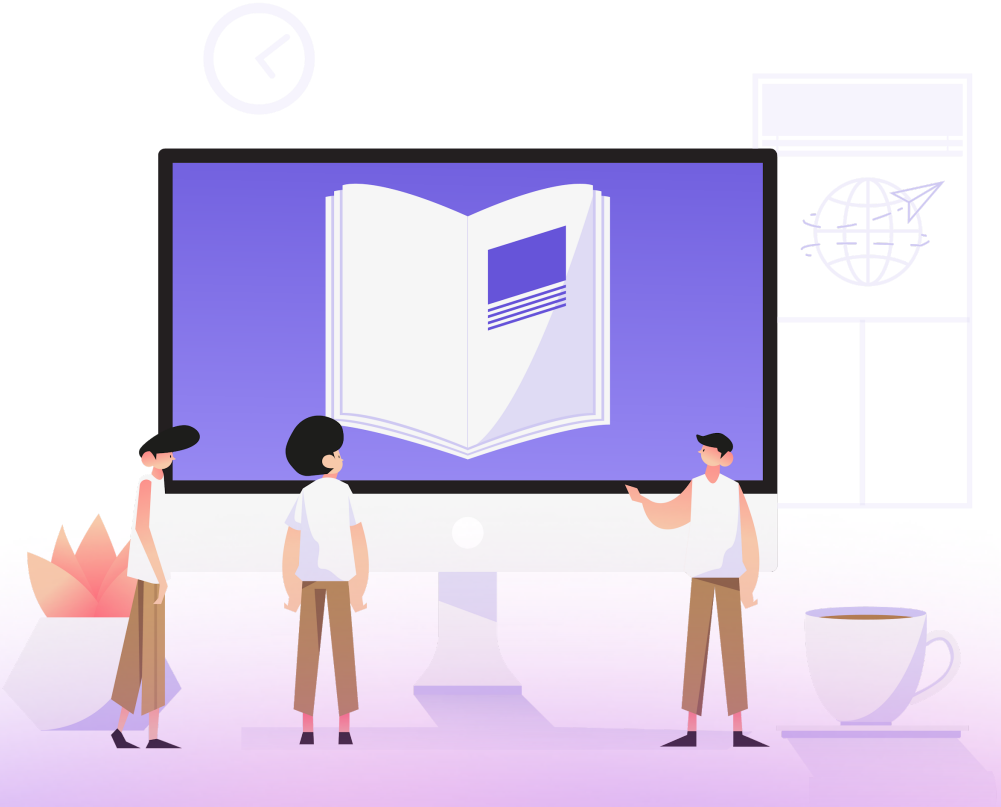
GeekBrains

Командная разработка по Agile-концепции с использованием Scrum.

Введение в Agile



На этом уроке

1. Дадим определения понятиям «продукт» и «проект».
2. Узнаем основные принципы и преимущества Waterfall — классического подхода к реализации проектов.
3. Ознакомимся с основными предпосылками зарождения Agile.
4. Ознакомимся с понятием Agile: дадим определение, разберём суть идеологии и 12 принципов гибкой разработки ПО.
5. Оценим выгоды от применения Agile.
6. Разберёмся, в каком случае Agile не подходит для использования на проекте.

Оглавление

Введение

Цель теоретического блока — получение необходимых знаний о командной работе и проектной деятельности по концепции Agile с применением фреймворка Scrum. На первых двух уроках мы рассмотрим историю возникновения, основные идеи, принципы и самые популярные методы организации работ в команде с использованием Agile-концепции. Третий и четвёртый уроки будут посвящены подробному разбору фреймворка Scrum и подготовке к его использованию на практике.

Первый урок начнём с определения того, что мы будем понимать под терминами «продукт» и «проект», затем перейдём к изучению классического подхода к реализации проектов и основным предпосылкам появления Agile. Вторая часть урока посвящена определению и подробному разбору Agile-концепции.

Продукт и проект

Продукт — это результат деятельности, который имеет определённую ценность и может быть материальным и нематериальным.

Материальный продукт — это физический объект, который может восприниматься на ощупь: здание, транспортное средство, одежда, гаджет или произведение искусства.

Нематериальным продуктом может быть услуга, идея (патент), результат научной деятельности в виде документа или программное обеспечение, имеющее цифровое выражение.

В рамках нашего курса под продуктом мы будем понимать программный продукт, например веб-сайт или мобильное приложение, который имеет самостоятельную ценность и отвечает потребностям конечного пользователя.

Программный продукт, как и любой другой продукт, появляется в результате деятельности, которую принято называть проектом.

Проект (англ. project от лат. projectus — «брошенный вперёд, выступающий, выдающийся вперёд») — это временная деятельность, которая направлена на достижение определённой цели: создание уникального продукта, услуги или результата.

Проекты бывают разными и имеют сложную классификацию, в основе которой лежит множество признаков. Интересующие нас проекты по разработке программного продукта относятся к IT-сфере и могут быть:

- маленькими и большими в зависимости от бюджета;
- простыми и сложными в зависимости от объёма работ;
- краткосрочными, среднесрочными и долгосрочными в зависимости от временных рамок начала и завершения;
- проектами разработки, внедрения, развития или поддержки по виду деятельности;
- внутренними, внешними или собственными (стартап) с точки зрения заказчика.

Независимо от того, с каким проектом вы столкнётесь в рамках своей профессиональной деятельности, он будет обладать рядом характеристик:

1. Носить временный характер. Любой проект имеет чёткие временные рамки: начало и завершение. Завершение наступает, когда:
    a. достигнуты цели проекта;
    b. или признано, что цели проекта не будут или не могут быть достигнуты;
    c. или исчезла необходимость в проекте.
2. Иметь установленную цель: чётко сформулированную и измеряемую, которая должна быть достигнута к концу проекта.
3. Обладать некоторой степенью уникальности, которая будет проявляться в работе над проектом или его результате.
4. Иметь ограниченные ресурсы: финансовые, человеческие, материальные.
5. Обладать специфической для проекта организацией работ: в проекте чаще всего участвуют специалисты с разными ролями, которые образуют команду и работают по определённым правилам.

У проекта также есть жизненный цикл, который всегда развивается по одному и тому же принципу:

1. Инициация: рождение идеи стартапа или получение заказа от клиента. Определение цели проекта, объёма и бюджета.
2. Планирование: формирование команды, организация процесса, определение сроков, оценка работ.
3. Выполнение работ — разработка.
4. Завершение проекта: передача продукта заказчику или вывод на рынок.
5. Чаще всего есть ещё одна стадия — поддержка и сопровождение.

Где заканчивается жизненный цикл проекта, там только начинается жизненный цикл его результата — программного продукта. Жизненный цикл программного продукта тоже конечен, но может быть гораздо длиннее проекта по его созданию и впоследствии может включать в себя множество других проектов по развитию.

С момента появления проектов по разработке программных продуктов IT-специалисты начали создавать и применять на практике различные подходы к управлению процессами в ходе жизненного цикла проекта. Они позволяют повысить эффективность и как можно быстрее получить качественный результат — продукт, готовый к использованию и выполнению своего главного предназначения: принесения прибыли или оптимизации затрат в случае с корпоративным программным обеспечением.

