# 17. Vývojové metodiky

Cílem těchto metod je zlepšení kvality softwaru a zefektivnění jeho vývoje. Základní myšlenkou je rozdělení do menších částí pro lepší kontrolu a snazší řízení. Odlišnosti v různých metodikách bývá hlavně ve způsobu dělení na části.

## Vodopádový model (Waterfall)

Základní rozdělení vývoje:

1. **Požadavky** – Na úplném začátku je potřeba ustálit zadání (počítá se s tím, že zadavatel přesně neví, jak vše probíhá a co přesně chce). Proto je třeba srozumitelně a přesně stanovit zadání (nejlépe psaná forma podepsaná oběma stranami). **Výstupní dokumentem často bývá SRS (Software Requirement Specification)**, kde bývá návrh systému, jeho údržba, popis vývoje (různé etapy) a celkově všechny podstatné věci pro vytvoření programu.
2. **Návrh** – Související požadavky se rozdělí do skupin. Popíše se jejich interakce a specifika. Vznikne návrh uložení dat a práce s nimi, požadované systémy (na kterých se bude pracovat a na které směřujeme), základní uživatelské rozhraní (wireframes) a vše se patřičně zdokumentuje. K tomu slouží **výstupní dokument ve formě SDD (Software Design Document)**.
3. **Implementace** – Návrh se zpracuje pro programátory do zvládnutelných bloků. Vzniknou základní objekty (databáze, funkce, formuláře), které se převedou do kódu. Vytvoří se tak samostatně fungující bloky, které se testují (na úrovni unit testů). **Výstupy jsou programátorská dokumentace, kód, provozní dokumentace, hardwarová infrastruktura, organizace provozu a podpory**
4. **Verifikace a validace** – Verifikace: kontrola jestli systém vyhovuje specifikacím (vytváříme systém správně?). Validace: kontrola jestli systém splňuje požadavky zadavatele (vytváříme správný systém?). Odhalují se chyby, upřesňují se požadavky, snaha o co nejpřesnější zpracování zadavatelových požadavků. Zlomový okamžik vývoje. Vše se spojuje, rozpočet roste, manažeři musí vše udržet jednotné, rozsah projektu se mění podle dostupných zdrojů. **Výstupem jsou povedené testy, validační dokumentace, akceptace systému – uvedení do provozu**
5. **Údržba** – Systém je nabídnut uživatelům. **Systém podstupuje dvěma změnám** Perfektivní: přizpůsobení uživatelům a vyhláškám. Korektivní: odstraňují se chyby a provozní incidenty.

## Rychlé prototypování

Větší důraz na uživatele a na rychlost vývoje. Zde je vývoj rozdělen do 4 fází:

1. **Plánování požadavků** – Dohoda o funkčních požadavcích, rozsahu, požadavcích na SW. **Výstupem je chválení požadavků**
2. **Uživatelský návrh** – Spolupráce mezi analytikem a uživateli. Vytvoření prototypů, prohloubení pochopení problematiky a složitosti řešení. Jde o rychlejší vývoj přímo na míru. **Výsledkem je schválení pracovního modelu.**
3. **Konstrukční fáze** – Spolupráce mezi programátory a uživateli. Další zadávání nových požadavků a testování SW
4. **Zakončení** – Integrace všech modulů, příprava na migraci na nové systémy, školí se uživatelé.

Při tomto rychlém vývoji vznikají chyby jako odvedení pozornosti od spolehlivosti a bezpečnosti (v budoucnu), programátoři tráví mnoho času plněním zbytečných úkolů a bez nějaké velké konceptualizace a jednotných základů lze těžko přidávat a obměňovat některé prvky.

## Spirální model

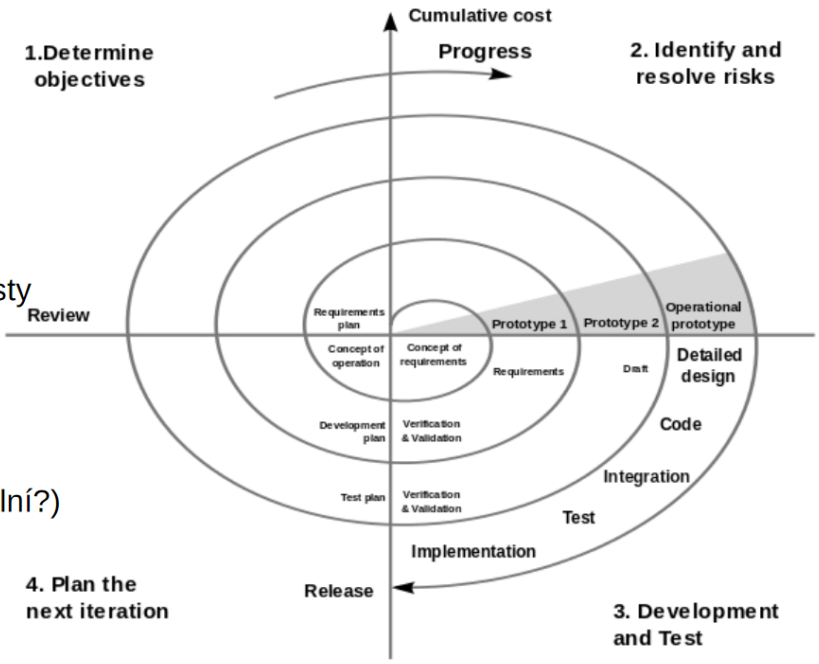
Snaha o zlepšení waterfall modelu. Pojmenování některých nereálných předpokladů a snaho o nápravu. Jednoduše řečeno se každý přírůstek kontroluje, jestli je správnou cestou řešení a jestli splňuje zadání. Hlavní myšlenkou je tedy vytváření malých přírůstků a vždy zvažovat:

