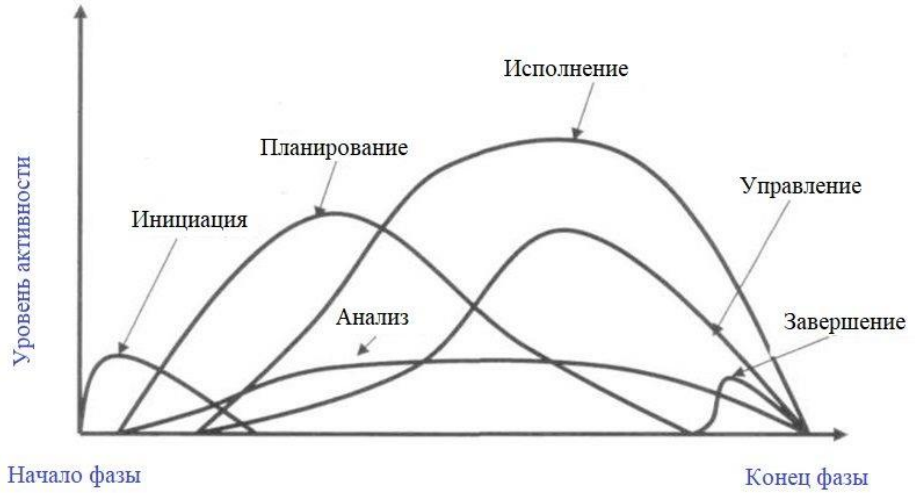
Лекция 4. Методологии и модели жизненного цикла программных средств

Основные модели жизненного цикла ПО

1. Понятие жизненного цикла ПО
2. Модели жизненного цикла
3. Методологии разработки ПО

Жизненный цикл проекта



Модели жизненного цикла и методологии

- ❖Каскадная модель
- ❖V-образная модель
- ❖Прототипирование (Макетирование)
- ❖Инкрементная модель
- ❖Быстрая разработка приложений
- ❖Спиральная модель
- ❖Компонентно-ориентированная модель
- ❖Экстремальное программирование

- ✓Rational Unified Process (RUP)
- ✓Microsoft Solutions Framework(MSF)
- ✓Семейство Agile