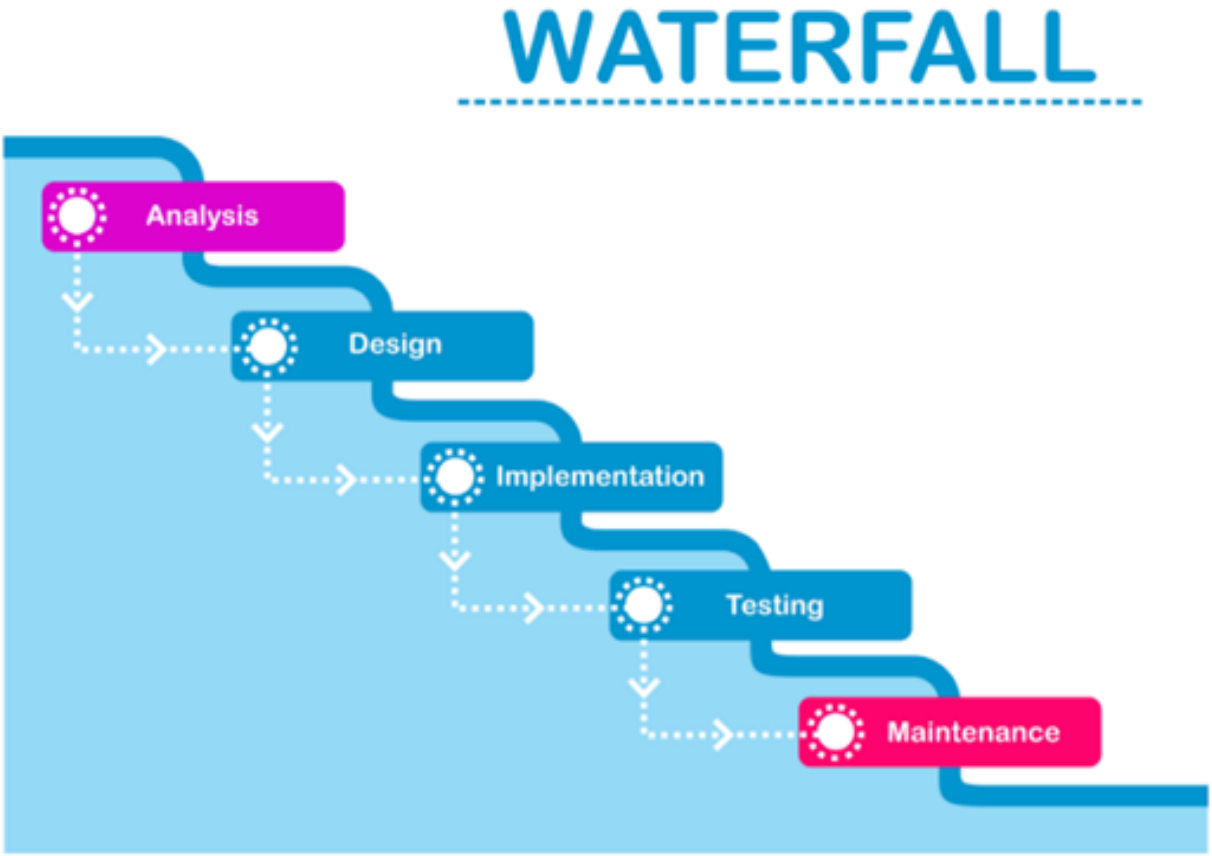
Одним из основных подходов к реализации IT-проектов стал каскадный метод или Waterfall. Его теперь принято называть традиционным. Долгие годы и даже десятилетия этот подход был основной методологией для реализации проектов в большинстве компаний мира.

Методология Waterfall

Методология — учение о методах, методиках, способах и средствах познания. В случае с разработкой программного продукта это способ реализации проектов, образующий систему организации процессов и имеющий определённые характеристики.

Каскадная модель (англ. waterfall model, модель «Водопад») — модель процесса разработки программного обеспечения, суть которой заключается в делении проекта на этапы и последовательной их реализации по принципу водопада:

- с отсутствием возможности вернуться в начало;
- с нарастающим накоплением мощности (стоимости и ценности);
- с получением общего результата проекта к завершению процесса.



В классической модели разработки программных продуктов выделяют следующие этапы:

1. **Подготовка проекта и анализ**. На первом этапе необходимо определиться с целью проекта, требованиями, сроками, составом команды и бюджетом.
    a. Команда собирает функциональные требования к продукту или анализирует полученные от заказчика документы.
    b. Делает уточнения и определяет общие технические требования, архитектуру, требования к безопасности.
    c. Создаёт подробное и исчерпывающее ТЗ.

    d. Делает оценку проекта с учётом всевозможных рисков.
    e. Создаёт календарный план-график работ с ключевыми датами проекта, началом и окончанием всех этапов.
2. **Проектирование.** Этот этап отвечает за подробное описание того, каким образом будет реализован программный продукт:
    a. Уточняются технические детали и инструменты: платформа, язык программирования, как будет взаимодействовать продукт с серверами, станут ли использовать API, какой будет логика внешнего и внутреннего интерфейса и т. д.
    b. Создаются схемы и фиксируется описание архитектуры, связи между компонентами.
    c. Продумывается интеграция, миграция данных и критерии безопасности: будет ли использоваться HTTPS, SSL-шифрование.
    d. Описываются роли пользователей и их полномочия: клиент, администратор, менеджер.
    e. Формулируются критерии надёжности, производительности и дальнейшей техподдержки продукта.
    f. Чаще всего результат этого этапа — прототип, дизайн-макеты и вся сопутствующая документация: обновлённое ТЗ, проектное решение по архитектуре и интеграции, подробный календарный план и т. д.
3. **Реализация.** На этом этапе пишут код продукта согласно макетам и требованиям, описанным в документации, со строгим соблюдением календарного плана. Результатом этапа становится продукт, готовый к тестированию, со всей сопутствующей документацией: спецификациями, сценариями тестирования, пользовательскими инструкциями и т. д.
4. **Тестирование и отладка продукта**. Тестовая эксплуатация продукта с целью оценки работоспособности и соответствия заявленным требованиям. Может включать в себя различные виды тестирования в зависимости от продукта: unit-тестирование, alfa- и beta-тестирование, нагрузочное тестирование, интеграционное тестирование, приёмочное тестирование и т. д. В случае выявления ошибок происходит их учёт и исправление. Результатами этапа должны быть готовый к запуску продукт, который прошёл приёмочное тестирование на ограниченном количестве пользователей, а также обновлённая проектная документация.
5. **Запуск и поддержка.** На этом этапе продукт запускается в коммерческую эксплуатацию для широкой аудитории пользователей. Чаще всего клиенты предпочитают подстраховаться и заручиться технической поддержкой на случай обнаружения ошибок или сбоя в работе продукта.

Также могут встречаться вариации набора этапов. Например, в случае работы нескольких команд над разными частями функционала может быть выделен этап **интеграции** для объединения компонентов, тестирования и настройки их совместной работы.

Все проекты, реализованные по традиционной каскадной методологии, объединяют следующие принципы:

1. Определённый объём работ, выраженный в перечне требований и зафиксированный документально.
2. Последовательное выполнение этапов без возможности их полного повторения.
3. Отношения между заказчиком и исполнителем строго формализованы и часто фиксируются в договоре или специальном документе.

4. Вертикальная иерархическая структура команды с соблюдением субординации.
5. Командно-административное управление: распределение обязанностей и ответственности, назначение задач и контроль работ.
6. Объёмная и сложная документация.
7. Строгий план-график.
8. Управление рисками: отслеживание изменений, которые влияют на срок и объём проекта.

Преимущества каскадной модели

1. Чётко выстроенная и прозрачная организация работ понятна всем участникам проекта.
2. Есть относительная стабильность требований и объёма работ.
3. Дисциплина, основанная на фиксированных сроках выполнения задач и контроле исполнения.
4. Устойчивость к изменению кадрового состава: благодаря тому, что ведётся подробная проектная документация, допускается, что состав команды на протяжении жизненного цикла проекта может меняться и это не повлияет на сроки исполнения проекта. Например, сбором требований и проектированием могут заниматься одни специалисты, а реализацией — другие. Очень часто допускается привлечение новых участников в команду на этапе тестирования, отладки и поддержки.
5. Подробная документация позволяет создавать базу знаний и лучших практик для последующих проектов.
6. Принятая в каскадной модели отчётность и визуализация процесса в виде диаграммы Ганта позволяет в любой момент показать заказчику или руководству, на какой стадии находится разработка продукта.
7. Зафиксированный объём работ позволяет контролировать бюджет проекта.
8. Чётко определены процедуры по контролю качества.
9. Контроль над изменениями позволяет минимизировать риски, связанные со срывом дедлайнов или расширением объёма проектов.
10. Формализация отношений между клиентом и исполнителем регулирует порядок взаимодействия в тех или иных спорных ситуациях на проекте.