1. zda je možné najít řešení, kdy budou všechny strany uspokojené
2. zda neexistuje lepší varianta (levnější, rychlejší), která vše splní
3. rizika vyplývající z toho přírůstku

4. jestli vše bylo splněno a můžeme pokračovat s dalším přírůstkem

## Spirální model

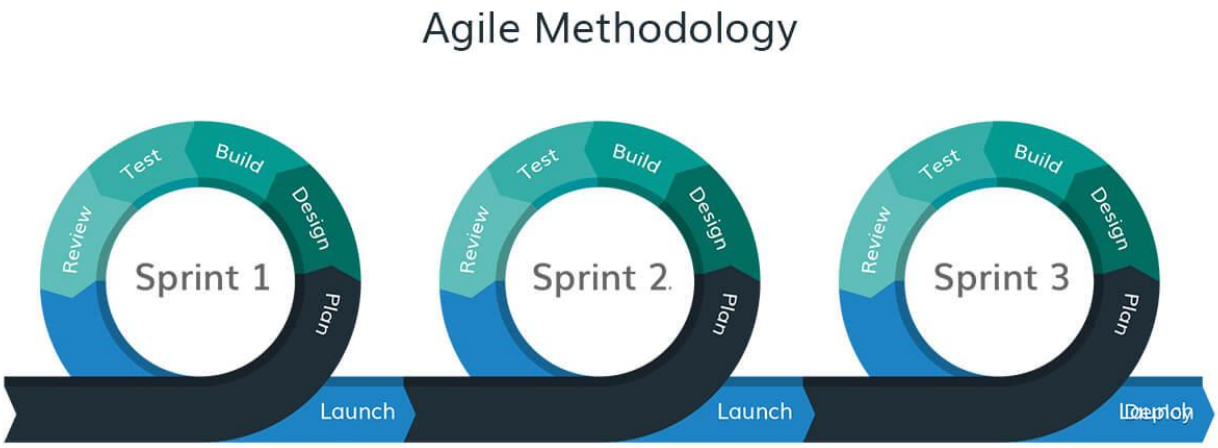
- sekvence vodopádů
- důraz na revize
- postupné prototypy
- opakované a včasné testy
- přeplánování v každém cyklu
- revize požadavků
- kontrola cílů (jsou aktuální?)




Ale i s tímto modelem přichází problémy. Není určen přesný rozsah programu a práce (nelze odhadnout cenu a čas). Spousta opakování a ztráta času. Pouze velké projekty (malé a střední by byly zdlouhavé a drahé). Základní chyby v požadavcích se najdou až později.

## Agilní přístup

Chtěná změna kvůli dlouhodobé nespokojenosti s rychlostí a spolehlivostí dodávek. Pokus o změnu priorit při vytváření SW. Agilní metoda vychází z častých iterací *sprintů* (malých waterfallů).




Základních 12 principů:

1. změna je po celou dobu vývoje vítaná
2. častá dodávka (řádově týdny, normálně totiž byly měsíce, rok)
3. častá komunikace mezi všemi týmy
4. důvěra v lidi a jejich motivace
5. komunikace v tváří v tvář (nejlepší možná)
6. zákaznická spokojenost rychlou dodávkou

7. zásadní v pokroku je funkční systém
8. konstantní tempo, stejná kvalita vývoje
9. stálý dohled na technickou kvalitu
10. jednoduchost (minimalizace zbytečné práce)
11. všichni musí hrát za jeden tým (lepší spolupráce a lepší nápady)
12. snaha vždy o zlepšení pro prospěch celku

Po čase ale i u tohoto modelu přišli problémy. Většinou zprofanovaná. Již moc nesouvisí s původními myšlenkami. Špatná plánovatelnost (malá vize do budoucna). Špatná dokumentace během vývoje (špatná udržitelnost).

## Shrnutí

Důležitá je týmový práce, připravenost, spolupráce s zadavatelem a uživateli, adekvátní volba řešení, metoda vývoje negarantuje kvalitu, dokumentace je potřeba a mizerná **kvalita nelze omluvit použitým modelem vývoje**.