



# ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ЛЕКЦИЯ № 3

ПРЕПОДАВАТЕЛЬ: ХУСТОЧКА А.В.



# МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- Методологии представляют собой ядро теории управления разработкой ПО. К существующей классификации в зависимости от используемой в ней модели жизненного цикла (каскадные и эволюционные) добавилась более общая классификация на прогнозируемые и адаптивные методологии.
- Прогнозируемые (предикативные) методологии фокусируются на детальном планировании будущего. Известны запланированные задачи и ресурсы на весь срок проекта. Команда с трудом реагирует на возможные изменения. План оптимизирован исходя из состава работ и существующих требований. Изменение требований может привести к существенному изменению плана, а также дизайна проекта.
- Адаптивные (гибкие) методологии нацелены на преодоление ожидаемой неполноты требований и их постоянного изменения. Когда меняются требования, команда разработчиков тоже меняется. Команда, участвующая в адаптивной разработке, с трудом может предсказать будущее проекта. Существует точный план лишь на ближайшее время. Более удаленные во времени планы существуют лишь как декларации о целях проекта, ожидаемых затратах и результатах. Среди адаптивных методологий: (Scrum, Crystal, Extreme Programming, Adaptive Software Development, DSDM, Feature Driven Development, Lean software development)

# RUP

Один из самых известных процессов, использующих итеративную модель разработки – RUP. Он был создан во второй половине 1990-х годов в компании Rational Software. Термином RUP обозначает как методологию, так и продукт компании IBM (ранее Rational) для управления процессом разработки. Методология RUP описывает абстрактный общий процесс, на основе которого организация или проектная команда должна создать специализированный процесс, ориентированный на ее потребности.

Основные характеристики:

- разработка требований, для описания требований в RUP используются прецеденты использования (use cases). Полный набор прецедентов использования системы вместе с логическими отношениями между ними называется моделью прецедентов использования. Каждый прецедент использования – это описание сценариев взаимодействия пользователя с системой, полностью выполняющего конкретную пользовательскую задачу. Согласно RUP все функциональные требования должны быть представлены в виде прецедентов использования.
- итеративная разработка, проект RUP состоит из последовательности итераций с рекомендованной продолжительностью от 2 до 6 недель. Перед началом очередной итерации определяется набор прецедентов использования, которые будут реализованы к её завершению. 3

# АРХИТЕКТУРА RUP

- Можно сказать, что RUP – ориентированная на архитектуру методология. Считается, что реализация и тестирование архитектуры системы должны начинаться на самых ранних стадиях проекта. RUP использует понятие исполняемой архитектуры (executable architecture) – основы приложения, позволяющей реализовать архитектурно значимые прецеденты использования. Основы исполняемой архитектуры должны быть реализованы как можно раньше.
- Это позволяет оценить адекватность принятых архитектурных решений и внести необходимые коррективы еще в начале проекта. Таким образом, для первых нескольких итераций необходимо выбирать прецеденты, которые требуют реализации большей части архитектурных компонентов. RUP поощряет использование визуальных средств для анализа и проектирования. Как правило, используется нотация и, соответственно, средства моделирования UML (такие как Rational Rose). Модель предметной области документируется в виде диаграммы классов, модель прецедентов использования – при помощи диаграммы прецедентов, взаимодействие компонентов системы между собой описывается диаграммой последовательности и т.д.

# ЖИЗНЕННЫЙ ЦИКЛ ПРОЕКТА

Жизненный цикл проекта RUP состоит из четырех фаз. Последовательность этих фаз фиксирована, но число итераций, необходимых для завершения каждой фазы, определяется индивидуально для каждого конкретного проекта. Фазы RUP нельзя отождествлять с фазами водопадной модели – их назначение и содержание принципиально различны.

Фазы RUP:

- Начало (Inception)
- Проектирование (Elaboration)
- Построение (Construction)
- Внедрение (Transition)

# НАЧАЛО (INCEPTION)

Стадия «начало» обычно состоит из одной итерации.

В ходе выполнения этой стадии необходимо:

- определить видение и границы проекта;
- создать экономическое обоснование;
- идентифицировать большую часть прецедентов использования и подробно описать несколько ключевых прецедентов;
- найти хотя бы одно возможное архитектурное решение;
- оценить бюджет, график и риски проекта.

Если после завершения первой итерации заинтересованные лица приходят к выводу о целесообразности выполнения проекта, проект переходит в следующую стадию. В противном случае проект может быть отменен или проведена еще одна итерация стадия «начало».

# ПРОЕКТИРОВАНИЕ (ELABORATION)

В результате выполнения этой стадии на основе требований и рисков проекта создается основа архитектуры системы. Проектирование может занимать до двух-трех итераций или быть полностью пропущенным (если в проекте используется архитектура существующей системы без изменений).

Целями этой фазы являются:

- детальное описание большей части прецедентов использования;
- создание оттестированной (при помощи архитектурно значимых прецедентов использования) базовой архитектуры;
- снижение основных рисков и уточнение бюджета и графика проекта.

В отличие от каскадной модели, основным результатом этой стадии является не множество документов со спецификациями, а действующая система с 20-30% реализованных прецедентов использования.

## ПОСТРОЕНИЕ (CONSTRUCTION)

- В этой стадии (длящейся от двух до четырех итераций) происходит разработка окончательного продукта. Вовремя ее выполнения создается основная часть исходного кода системы и выпускаются промежуточные демонстрационные прототипы.



## ВНЕДРЕНИЕ (TRANSITION)

- Целями стадии «внедрения» являются проведение бетатестирования и тренингов пользователей, исправление обнаруженных дефектов, развертывание системы на рабочей площадке, при необходимости – миграция данных.
- Кроме того, на этой стадии выполняются задачи, необходимые для проведения маркетинга и продаж. Стадия «внедрения» занимает от одной до трех итераций. После ее завершения проводится анализ результатов выполнения всего проекта: что можно изменить для улучшения эффективности в будущих проектах.

# РАБОЧИЙ ПРОЦЕСС RUP

В терминах RUP участники проектной команды создают так называемые артефакты (work products), выполняя задачи (tasks) в рамках определенных ролей (roles). Артефактами являются спецификации, модели, исходный код и т.п. Задачи разделяются по девяти процессным областям, называемым дисциплинами (discipline). В RUP определены шесть инженерных и три вспомогательные дисциплины. В них входят:

- Бизнес-моделирование (Business Modeling) – исследование и описание существующих бизнес-процессов заказчика, а также поиск их возможных улучшений.
- Управление требованиями (Requirements Management) – определение границ проекта, разработка функционального дизайна будущей системы и его согласование с заказчиком.
- Анализ и проектирование (Analysis and Design) – проектирование архитектуры системы на основе функциональных требований и ее развитие на протяжении всего проекта.
- Реализация (Implementation) – разработка, юнит-тестирование и интеграция компонентов системы.