Типы жизненных циклов

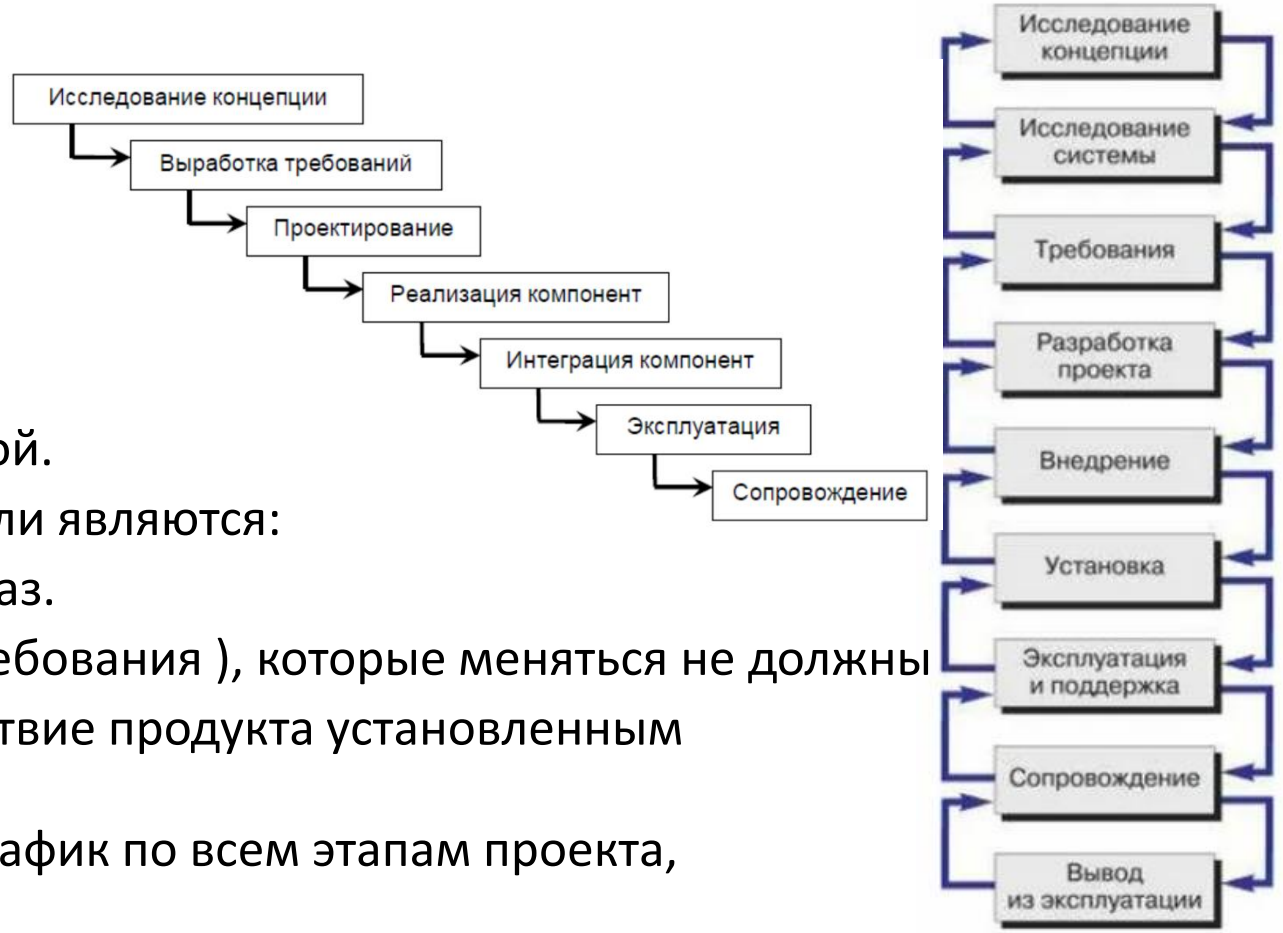
Прогнозирующие ЖЦ

- ставят оптимизацию процесса выше адаптивности
- Водопад, макетирование, RAD, инкремент, спираль

Адаптивные ЖЦ

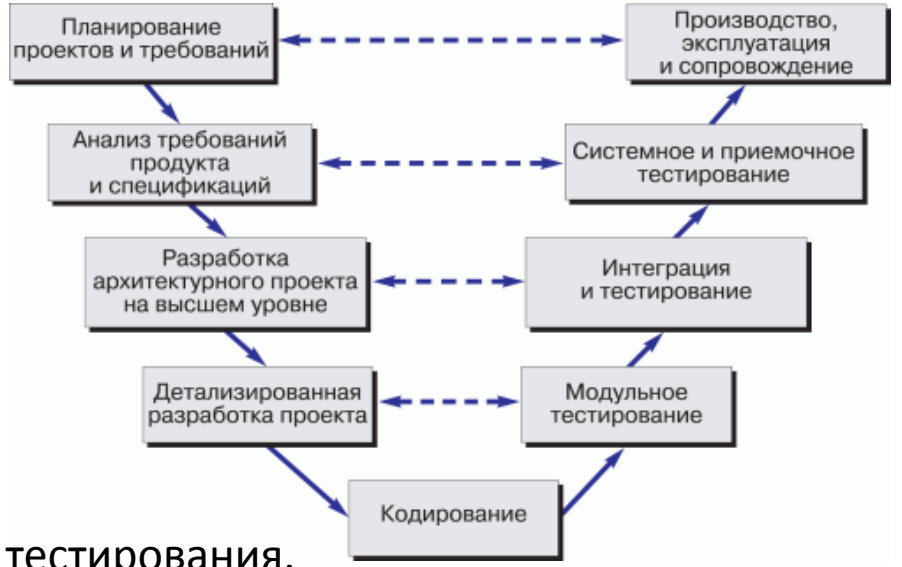
- принимают и схватывают изменения в ходе процесса разработки и отвергают детальное планирование
- Agile: экстремальное программирование XP, SCRUM и т.д.