Перечисленные преимущества позволяют использовать каскадную модель многие десятилетия и по сей день для реализации крупномасштабных проектов по внедрению и тиражированию стандартного ПО для корпоративных нужд организаций из разных сфер и государственной системы. Однако в связи с очень активным развитием ПО и интернета традиционная модель стала несостоятельной для многих проектов, связанных с современными информационными технологиями и инновациями.

Предпосылки появления гибкого подхода к реализации проектов

Основной причиной, по которой традиционная модель не отвечала новым потребностям в разработке ПО, стало противостояние изменениям в процессе реализации и отсроченный во времени результат, который мог не успеть за развивающимися технологиями и устареть до выпуска продукта. Кроме того, каскадная модель в силу строгой фиксации объёма и сроков требует больших усилий по управлению рисками и жёсткого контроля над исполнителями, что очень часто приводит к снижению мотивации и работоспособности сотрудников. Перечисленные факторы и отсутствие достаточной вовлечённости конечных пользователей в процесс реализации проекта могли привести к тому, что результат не соответствовал ожиданиям и потребностям клиента, терял свою ценность.

Новые проекты требовали скорости отработки гипотез и поиска нестандартных решений. Важно было как можно быстрее получить продукт, максимально удовлетворяющий потребности конечного пользователя. Это стало основной предпосылкой для поиска новых методов организации процесса разработки и подходов к управлению проектами.

Считается, что описание Waterfall-методологии впервые появилось в 1970 году в статье Винстона Ройса (Dr. Winston D. Royce, директора Lockheed Software Technology Center) Managing the development of large software system, поэтому именно его стали называть автором этого подхода. Однако часто не учитывают, что поводом для статьи был назревший в то время первый кризис программного обеспечения, и в своей статье Ройс говорил, как можно адаптировать традиционную модель реализации проектов к новым условиям.

Ознакомившись с содержанием статьи Ройса, можно сделать вывод, что он предусматривал обратные связи между этапами на одном уровне: к примеру, дизайн-кодирование, кодирование-тестирование и т. д. Речь идёт о параллельных работах по двум последовательным этапам, что даёт возможность на более ранних стадиях выявить ошибки предыдущего этапа жизненного цикла. Это допущение устраняет один из весомых недостатков каскадной модели — невозможность возврата на предыдущий этап — и предполагает, пусть и не в чистом виде, использование итеративного подхода, который позже стал одним из принципов гибкой концепции.

Предпосылками для появления нового гибкого подхода в реализации проектов можно также считать революционные перемены в самой организации ведения бизнеса и труда, которые произошли в Японии в середине XX века. Можно предположить, что некоторые важные принципы, вошедшие в основу философии гибких методологий, были заимствованы из системы управления качеством, одним из авторов которой был Эдвард Деминг. Она была известна по всему миру как цикл Деминга-Шухарта «Планируй-Делай-Изучай-Действуй» (Plan-Do-Study-Act) и включала основные постулаты производственной системы компании Toyota (англ. Toyota Production System — TPS).

Деминг начал активно популяризировать свою систему в Японии во время восстановления японской промышленности после окончания Второй мировой войны. Он показал, как путём применения соответствующих принципов управления организации могут существенно повысить качество и

одновременно снизить затраты за счёт сокращения количества отходов, переделывания, изнурения персонала, судебных разбирательств при одновременном повышении лояльности клиентов. Суть подхода заключалась в том, чтобы практиковать непрерывное совершенствование (кайдзен) и рассматривать производство как единую систему, а не как отдельные части.

Деминг сформулировал 14 управленческих принципов, которые касались не только правил оптимизации производства и улучшения качества, но и взаимоотношений между сотрудниками:

- убрать границы между подразделениями и формировать команды, состоящие из сотрудников разной специализации;
- приветствовать лидерство, стремление к образованию и ввести обучение и повышение квалификации для сотрудников на постоянной основе;
- отказаться от командно-административного управления, тотального контроля, проверок и аттестаций, что поможет искоренить страх сотрудников перед руководством, повысить эффективность труда, мотивацию и инициативность.

Перечисленные выше тезисы, как и итеративный подход, предложенный Ройсом, нашли своё отражение в принятых позже принципах гибкой концепции.

Производственная система компании Toyota не только подарила миру понятие поточного и бережливого производства, но и сыграла, возможно, самую существенную роль в появлении новой философии управления командами и проектами, в том числе подарив такую технику организации процесса разработки, как Kanban.

Toyota сформулировала собственные 14 универсальных принципов управления, и вот основное, на что нам стоит обратить внимание:

1. Организовать процесс в виде непрерывного потока. Речь идёт не только об организации потокового производства, но и о непрерывном обмене информацией, непрерывном совершенствовании и развитии.
2. Равномерно распределять объём работ и нагрузки (хейдзунка — «выравнивание»), это призвано устранить череду авралов и простоев.
3. Использовать визуальный контроль, чтобы ни одна проблема не осталась незамеченной, путём создания простых систем визуального контроля на рабочих местах.
4. Формировать межфункциональные группы, чтобы повысить качество, производительность и усовершенствовать процесс.
5. Неустанно обучать людей работать в команде на общую цель. Освоить работу в команде должен каждый.
6. Немаваси — это процесс совместного обсуждения проблем и потенциальных решений, в котором участвуют все. Его задача — собрать все идеи и выработать единое мнение, куда двигаться дальше.
7. Стать обучающейся структурой за счёт неустанного самоанализа (хансей) и непрерывного совершенствования (кайдзен).

Эти принципы органично сочетают важнейшие организационные и мотивационные аспекты командной работы, последовательное применение которых позволило компании Toyota приумножить капитал и поддерживать корпоративную культуру здоровой и эффективной на протяжении десятилетий. Неудивительно, что эти принципы также легли в основу нового гибкого подхода.

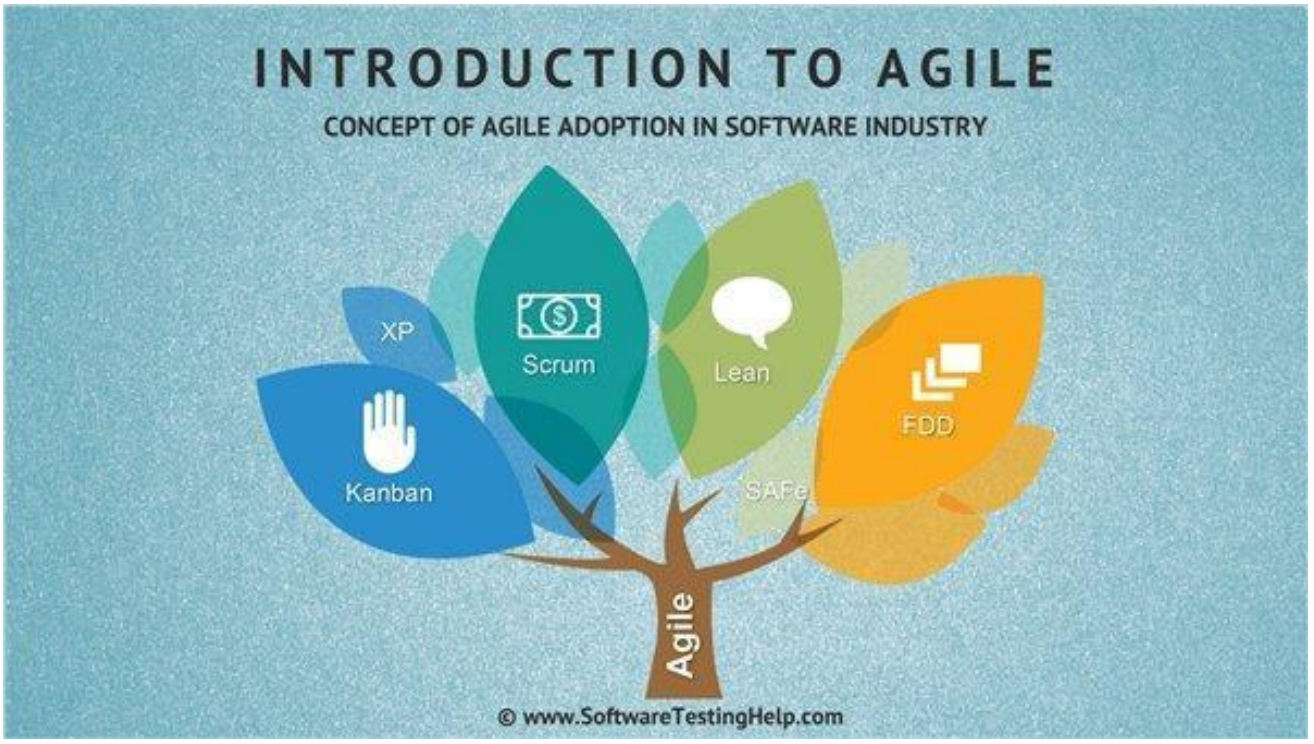
Несмотря на то, что ещё в середине XX века были выработаны столь очевидные на сегодняшний день принципы организации рабочего процесса и взаимодействия команд, потребовалось не одно десятилетие на эволюцию подхода к управлению проектами. Только в 90-х годах XX века появились Scrum, XP и многие другие методы, принципы которых в начале XXI века были объединены в концепцию, которой дали название Agile.