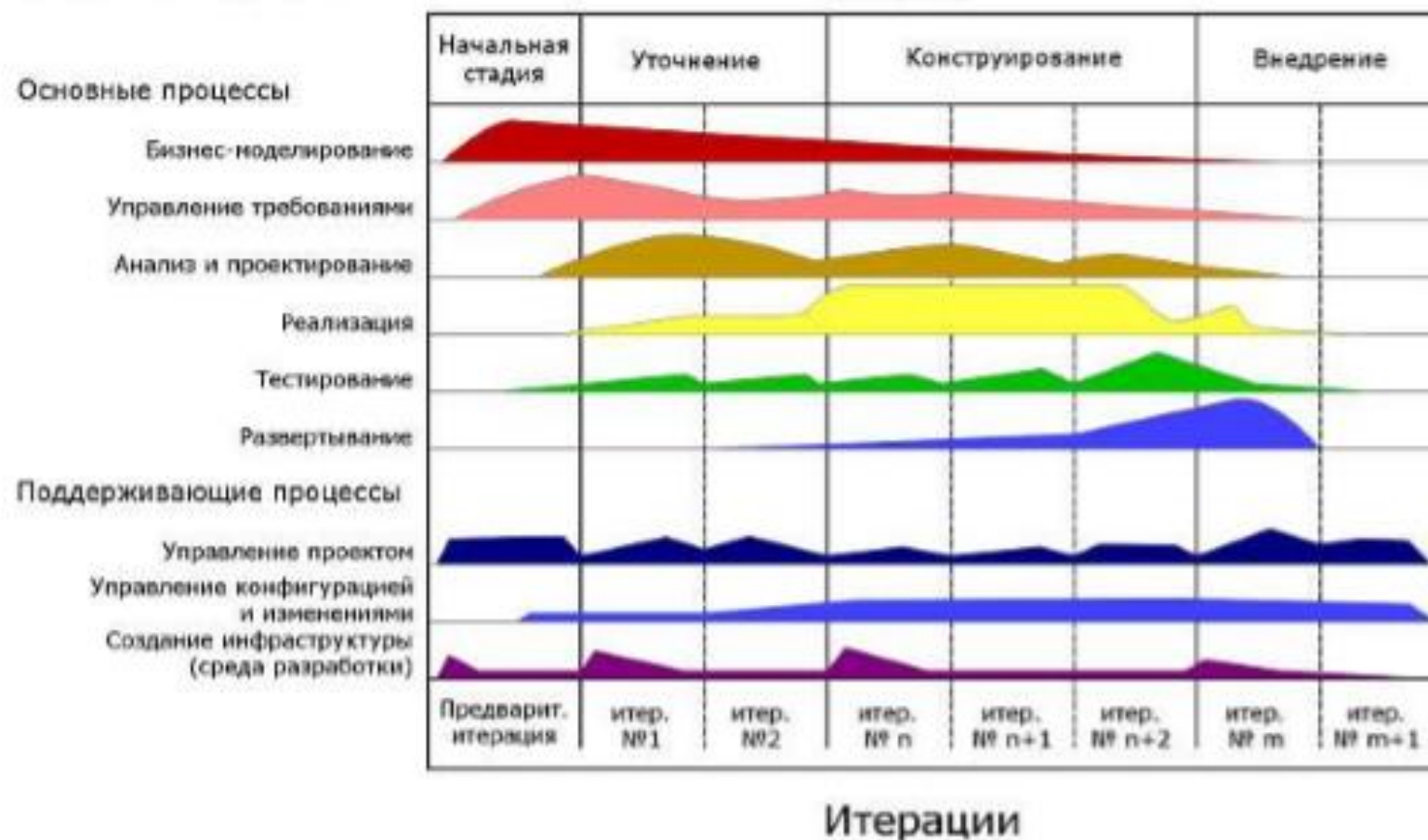
# РАБОЧИЙ ПРОЦЕСС

- Тестирование (Test) – поиск и отслеживание дефектов в системе, проверка корректности реализации требований.
- Развертывание (Deployment) – создание дистрибутива, установка системы, обучение пользователей.
- Управление конфигурациями и изменениями (Configuration and Change Management) – управление версиями исходного кода и документации, процесс обработки запросов на изменение (Change requests).
- Управление проектом (Project Management) – создание проектной команды, планирование фаз и итераций, управление бюджетом и рисками.
- Среда (Environment) – создание инфраструктуры для выполнения проекта, включая организацию и настройку процесса разработки.

# РАСПРЕДЕЛЕНИЕ УСИЛИЙ ПРИ ВЫПОЛНЕНИИ ПРОЕКТА

Рабочие процессы

Стадии



## RUP. РЕЗЮМЕ

- В ходе жизненного цикла проекта распределение усилий проектной команды между дисциплинами постоянно меняется. Например, как правило, в начале проекта большая часть усилий затрачивается на анализ и дизайн, а ближе к завершению – на реализацию и тестирование системы. Однако в общем случае задачи из всех девяти дисциплин выполняются параллельно.
- Для полноценного внедрения RUP организация должна затратить значительные средства на обучение сотрудников. При этом попытка обойтись своими силами скорее всего будет обречена на неудачу – необходимо искать специалиста по процессам (process engineer) с соответствующим опытом или привлекать консультантов.

# MICROSOFT SOLUTIONS FRAMEWORK (MSF)

- Данная методология описывает подход и организацию работы при создании программных продуктов. Подробно про методологию MSF вы можете более подробно прочитать в переводе Microsoft Solutions Frameworks for Agile Software Development, которая входит в поставку Microsoft Team Foundation Server
- Но это надо скачивать тут <https://www.microsoft.com/en-us/download/details.aspx?id=5365>
- Поэтому дальше «обзор методологии»

# ИСТОРИЯ MSF

- История создания MSF берет начало в 1993 году. Компания Microsoft уже была ведущей в IT-сфере и искала способы повысить эффективность и отдачу от своих проектов.
- 90-е годы стали временем расцвета новых подходов к разработке. Модель «Водопад», которую использовали больше двух десятилетий, уже в полной мере не отвечала требованиям в IT: была слишком жесткой и формализованной, медленно реагировала на новые потребности пользователей.
- У Microsoft был обширный опыт в создании программных продуктов и продвижении масштабных IT-проектов: были уже выпущены Windows 3.11, Office 3.0 и многое другое. Компания суммировала накопленные знания и навыки, проанализировала опыт конкурентов и в 1993 году выпустила серию руководств, посвященных организации труда разработчиков — «белые книги» MSF.
- Пять лет спустя, в 1998, была выпущена вторая ревизия MSF. В 2001 за ней последовала третья, а в 2005 вышла последняя на данный момент версия — MSF 4.0.
- Отличительными чертами Microsoft Solutions Framework стали гибкость и масштабируемость. Эта методология подходит для работы в проектной группе или организации любого масштаба. MSF включает основополагающие принципы, модели и дисциплины управления персоналом, процессами и технологиями. Другим преимуществом методологии стала ее демократичность и отсутствие иерархических отношений «начальник — подчиненный».

# ОСНОВОПОЛАГАЮЩИЕ ПРИНЦИПЫ MSF

Принципов, на которых базируется MSF, всего восемь:

- Способствуйте открытому общению
- Работайте над общим видением
- Расширяйте полномочия членов команды
- Разделяйте ответственность
- Сотрудничайте с клиентом и сосредоточьтесь на предоставлении бизнес-ценности
- Будьте готовы к переменам и оставайтесь гибкими
- Инвестируйте в качество
- Учитесь на опыте



# СПОСОБСТВУЙТЕ ОТКРЫТОМУ ОБЩЕНИЮ

- Модель MSF предусматривает свободный и открытый обмен информацией между всеми членами команды и заинтересованными сторонами. Это помогает исключить недопонимание между заказчиком и исполнителем и снижает вероятность того, что работу придется переделывать. Все этапы разработки обстоятельно описываются, обеспечивается доступность документации для всех участников проекта — так налаживается эффективное взаимодействие.

