



**SPŠE
JEČNÁ**

**STŘEDNÍ
PRŮMYSLOVÁ
ŠKOLA
ELEKTROTECHNICKÁ**

Metodiky a životní cyklus vývoje softwaru

- **Anki Flashcards:** [Anki](#) [Flashcards](#)
- **Autoři:** @danielkahoun @MaksymDoremi @OndrejKulhavy
- **Video:**

Shrnutí

Tento text se zabývá metodikami a životním cyklem vývoje softwaru, konkrétně Software Development Life Cycle (SDLC). Obsahuje analýzu a plánování, návrh, implementaci, testování, nasazení a údržbu. Dále porovnává různé metodiky vývoje, jako je Waterfall, Spirální model, Agilní přístup (s důrazem na Scrum), Extrémní programování, Prototypovací model a RUP (Rational Unified Process).

SDLC slouží jako rámec pro plánování, návrh a údržbu softwarových systémů, s každou fází zaměřenou na specifické cíle. Porovnání metodik ukazuje, že Agilní přístup a Scrum nabízejí rychlou zpětnou vazbu a flexibilitu, ale vyžadují úzkou spolupráci se zákazníkem. Spirální model kombinuje designový a prototypový přístup, zatímco RUP nabízí iterativní a inkrementální přístup vhodný pro velké projekty.

Text rovněž prezentuje výhody a nevýhody jednotlivých metodik, například jednoduchost Waterfallu a jeho nepružnost vůči změnám, flexibilitu Spirálního modelu na úkor nákladů a časové náročnosti, a agilní přístup s jeho rychlým dodáním, ale náročným plánováním. RUP zdůrazňuje důkladnou dokumentaci a schopnost odhalit chyby, ale může být nákladný a nevhodný pro malé projekty.

Metodiky a životní cyklus vývoje softwaru

SDLC (Software Development Life Cycle) je soubor procesů nebo fází, které softwarový produkt prochází od jeho návrhu až po jeho údržbu a odstranění z provozu. SDLC je rámec, který organizace používají k plánování, návrhu, vytváření, testování a údržbě softwarových systémů. Každá fáze SDLC má své vlastní cíle a úkoly, a celkově pomáhá zajistit, že vývoj softwaru probíhá systematicky a efektivně.

Často používané fáze v rámci životního vývoje softwaru (SDLC)

Fáze popsané níže se mírně liší na základě vybrané metodiky vývoje. V základu jsou ale všechny stejné.



Analýza a plánování

- Rozpoznání problémů k vyřešení
- Stanovení požadavků
- Definice cílů



Návrh

- Vytvoření technického návrhu podle zákaznických požadavků
- Zpracování analýzy včetně UML diagramů a schémat databáze
- Výběr frameworků, programovacích jazyků a vývojového prostředí
- Odhad ceny pro celý projekt (Waterfall) nebo iteraci (např. Scrum)



Implementace

- Tvorba zdrojového kódu a databáze
- Optimalizace pro výkon



Testování

- QA tým naplní demo verzi testovacími daty
- Ověření správnosti fungování produktu
- Chyby zaznamenány do logu pro opravu