Agile: определение, основные принципы

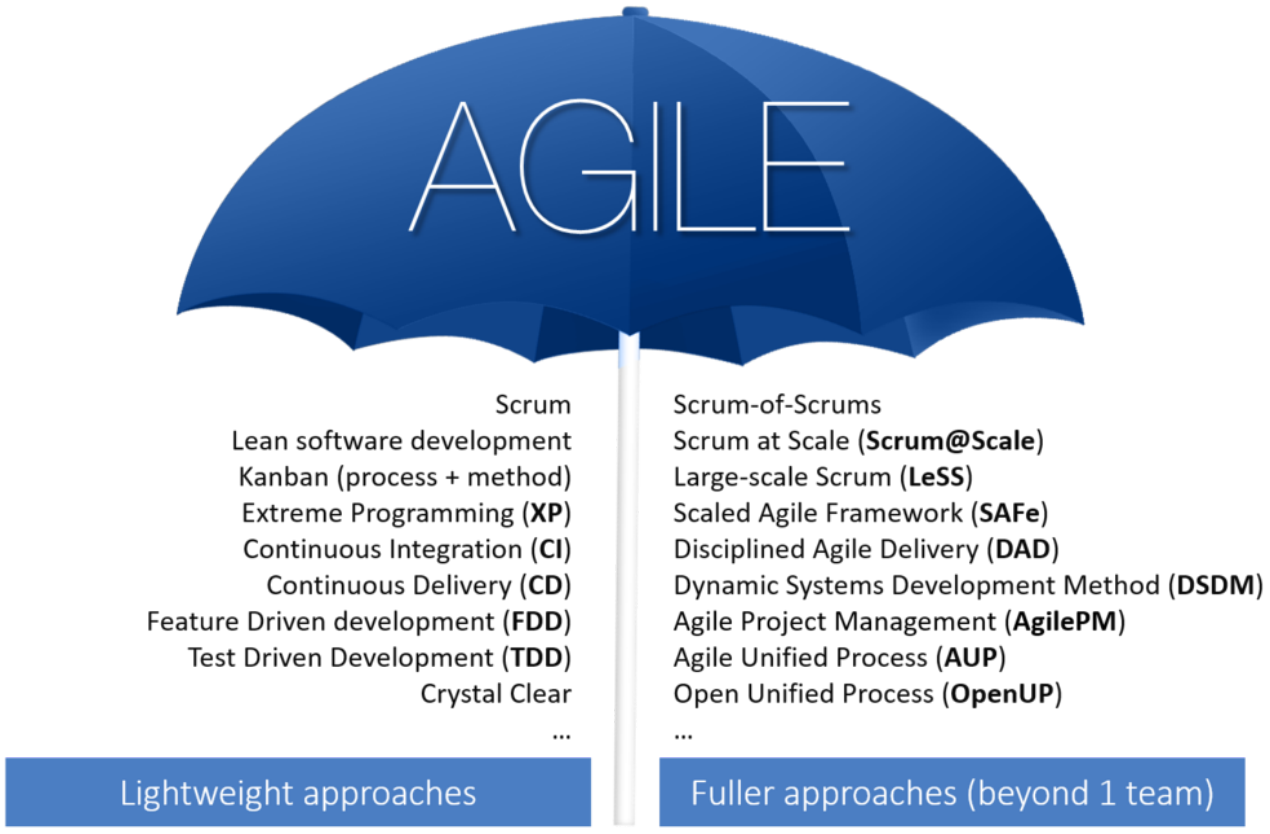
В 2001 году 17 единомышленников встретились на горнолыжном курорте Сноубёрд (Snowbird, Utah, USA) недалеко от Солт-Лейк-Сити, чтобы поделиться идеями повышения эффективности при разработке программного обеспечения и найти решение, позволяющее избежать проблем, с которыми они сталкивались на протяжении всей своей карьеры. В группу вошли авторы и приверженцы уже существовавших на тот момент гибких методов разработки и фреймворков. Среди них был Джефф Сазерленд (автор Scrum) и представители других конкурирующих подходов: экстремального программирования (XP), Crystal, Adaptive Software Development (ASD), Feature Driven Development (FDD) и Dynamic Systems Development Method (DSDM). В результате этой встречи были сформулированы главные идеи, которые впоследствии стали основой всей философии, и главные принципы гибких подходов к разработке программного обеспечения (ПО). Появились название концепции (Agile) и документ — Agile-манифест, который объединил и зафиксировал все тезисы.

Английское слово agile имеет несколько значений в зависимости от контекста: «гибкий», «проворный», «юркий», «подвижный», «ловкий», «быстро переналаживаемый», «шустрый», «сообразительный» (agile mind — «живой ум»). Это великолепно передаёт суть Agile-концепции.

Важно отметить, что Agile или гибкие методологии разработки (англ. Agile Software Development) — это обобщающий термин для целого ряда подходов и практик, основанных на ценностях Agile-манифеста. Эти подходы называют фреймворками, методами, методиками или гибкими методологиями и на инфографиках изображают в виде дерева, где Agile — ствол, а ветви — разные методики и фреймворки:



Также семейство Agile схематично можно изобразить в виде зонтика, под которым «прячутся» гибкие методики и фреймворки:



Список основных фреймворков, методов и методик семейства Agile

Ниже перечислены основные методики и фреймворки семейства Agile, которые используются как при организации работы отдельных команд (Lightweight approaches), так и при масштабировании Agile-подхода на целую организацию (Fuller approaches beyond 1 team).

Лёгкие подходы семейства Agile (Lightweight approaches) используют простые правила и процессы для быстрой адаптации к изменяющейся среде:

1. Scrum.
2. Kanban.
3. Scrumban.
4. Lean software development.
5. Extreme Programming (XP).
6. Continuous Integration (CI).
7. Continuous Delivery (CD).
8. Feature Driven development (FDD).
9. Test Driven development (TDD).
10. Crystal clear.

Комплексные подходы семейства Agile для нескольких команд (Fuller approaches beyond 1 team) отвечают за выстраивание систем взаимодействия нескольких команд внутри организаций или на сложных крупных проектах.

Перечисленные ниже более сложные модели разработки ПО также часть семейства Agile:

1. Scrum-of-Scrums.
2. Scrum at Scale (Scrum@Scale).
3. Large-scale Scrum (LeSS).
4. Scaled Agile Framework (SAFe).
5. Disciplined Agile Delivery (DAD).
6. Agile Unified Process (AUP).
7. Dynamic Systems Development Method (DSDM).
8. Agile Project Management (AgilePM).
9. Open Unified Process (OpenUP).