Каскадная модель, водопад (Waterfall)



- Модель с 1970 года, является самой первой.
- Основными принципами каскадной модели являются:
- - Строго последовательное выполнение фаз.
- - Основа модели – сформулированные требования), которые меняться не должны
- - Критерий качества результата – соответствие продукта установленным требованиям.
- *Достоинства:* дает план и временной график по всем этапам проекта, упорядочивает ход конструирования.
- *Недостатки*:
- 1) реальные проекты часто требуют отклонения от стандартной последовательности шагов;
- 2)цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично);
- 3) результаты проекта доступны заказчику только в конце работы.

V-образная модель

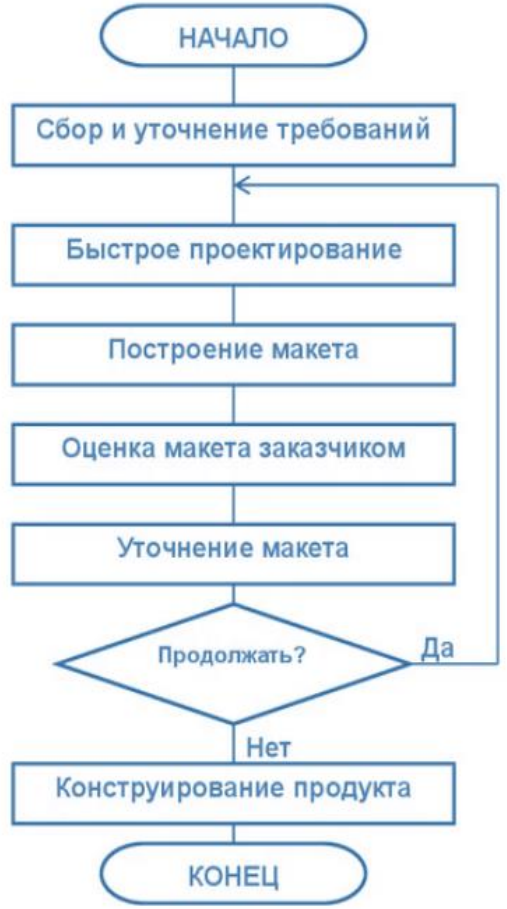


- V-образная модель была создана как итерационная разновидность каскадной модели.
- Целями итераций в этой модели является обеспечение процесса тестирования.
- Также известна как модель верификации и валидации.
- Тестирование продукта обсуждается, проектируется и планируется на ранних этапах жизненного цикла разработки. План испытания приемки заказчиком разрабатывается на этапе планирования, а компоновочного испытания системы - на фазах анализа, разработки проекта и т.д.
- Достоинства:
  - Пользователи V-модели участвуют в разработке и поддержке модели.
  - V-model позволяет разбить деятельность на отдельные шаги, каждый из которых будет включать в себя необходимые для него действия, инструкции к ним, рекомендации и подробное объяснение деятельности.
- Ограничения:
  - Организация и выполнение управления, обслуживания, ремонта и утилизации системы не учитываются в V-модели. Однако, планирование и подготовка к этим операциям моделью рассматриваются.
  - V-образная модель больше касается разработки программного обеспечения в проекте, чем всей организации процесса.

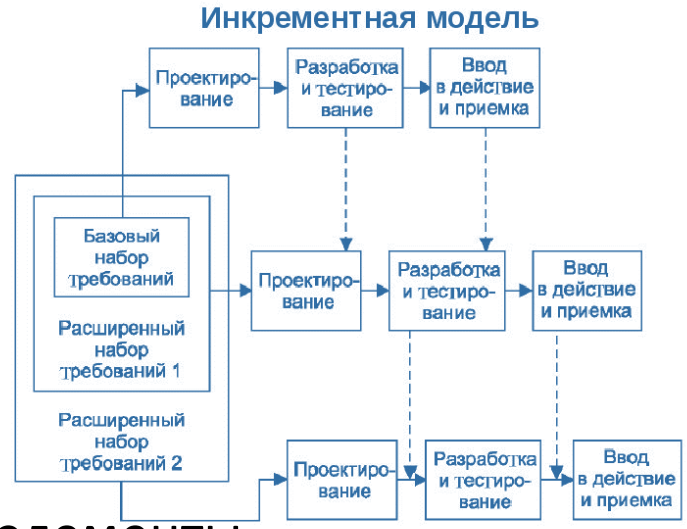
Прототипирование (Макетирование)

