы, такие как изменение рыночных условий, новые требования клиентов или технологические инновации. Благодаря этому, Agile-методологии становятся неотъемлемой частью стратегий успешных компаний, нацеленных на долгосрочный рост и устойчивое развитие в условиях высокой конкуренции.

**Список использованных источников**

1. Официальный сайт Манифеста Agile: <https://agilemanifesto.org/>. Дата обращения: 9.03
2. Официальное руководство по Scrum: <https://www.scrumguides.org/>. Дата обращения: 10.03
3. Kanban – Дэвид Андерсон. Дата обращения: 11.03
4. Extreme Programming Explained: Embrace Change – Кент Бек. Дата обращения: 11.03
5. Agile Project Management with Scrum – автор Кен Швабер. Дата обращения: 10.03
6. Lean Software Development: An Agile Toolkit – Мэри и Том Поппендик. Дата обращения: 9.03
7. Agile Estimating and Planning – Майк Кон. Дата обращения: 9.03