# Metodiky a životní cyklus vývoje softwaru

*Poznámka: Základní princip je u všeho strašně podobný, takže se jako první naučte těch zhruba 5-6 kroků (požadavek, analýza, návrh, implementace,...), ty se opakují v různé formě a pořadí. Pak se jenom doučte specifika těch kterých věcí a snažte se rozumět tomu, jak to funguje a představit si to, ne jenom to jet zpaměti - když to budete chápat, budete to umět říct daleko líp.*

*Marek Gauss*

## Nejvýznamnější metodiky

### Vodopádový model

Vodopádový model je tzv. **sekvenční lineární proces**, kde se na vývoj hledí jako na „vodopád“ n fází. Rozbíjí projekt na jednotlivé fáze, kde každá další může začít až ve chvíli kdy předchozí skončila. Navíc se nemůžu vracet k dokončeným fázím nebo přeskakovat mezi fázemi. V opozici vůči němu stojí novodobá agilní metodika (např. Scrum ...). Jedná se o velmi zastaralou metodiku, kterou se v praxi v dnešní době nevyplatí využívat. Vyžaduje aby vše probíhalo dokonale.

#### Waterfall Methodology. Waterfall Methodology | by Chathmini Jayathilaka | Medium

#### Fáze vodopádu:

1. Specifikace požadavků
2. Návrh
3. Vývoj
4. Testování a ladění (validace)
5. Instalace, nasazení
6. Údržba

U vodopádu se skutečně přísně postupuje podle sekvenčního způsobu. Připraví se požadavky, které jsou dané a během vývoje se nemění. Když jsou požadavky kompletní, přechází se do fáze návrhu. Software je naprojektován a návrh je předán implementátorům; návrh by měl sloužit jako plán pro implementaci. Programátoři zodpovědní za implementaci návrh převezmou a podle něj stvoří software. V pozdějších fázích pak dochází ke skládání a kombinaci jednotlivých stvořených částí softwaru, modifikacím, opravám a zavádění nových funkcionalit.

Vodopádový model vyžaduje, aby se k další fázi přistoupilo jedině za podmínky, že předchozí fáze je kompletní a bezchybná. Existují i upravené modely na základě vodopádu, které se mohou výrazně lišit.

* pokud se věnuje dostatek času přípravě, může se čas ušetřit v pozdějších etapách vývoje
* klade stejný důraz na dokumentaci jako na zdrojový kód => noví členové týmu mají možnost se lépe začlenit a uchytit
* je jednoduchý a dobře pochopitelný, snadno se v něm hledají konkrétní cíle a milníky
* pokud máme jasně daný cíl (NASA či americké vládě) a víme přesně co a jak dělat nebo pracujeme na malém projektu kde by byly agilní metodiky overkill
* problém může nastat v případě, že klient nemá stoprocentně ujasněné požadavky
* panuje všeobecné přesvědčení, že nelze dovést jednu fázi vývoje do konečného stavu před tím, než bude započata fáze další
* je velice těžké v tomto modelu provádět změny nebo počítat s neočekávanou skutečností (změna požadavků klienta, obtížnost implementace určité funkcionality apod.); u vodopádového modelu je lepší při takové situaci rovnou utvářet nový model a spousta práce přijde vniveč

### Prototypovací model (software prototyping)

Prototypování je realizace projektu pomocí vytváření prototypů aplikací = neúplných verzí směřujících ke konečnému výsledku. Z obecného hlediska se prototypování považuje za široce známou metodu, která se používá nejen v IT, ale i například ve strojírenství nebo výrobě. Obvykle je smyslem prototypování simulovat určitou část hotového produktu; může dojít i k tomu, že se prototyp bude výrazně lišit od celku.

Původním účelem prototypování je umožnit budoucím uživatelům SW hodnotit návrh designu a funkcí pomocí skutečného použití, nejen interpretace a komentování popisu produktu.

#### Obecné fáze prototypování

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

Kroky 3 a 4 se opakují podle potřeby několikrát.



#### Druhy prototypů

*horizontální prototyp* - termín běžně poukazující na první návrh UI; slouží k náhledu na celý systém nebo subsystém a hlavním účelem je demonstrace komunikace s uživatelem spíše než funkčnost; hodí se pro potvrzení požadavků na UI, ukázku systému pro získání zpětné vazby a případných objednávek od potenciálních zákazníků a na předběžný odhad času, nákladů a náročnosti projektu

*vertikální prototyp* - úplnější propracování jednoho podsystému nebo funkce produktu; je vhodné pro získání lepších požadavků na danou funkci: zjemnění návrhu databáze, informace o objemu dat a výkonnostních potřebách (pro určení kapacity sítě apod.), zjednodušení složitých požadavků

#### Typy prototypování

Prototypování má více variant, všechny ale vycházejí ze dvou hlavních forem: **zahazovacího prototypování** a **evolučního prototypování**.

**Zahazovací prototypování** (také nazýváno uzavřené nebo rapidní prototypování) znamená stvoření modelu, který se ve finální fázi přímo nestane součástí projektu a bude zahozen, zatímco požadavek, který reprezentoval, se zavede jako samostatná, propracovanější implementace. Primární účel tohoto je vyobrazit zákazníkovi, jak zhruba bude požadovaná funkce/požadovaný subsystém vypadat.