# РАБОТАЙТЕ НАД ОБЩИМ ВИДЕНИЕМ

- Данный принцип Microsoft Solutions Framework подразумевает, что все члены команды должны детально понимать цели и задачи, над которыми работает коллектив. Общий взгляд на то, каким должен быть результат, гарантирует, что усилия разработчиков будут согласованными.
- Зачастую к успеху приводит именно умение правильно понять и сформулировать достоинства выработанного решения. Кроме того, эффективность труда в команде повышается, когда все сотрудники видят картину проекта в целом. Это стимулирует их интересоваться даже теми областями проекта, которые не относятся непосредственно к их задачам. Благодаря этому сотрудники глубже вникают в принципы работы программного продукта, обмениваются идеями и решениями, помогают друг другу и набираются новые компетенции и командный опыт.

# РАСШИРЯЙТЕ ПОЛНОМОЧИЯ ЧЛЕНОВ КОМАНДЫ

- Каждому члену команды должны быть предоставлены все полномочия, необходимые для выполнения его обязанностей. Если его работа зависит от коллег, он должен быть уверен, что с их стороны не будет задержек и проволочек. Для дисциплины следует использовать графики, в которых будут обозначены сроки для каждой задачи.

# РАЗДЕЛЯЙТЕ ОТВЕТСТВЕННОСТЬ

- Все участники проекта принимают и осознают свою ответственность за порученную задачи и собственные решения.
- В MSF проект делят на равноценные и уникальные сегменты. За успешную реализацию каждого в равной степени отвечают все специалисты, работающие над ним. Участник подотчетен своей рабочей группе, она — всей организации, а та, в свою очередь, — заказчику. При этом ответственность распределяется на каждом уровне равномерно между всеми сотрудниками.
- Такой подход к разделению ответственности обусловлен тем фактом, что в общем решении зачастую сложно выделить вклад отдельного специалиста. Согласно принципу, все успехи и неудачи проекта участники делят поровну. Никто не может приписать себе единоличные заслуги или стать козлом отпущения.

# СОТРУДНИЧАЙТЕ С КЛИЕНТОМ И СОСРЕДОТОЧЬТЕСЬ НА ПРЕДОСТАВЛЕНИИ БИЗНЕС-ЦЕННОСТИ

- Всегда нужно помнить о главном: программное решение должно представлять ценность для бизнеса заказчика. MSF требует от команды ориентироваться на клиента и вовлекать его в работу.
- В MSF это значит понимать его цели и проблемы: зачастую заказчик совсем не разбирается в разработке программного обеспечения или даже в компьютерах, но это не мешает ему быть экспертом в своем бизнесе. Только он знает точно, каковы его потребности; какая функциональность жизненно необходима, а какая избыточна; что имеет ценность для бизнеса, а что — нет. Поэтому необходимо, чтобы клиент был вовлечен в работу над проектом, и если он доволен результатами — все идет как надо.

# БУДЬТЕ ГОТОВЫ К ПЕРЕМЕНАМ И ОСТАВАЙТЕСЬ ГИБКИМИ

- Любая разработка программного обеспечения требует времени. Это означает, что на каком-то этапе работы требования заказчика могут измениться — к этому стоит быть готовыми.
- Все члены коллектива должны быть вовлечены в принятие решений об изменениях в проекте. Благодаря этому новые задачи и цели будут осмыслены и рассмотрены с учетом всех точек зрения и идей по реализации.

# ИНВЕСТИРУЙТЕ В КАЧЕСТВО

- Успех команды зависит от того, насколько каждый специалист осознает свою ответственность за продукт. Чтобы обеспечить высокое качество решения, на протяжении всего проекта работает группа тестирования — в ее задачи входит раннее выявление ошибок и недочетов. Обнаруженные баги надо исправлять как можно скорее, чтобы они не повлияли на разработку. Microsoft Solution Framework требует, чтобы в планы и графики изначально вносилось время на устранение недостатков. Только в этом случае можно уложиться в срок.

## УЧИТЕСЬ НА ОПЫТЕ

- Любой коллектив разработчиков, если только он не вчера появился, обладает наработанным опытом. У каждого хорошего специалиста есть свои испытанные временем приемы. Совокупные знания команды выходят далеко за пределы компетенций отдельного сотрудника. Более того, новый проект и каждая его итерация — это источник опыта.
- В коллективе важно создать и поддерживать среду, в которой каждый член команды чувствует себя востребованным, полезным и свободным. В психологически комфортных условиях возрастает не только эффективность сотрудника, но и его мотивация к самосовершенствованию и обмену знаниями с коллегами.
- В график работы необходимо закладывать время на обучение и общение — для этого в MSF существует дисциплина управления готовностью, о которой речь пойдет ниже. Периодически следует совместно анализировать выполненную работу. Необходимо поддерживать атмосферу доброжелательности и взаимопомощи. Открытость, постоянный обмен информацией и опытом — залог успеха.



# ПЯТЬ «БЕЛЫХ КНИГ» MSF

MSF разработана как комплекс отдельных компонентов — моделей и дисциплин. Всего их пять, и они описаны в пяти «белых книгах» (white papers) MSF.

Моделей используется две:

- модель команды;
- модель процесса.

А дисциплин в MSF три:


- управление проектами;
- управление рисками;
- управление готовностью.

# МОДЕЛЬ КОМАНДЫ MSF

- Команда разработчиков — главный стратегический ресурс компании, определяющий успех проекта. В традиционной практике команды организованы иерархически — от руководителя до работников низшего звена, например:



*Пример традиционной иерархической организации труда*

- 
- При такой организации работы вес мнения отдельного сотрудника определяется не его компетенциями и знаниями, а положением в иерархии. MSF предлагает более демократическую модель команды, и поэтому не испытывает проблем классической.
  - Команда проекта в MSF — это коллектив равноправных сотрудников. Они разделяют ответственность и свободно обмениваются опытом и информацией. Внутри команды есть ролевые кластеры (роли), отражающие функциональные обязанности конкретных специалистов. У каждой роли — свои цели и задачи, и все они считаются равноценными и одинаково важными. Роли дополняют друг друга и вместе служат единой цели — созданию качественного продукта.

## Модель команды в MSF



## РОЛЬ: БИЗНЕС-АНАЛИТИК

- Это главный посредник между командой разработчиков и клиентом. Он должен детально разобраться в потребностях заказчика, определить бизнес-ценность продукта и понять, какая именно функциональность необходима в программе. Он трансформирует эту информацию в конкретные определения и требования к качеству и доводит их до разработчика. Можно считать, что бизнес-аналитик — представитель клиента в коллективе. Он управляет продуктом, чтобы тот соответствовал потребностям бизнеса и ожиданиям заказчика.

## РОЛЬ: МЕНЕДЖЕР ПРОЕКТА

- В модели MSF главная задача менеджера проекта — контролировать график работ и бюджет. Он отвечает за то, чтобы все задачи выполнялись своевременно и не выходили за пределы сметы. Занимается планированием работы и составляет отчеты, оценивает риски и вырабатывает меры по их снижению. Тесное сотрудничество с другими ролями проекта позволяет ему быть в курсе событий и оперативно решать административные проблемы.

## РОЛЬ: АРХИТЕКТОР

- Отвечает за архитектуру программного продукта — за основные принципы, по которым будет работать приложение, организационную конфигурацию системы, ее физическую структуру и интерфейсы. Архитектор стремится сделать программное решение проще и удобнее — как для пользователя, так и для разработчика, в том числе на этапе поддержки. Он участвует в обсуждении всех нововведений и изменений в ходе проекта, определяя, как именно новая функциональность будет вписываться в систему.