Nasazení

- Uvedení hotového produktu do reálného provozu

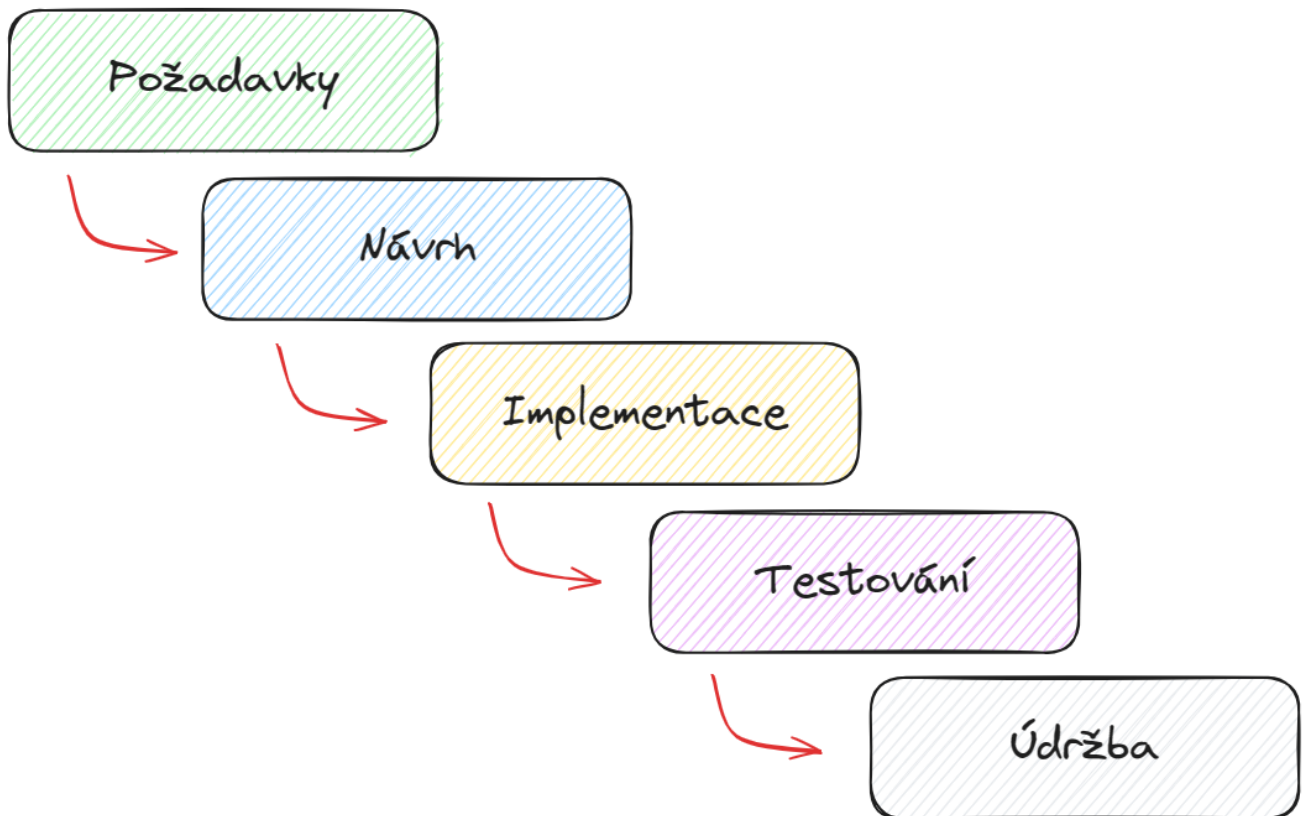


Údržba

- Oprava chyb a řešení zákaznických problémů
- Správa změn v softwaru
- Monitoring výkonu, bezpečnosti a uživatelské zkušenosti pro neustálé zlepšování.

Klasické přístupy

Waterfall



- Jedná se o základní model, první co vznikl v rámci vývoje software.
- Po dokončení jedné fáze, hned přechází dále.
- Je třeba důkladně pracovat na prvních fázích, aby nedošlo ke zmatku potom.