Философия и основные принципы Agile

Все вышеперечисленные методы, методики и фреймворки, которые относят к семейству Agile, объединяет общая философия, которая опирается на четыре главных идеи и двенадцать основных принципов, зафиксированных в Agile-манифесте. Он вобрал в себя всё лучшее, что когда-то сформулировали его предшественники.

Основывая свои выводы на недостатках традиционного каскадного подхода, авторы манифеста подчеркнули, что не надо отрицать важности используемых инструментов, чётко выстроенных процессов, следования первоначальному плану, исчерпывающей проектной документации и согласования условий договора. Однако гораздо важнее ценить людей и взаимодействие между ними, сотрудничество между командой и клиентом, готовность к изменениям. И не забывать, что главный результат работы — это работающий продукт.

4 главные идеи Agile-философии

Эффективная работа с Agile начинается с понимания и принятия основных ценностей. Разберём подробнее, какой смысл авторы вкладывают в каждую из идей.

«Люди и взаимодействие важнее процессов и инструментов»

В первую очередь необходимо сосредоточиться на людях и их общении, а затем уже обратить внимание на методы и инструменты, которыми они пользуются. Частое заблуждение командно-административных руководителей состоит в том, что чётко выстроенные предсказуемые процессы и подбор правильных инструментов в сочетании с регламентированным порядком взаимодействия и должным контролем

повысят эффективность работы команды. На практике оказывается, что это снижает мотивацию участников, лишает их свободы выбора и действий, приводит к неэффективной коммуникации, слепому следованию указаниям и, как результат, к некачественному продукту.

Намного важнее выстраивать для команд безбарьерную коммуникативную среду, основанную на сотрудничестве и доверии, создав условия для обмена мнениями и поиска лучших решений. Важно позволить участникам самостоятельно руководить процессом и поощрять развитие, инициативу и лидерство. Именно свобода выбора и действий повышает уровень ответственности и позволяет увеличить эффективность команд.

«Работающий продукт важнее исчерпывающей документации»

Своевременная поставка работающего продукта, который полностью удовлетворяет потребности заказчика и выполняет свою главную коммерческую функцию, гораздо важнее подробной спецификации к нему. Привести в порядок документацию в конце проекта намного проще, чем доработать некорректно работающее ПО, когда все сроки вышли.

Документация важна для накопления базы знаний, лучших практик и передачи опыта. Это материальный результат труда и юридическое подтверждение выполненных работ. Во всех компаниях, а особенно в крупных организациях, есть список необходимой проектной документации, шаблоны и регламент оформления.

Порой необходимой документации так много, что участники команды, включая разработчиков, тратят до 30% проектного времени на подготовку проектной документации. Часто это обусловлено бюрократической системой и по факту большая часть документации избыточна, не несёт никакой ценности для проекта. В работе над таким объёмом документации очень часто смещается фокус внимания с главной цели любого проекта — создания работающего продукта, ведь первоочередной задачей становится сдача документации. К сожалению, многие компании пока не могут изменить эту ситуацию, и чаще всего это связано с политикой безопасности или сложной системой отчётности, особенно в компаниях с государственным участием.

Авторы идеи говорят, что если наличие большого объёма сложной документации не предусмотрено обязательными правилами организации, то Agile-команда вправе сама выбирать, каким образом и в каком количестве вести документацию по проекту. Главное — не терять фокус и сосредоточиться на достижении общей цели проекта, на создании качественного работающего продукта.

Неверно, что Agile призывает вовсе отказаться от документации. Многие документы по-прежнему остаются полезными для проекта, например своей актуальности не теряет ТЗ, которое помогает понять идею и потребности клиента, а также каркасная визуализация архитектуры, диаграммы последовательности и всё то, что призвано помочь в создании продукта, а не отнять время на разработку.

«Сотрудничество с заказчиком важнее согласования условий контракта»

В классической модели организации работ на проекте чаще всего работают команда заказчика и команда исполнителя, отношения между которыми регулируются условиями договора, и все вопросы решаются путём переговоров на основе этого юридического документа. Agile призван изменить такой порядок взаимодействия и предлагает рассматривать клиента и исполнителя как единую команду, которая преследует общую цель, где важнее не заботиться о том, чтобы защитить себя или отстоять свои права, а сотрудничать друг с другом в атмосфере дружелюбия и равноправия. Продукт, разработанный при постоянном участии клиента в процессе реализации, всегда будет отвечать заявленным требованиям и соответствовать коммерческой ценности.

По замыслу авторов, «контракт» в данном случае имеет и более широкое значение, чем договор в виде документа. Речь идёт и о договоренностях и регулировании взаимоотношений между подразделениями и отделами внутри компании, которые порой препятствуют сотрудничеству не меньше, чем условия юридического контракта. Необходимо помнить, что в Agile живое общение между людьми ценится выше соблюдения бюрократических формальностей, потому что это в разы увеличивает эффективность коммуникации.

«Готовность к изменениям важнее следования первоначальному плану»

Эта идея имеет непосредственное отношение к гибкости, которой должны обладать компания и команда. В современном мире всё настолько быстро меняется, что проявлять гибкость — это необходимость. Если обстоятельства меняются, значит, проекту нужен новый план. Не противостоять изменениям в попытке во что бы то ни стало следовать первоначальному плану в ущерб качеству, а принять перемены и адаптироваться к ним — вот одна из главных ценностей Agile.

12 принципов Agile-разработки

Перечисленные выше идеи — основа Agile-философии, на которую опираются 12 принципов Agile-разработки, призванные указать направление работы и дать понимание методов, которые стоит использовать. Эти принципы можно разделить на три группы: взаимоотношения с клиентом, организация процесса и требования к команде.

Взаимоотношения с клиентом

1. **Наивысший приоритет для нас — удовлетворение потребностей заказчика** благодаря регулярной и ранней поставке ценного программного обеспечения.

   В первую очередь этот принцип связан с получением обратной связи, по которой можно оценить соответствие продукта ожиданиям и требованиям клиента. Команда должна стремиться к ранней поставке результата своей работы, даже если это только одна реализованная функция, и делать это на регулярной основе, чтобы не потерять связь с клиентом.

2. **Изменение требований приветствуется даже на поздних стадиях разработки**. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

   Это один из самых сложных принципов, который очень часто сталкивается с сопротивлением не только на уровне руководства, которое отвечает за соблюдение сроков и объём работ, но и часто со стороны участников команды, которой необходимо внести определённые правки или переделать целый кусок кода. Чаще всего изменения связаны с непредвиденными обстоятельствами, которые нельзя было учесть в начале проекта или при планировании. Это может вызвать огорчение, но принять это легче, чем изменения, которые связаны с человеческим фактором: например, если клиент что-то упустил или осознал, что бизнес-процесс или функция требует некоторой доработки. Если такие изменения приводят к переписыванию кода или, ещё хуже, к изменению архитектуры всего продукта, разработчик однозначно будет недоволен и конфликт интересов неизбежен.

   Всем участникам проекта следует изменить своё отношение к переменам и перестать относиться к этому как к проблеме или обесцениванию проделанной работы. Необходимо чаще смотреть на ситуацию с точки зрения клиента и принять тот факт, что вынужденные изменения всегда делают продукт лучше.

3. **Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.**

   Этот принцип отвечает за итеративность и заключается в регулярной поставке работающего ПО с постепенным наращиванием функционала в каждом новом релизе. Этот принцип позволяет уйти от авралов и простоев в команде, оперативно выявить и устранить препятствия, адаптироваться и организовать процесс разработки таким образом, чтобы поставлять работающий продукт частями на основе приоритетов клиента.

4. **Работающий продукт — основной показатель прогресса.**

   Лучшее подтверждение выполненной работы — не отчёт о статусе проекта или процент выполненной задачи, а демонстрация текущего результата. Наполовину выполненная задача — это не выполненная задача, так как она не несёт никакой ценности для клиента. Самый эффективный способ сообщить о достижениях команды — это показать работающий продукт. Выполнению этого принципа способствует итеративный подход.