## РОЛЬ: РАЗРАБОТЧИК

- Кажется, что роль разработчика — самая незатейливая. Рабочая лошадка: создает код и воплощает идеи клиента, трансформированные в техническое задание усилиями бизнес-аналитика и архитектора. И придерживается сроков, установленных проект-менеджером.
- Но в MSF разработчик — полноправный участник всех обсуждений. Он определяет, какое время потребуется ему для создания функциональных блоков программы. Помогает архитектору выстраивать удачную структуру проекта с точки зрения его реализации, основываясь на своем глубоком знании языка программирования. Участвует в создании дизайна приложения. В проекте считается экспертом по всем техническим вопросам, и за ним зачастую остается последнее слово в совещаниях о том, «как все это реализовать в рамках используемых технологий».



## РОЛЬ: ТЕСТИРОВЩИК

- Задача тестировщика — выявить в продукте баги, проблемы и неудачные решения, которые могут негативно сказаться на качестве и снизить ценность приложения для клиента. Тестировщик обязан понимать и учитывать контекст применения программного продукта: кто, как и для чего будет его использовать на стороне заказчика. Если функция X выполняется корректно и выдает правильный результат, но его получение занимает в среднем час, тестировщик может признать ее работу неудовлетворительной — зная, что клиенту требуется выполнять эту функцию 20–30 раз в день.
- Ошибки и отклонения от заданных параметров фиксируются и документируются, после чего тестировщик передает их разработчику для исправления. Тестировщик также участвует в выработке стандартов качества и создании тестовых задач, нагрузочных тестов и подобного.

## РОЛЬ: РЕЛИЗ-МЕНЕДЖЕР

- В Microsoft Solutions Framework, что касается выпуска готового продукта или любой его работоспособной версии, — на релиз-менеджере. Он создает план выпуска версий, сертифицирует готовый продукт и следит за тем, чтобы программа дошла до клиента. Кроме того, в его обязанности входит информационное сопровождение версий, например описание новых функций, изменений и нововведений.

## РОЛЬ: АДМИНИСТРАТОР БАЗ ДАННЫХ

- Если в составе продукта есть база данных, то команде не обойтись без администратора БД. Он следит за порядком в базе, ее целостностью и состоянием данных, работоспособностью серверов. В его обязанности входят все рутинные операции, обеспечивающие безопасность и сохранность информации, например регулярное создание бэкапов.
- В отдельных проектах может принимать участие и **разработчик баз данных**. Он несет ответственность за создание структуры БД и функциональности, обеспечивающей ее работу. Частично этот специалист может пересекаться по задачам с разработчиком проекта. Есть смысл выделить эту роль, если проект сложный и базы данных играют в нем значительную роль.

Несмотря на множество ролей, модель MSF подходит не только большим коллективам, но и маленьким командам: в этом случае каждый участник берет на себя несколько ролей.

Методология MSF подчеркивает, что все роли — равноправны, и ни один член команды не может считаться более влиятельным или принимать решения единолично.

# МОДЕЛЬ ПРОЦЕССА MSF

- Еще один важный компонент методологии MSF — модель процесса, то есть последовательность действий, необходимых для построения IT-решения. Модель не предписывает конкретных процедур и не содержит жестких формализованных требований к процессу — при создании MSF компания Microsoft стремилась сделать ее гибкой и адаптируемой к условиям любого проекта. В MSF объединились две концепции разработки: «Водопад» и спиральная модель.
- От «Водопада» MSF досталась система вех (milestones) — особых точек в конце каждой фазы процесса, отвечающих заданным критериям завершения фазы. В этих точках команда рассматривает результаты своего труда и отвечает на вопросы «Сделали ли мы все, что планировали?», «Работает ли решение так, как нужно заказчику?», «Готовы ли мы двигаться дальше или необходимо уделить внимание доработкам?». Чтобы перейти на следующий этап, необходимо дать большинство положительных ответов.
- От спиральной модели MSF унаследовала фокусировку на уточнениях требований к проекту. Разработчик должен постоянно быть готов к тому, что задачи, а порой и цели клиента могут измениться на любом этапе работы.
- Тем не менее MSF — это не просто компиляция двух систем. Инновационность методологии заключается в том, что она охватывает жизненный цикл решения от начала проекта до развертывания в реальном времени. Это помогает проектным группам сосредоточиться на бизнес-ценности приложения, поскольку она не будет реализована до тех пор, пока решение не развернуто и не запущено в эксплуатацию.
- Весь процесс разработки в MSF разбит на отдельные итерации. Каждая проходит несколько этапов (фаз).



# ЭТАПЫ MSF

## 1. Выработка концепции (Visioning)

- Команда вырабатывает единое видение проекта или его части. Совместными усилиями коллеги решают, какая именно функциональность будет разрабатываться в ходе итерации, определяют основные концепции, которые лягут в основу разработки. Этап завершается вехой «Концепция утверждена».

## 2. Планирование (Planning)

- Задачи, которые необходимо выполнить в ходе итерации, разбиваются на подзадачи, определяется сложность их реализации, устанавливаются сроки и назначаются ответственные. В планы закладывается время на тестирование и исправление дефектов, а также предварительно намечается, как именно будет проходить тестирование.

## 3. Разработка (Developing)

- На данном этапе MSF создается программный код новой функциональности в соответствии с концепцией и утвержденными планами.

## 4. Стабилизация (Stabilizing)

- К делу подключаются тестировщики. После тестирования выявленные баги и недочеты возвращаются разработчикам для исправления.

## 5. Внедрение (Deploying)

- Очередной релиз программного продукта передается заказчику и устанавливается на клиентских компьютерах.

В конце каждой итерации клиент должен получить работоспособную версию приложения. По завершении этапа внедрения и итерации в целом немедленно начинается новая итерация.

# ДИСЦИПЛИНЫ MSF: УПРАВЛЕНИЕ ПРОЕКТОМ

В MSF это целый набор навыков и компетенций, в том числе:

- комплексное планирование всех этапов и аспектов проекта;
- управление бюджетом, расходами и ресурсами;
- подготовка графиков и контроль за их соблюдением;
- ведение административной документации.

Роль, ответственная за выполнение этого сегмента работы, — менеджер проекта (программы). По мере того как масштаб проекта растет, управление проектом может разделиться на две специализированные ветви: одна будет связана с архитектурой программного решения и спецификациями, а другая — собственно с управлением проектом.

Когда большие коллективы разрабатывают крупные проекты, управление может выполняться на нескольких уровнях: задачи распределяются между руководителями рабочих групп (команд, каждая из которых отвечает за ту или иную роль в проекте), а роль управления программой отвечает за координирование руководителей и в целом курирует проект.

MSF стремится избавиться от иерархической структуры. Поэтому и при управлении проектом нет диктатуры. Демократичные обсуждения, при которых рассматриваются все точки зрения и достигается консенсус, способствуют выработке наиболее удачных решений. Когда члены команды не могут прийти к соглашению, менеджер проекта выступает арбитром: он обязан принять решение, максимально удовлетворяющее клиента и ориентированное на его бизнес-ценности.

# ДИСЦИПЛИНЫ MSF: УПРАВЛЕНИЕ РИСКАМИ

Риски — это факторы и события, которые могут оказать негативное влияние на проект в перспективе. В MSF есть специальный процесс, который помогает выявлять, отслеживать и минимизировать риски. Он состоит из шести шагов.

## 1. Определение рисков

- Любой член команды в любое время может (и должен) сообщать о рисках, которые он выявил. Например, если обнаружил ошибку в сторонней библиотеке кода, которая на данный момент не беспокоит, но грозит привести к проблемам, когда будет использоваться соответствующая функциональность библиотеки.