- Основная цель прототипирования — снять неопределенности в требованиях заказчика.
- Макетирование (прототипирование) — это процесс создания модели требуемого программного продукта.
- Макетирование основывается на многократном повторении итераций, в которых участвуют заказчик и разработчик.
- *Достоинство:*
- обеспечивает определение полных требований к ПО.
- *Недостатки:*

- заказчик может принять макет за продукт;
- разработчик может принять макет за продукт.



Инкрементная модель

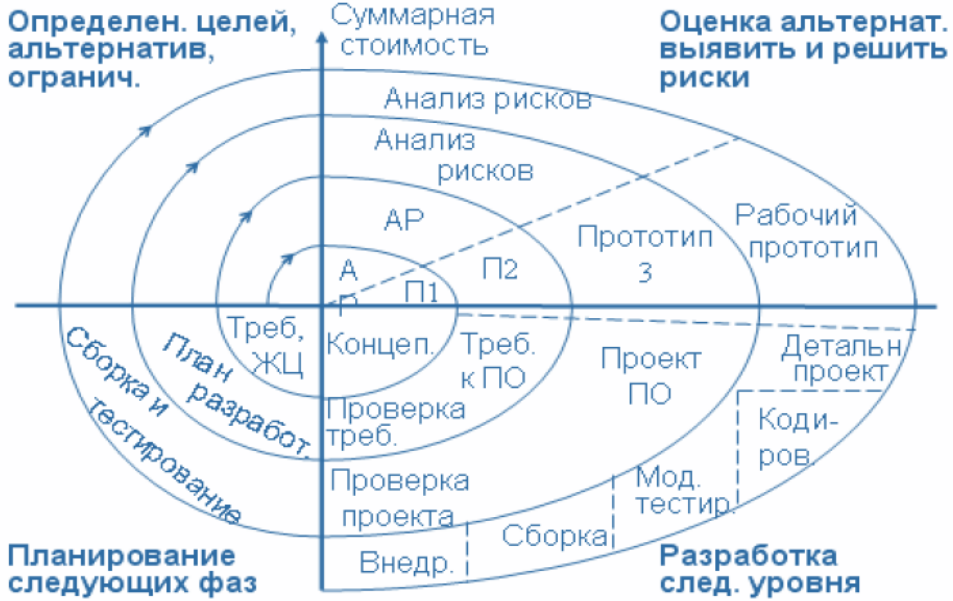


- Инкрементная модель является классическим примером инкрементной стратегии конструирования. Она объединяет элементы последовательной водопадной модели с итерационной философией макетирования.
- Еще называют модель расширения системы.
- Достоинства: затраты, которые получаются в связи с изменением требований пользователей, уменьшаются, повторный анализ и совокупность документации значительно сокращаются по сравнению с каскадной моделью; легче получить отзывы от клиента о проделанной работе — клиенты могут озвучить свои комментарии в отношении готовых частей.
- Недостатки: менеджеры должны постоянно измерять прогресс процесса, в случае быстрой разработки не стоит создавать документы для каждого минимального изменения версии; структура системы имеет тенденцию к ухудшению при добавлении новых компонентов — постоянные изменения нарушают структуру системы. Чтобы избежать этого требуется дополнительное время и деньги на рефакторинг.

