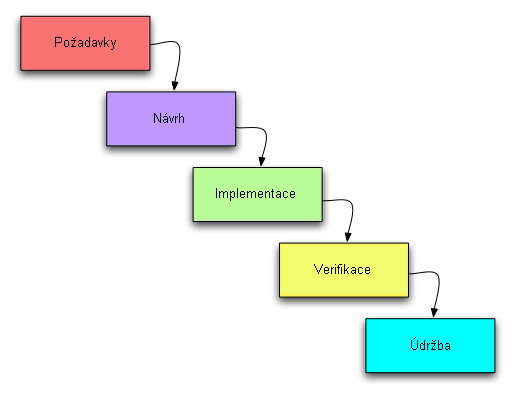
**Metodiky a životní cyklus vývoje softwaru**

**Metodiky**

* postup, pravidla, nástroje (framework) užitý týmem
  + Pro návrh, plánování, řízení vývoje SW
* **Vodopádový (waterfall)**
  + **Hodně zastaralý**
  + **Špatný**
  + **Kdy použít?**
    - Požadavky se často nemění a jsou jasné
    - Krátký nekomplikovaný projekt
    - Zajištěný stabilní prostředí
  + **Výhody**
    - Nevracet se zpět po ukončení fáze
    - Přísná kontrola na konci každé fázi
  + **Nevýhody**
    - Chyby opraveny jen v konkrétní fázi
    - Testing pozdě
    - Nezahrnuje feedback klienta
    - Malé chyby mohou způsobit problémy v hotovém SW
    - Hotový projekt musí být **NAPROSTO DOKONALÝ**
  + Projekt rozdělen na fáze jdoucí postupně za sebou
  + Realizace celého systému najednou
    - Požadavky
    - Návrh
    - Implementace/Vývoj
    - Testování
    - Integrace
    - Údržba
  + Používá se například u NASA
    - Vláda přesně ví, co chce
* **Prototypový**
  + **Kdy použít?**
    - Nejasné požadavky
  + **Výhody**
    - Feedback uživatelů
    - Naleznutí chybějící funkcionality
  + **Nevýhody**
    - Pomalý
    - Klient se někdy nechce zapojovat do feedbacku
  + **Není to kompletní metodika vývoje**
    - Doplnění ve větších metodikách
    - Vývoj neúplných verzí
  + **Rozdělení projektu na menší části**
    - Snížení projektových rizik
    - Zjednodušení možností změn v průběhu
  + Fáze
    - Požadavky
    - Rychlý návrhový design
    - Prototyp SW z požadavků
    - Prezentování klientovi – LOOP
      * Feedback
    - Předělání podle feedbacku – LOOP
      * Po splnění, kompletní dodělání podle prototypu
    - Implementace/integrování do produkce/projektu