## 2. Анализ и расстановка приоритетов

- Насколько серьезную угрозу представляет выявленный риск для проекта? Возможно ли избежать проблем? Требуются немедленные действия или время терпит? Необходимы ли дополнительные ресурсы, например время на поиск решения? Все эти вопросы обсуждаются коллегиально.



### 3. План и график

- Информация, собранная при анализе рисков, должна быть преобразована в конкретные планы, стратегии и действия. Следует сформулировать четкие руководства, что необходимо делать и что запрещено, чтобы снизить или исключить риски.

### 4. Отслеживание и отчет

- Придерживаться принятого плана неукоснительно — половина дела в MSF. Любое изменение в проекте может повлечь новые риски или рецидив старых. Отслеживание рисков помогает держать их под контролем и своевременно пересматривать тактику борьбы с ними.

### 5. Контроль

- Контроль — это исполнение планов в отношении рисков и связанных с ними отчетов.

### 6. Знание

- Изучая риски, команда получает новую информацию о них и о способах преодоления сопутствующих сложностей. Эти знания необходимо фиксировать и помещать в базу данных, чтобы иметь к ним доступ в будущем.



## ДИСЦИПЛИНЫ MSF: УПРАВЛЕНИЕ ГОТОВНОСТЬЮ

- Эта дисциплина занимается вопросами профессионального роста и подготовки специалистов.
- С точки зрения организации, знания и навыки сотрудника — это ценный ресурс, так что обучение и повышение квалификации можно рассматривать как улучшение качества ресурса. В MSF под готовностью понимается отношение текущего объема знаний и навыков к тому уровню, который желателен или необходим для конкретного специалиста. От готовности зависит, например, способность сотрудника выполнять ту или иную роль в команде.
- Для небольших или краткосрочных проектов подход к управлению готовностью может быть простым — просто оценить знания сотрудников и затем распределить роли в команде. Но организации, занимающиеся долгосрочными проектами, могут извлечь из этой дисциплины наибольшую выгоду, поскольку она предлагает комплексную программу подготовки и повышения квалификации.



Процесс управления готовностью включает четыре этапа:

1. Определение

- На этом этапе выстраивается структура команды. Для каждой роли определяются уровни квалификации и компетенции, необходимые для успешной работы специалистов. Кроме того, вырабатываются сценарии — типичные виды деятельности, которые потребуются для разработки проекта.

2. Оценка

- Здесь внимание сосредоточено на каждом члене команды. Проводится анализ компетенций и навыков, связанных с должностными обязанностями, и определяется, насколько они соответствуют желаемым показателям для каждой конкретной роли. Это позволяет выявить разницу между текущим уровнем знаний и требуемым. В результате можно разработать планы индивидуального обучения для каждого сотрудника, которые позволят ему приобрести нужный уровень компетенций.

3. Изменение

- В процессе обучения специалисты совершенствуют знания, чтобы преодолеть разрыв между нынешним и желаемым уровнем квалификации. При этом используются учебные планы со списками ресурсов и учебных материалов — учебников, технических документов. Учебный план может предусматривать и самостоятельное изучение, и под руководством наставника.

4. Подведение итогов

- На этом этапе проводится повторная оценка знаний и компетенций, чтобы определить, были ли планы обучения эффективными и не требуются ли дополнительные занятия. Рассматриваются не только теоретические знания, но и способность сотрудника использовать их на практике.

Управление готовностью — процесс, в идеале не завершающийся на протяжении всего проекта. Непрерывное совершенствование знаний и умений каждого члена команды — путь к повышению качества и успешности проекта в целом.

# SCRUM

Scrum предоставляет эмпирический подход к разработке ПО. Этот процесс быстр, адаптивен, умеет подстраиваться и отличен от каскадной модели. Scrum основан на повторяющихся циклах, это делает его более гибким и предсказуемым.

Для начала определим роли, которые участвуют в процессе:

- Scrum мастер (Scrum Master)
- Владелец продукта (Product Owner)
- Команда (Team).

# SCRUM МАСТЕР

Самая важная роль в методологии. Scrum Мастер отвечает за успех Scrum в проекте. Как правило, эту роль в проекте играет менеджер проекта или лидер команды (Team Leader). Важно подчеркнуть, что Scrum Мастер не раздает задачи членам команды. В Scrum команда является самоорганизующейся и самоуправляемой.

Основные обязанности Scrum Мастера таковы:

- создает атмосферу доверия
- участвует в митингах в качестве фасилитатора - человека, обеспечивающий успешную групповую коммуникацию
- устраняет препятствия
- делает проблемы и открытые вопросы видимыми
- отвечает за соблюдение практик и процесса в команде

Scrum Мастер отслеживает прогресс команды при помощи Sprint Backlog, отмечая статус всех задач в спринте. Scrum Мастер может также помогать заказчику создавать список задач для команды

# PRODUCT OWNER

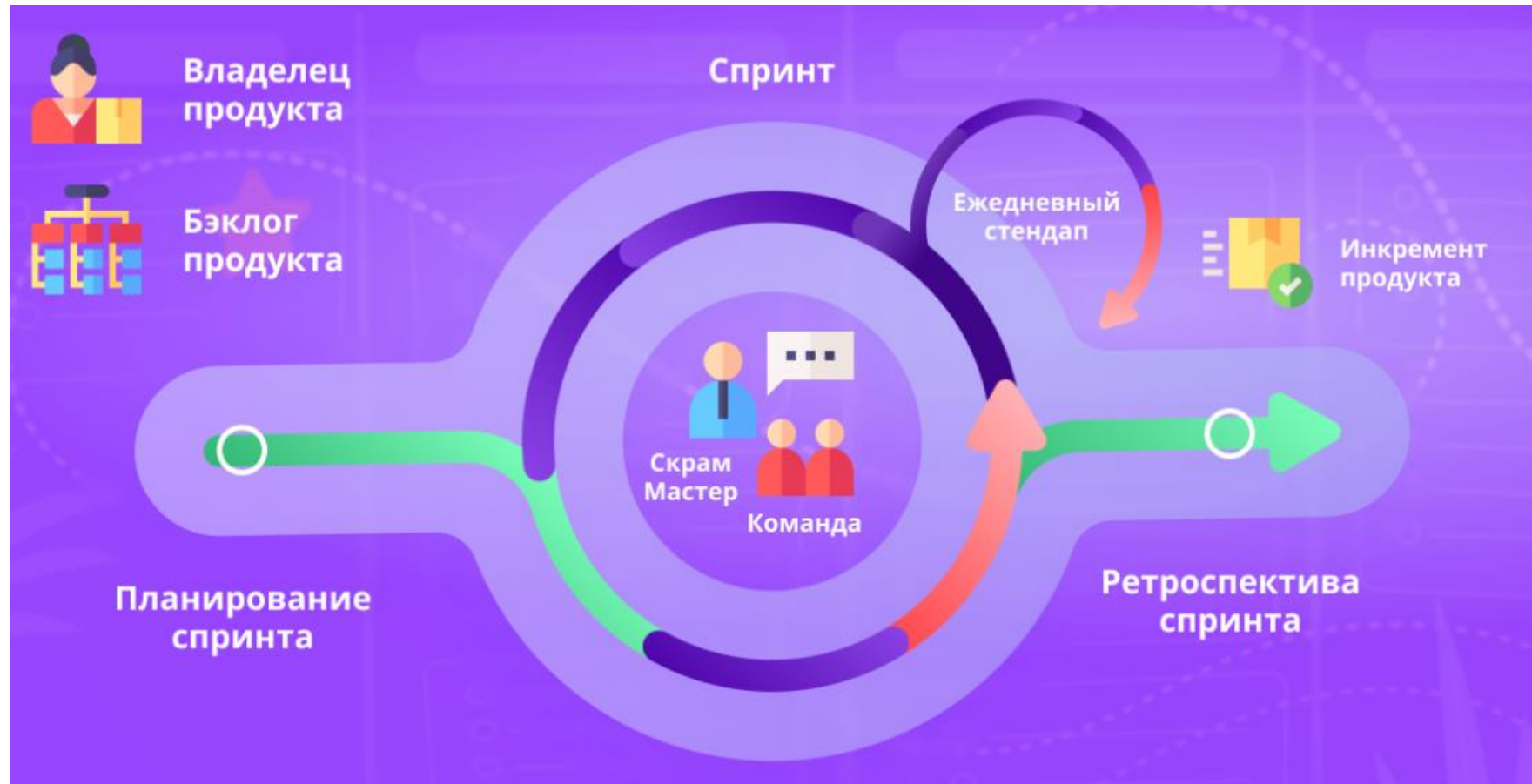
- Это человек, отвечающий за разработку продукта. Как правило представитель заказчика для заказной разработки. Владелец продукта - это единая точка принятия окончательных решений для команды в проекте, именно поэтому это всегда один человек, а не группа или комитет.