Организация процесса

5. **Непосредственное общение** — наиболее практичный и эффективный способ обмена информацией как с самой командой, так и внутри команды.

   Нет ничего лучше, чем обмениваться знаниями, опытом и мнениями при личном общении. Это гораздо эффективнее и быстрее, чем использовать для этого документацию или электронные средства коммуникации вроде почты или мессенджера. Личное общение также наилучший способ разобрать спорные моменты и найти общее решение.

6. **На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.**

   Если команда хочет создать действительно ценный продукт, который будет отвечать потребностям клиента и соответствовать требованиям рынка, полноценное вовлечение представителей клиента в командную работу обязательно. Только клиент знает, что является приоритетом для бизнеса и какие функции имеют наибольшую ценность. Когда представители бизнеса и разработчики трудятся в одной команде, это в разы увеличивает эффективность работы и повышает качество продукта.

7. **Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.**

   Обеспечение этого принципа осуществляется посредством ограничения объёма работ в рамках итераций. Команда начинает двигаться от итерации к итерации с определённой средней скоростью, которая позволяет держать ритм работы на протяжении долгого времени. Устойчивый темп позволяет исключить переработки.

8. **Простота — искусство минимизации лишней работы — крайне необходима.**

   Этот принцип имеет отношение не только к организации работ или проектной документации, но и в целом к написанию кода. Гибкие команды создают максимально простые решения, избегая ненужных функций и сложной архитектуры с множеством зависимостей между системами и объектами. Лучше создать систему без большого количества зависимостей и ненужного кода: в неё будет проще вносить изменения.

Команда

9. **Над проектом должны работать мотивированные профессионалы**. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.

   В этом пункте речь идёт в первую очередь о нематериальной мотивации и создании благоприятных условий для работы команды. Необходимо уйти от тотального контроля и назначения задач, вместо этого поддержать самостоятельность сотрудников. Выстраивать отношения, основанные на доверии и командной ответственности.

10. **Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.**

    Качество важнее скорости, качество вообще превыше всего. Хорошо разработанный и реализованный продукт намного быстрее поставляется клиенту и обладает необходимой гибкостью для внесения изменений.

11. **Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.**

Важно дать команде возможность самостоятельно заниматься планированием, распределением задач и поиском подходящих решений в сложных ситуациях. Это способствует развитию коллективной ответственности, где каждый участник болеет за общий результат, а не за свой личный успех.

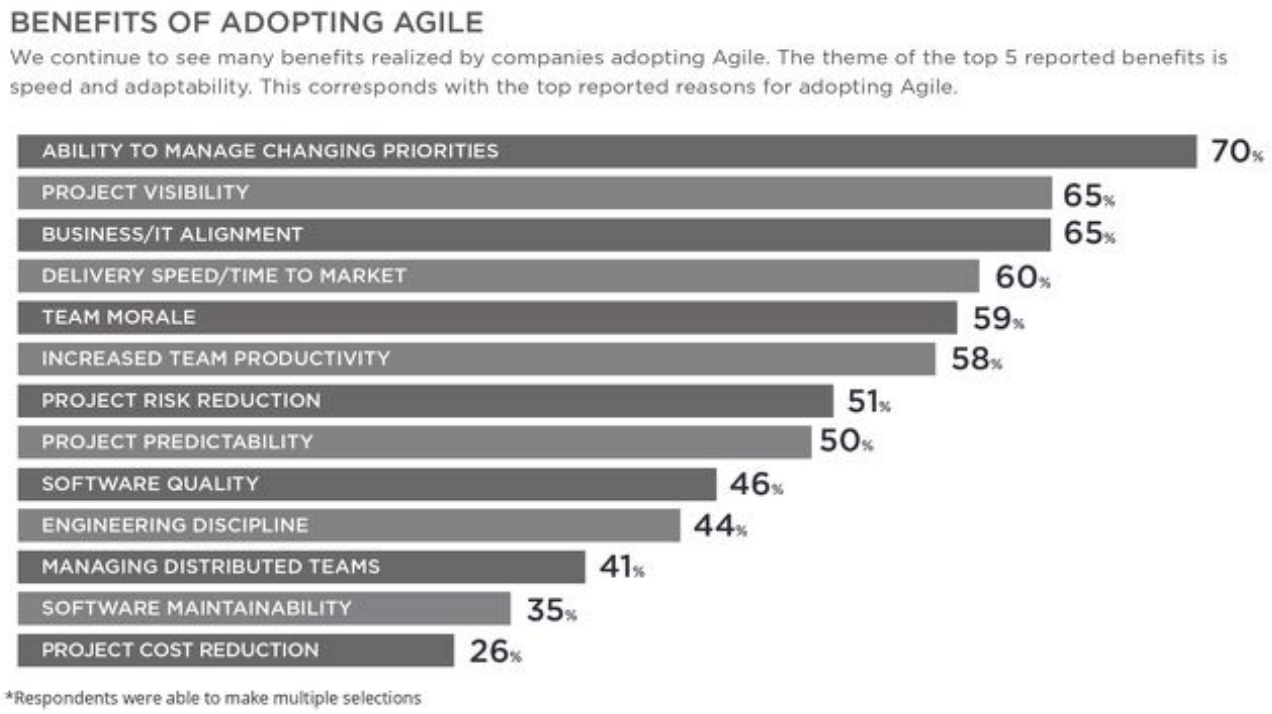
12. **Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.**

    Единственный путь к постоянному развитию и совершенствованию — это регулярный анализ проделанной работы. Самостоятельное выявление и открытое признание ошибок — это путь к выходу из сложившейся ситуации и адаптации к новым условиям. Agile-команды постоянно улучшают свои компетенции в области разработки, регулярно уделяя время обсуждению уроков, которые они получили после каждой итерации и в конце проекта.

После погружения в философию и основные принципы Agile-концепции любопытно узнать, какие преимущества это даёт на практике.

Преимущества применения Agile

Один из основных мотивов внедрения и применения Agile на проектах — экономические выгоды. Многие компании и клиенты считают, что этот подход позволит сэкономить бюджет. Однако на практике очень часто оказывается, что содержание Agile-команд обходится дороже, чем использование стандартного каскадного подхода. То, что экономия бюджета — не основная выгода от применения этого подхода подтверждают и ежегодные исследования State of Agile, согласно которым только 26% компаний, участвующих в исследовании, отметили снижение затрат благодаря внедрению у себя Agile. Это самый низкий результат среди всех преимуществ, которые выделили респонденты.



Согласно исследованию, преимущества применения Agile распределились следующим образом:

- способность управлять изменением приоритетов — 70%;
- прозрачность (видимость) проекта — 65%;
- эффективность сотрудничества бизнеса и сферы IT — 65%;
- время и скорость выпуска готового продукта на рынок — 60%;
- улучшение психологического климата в команде — 59%;
- повышение производительности команды — 58%;
- снижение рисков на проектах — 51%;
- повышение прогнозируемости проектов — 50%;
- повышение качества ПО — 46%;
- улучшение дисциплины в команде — 44%;
- улучшение управления распределёнными командами — 41%;
- сокращение издержек на обслуживание ПО — 35%;
- снижение расходов — 26%.

Приведённые выше данные показывают, насколько может улучшиться положение дел в компании, которая решится внедрить у себя культуру Agile и применить на практике основанные на её принципах методы организации процесса и техники разработки. Главный вопрос, на который необходимо ответить в заключение: как определить, подходит ли Agile конкретно вам и вашему проекту.

Когда Agile не подходит

С появлением Agile многие компании стали внедрять его философию и практику в свои организации и проекты, надеясь, что это повысит скорость разработки, улучшит качество результата, минимизирует риски и сэкономит бюджет, станет спасением от всех проблем, которые были связаны с традиционным

подходом. Неудивительно, что многие компании ждало большое разочарование. Оказалось, что всё не так просто: мало выбрать подходящие под работу инструменты, необходимо понимать, зачем они нужны и какую ценность несут. Многие компании и специалисты были готовы смело использовать прикладные методы в своей работе, но не готовы менять своё мировоззрение. Удивительным открытием стало и то, что Agile подходит не всем: не всем компаниям, не всем специалистам и не всем проектам. Необходим целый ряд условий, чтобы применение Agile было возможным и успешным.