Быстрая разработка приложений

- RAD (rapid application development) — концепция организации технологического процесса разработки программных продуктов, ориентированная на максимально быстрое получение результата в условиях сильных ограничений по срокам и бюджету и нечётко определённых требований к продукту. Это второй пример применения инкрементной стратегии конструирования.
- RAD — высокоскоростная адаптация линейной последовательной модели, в которой быстрая разработка достигается за счет использования компонентно-ориентированного конструирования. Если требования полностью определены, а проектная область ограничена, RAD-процесс позволяет группе создать полностью функциональную систему за очень короткое время (60-90 дней).
- Достоинства: благодаря принципу временного блока уменьшаются затраты и риск, связанный с соблюдением графика; в состав каждого временного блока входит анализ, проектирование и внедрение (фазы отделены от действий); основное внимание переносится с документации на код, причем при этом справедлив принцип "получаете то, что видите" (Whatyouseeiswhatyouget, WYSIWYG).
- Недостатки: пользователь должен всегда принимать участие на всех этапах разработке; необходимо достаточное количество высококвалифицированных и хорошо обученных разработчиков; использование модели может оказаться неудачным в случае, если отсутствуют пригодные для повторного использования компоненты; жесткие временные ограничения.

Спиральная модель



- Спиральная модель — классический пример применения эволюционной стратегии конструирования.
- Спиральная модель (автор Барри Боэм, 1988) базируется на лучших свойствах классического жизненного цикла и макетирования, к которым добавляется новый элемент — анализ риска, отсутствующий в этих парадигмах.
- Модель определяет четыре действия, представляемые четырьмя квадрантами спирали.
- С каждой итерацией по спирали (продвижением от центра к периферии) строятся все более полные версии ПО.
- Достоинства спиральной модели: наиболее реально (в виде эволюции) отображает разработку программного обеспечения; позволяет явно учитывать риск на каждом витке эволюции разработки; включает шаг системного подхода в итерационную структуру разработки; использует моделирование для уменьшения риска и совершенствования программного изделия.
- Недостатки спиральной модели: повышенные требования к заказчику; трудности контроля и управления временем разработки.

Компонентно-ориентированная модель

- Компонентно-ориентированная модель является развитием спиральной модели и тоже основывается на эволюционной стратегии конструирования. В этой модели конкретизируется содержание квадранта конструирования — оно отражает тот факт, что в современных условиях новая разработка должна основываться на повторном использовании существующих программных компонентов.
- Достоинства компонентно-ориентированной модели:
- 1) уменьшает на 30% время разработки программного продукта;
- 2) уменьшает стоимость программной разработки до 70%;
- 3) увеличивает в полтора раза производительность разработки

Extreme Programming (XP)

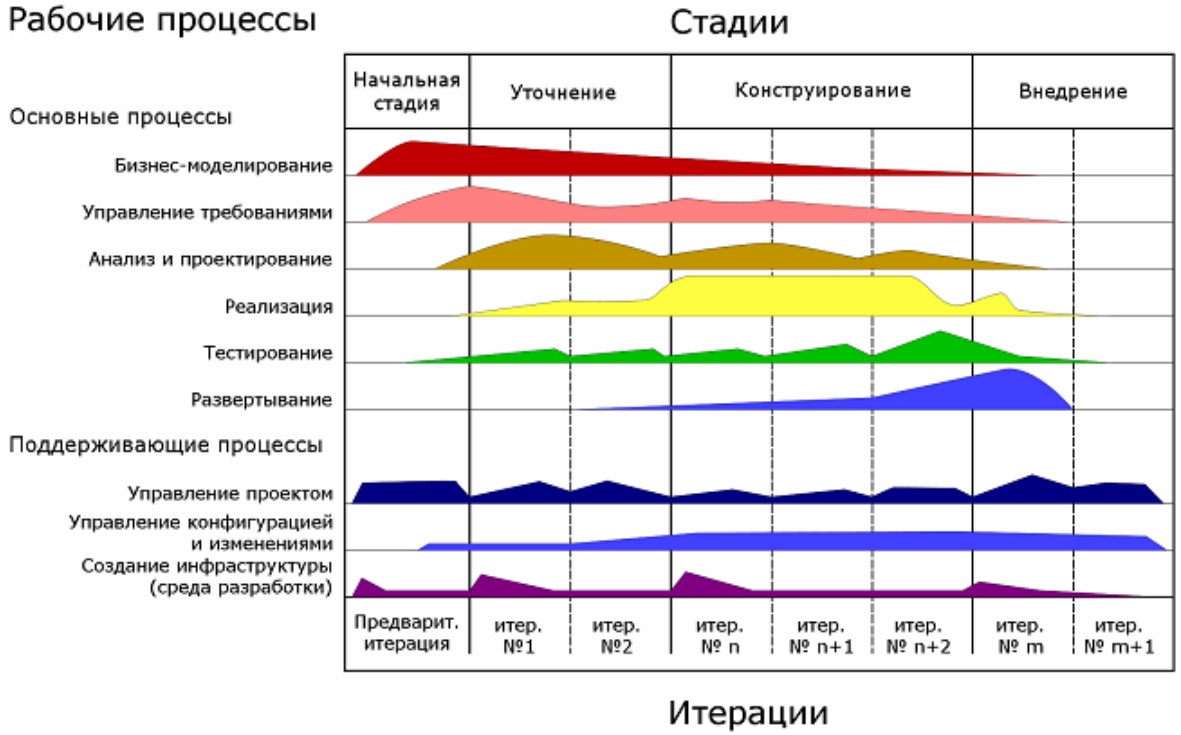
- Экстремальное программирование является примером так называемого метода «живой» или гибкой разработки.
- Модель жизненного цикла XP является итерационно-инкрементной моделью быстрого создания (и модификации) протопопов продукта, удовлетворяющих очередному требованию