# КОМАНДА (TEAM)

- В методологии Scrum команда является самоорганизующейся и самоуправляемой. Команда берет на себя обязательства по выполнению объема работ на спринт перед Владелъцем продукта. Работа команды оценивается как работа единой группы.
- В Scrum вклад отдельных членов проектной команды не оценивается, так как это разваливает самоорганизацию команды. Размер команды ограничивается размером группы людей, способных эффективно взаимодействовать лицом к лицу.
- Типичные размер команды - 7 плюс минус 2. Команда в Scrum кроссфункциональна. В нее входят люди с различными навыками - разработчики, аналитики, тестировщики. Нет заранее определенных и поделенных ролей в команде, ограничивающих область действий членов команды. Команда состоит из инженеров, которые вносят свой вклад в общий успех проекта в соответствии со своими способностями и проектной необходимостью



# ЭТАПЫ SCRUM



- В основе лежат короткие ежедневные встречи – Scrum и циклические 30-дневные встречи, называемые спринтом. Результатом спринта является готовый продукт, который можно передавать заказчику (по крайней мере, система должна быть готова к показу заказчику).
- Короткие спринты обеспечивают быструю обратную связь проектной команды с заказчиком. Заказчик получает возможность гибко управлять системой, оценивая результат спринта и предлагая улучшения к созданной функциональности.
- Такие улучшения попадают в список имеющихся на данный момент бизнес-требований и технических требований к системе (Product Backlog), приоритезируются наравне с прочими требованиями и могут быть запланированы на следующий (или на один из следующих) спринтов.
- Каждый спринт представляет собой маленький «водопад». В течение спринта делаются все работы по сбору требований, дизайну, кодированию и тестированию продукта.
- Цель спринта должна быть фиксированным. Это позволяет команде давать обязательства на тот объем работ, который должен быть сделан в спринте. Это означает, что Sprint Backlog не может быть изменен никем, кроме команды

# ЭКСТРЕМАЛЬНОЕ ПРОГРАММИРОВАНИЕ (EXTREME PROGRAMMING)

- Методология XP, разработанная Кентом Беком (Kent Beck), Уордом Каннингемом (Ward Cunningham) и Роном Джеффрисом (Ron Jeffries), является сегодня одной из самых популярных гибких методологий.
- Она описывается как набор практик: игра в планирование, короткие релизы, метафоры, простой дизайн, переработки кода (refactoring), разработка «тестами вперед», парное программирование, коллективное владение кодом, 40-часовая рабочая неделя, постоянное присутствие заказчика и стандарты кода.
- Интерес к XP рос снизу вверх – от разработчиков и тестировщиков, замученных тягостным процессом, документацией, метриками и прочим формализмом. Они не отрицали дисциплину, но не желали бессмысленно соблюдать формальные требования и искали новые быстрые и гибкие подходы к разработке высококачественных программ.

- При использовании XP тщательное предварительное проектирование ПО заменяется, с одной стороны, постоянным присутствием в команде заказчика, готового ответить на любой вопрос и оценить любой прототип, а с другой – регулярными переработками кода (так называемый рефакторинг).
- Основой проектной документации считается тщательно прокомментированный код. Очень большое внимание в методологии уделяется тестированию.
- Как правило, для каждого нового метода сначала пишется тест, а потом уже разрабатывается собственно код метода до тех пор, пока тест не начнет выполняться успешно. Эти тесты сохраняются в наборах, которые автоматически выполняются после любого изменения кода. Хотя парное программирование и 40-часовая рабочая неделя и являются, возможно, наиболее известными чертами XP, но все же носят вспомогательный характер и способствуют высокой производительности разработчиков и сокращению количества ошибок при разработке.

# CRYSTAL CLEAR

Легковесная гибкая методология, созданная Алистером Коуберном, которая предназначена для небольших команд в 6-8 человек для разработки некритичных бизнес-приложений. Как и все гибкие методологии, Crystal Clear больше опирается на людей, чем на процессы и артефакты.

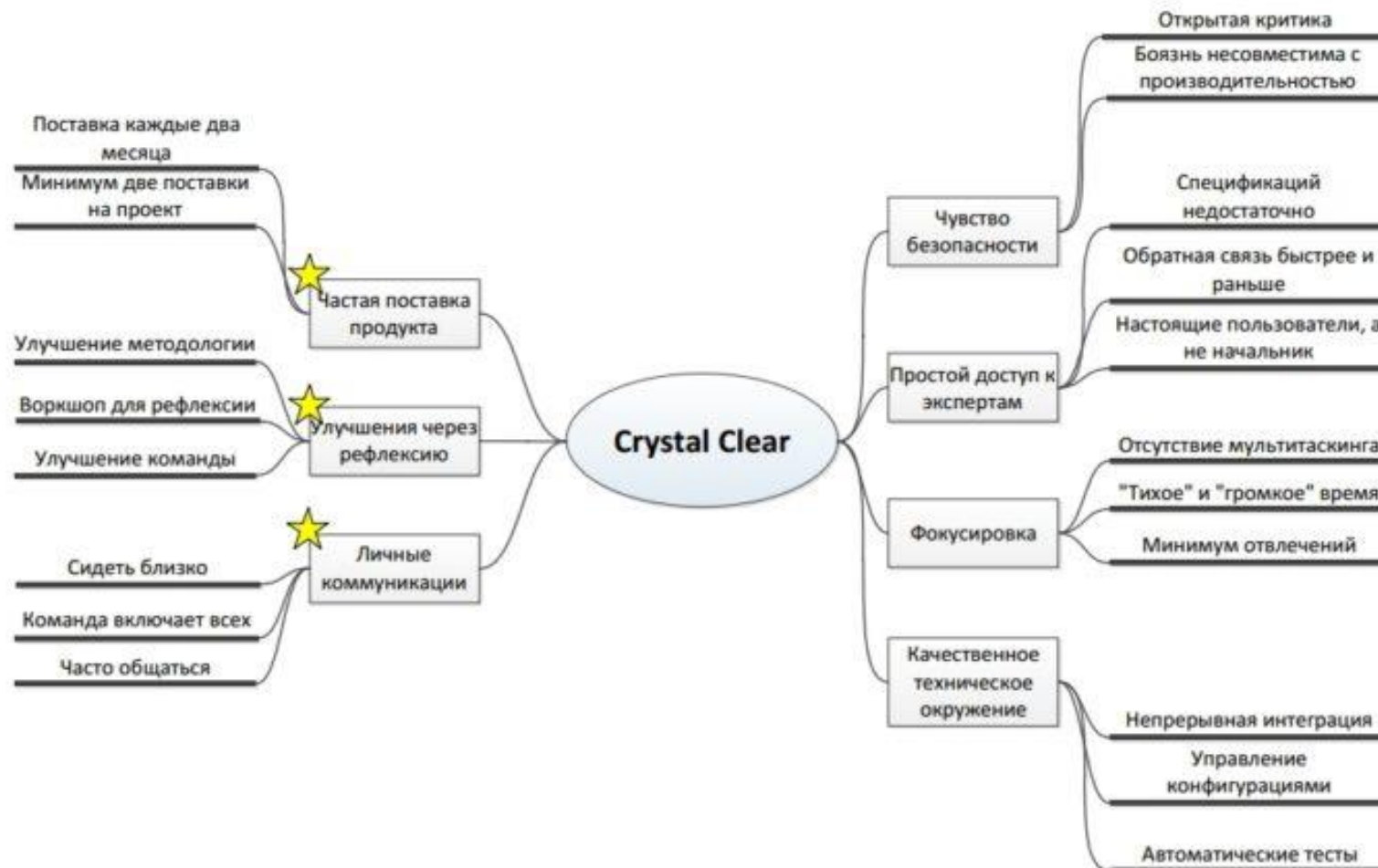
Crystal Clear использует семь методов/практик, три из которых являются обязательными:

- частая поставка продукта;
- улучшения через рефлексию;
- личные коммуникации;
- чувство безопасности;
- фокусировка;
- простой доступ к экспертам;
- качественное техническое окружение.

# CRYSTAL CLEAR

- Методология Crystal Clear уступает XP по производительности, зато максимально проста в использовании. Она требует минимальных усилий для внедрения, поскольку ориентирована на человеческие привычки.
- Считается, что эта методология описывает тот естественный порядок разработки ПО, который устанавливается в достаточно квалифицированных коллективах, если в них не занимаются целенаправленным внедрением другой методологии.
- Основные характеристики Crystal Clear:
  - итеративная инкрементная разработка;
  - автоматическое регрессионное тестирование;
  - пользователи привлекаются к активному участию в проекте;
  - состав документации определяется участниками проекта;
  - как правило, используются средства контроля версий кода.

# ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ CRYSTAL CLEAR



# KANBAN

- Канбан – это визуальная система управления работой команды, одна из самых популярных методологий управления наравне со Scrum.
- Чаще Agile используется в IT, но именно канбан можно применить ко всем сферам бизнеса. В конце концов, канбан – это способ визуализировать задачи для повышения продуктивности команды, и неважно – команды разработчиков, продаж, врачей или строителей.
- Карточки в современном канбане применяются для визуализации потока задач, сокращения незавершенной работы, выстраивания приоритетов. Это позволяет сделать сроки предсказуемыми и регулируемыми. Все участники команды видят, на каком этапе находится задача, что уже сделано и что предстоит сделать. Это помогает повысить продуктивность, выстроить процессы, отрегулировать нагрузку сотрудников и соблюдать дедлайны.



# KANBAN

Суть kanban-методологии заключается в следующем:

- Есть план того, что нужно сделать, он называется backlog (бэклог). В нем список задач отсортирован по приоритету, при необходимости его можно и нужно корректировать, меняя важность карточек.
- Есть ограничения по количеству задач «В процессе», чтобы регулировать нагрузку сотрудников или отделов, избегать завалов и простоев. Это ограничение называется WIP -лимит.

При необходимости для задач можно выставить дедлайн, но это необязательно. Приоритетные задачи всегда находятся вверху бэклога – это значит, что они будут сделаны как можно скорее.

# ЦЕННОСТИ МЕТОДА

Методология базируется на культуре взаимного уважения и работе в команде, что обеспечивает успех, целесообразность работы и высокую вовлеченность сотрудников. К этому сводятся все девять ценностей канбана:

- Прозрачность – открытый обмен информацией;
- Баланс – равновесие между нагрузкой и возможностями;
- Сотрудничество – совместная работа участников команды и ее совершенствование;
- Фокус на заказчике и его потребностях – создание продукта, который нужен клиенту;
- Поток – непрерывная работа;
- Лидерство – вдохновение своим примером других участников. При этом нет иерархии, понятие применимо на всех уровнях;
- Понимание – знание всеми участниками целей развития команды;
- Согласие – совместное движение к целям и совершенствованию;
- Уважение – понимание и положительная оценка всех участников команды.

Если отступить хотя бы от одной из ценностей, у команды ничего не получится – так считают создатели краткого руководства по канбану Дэвид Дж. Андерсон и Энди Кармайл .

# ПРИНЦИПЫ

Чтобы успешно использовать систему в своей команде, нужно придерживаться основных принципов канбана:

- визуализировать работу – разделить задачи на этапы;
- систематизировать доску – создать колонки, которые будут отражать текущий этап работы над задачей. Например: «надо сделать», «в работе», «сделано»;
- актуализировать задачи – постоянно обновлять статус, перемещая карточки из одной колонки в другую на доске, и выстраивать приоритеты в бэклоге;
- контролировать течение задач – если выполнение каких-то операций затягивается и карточка долго не продвигается по доске, важно проанализировать причины и при необходимости перераспределить ресурсы или помочь в решении;
- постоянно совершенствовать систему – визуализация помогает выявлять проблемные этапы и задачи. Процесс можно и нужно корректировать, устраняя уязвимые места.