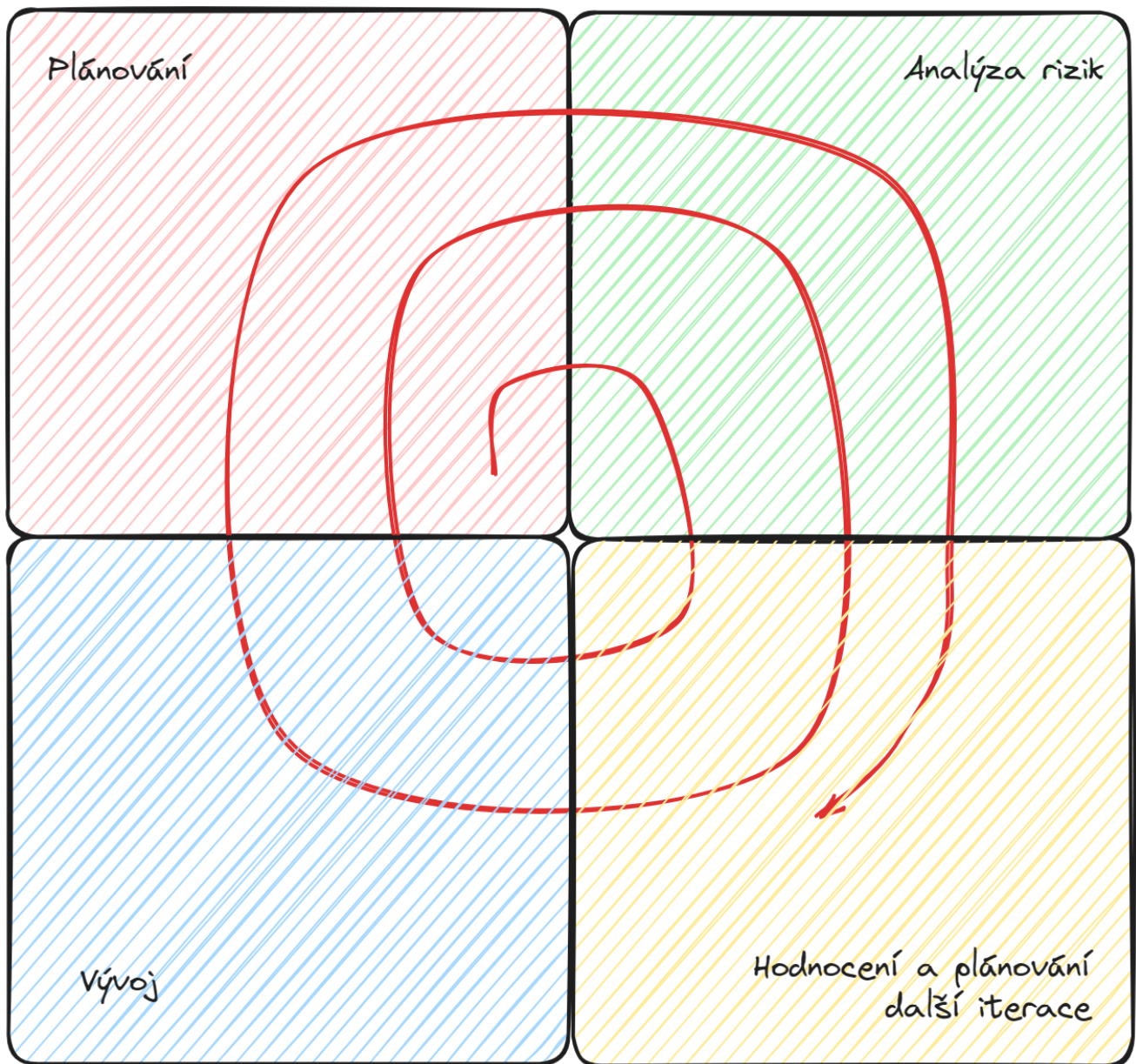
✓ Výhody

- Jednoduchý model
- Vhodný pro malé projekty

⚡ Nevýhody

- Není flexibilní
- Nevhodný pro velké projekty
- Není cesty zpět na předchozí fáze
- Těžko se odhadují náklady a časová náročnost projektu

Spirální model



Spirální model vývoje software obsahuje několik fází, které se pravidelně opakují, aby zajistily postupné zdokonalování projektu. Tyto fáze jsou:

1. Plánování:

- Stanovení cílů projektu.
- Určení omezení a požadavků.
- Plánování zdrojů a časového rozvrhu.
- Identifikace možných rizik a jejich řízení.

2. Analýza rizik:

- Identifikace rizik spojených s projektem.
- Zhodnocení dopadu rizik.
- Plánování opatření na minimalizaci rizik.

3. Vývoj:

- Vytvoření a vývoj softwaru podle specifikací.
- Implementace a testování jednotlivých částí software.
- Monitorování a kontrola kvality.

4. Hodnocení a plánování další iterace:

- Získání zpětné vazby od zákazníka nebo uživatelů.
- Hodnocení výsledků a kvality softwaru.
- Plánování další fáze v závislosti na zpětné vazbě a potřebách projektu.

✓ Výhody

- **Flexibilita:** Umí se přizpůsobit měnícím se požadavkům během vývoje.
- **Iterativnost:** Opakované iterace umožňují postupné zlepšování a zdokonalování produktu.
- **Zahrnutí zákazníka:** Průběžné prezentace a zpětná vazba od zákazníka jsou integrované do procesu.
- **Postupné zvyšování složitosti:** Umožňuje postupné zvyšování složitosti produktu s každou iterací.
- **Včasná detekce chyb:** Díky průběžnému testování mohou být chyby detekovány a opraveny dříve v procesu vývoje.