Diagram

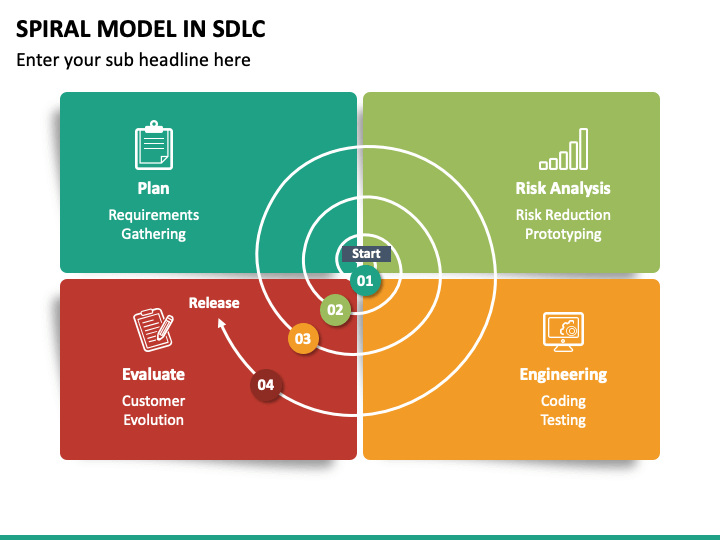
Description automatically generated

* **Inkrementální (přírůstkový) / Iterativní**
  + Požadavky rozděleny do několika oddělených modulů SW Dev cyklu
    - Požadavky
    - Návrh
    - Implementace/Vývoj
    - Testování
  + Každá iterace prochází všema fázemi až po testování
  + Nový release systému přidává funkcionalitu předešlému releasu
  + **Kdy použít?**
    - Požadavky jsou pochopitelné
    - Potřeba early release
    - Funkcionalita s vysokým rizikem
  + **Výhody**
    - Méně stojí
    - Klient může reagovat na každý doplnění
    - Chyby lehce naleznout
  + **Nevýhody**
    - Potřeba dobrý plán/design
    - Problémy v architektuře kvůli neprocházení celýho životníh cyklu