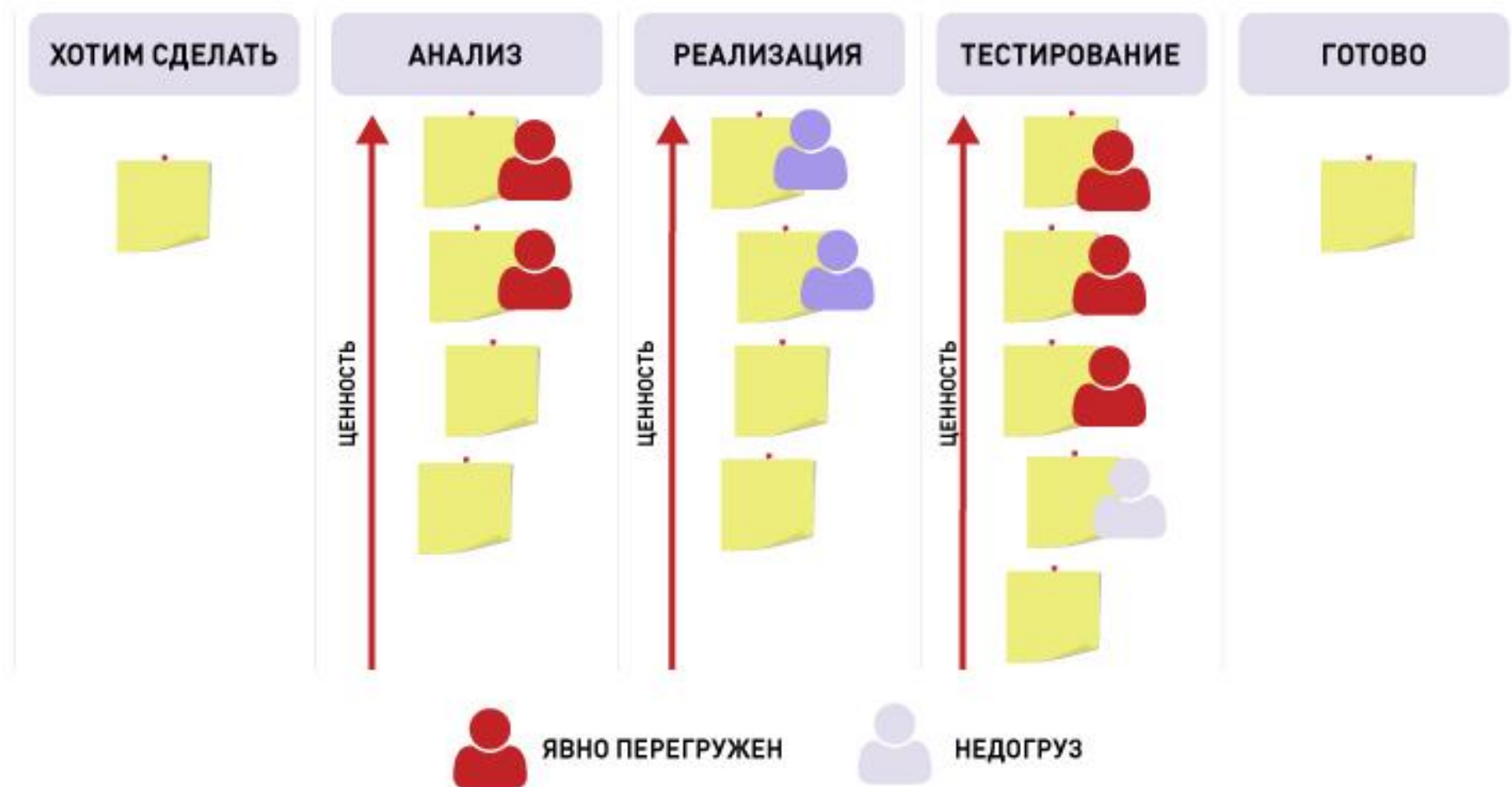
# ИНСТРУМЕНТЫ

Главный инструмент канбана – доска с карточками. Это может быть физическая меловая доска, магнитная, со стикерами или электронная. К ней должны иметь доступ все участники команды в любой момент времени.

Колонки доски:

- «Бэклог» – поле для всех карточек, пул задач, который может пополняться, сортироваться по приоритетности;
- «В процессе» – включает несколько видов внутренних колонок, адаптированных под команду и обозначающих разные этапы работы над карточкой;
- «Готово» – полностью выполненные задачи, которые не требуют от команды дальнейших действий.
- На одной доске можно вести сразу несколько проектов, для этого используют карточки разных цветов или swimlanes – горизонтальные разделители. Каждая карточка в канбане может содержать дополнительную информацию с описанием задачи, именем того, кто над ней работает, ее приоритет, дедлайн. Задачи могут быть ежедневными, еженедельными, ежемесячными.

# ДОСКА КАНБАН. ПРИМЕР 1



## ДОСКА КАНБАН. ПРИМЕР 2



# ПРАВИЛА РАБОТЫ С КАРТОЧКАМИ

Основные правила Kanban при работе с карточками направлены на непрерывное течение процесса, регулирование сроков и внимание к задачам, которые по каким-то причинам не движутся по потоку:

- WIP-лимит может быть разным для конкретных специалистов или отделов в зависимости от их ресурсов. Цель применения лимита – направить фокус сотрудника на одну задачу, вместо того, чтобы он пытался делать несколько сразу.
- Максимальным лимитом регулируется количество карточек в каждом столбце. Лимит основывается на реальных возможностях команды, в него входят все карточки, которые находятся в работе.
- Нельзя начинать новую карточку, если не сделана предыдущая. Если задача по каким-то причинам не может быть завершена, ее нужно перенести в колонку Blocked и искать другие способы ее завершения.

# ПРАВИЛА РАБОТЫ С КАРТОЧКАМИ

Приоритетность задач в канбане зависит от их важности для бизнеса или клиента, размера недополученной прибыли или издержек в случае, если они не будут сделаны в срок. Чтобы участникам команды было понятнее, какая работа важнее, внедряют классы обслуживания, на карточках их обозначают символами:

- срочный – нельзя откладывать;
- с фиксированной датой – нужно сделать к определенному сроку;
- стандартный – издержки растут пропорционально задержке, желательно сделать вовремя;
- нематериальный – стоимость задержки растет медленно, задача несрочная, делать ее сейчас необязательно, если есть более важные.



# РИТМ РАБОТЫ КОМАНДЫ

В канбане есть рекомендованные регулярные встречи для координации работы команды и получения обратной связи. Они проводятся на уровне команды и на уровне компании.

Встречи на уровне команды:

- канбан-митинг – ежедневные встречи по 15 минут для обсуждения текущих задач на сегодня;
- встречи для обновления бэклога – один раз в неделю по 30 минут для добавления и приоритизирования новых задач;
- встреча с клиентом – собрание на 30 минут вместе с заказчиком, на котором команда выясняет, доволен ли он качеством и скоростью работы;
- обзор рисков – ежемесячная встреча для обсуждения прошлых неудач и поиска вариантов их устранения.

Встречи на уровне компании:

- обзор операций – проводится ежемесячно для оценки и поиска способов общего повышения эффективности всех команд и отделов;
- обзор стратегии – ежеквартальная встреча для оценки деятельности всей компании, выявления масштабных проблем. В ней принимают участие ключевые лица команды и руководство.
- Некоторые виды встреч можно объединять в одну, чтобы не нагружать участников большим количеством совещаний. Некоторые из них могут не иметь смысла конкретно для вашего бизнеса.

# ЧЕМ КАНБАН ОТЛИЧАЕТСЯ ОТ СКРАМА

- Скрам и канбан – методологии Agile, в обеих применяются доски с карточками и общие принципы и ценности гибкого управления. Но они не взаимозаменяемы и используются в командах с разными целями и задачами.
- В скраме работа над продуктом делится на запланированные спринты – отрезки времени на выполнение заранее сформированного списка задач, чаще всего это две недели. В процессе спринта не могут добавляться в работу новые карточки из бэклога. Все новые цели и задачи добавляются в следующие спринты. Скрам подходит для команд, разрабатывающих продукт, который требует планирования, и не подходит для команд, в которых приоритеты меняются каждый день.
- В канбане карточки движутся по доске в непрерывном потоке на базе приоритетов. В любой момент времени приоритеты могут меняться, если этого требуют обстоятельства. Это обеспечивает большую, чем в скраме, гибкость.
- Kanban – это методология управления командами, где запланировать невозможно. Например, это может быть техническая поддержка: если клиент позвонил и зарегистрировал проблему, команда не может запланировать разрешить ее в следующем спринте через две недели. Важно разрешить проблему как можно скорее и не потерять лояльность клиента, а значит, планирование и расстановка приоритетов должны происходить гораздо динамичнее по сравнению со Scrum. Применяя канбан в своей команде поддержки, вы повышаете лояльность и удовлетворенность своих клиентов.

# ВСПОМОГАТЕЛЬНЫЕ ИНСТРУМЕНТЫ

Существует ряд инструментов, упрощающих процесс планирования разработки, например:

- TFS
- JIRA
- Yandex Tracker
- ClickUp
- Trello