Agile методологии-Резюме

Основни концепции:

- **Agile методи:** Възникнали в отговор сложността на традиционните методологии, Agile-м-те поставят акцент на:
    - **Бързо разработване** и промяяна на софтуер **-** Спецификацията, проектирането и внедряването се извършват едновременно; Системата се разработва на етапи, под формата на няколко версии, като заинтересованите страни участват в оценката на всяка версия; Потребителските интерфейси често се създават с помощта на интегрирана среда за разработка (IDE) и графични инструменти.
    - **Минимизиране на документирането**
    - Интерактивно **участие на клиентите** и бързо реагиране на промени
    - Фокус върху кода
    - Итерации
    - Целта на гъвкавите методи е да се намали сложността в софтуерния процес (например чрез ограничаване на документацията) и да се реагира бързо на променящите се изисквания без прекомерна повторна работа

    Манифест за гъвкави методи (Agile Manifesto):

    - **Хората и взаимодействията** пред процесите.
    - **Работещ софтуер** пред изчерпателната документация.
    - **Сътрудничество с клиента пред договорите**.

- **Реагиране на промени пред следване на предварителни планове**.

Практики и принципи:

- **Клиентско участие:** Клиентите са включени активно в процеса, предоставяйки обратна връзка и приоритети.
- **Инкрементна доставка:** Софтуерът се разработва и предоставя на малки стъпки.
- **Хора, а не процеси**
- **Приемане на промени**
- **Поддържане на простота:** Фокус върху намаляване на сложността в процесите и кода.

Приложимост: при малък и среден продукт; фокуса върху малки, тясно интегрирани екипи- не подходящ за разработка на големи системи; при ясно изразен ангажимент от страна на клиента да участва активно в процеса на разработка и когато няма множество външни правила и регулации

Методи:

- **Extreme Programming (XP):**
  - **Чести обновления:** Нови версии на софтуера могат да бъдат изграждани няколко пъти на ден;
  - Инкременти се доставят на клиентите на всеки 2 седмици;
  - **Тестове преди написване на кода** или всички тестове трябва да бъдат изпълнени за всяки build, като build-а се приема само ако тестовете преминат успешно.
  - **Работещ софтуер на всеки етап.**

- **Scrum:** Модел за управление с фокус върху итеративно разработване чрез "спринтове" (цикли от 2-4 седмици).

Практики на екстремно програмиране:

- **Инкрементално планиране**: Изискванията се записват върху карти на **историите**, а историите, които трябва да бъдат включени в издание, се определят от наличното време и тяхната относителна приоритетност. Разработчиците разбиват тези истории на **задачи**. Тези задачи са основата за оценки на графика и разходите.
- **Малки издания**: Разработва се минимално полезен комплект функционалности, който предоставя бизнес стойност, като първото издание съдържа основната функционалност. Изданията на системата са чести и инкрементално добавят нови функционалности към първоначалното издание.
- **Прост дизайн**: Извършва се достатъчно проектиране, за да се изпълнят текущите изисквания, и нищо повече.
- **Развитие чрез тестове**: Използва се автоматизирана рамка за единични тестове, за да се напишат тестове за нова функционалност преди тя да бъде внедрена.
- **Рефакториране**: Всички разработчици се очаква да **рефакторират кода непрекъснато**, веднага щом се открият възможности за подобрение на кода. Това поддържа кода прост, поддържим и улеснява въвеждането на промени, когато те трябва да бъдат реализирани; **добре структуриран** и **ясен код; архитектурно рефакториране** понякога
- **Паралелно програмиране**: по двойки
- **Колективна собственост**: Двойките разработчици работят върху всички области на системата и всички разработчици носят отговорност за целия код
- **Непрекъсната интеграция**: Веднага щом задачата е завършена, тя се интегрира в цялата система. Тестовете след това да минат успешно

- **Устойчив темп**: Дългите извънредни часове не се считат за приемливи
- **Клиент на място**: клиентът член на екипа; Клиентът избира историите, които ще бъдат включени в следващото издание, въз основа на техните приоритети и оценките на графика.

Примери за рефакториране:

- **Реорганизация на йерархията на класовете** с цел премахване на дублиран код.
- **Подреждане и преименуване** на атрибути и методи, за да се направят по-лесни за разбиране.
- **Заменяне на вградения код с извиквания на методи**, които са включени в библиотека на програмата.

Характеристики на тестването в XP:

- **Развитие на тестове преди код (Test-first development).**
- **Инкрементално развитие на тестове** от сценарии.
- **Участие на потребителя** в разработката и валидирането на тестовете.
- Използване на **автоматизирани тестови системи**, които стартират всички тестове на компоненти всеки път, когато бъде изградена нова версия; тестовете са програми
- Обикновено се използва тестова рамка като **JUnit**.

Процесът на Scrum

- **Спринт цикъл**: Спринтовете са с фиксирана продължителност, обикновено 2-4 седмици. Те съответстват на разработката на нова версия на системата в XP.
- Началната точка за планиране е **продуктовия беклог**, който представлява списък с работата, която трябва да бъде извършена по проекта.

- **Фазата на избор** включва целия проектен екип, който работи с клиента, за да изберат функциите и функционалностите, които ще бъдат разработени по време на спринта.

Ползи от Scrum

- Продуктът е разделен на набор от управляеми и разбираеми части.
- Нестабилните изисквания не спират напредъка.
- Целият екип има видимост върху всичко, което подобрява комуникацията в екипа.
- Клиентите виждат навременното доставяне на нови версии и получават обратна връзка за работата на продукта.
- Създава се доверие между клиентите и разработчиците, което създава положителна култура, в която всички очакват успеха на проекта.

Мащабиране на Agile методите (Scaling agile methods)

- Agile методите са доказали своята успешност при малки и средни проекти, които могат да бъдат разработвани от малък съвместно работещ екип.
- **Мащабирането на Agile методи** включва промени в тези методи, за да се справят с по-големи и по-дълги проекти, при които има множество екипи, работещи на различни места.

Управление на проекти:

- **Гъвкаво управление:** Подход, адаптиран за итеративно разработване, включващ редовни срещи, видимост на напредъка и бърза обратна връзка.

- **Scrum мастер:** Роля, която гарантира защита на екипа от външни смущения и подпомага комуникацията.

Заключение:

Agile е подход, насочен към ускоряване на разработката, активно включване на клиентите и справяне с променящите се изисквания. Методите като XP и Scrum демонстрират как Agile може да се приложи успешно, особено за малки и средни проекти.