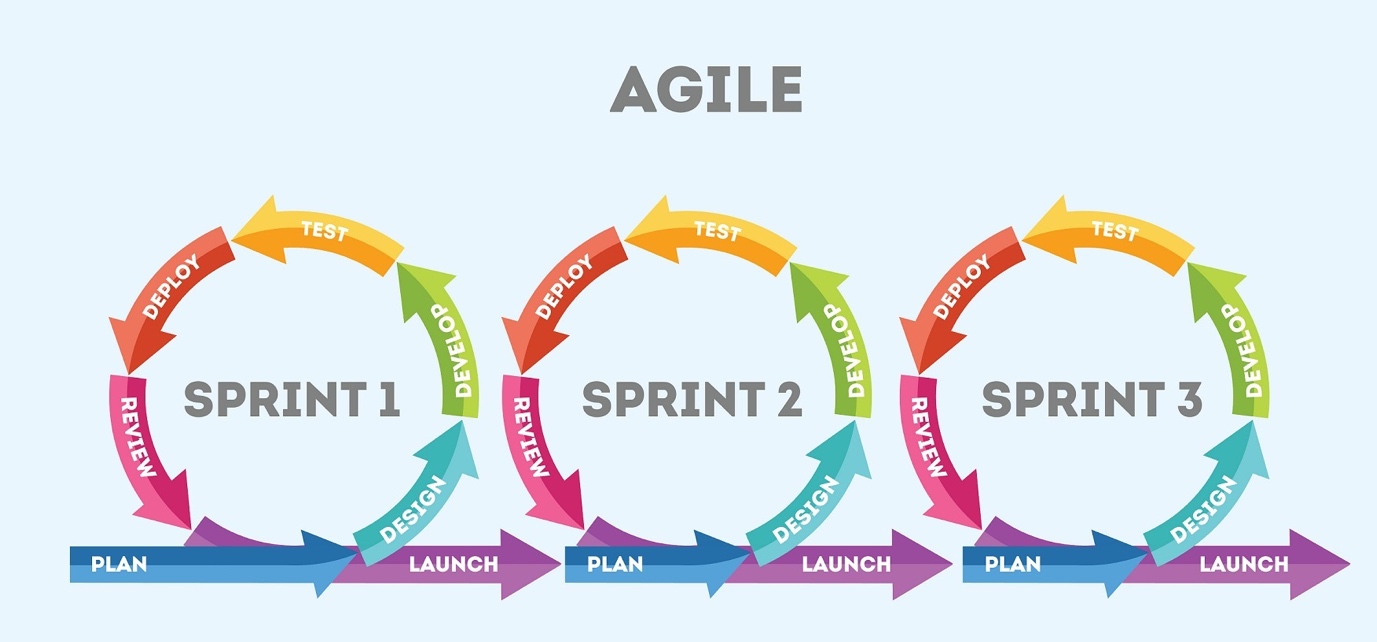
Graphical user interface, application

Description automatically generated

* **Spirálový**
  + **Kdy použít?**
    - Velký projekt
    - Potřeba častých releasů
    - Není zakázané tvoření prototypů
    - Středně-vysoce rizikové projekty
    - Požadavky nejasné a složité
    - Požadovány změny kdykoliv
  + **Výhody**
    - Povolené změny i v pozdější stadii vývoje
    - Zamezení rizik díky opakování
    - Rychlý vývoj
    - Feedback klientů
  + **Nevýhody**
    - Možnost nevejít se do budgetu
    - Víc dokumentace
    - Pouze velký projekty (nevýhodný pro malý)
  + **Zaměřen na** 
    - Analýzu rizik
    - Minimalizace projektových rizik
  + Každý cyklus (iterace) spirály spouští stejný kroky
  + Fáze
    - **Plánování**
      * Cíle/požadavky
      * Budget
      * Definování risků
    - **Analýza risků**
      * Analýza
      * Zamezení risků
    - **Vývoj**
      * Design a vývoj SW
      * Testing SW
      * Cílem je vývoj prototypu
    - **Vyhodnocení**
      * Zda SW splňuje požadavky
      * Feedback od uživatelů
      * Změny v SW
      * Pokud nesplňuje – Nový cyklus



* **AGILE**
  + Skládá se ze SPRINTŮ
    - Jeden sprint se časově pohybuje kolem 1-4 týdnů
    - Projekt rozdělen do menších lehčích kroků
    - Každý sprint může být přednesen KLIENTOVI jako progres a posun projektu
    - Je brát v potaz feedback KLIENTA a jsou na jeho základě většinou prováděny změny v PLÁNU/DESIGNU
  + Je prioritou KLIENT, hotový produkt a týmová práce
    - Obsahuje v sobě stálou komunikaci s KLIENTEM
  + Do komunikace jsou zapojeny i VÝVOJÁŘI
  + Týmy v AGILE mezi sebou neustále komunikují
    - Efektivnější
    - Lepší pochopení
    - Rychlejší zpětná vazba
  + Agile se počítá jako FLEXIBILNÍ, RESPONZIVNÍ A ADAPTATIVNÍ
  + Produktem je často VYSOKÁ KVALITA
  + Jednotlivé fáze nemusí jít chronologicky
    - Často se prolínají navzájem (komunikace)



* + **Extrémní programování (XP) (počítá se jako AGILE)**
    - **Extremizuje postupy**
    - Neustálá revize kódu
      * Jeden ze způsobu je např. **Programování v páru**
        + U jednoho počítače sedí dva programátoři
        + Jeden píše kód
        + Druhý kód kontroluje
        + Pravidelné střídání rolí
        + Toto řešení může zabrat více času
        + Výsledný kód bude mít ale výrazně méně defektů a bude více efektivní
    - **Testování v XP**
      * Testuje se pořád a neustále
      * Nejvyšší užití Unit testů
        + Otestování každé možnosti, na který by program mohl selhat
      * Testování PŘED, PŘI a PO změně kódu
    - **Kdy použít?**
      * Programování opravdu nezbytné části
      * Rychlá potřeba
        + Malá timeline
      * Komunikace se zákazníkem
      * Malý až střední tým (2-12 lidí)
      * Připravený ke změnám
    - **Výhody**
      * Větší komunikace a spolupráce
      * Rychlý feedback
      * Flexibilita a přizpůsobování k požadavkům
      * Malý risky
        + Vysoký počet releasu
        + Vysoký počet testování
      * Vysoká produktivita
      * Větší kvalita konečného produktu
    - **Nevýhody**
      * Potřeba zkušených programátorů
      * Je potřeba mít členy týmu fyzicky vedle sebe
      * Zabírá více času v případě PROGRAMOVÁNÍ V PÁRU
    - **Zaměřen na**
      * Vysokou kvalitu
      * Splnění očekávání zákazníka
      * Vylepšení komunikace mezi týmem
      * Jednoduchost a správnost

A diagram of a process

Description automatically generated with low confidence

V-model

A picture containing text, screenshot, font, line

Description automatically generated