Metodiky a životní cyklus vývoje softwaru

Životní cyklus vývoje softwaru
strukturovaný proces, který popisuje všechny fáze, kterými software prochází od myšlenky až po ukončení.

Jednolitvé fáze
Analýza požadavků - shromažďují se všechny požadavky od zákazníka nebo uživatelů, účastní se analytici, zákazníci, manažeři
Návrh systému - plánuje se jak to bude všechno celý fungovat - architektura, diagramy UML, UI, databáze
Implementace - psaní kódu podle návrhu, rozděluje se program na různé části podle funkčnosti, github třeba
Testování - testování všech funkcí, ověřování funkcionalit
Nasazení - software běží v produkčním prostředí,
Údržba a podpora - oprava chyb, updaty, monitorování

Hlavní metodiky při vývoji

1. Vodopádový model

Lineární sekvenční proces, kde každá fáze musí být dokončena před začátkem další. Používá například NASA při vývoji softwaru nebo lékařské přístroje. Požadavky musí být jasně a přesně definovány na začátku při analýze. Následné změny v pozdějších fázích jsou velmi složité a nákladné. Zákazník vidí až hotový produkt v posledních fázích projektů

+ jednoduchý na pochopení a řízení
+ jasně definované milníky a výstupy
+ důkladná dokumentace každé fáze
+ definovaná zodpovědnost v každé fázi
- Nulová flexibilita
- Extrémně závislá na analýze
- pozdní probémy jsou špatná věc
- zákazník nemá průběžný přehled o vývoji

modifikace vodopádu
Inkrementální model - pricnip verzování - rozděluje vývoj na menší částim které se vyvíjí postupně
Spirálový model - kombinuje inkrementální model s důrazem na analýzu rizik, mezi jednotlivé kroky vývoje vkládá další procesy
Průzkumné programování - zaměřuje se na pochopení potřeb zákazníka prostřednictvím rychlého prototypy

2. Agilní model

Vyvíjely se postupně jako reakce na problémy tradičních přístupů k vývoji softwaru. Zaměřují se na flexibilitu, spolupráci a rychlé dodávání fungujícího produktu.

SCRUM
Jedná se o framework pro řízení vývoje, které rozděluje práci do krátkých časově ohraničených intervalů (sprinty)

Role
- Product Owner - komunikuje se zákazníkem
- Scrum Master - vedoucí vývojářů, rozdává úkoly- tvoří sprinty
- Scrum team member - vývojář v týmu

Klíčové události
- Sprint - základní časová jednotka, 1-4 týdny
- Sprint Planning - plánování práce na začátku sprintu, odpovída ná na otázky Co se bude dělat a Jak se to bude dělat
- Daily scrum meeting - denní schůzky, členové týmu popisují svůj kousek práce, nějaký problémy atd..
- Sprint review - probíhá na konci sprintu, získání feedbacku
- Sprint retrospective - zhodnocení práce, co bylo good co ne co dělat jinak atd…

Jednoduchý příklad

Tým vyvíjí mobilní aplikaci pro sledování výdajů:

1. Product Backlog obsahuje prioritizované funkce jako:
    - Uživatel se může registrovat
    - Uživatel může přidat výdaj
    - Uživatel může zobrazit měsíční statistiky
    - atd.
2. Pro první dvoutýdenní sprint:
    - Na Sprint Planning tým vybere funkce "registrace" a "přidání výdaje"
    - Tým rozdělí tyto funkce na konkrétní úkoly a vytvoří Sprint Backlog
    - Každý den se tým schází na Daily Scrum a diskutuje o pokroku
    - Po dvou týdnech tým na Sprint Review předvede fungující registraci a přidávání výdajů
    - Na Sprint Retrospective tým identifikuje, že měli problémy s testováním, a naplánují zlepšení
3. Pro druhý sprint:

- Tým vybere další funkce z Product Backlogu
- Implementují zlepšení identifikované v retrospektivě
- A celý cyklus se opakuje

3. Extrémní programování

Agilní metodika vývoje softwaru, která dovádí běžné praktiky do extrému, jde hlouběji do technických aspketů. Záleží na rychlost a správnosti.

Nejdůležitější hodnoty:
Komunikace - přímá, osobní komunikace se všemi členy týmu, problémy se řeší okamžitě a přímo
Jednoduchost - programuje se vždycky to co je potřeba, zaměřuje se na současné požadavky, výhýba se spekulativnímu kódu

Praktiky XP
Párové programování
Dva programátoři jsou u jednoho počítače - jeden píše kód, druhý přemýšlí strategicky a kontroluje. Navzájem se střídají. Průběžně si navzájem kontrolují kód, sdílí se znalosti a návrhy.

Test-driven Development
1. Nejprve napsat test pro novou funkci (test selže)
2. Implementovat nejjednodušší kód, který způsobí, že test projde
3. Refaktorovat kód pro lepší čitelnost a efektivitu

Kdy použít XP

Extrémní programování je nejvhodnější v situacích, kdy:

- Požadavky se často mění nebo nejsou na začátku jasné
- Projekt má vysoké technické riziko
- Kvalita kódu je kritická pro úspěch projektu
- Tým je malý a soudržný (typicky 3-10 vývojářů)
- Zákazník je ochoten aktivně spolupracovat s vývojovým týmem