⚡ Nevýhody

- **Náklady:** Implementace spirálového modelu může být nákladnější v porovnání s jednoduššími modely vývoje.
- **Časová náročnost:** Kvůli opakovaným iteracím a důrazu na analýzu a plánování může trvat déle než jiné modely dostat produkt na trh.
- **Nemusí být vhodný pro malé projekty:** Pro menší a méně komplexní projekty může být spirálový model příliš robustní a zdánlivě překomplikovaný.
- **Nepředvídatelnost:** Z důvodu neustálých změn a iterací může být obtížné předvídat časové a finanční nároky projektu

Agilní přístup

- Agilní vývoj probíhá v krátkých cyklech, nazývaných iterace, které obvykle trvají několik týdnů.
- Každá iterace produkuje funkční část produktu, která může být okamžitě nasazena.
- Týmy pracují společně a pravidelně komunikují s klientem nebo uživateli, aby lépe porozuměli jejich potřebám a získali zpětnou vazbu.
- Agilní vývoj se snaží dodávat funkční části produktu od začátku vývoje, což umožňuje rychlou zpětnou vazbu od zákazníka.

✓ Výhody

- Rychlé dodání
- Průběžná zpětná vazba
- Flexibilita

- Efektivní spolupráce

⚡ Nevýhody

- Složité vést dokumentaci
- Potřeba úzké spolupráce se zákazníkem
- Náročné plánování

Scrum

- Agilní, iterativní metodika vývoje.

Role

- Product Owner

- Komunikace se zákazníkem je hlavní odpovědností.
- Zabývá se obchodní stránkou projektu a nezasahuje do samotného vývoje.
- Informuje zákazníka o nových verzích, stanovuje priority, časový plán a financování.

- Scrum Master

- Scrum Master je lídrem týmu.
- Zajišťuje dodržování dohodnutých procesů a odpovídá za dosažení produktových cílů.
- Organizuje denní schůzky (denní scrum) a aktivně se vyjadřuje k průběhu vývoje.
- Jeho prioritou je odstraňování překážek bránících týmu v dodání softwaru.

- Scrum Team Member

- Člen vývojového týmu s odpovědností za vývoj softwaru.

Eventy

- **Sprint Planning Meeting**
 - Plánování rozvržení práce pro Sprint na základě požadavků zákazníka (tzv. User Stories).
- **Sprint**
 - Iterace trvající obvykle dva až čtyři týdny, začínající okamžitě po závěrech předchozího Sprintu.
- **Daily Scrum (stand-up)**
 - Každodenní 15 minutové setkání, konané ve stejný čas a na stejném místě.
- **Sprint Review**
 - Scrum Team prezentuje zákazníkovi výsledný software, a zákazník se vyjadřuje k verzi. V případě nespokojenosti se spouští nový Sprint.
- **Sprint Retrospective**

- Cílem je zjistit, jak lze zefektivnit a zkvalitnit vývoj, hodnotit se průběh a hledají se zlepšení.

Extrémní programování (XP)

- Všechno je v extrémech, představuje odvážný přístup k vývoji s důrazem na rychlé iterace, někdy v řádu minut či hodin.
- **Párové programování**
 - Dva nebo více vývojářů pracuje současně u jednoho počítače, což podporuje spolupráci a sdílení znalostí.
- **Využití v případě nedostatku času na dokončení zakázky**
 - XP se často používá, když je potřeba rychle reagovat na změny v požadavcích nebo pokud hrozí, že se projekt nedokončí včas.
- **Intenzivní testování**
 - Klade důraz na průběžné a systematické testování, což má za cíl zajištění kvality a minimalizaci chyb.
- **Není zaměřen na management**
 - XP klade méně důrazu na tradiční manažerské procesy a hierarchie, spíše se zaměřuje na pružnost a rychlou odezvu na změny.

Prototypovací model

Prototypování představuje realizaci projektu prostřednictvím vytváření prototypů aplikací, což jsou neúplné verze směřující k dosažení konečného výsledku.

Tato metoda je široce známá a používá se v různých odvětvích, jako je strojírenství či výroba.

Obvykle má prototyp za úkol simulovat určitou část hotového produktu, ačkoliv může dojít k tomu, že se prototyp výrazně liší od celkového provedení.

Účel prototypování:

Umožňuje budoucím uživatelům softwaru hodnotit návrh designu a funkcí prostřednictvím skutečného použití, nikoli pouze interpretací a komentováním popisu produktu.