U této metody není nejdůležitější částí vývoje formalita modelu, ale spíše rychlost, jakou je vytvořen. Model pak slouží jako východisko pro zpětnou vazbu od zájemce, podle kterého může dále upravit svoje požadavky. Když je dosaženo dostatečně jasných požadavků, model je zahozen a na základě sesbíraných poznatků se tvoří finální subsystém.

Souhrn:

1. Sepsání předběžných požadavkům
2. Návrh prototypu
3. Uživatelské ověření prototypu, definice nových požadavků
4. Opakování postupu, pokud je třeba
5. Sepsání výsledných požadavků

**Evoluční prototypování** se výrazně liší od zahazovacího. Jeho principem je strukturovaná tvorba robustního prototypu, který slouží jako jádro výsledného systému a poté pouze dochází k opravám a rozšířením. Při takovém vývoji dochází neustále k zpřesnění a zjemnění systému. „Evoluční prototypování uznává, že nerozumíme všem požadavkům a implementujeme pouze ty, které dobře chápeme.“

V evolučním prototypování teoreticky výrobek není nikdy hotov, pouze se rozvíjí s tím, jak se mění nároky na použití. Není neobvyklé podat uživateli nekompletní prototyp rovnou k praktickému používání; uživatelé často zjistí, že nedokončený systém je lepší než žádný. Aby se minimalizovalo riziko, implementují se pouze vlastnosti, které jsou dobře pochopeny. U zákazníka se nasazuje částečný systém, takže si zákazník může zažádat o nové funkce a vývojáři je pak zařadí mezi své požadavky a mohou je bez problémů realizovat.

**Inkrementální prototypování** znamená, že se produkt skládá z dílčích prototypů, které na závěr slučujeme na výsledné řešení.

**Extrémní prototypování** se používá především na vývoj webových aplikací. Vývoj se rozloží do tří fázi a každá vychází vždy z té předchozí. První fází je jednoduchý statický prototyp, který vychází z obyčejné HTML stránky. Druhá fáze je kódování obrazovek a plně funkčního UI se simulovanými službami. Třetí fáze je implementování potřebných/požadovaných služeb.

#### Výhody a nevýhody prototypování

* snížení času a nákladů: zvýší kvalitu požadavků a standardů pro vývojáře; protože při vývoji cena roste exponenciálně, včasné stanovení toho, co zákazník chce, zpravidla vede k rychlejšímu a méně nákladnému vývoji
* zlepšené zapojení uživatele: prototypování vyžaduje větší zapojení uživatele tak, aby mohl vidět prototyp a komunikovat s ním; to přináší lepší, úplnější a užitečnější zpětnou vazbu; výsledný produkt tak spíše bude vyhovovat požadavkům zákazníka
* nedostatečná analýza: soustředění se na konkrétní část prototypu může odvádět pozornost od pohledu na záležitost jako celek; lze tak přehlédnout lepší možnosti řešení, nebo např. přeměny prototypu na špatný návrh výsledku, který je těžké udržovat
* zmatení prototypu a výsledného systému: uživatel se může chybně domnívat, že prototyp, který je určen k zahození, je nevyladěná verze konečného systému
* nepochopení potřeb uživatele ze strany vývojářů: vývojáři mohou předpokládat, že uživatel rozumí jejim cílům a sdílí je
* lpění vývojářů na prototypu: tj. potenciální snaha převést prototyp na funkční model, přestože má například nevhodnou podkladovou architekturu;
* nadměrný čas vývoje prototypu: pokud vývojáři zapomenou na fakt, že je cílem prototypu vyvinout jej rychle, může dojít k výraznému zdržení
* náklady na implementaci prototypingu: počáteční náklady mohou být vysoké; mnoho společností má již funkční workflow a změna na prototypování může znamenat přeškolení, změnu nástrojů apod.