Модель «Кеневин» и матрица Стейси

Последние десять лет для верного выбора модели управления и организации процесса разработки очень часто используют модель «Кеневин» (Cynefin framework), автор которой Дейв Сноуден (Dave Snowden), в прошлом возглавлявший Институт управления знаниями на базе IBM.

Термин Cynefin переводится с английского языка как «среда обитания» или «место». Используется для объяснения эволюционной природы любых сложных систем.

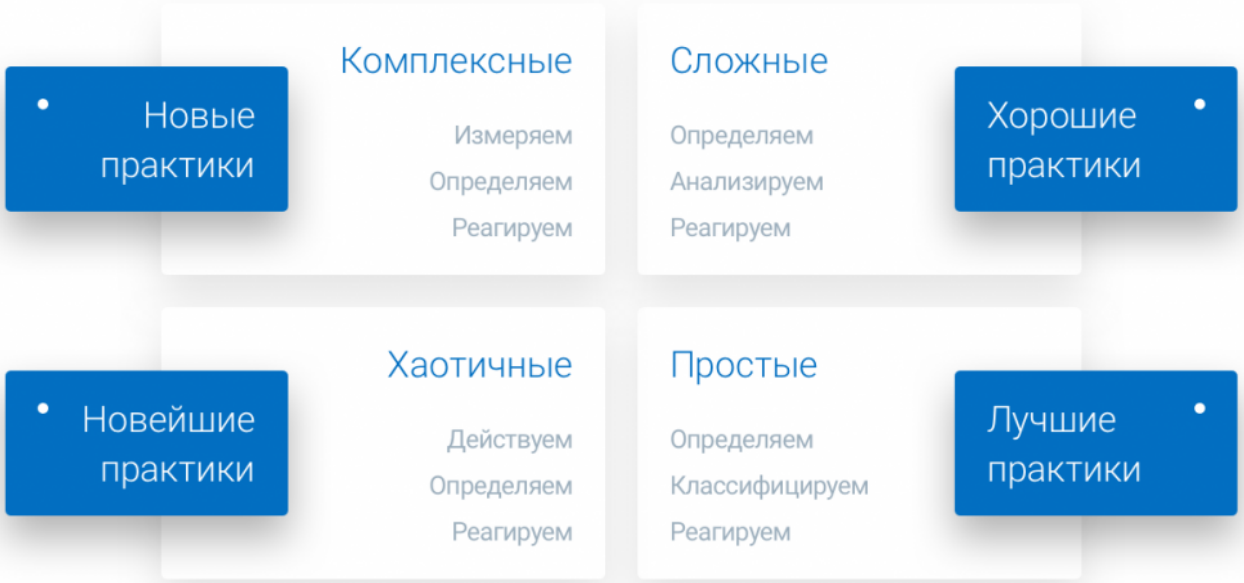
Система (др.-греч. σύστημα «целое, составленное из частей; соединение») — множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство.

Метод «Кеневин» позволяет руководителю определить, в какой «среде обитания» или системе существует проект, и выбрать верную модель управления, коррелирующую с компетенцией команды.

Системы бывают простыми, сложными, комплексными и хаотическими, и определяются они наличием или отсутствием причинно-следственных связей между элементами.

Цель методики Сноудена — помочь руководителям выработать управленческую тактику в ситуациях, когда привычные способы решения проблем не работают.

Выбор тактики управления в зависимости от типа системы

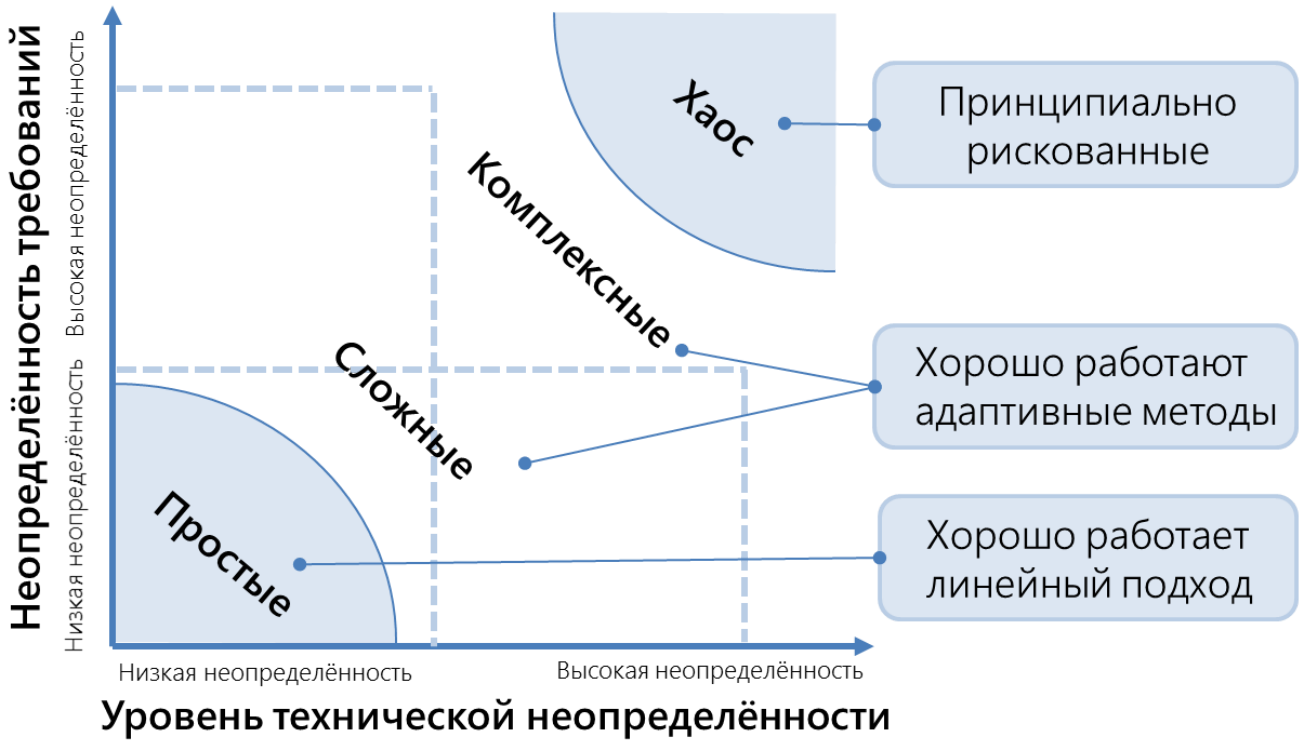


1. Проекты, связанные с **простыми системами**, характеризуются прозрачностью. Задачи имеют очевидные причинно-следственные связи, понятен объём работ, срок, бюджет, у команды есть опыт решения подобных задач.
    a. **Формула принятия решений на таких проектах:** «Определяем – Классифицируем – Реагируем». Используем лучшие практики.
    b. **Метод:** используем инструкцию или лучшие практики.
2. Проекты, связанные со **сложными системами**, характеризуются некоторой степенью неопределённости. Задачи не уникальны, но их очень много, на определение причинно-следственных связей нужны время и усилия нескольких специалистов. У команды есть некоторый опыт в решении подобных задач и есть представление, в каком направлении работать. Объём работы, сроки и бюджет можно спрогнозировать.
    a. **Формула принятия решения:** «Определяем – Анализируем – Реагируем». Используем хорошие практики.
    b. **Методы:** каскадная модель или адаптивный вариант каскадной методологии, смешение методов.
3. Проекты, связанные с **комплексными или запутанными системами**, характеризуются высокой степенью неопределённости. Способ решения задач до конца не ясен, причинно-следственные связи не очевидны и требуют установления, спрогнозировать объём работ и срок очень сложно. Поиск подходящих решений чаще всего происходит в процессе.
    a. **Формула принятия решения:** «Измеряем – Определяем – Реагируем». Новая практика изобретается в процессе.
    b. **Метод:** Agile