Жизненные циклы

Наименование	Краткое описание
Водопад	Основные операции по разработке программного продукта линейно упорядочены, каждая фаза обычно завершается до начала следующей.
V-образная	Является расширением модели водопада и основана на связи фазы тестирования для каждой соответствующей стадии разработки.
Прототипирование	Функциональные требования и проектно-конструкторские спецификации генерируются одновременно
Инкремент	Разбиение большого объема проектно-конструкторских работ на ряд меньших составных частей
Быстрая разработка приложений	Предполагает групповую работу и применение инструментальных средств визуального моделирования и генерации приложения
Спираль	Повторение одного и того же набора фаз ЖЦ, включающего планирование, проектирование, построение и оценивание и так до тех пор, пока разработка продукта не будет завершена
Компонентно-ориентированнная	Разновидность спиральной модели, предполагает ускорение хода работ за счет повторного использования компонент
Extreme Programming	один из фреймворков Agile подхода, который позволяет создавать программное обеспечение высокого качества

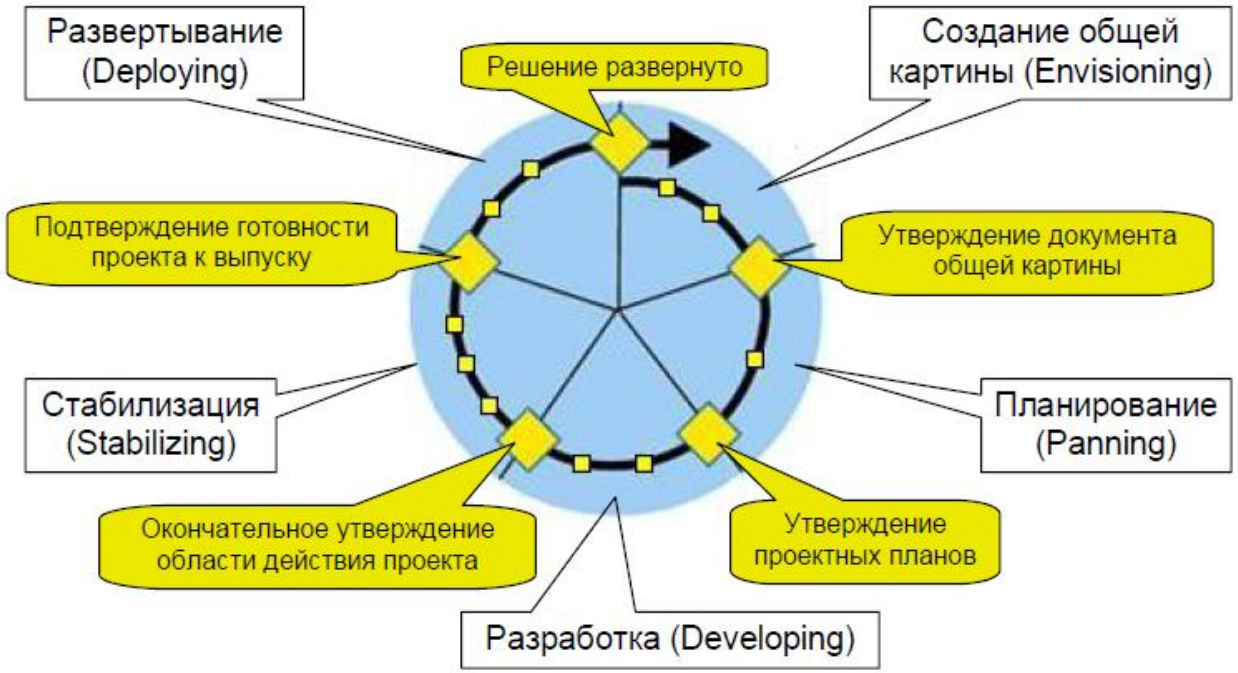
Rational Unified Process (RUP)