### Agilní model (Inkrementální model)

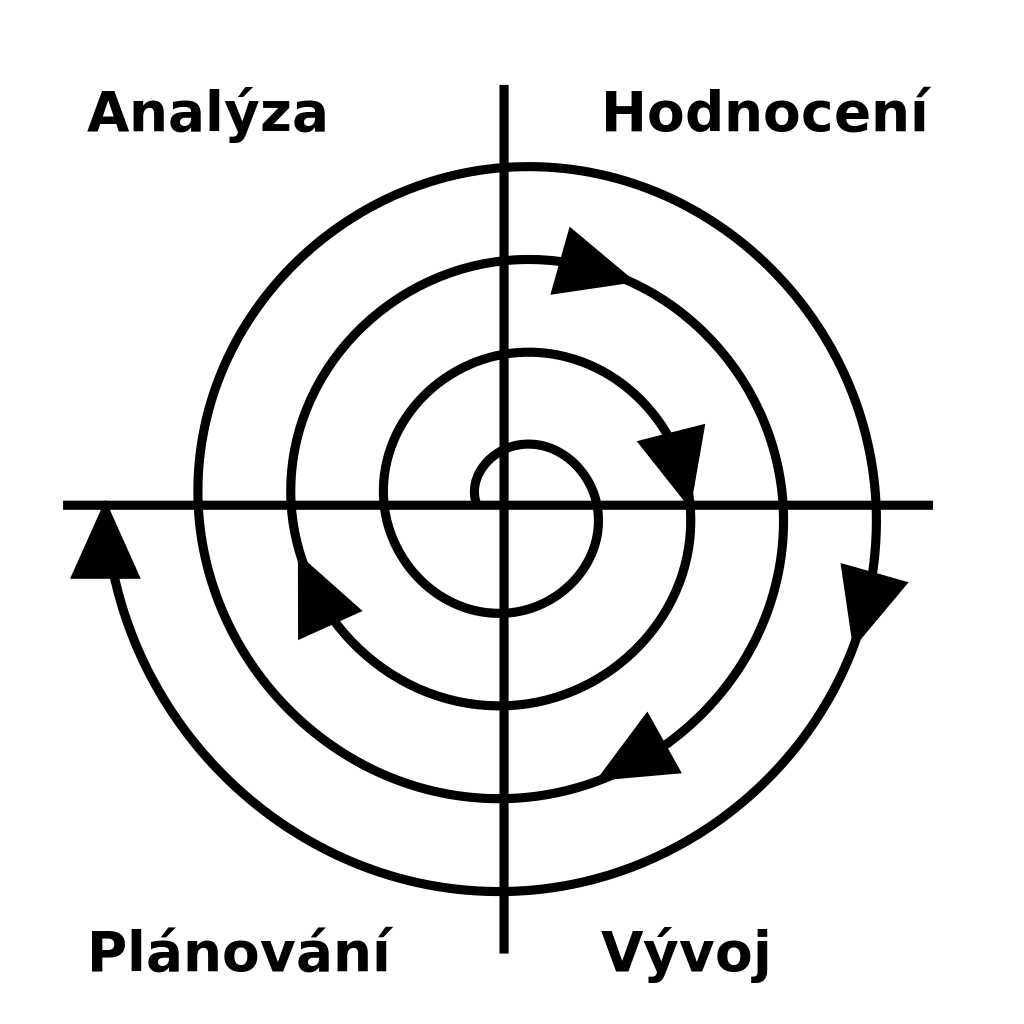
Je schopen se přizpůsobit uživateli. Dělíme vývoj do sprinty nebo jinak označených období. Celý development je rozdělen na více menších částí a sprinty jsou zde pro to abychom mohli průběh vývoje kontrolovat. Je pro nás důležité zůstávat v kontaktu s uživatelem a představovat mu progres abychom věděli že se neoddalujeme od jeho původní idey.

Inkrementální model je vhodný ke kombinaci sekvenčních (např. vodopád) a iteračních (např. prototyping) metodik softwarového vývoje. Zavádí zjednodušení implementace změn přímo během vývoje.

Principy inkrementálního modelu:

1. Provádí se série malých vodopádů, kde každý vodopád je určen jen pro malou část systému a je dokončen před započetím práce na dalším přírůstku.
2. Obecné požadavky definujeme dříve, než přikročíme k evolučnímu vývoji pomocí malých vodopádů.
3. Koncept, analýza, design architektury a systémové jádro se definují pomocí vodopádu, poté přecházíme na prototypování, které vrcholí instalací finálního prototypu jako funkčního produktu.

### Spirálový model



Proces vývoje, který kombinuje designový a prototypový přístup tak, aby postupoval v kombinování výhod od shora dolů (prototypování) a zdola nahoru (designování).

Základní čtyři kvadranty modelu (iterace = jedno projití spirálou):

1. Analýza - stanovení cílů, alternativ a velikosti iterace
2. Vyhodnocení - probrání alternativ, identifikace a řešení rizik
3. Vývoj - vývoj a kontrola produktu, součástí vývoje je i testování
4. Plánování - plán pro příští iterace

Extrémní programování (agilní/iterativní metodika):

* všechni části jsou řešené až extrémně do detajlu
* mezi tyto metodyky spadá například programování ve dvou
  + kde jeden programuje a druhý ho kontroluje a snaží se oběvit chyby které by později mohli být problémové či poradit jakým způsobem danou problematiku řešit
  + mohou se vyměnit, výc očí víc vidí, víc hlav víc ví
* důležitá je komunikace, jednoduchost, zpětná vazba, odvaha
* programujeme vždy to co je potřeba
* jedná se o iterativní model ale "sprinty" jsou mnohem kratší
* snažíme se dělat vše co nejrychleji a do posledního detailu
* to co je pro nás důležité tomu se budeme stále věnovat to, co se pro nás osvědčilo tomu věnujeme nejvíce času

### Další postupy při vývoji softwaru

* Rapid Application Development (RAD)
* Objektově orientované metodické přístupy
* Metodou shora-dolů (dekompozice)
* Unified Process (založeno na UML)
* Agilní metodika (iterativní vývoj prostřednictvím spolupráce mezi týmy)
* Integrovaný vývoj softwaru