Obecné fáze prototypování

1. Identifikace základních požadavků
2. Vývoj prvotního prototypu
3. Revize, zhodnocení, zpětná vazba
4. Úpravy a vylepšení prototypu

Druhy prototypů

- **Horizontální prototyp:**
 - První návrh uživatelského rozhraní.
 - Poskytuje náhled na celý systém nebo subsystém.

- Hlavním cílem je demonstrovat komunikaci s uživatelem spíše než funkčnost.
- **Vertikální prototyp:**
 - Detailnější propracování jednoho podsystému nebo funkce produktu.
 - Vhodný pro získání lepších požadavků na danou funkci.
 - Zjednodušuje složité požadavky.

Typy prototypování

- **Zahazovací prototypování (Rapidní prototypování):**
 - Vytvoření modelu, který není součástí finálního projektu.
 - Model je zahozen, ale slouží k vizualizaci požadované funkcionality pro zákazníka.
 - Poslouží jako návrh pro ostrou verzi projektu.
 - Model slouží jako východisko pro zpětnou vazbu od zákazníka.
 - Na základě získaných poznatků vzniká finální verze.
- **Evoluční prototypování:**
 - Vyvinutý model slouží jako jádro.
 - Zákazník obdrží verzi a může poskytnout zpětnou vazbu.
 - Postupně jsou provedeny nové úpravy a přidávány nové funkce.
 - Nejde o konečný produkt; vždy je možné najít chyby nebo přidat nové požadavky.
- **Inkrementální prototypování:**
 - Produkt se skládá z dílčích prototypů, které jsou nakonec spojeny do výsledného řešení.
- **Extrémní prototypování:**
 - Dělí se do tří fází, každá vychází z předchozí.
 - Zaměřuje se zejména na vývoj webových aplikací.
 - První fáze zahrnuje vytvoření statického prototypu.
 - Druhá fáze spočívá v kódování funkčních částí.
 - Třetí fáze zahrnuje implementaci funkcí.
 - Opět nejde o konečný produkt; vždy je možné odhalit chyby nebo přidat nové požadavky.

Rational Unified Process

- Iterativní a inkrementální, tradiční a striktní přístup k vývoji.
- Vhodný zejména pro rozsáhlé projekty v korporátním prostředí, například v korporacích nebo bankách.
- Striktně definuje požadavky, a nové požadavky jsou stanoveny až po dokončení každé iterace.
- Obsahuje čtyři hlavní fáze:
 - **Zahájení:** Sjednání potřeb celého projektu, posouzení proveditelnosti a vhodnosti. Nápad a struktura projektu jsou definovány.
 - **Rozpracování:** Analýza požadavků, tvorba detailních specifikací, návrh architektury a vytvoření plánu projektu.
 - **Konstrukce:** Implementace, testování, integrace a vytvoření dokumentace.
 - **Předání:** Nasazení, údržba, podpora a finalizace dokumentace.

- Každá iterace začíná až po dokončení předchozí a pokračuje, dokud není zákazník spokojen s výsledkem.

Popis činnosti

- **Business modelování**
 - Popisuje strukturu a dynamiku podniku nebo organizace, identifikuje aktivity a role jednotlivých účastníků (kdo co dělá).
- **Požadavky**
 - Stanovuje a dokumentuje požadavky od zákazníka, definuje funkční a nefunkční specifikace.
- **Analýza a design**
 - Zahrnuje návrh software, vytváření UML diagramů, a design databáze. Tato fáze připravuje podrobný plán implementace.
- **Implementace**
 - Zahrnuje samotnou tvorbu zdrojového kódu (v případě softwaru) nebo implementaci databáze. Realizuje plány vytvořené v předchozí fázi.
- **Testování**
 - Provádí testování softwaru, zajišťuje, že vytvořený produkt splňuje stanovené požadavky a je plně funkční.
- **Nasazení**
 - Zahrnuje nasazení hotového softwaru nebo produktu do reálného prostředí, aby byl k dispozici koncovým uživatelům.

✓ Výhody

- Iterativní, snadné odhalení chyb
- Důkladné vypracování dokumentace, UML
- Snazší správa změn

⚡ Nevýhody

- Nevhodný pro malé projekty
- Nákladný na implementaci

Zdroje a Zajímavé Odkazy

- [Software development process - Wikiwand](#)