- ✓ Rational Unified Process (RUP) — методология разработки программного обеспечения, созданная компанией Rational Software. Основана на процессах объектно-ориентированного программирования и дополняется видами деятельности как бизнес-моделирование, управление проектом и управление конфигурациями.

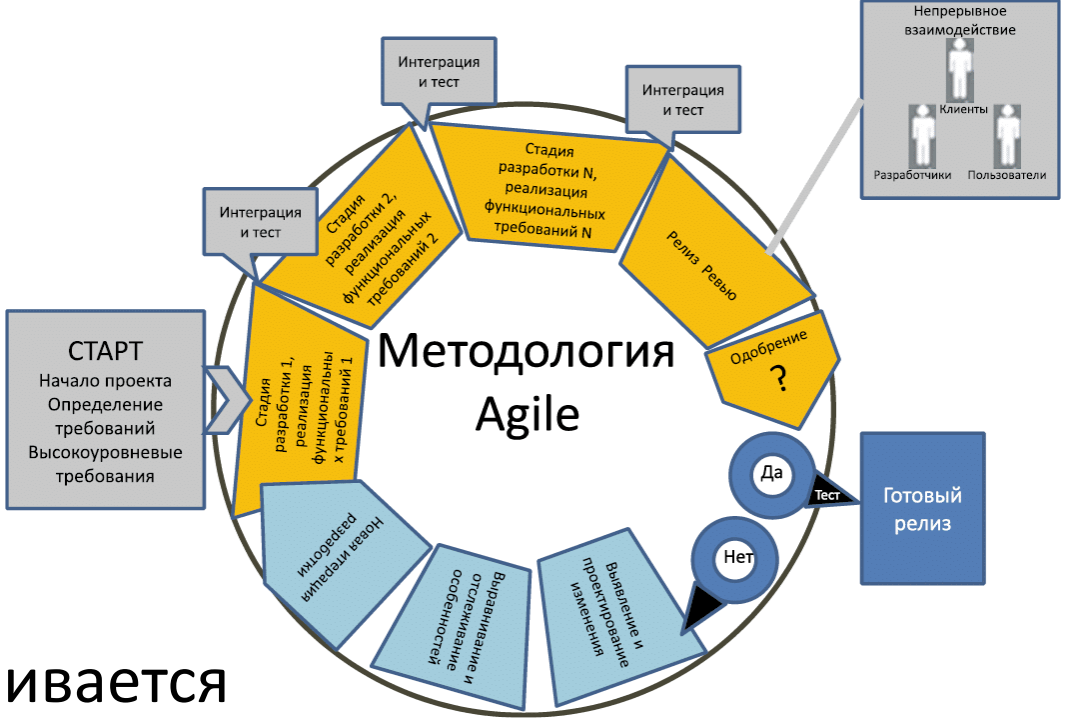


Microsoft Solutions Framework (MSF)

- Методология разработки программного обеспечения от Microsoft. MSF опирается на практический опыт корпорации Майкрософт и описывает управление людьми и рабочими процессами в процессе разработки решения.
- MSF состоит из принципов, моделей и дисциплин по управлению персоналом, процессами, технологическими элементами и связанными со всеми этими факторами вопросами, характерными для большинства проектов.