4. Проекты, находящиеся в зоне **хаотичных систем**, определяются полной неизвестностью. Это абсолютно новые задачи, которые никто и никогда не решал раньше, причинно-следственные связи неизвестны. Попытка разобраться с такой системой — путь к инновациям. Здесь хорошо работает экспериментальный подход, и любой способ решения (стабилизации системы) будет новым.
    a. **Формула принятия решения:** «Действуем – Определяем – Реагируем». Действуем, исходя из ситуации, потом оцениваем результат.
    b. **Метод:** новый

Пользуясь этой методологией, легко понять, что инновационный стартап будет относиться к хаотичным системам и требует поиска новых решений, а установка стандартного ПО в рамках организации — это область простых систем, так как всё можно сделать по инструкции. Внедрение стандартного, но сложного ПО, которое требует доработки под индивидуальные бизнес-процессы клиента, подразумевает интеграцию с новыми системами, миграцию данных и новый дизайн интерфейса, можно отнести к сложным системами или даже к пересечению областей сложных и комплексных систем, так как работа подразумевает анализ, проектирование и поиск оптимальных решений. В ситуации, когда есть бюджет, но клиент не знает, что он хочет получить, или не уверен в наборе функционала, команда сразу попадает в запутанную систему, где удовлетворяющее потребности клиента решение будет появляться постепенно в процессе реализации проекта.

Модель «Кеневин» отлично дополняет матрица Стейси, где выбор подхода к реализации проекта зависит не только от системы, в которой вы находитесь, но в первую очередь от факторов неопределённости: неопределённости требований и технической неопределённости.



Вертикальная ось показывает, понятно ли, что мы хотим сделать. Горизонтальная ось показывает, понимаем ли мы, как этого достичь. Оси образуют четыре домена модели «Кеневин», в которые может

попасть проект: простые, сложные, комплексные и хаос. После определения домена, с которым напрямую связана степень неопределённости, можно подбирать подход. Исходя из схемы, очевидно, что Agile подходит для самой обширной системы — комплексной.

Выбор подхода к реализации проекта в этом случае выглядит просто, так как метод «Кеневин» и модель Стейси помогают ответить на один из главных вопросов: подходит ли проект под Agile-концепцию. Однако, если проект подходит, а компания и команда оказались не готовы к работе с Agile-подходами и не придерживаются Agile-философии, то результатом будет неудача и разочарование.

Есть ещё два вопроса, которые требуют ответа:

1. Подходит ли компания?
2. Подходит ли команда?

Agile не подходит, если:

1. Компания работает в строго регламентируемой индустрии.
2. Компания имеет централизованный бюрократизированный аппарат.
3. В компании приветствуется командно-административное управление.
4. Отсутствует лояльность со стороны менеджмента или клиента.
5. Отсутствуют компетентные трудовые ресурсы.
6. У сотрудников низкая мотивация, отсутствие командного духа и командной ответственности.

Agile подходит:

1. Для компаний, которые понимают, принимают и поддерживают Agile-философию.
2. Для клиентов, которые готовы к сотрудничеству и лояльны к изменениям в сроках, объёме и бюджете.
3. Когда все участники команды обладают необходимой квалификацией.
4. Для команд, которые готовы взять на себя ответственность за результат и тесно сотрудничать с заказчиком.
5. Для команд, ориентированных на самостоятельную работу, сотрудничество и взаимопомощь.
6. Для команд, которые позитивно воспринимают перемены, направленные на совершенствование их работы и улучшение продукта.

Заключение

У всех методологий и подходов есть свои преимущества и недостатки. Выбор зависит от множества факторов и должен быть индивидуальным. В рамках курса мы попробуем на практике применить Agile-подход к разработке и создать простой программный продукт. Наш проект будет ограничен сроком в 2 месяца, и нашей задачей будет работать в рамках недельных итераций с постепенным наращиванием готового к работе функционала.

На следующем уроке мы подробно познакомимся с фреймворком Scrum, техникой Kanban, узнаем, почему среди команд стало популярным применение гибрида Scrumban.

Глоссарий

Манифест Agile (Agile Manifesto), манифест гибкой разработки программного обеспечения — основной документ, содержащий описание ценностей и принципов гибкой разработки программного обеспечения, разработанный в феврале 2001 года на встрече 17 независимых практиков нескольких методик программирования, именующих себя Agile Alliance.

Методология разработки — набор методов по управлению разработкой ПО, практических правил и техник разработки. Или, по-другому, методология — детализированный набор правил, практик и принципов как способ реализации той или иной модели.

Фреймворк (англ. framework — каркас, структура, рамки, основа) — набор базовых элементов и правил, своего рода каркас, на котором строится процесс разработки; программная платформа, определяющая структуру программной системы.

Метод (от др.-греч. μέθοδος — путь исследования или познания, от μετά- + ὁδός «путь») — способ достижения какой-либо цели. В отличие от области знаний или исследований, он авторский, то есть создан конкретной персоной или группой персон, научной или практической школой. В силу своей ограниченности рамками действия и результата, методы имеют тенденцию устаревать, преобразовываясь в другие методы, развиваясь в соответствии со временем, достижениями технической и научной мысли, потребностями общества. Совокупность однородных методов принято называть подходом. Развитие методов — естественное следствие развития научной мысли.

Методика — это, как правило, некий готовый «рецепт», алгоритм, процедура для проведения каких-либо нацеленных действий. Методика отличается от метода конкретизацией приёмов и задач. Например, математическая обработка данных эксперимента может объясняться как метод (математическая обработка), а конкретный выбор критериев, математических характеристик — как методика.

Модель (фр. modèle от лат. modulus «мера, аналог, образец») — система, исследование которой служит средством для получения информации о другой системе; представление некоторого реального процесса, устройства или концепции; форма отображения определённого фрагмента действительности (предмета, явления, процесса, ситуации) — оригинала модели, которое содержит существенные свойства моделируемого объекта и может быть представлено в абстрактной (мысленной или знаковой) или материальной (предметной) форме.

Концепция (от лат. conceptio – «система понимания») — комплекс взглядов на что-либо, связанных между собой и образующих систему; определённый способ понимания, трактовки каких-либо явлений; основная точка зрения, руководящая идея для их освещения.

Система (др.-греч. σύστημα «целое, составленное из частей; соединение») — множество элементов, находящихся в отношениях и связях друг с другом, которое образует целостность, единство.

Дополнительные материалы

1. Генри Нив «Организация как система. Принципы построения устойчивого бизнеса Эдвардса Деминга» (книга).
2. Джеффри Лайкер «Дао Toyota. 14 принципов менеджмента ведущей компании мира» (книга).
3. Инструкция: как перейти на Agile (статья).
4. Хватит переходить к Agile! Три беседы, с которыми необходимо ознакомиться, перед тем как переходить к действиям (статья).
5. Видео (9 мин.): The Cynefin Framework / Модель Кеневин от автора фреймворка Дейва Сноудена, с переводом.
6. Статья: Why Agile? - The Stacey complexity model.

Используемые источники

1. Agile-манифест разработки программного обеспечения (официальный сайт).
2. Неизвестная история Agile (статья).
3. 14 управленческих принципов Эдварда Деминга (статья).
4. 14 управленческих принципов ДАО Тойота — «сердце» Toyota Production System (статья).
5. Как внедрить Agile в компании, не разрушив её (статья).
6. Глобальная статистика использования Agile-методологий в мире (отчёт).
7. Эндрю Стеллман, Дженнифер Грин «Постигая Agile. Ценности, принципы, методологии» (книга).

Практическое задание

1. Пройти тестирование к Уроку 1.
2. Ознакомиться с методологическим пособием к уроку.