Agile



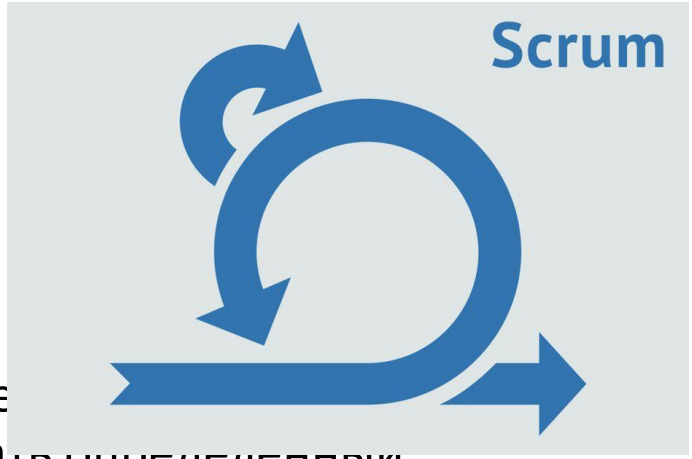
- Agile–семейство гибких итеративно-инкрементальных методов к управлению проектами и продуктами. Согласно данному подходу, проект разбивается не на последовательные фазы, а на маленькие подпроекты, которые затем «собираются» в готовый продукт.
- К гибким методологиям относят Extreme Programmin, DSDM, Канбан, Scrum, FDD и др.
- Принят в 2001 году на основе Agile Manifesto, который содержит 4 основные идеи и 12 принципов.

➢ люди и взаимодействие важнее процессов и инструментов;
➢ работающий продукт важнее исчерпывающей документации;
➢ сотрудничество с заказчиком важнее согласования условий контракта;
➢ готовность к изменениям важнее следования первоначальному плану.

ПРИНЦИПЫ agile

- наивысшим приоритетом признается удовлетворение заказчика за счёт ранней и бесперебойной поставки ценного программного обеспечения;
- изменение требований приветствуется даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
- частая поставка работающего программного обеспечения (каждые пару недель или пару месяцев с предпочтением меньшего периода);
- общение представителей бизнеса с разработчиками должно быть ежедневным на протяжении всего проекта;
- проекты следует строить вокруг заинтересованных людей, которых следует обеспечить нужными условиями работы, поддержкой и доверием;
- самый эффективный метод обмена информацией в команде — личная встреча;
- работающее программное обеспечение — лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
- постоянное внимание к техническому совершенству и хорошему проектированию увеличивают гибкость;
- простота как искусство не делать лишней работы очень важна;
- лучшие требования, архитектура и проектные решения получаются у самоорганизующихся команд;
- команда регулярно обдумывает способы повышения своей эффективности и соответственно корректирует рабочий процесс.

SCRUM



- SCRUM выполняется на итеративной основе, итерации носят название спринт продолжаются порядка 30 дней: каждый спринт на выходе должен дать определенный результат — обеспечить некоторую функциональность продукта.
- Scrum разбивает проект на части, которые сразу могут быть использованы Заказчиком для получения ценности, называемые заделами продуктов (product backlog).

Scrum процесс

Kanban

- Kanban намного менее строгий, нежели Scrum – он не ограничивает время спринтов, нет ролей, за исключением владельца продукта. Kanban даже позволяет члену команды вести несколько задач одновременно, чего не позволяет Scrum.
- Необходимо определить этапы потока операций (workflow) – столбцы в таблице.
- Для каждой задачи создаётся индивидуальная карточка, в которую заносится вся необходима информация о задаче. Карточка перемещается по этапам.
- Количество карточек на одном этапе строго регламентировано. Благодаря этому сразу становится видно, когда в потоке операций возникает «затор», который оперативно устраняется.
- Задачи из беклога попадают в поток в порядке приоритета. Таким образом, работа никогда не прекращается.
- Постоянное улучшение (kaizen): концепция постоянного улучшения появилась в Японии в конце XX века. Её суть в постоянном анализе производственного процесса и поиске путей повышения производительности.