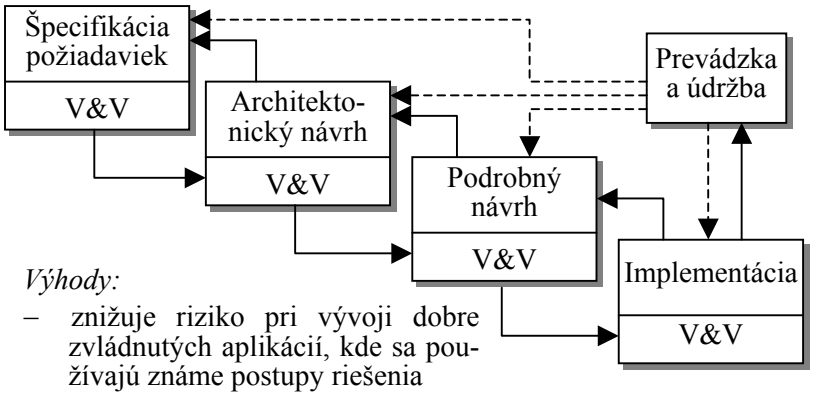
Modely životného cyklu softvéru

- veľké množstvo modelov
- najrozšírenejšie
  - vodopádový
  - inkrementálny
  - evolučný
- pre všetky podsystémy systému sa nemusí použiť ten istý model, napr. ak ide o podsystém so slabou špecifikáciou, použije sa prototypovanie; iný systém môže mať dobrú špecifikáciu, použije sa vodopádový model

Vodopádový model

Problémy:

- reálne projekty nedodržujú jednotlivé kroky v predpísanom poradí
- používateľ nedokáže v počiatočných etapách formulovať úplné požiadavky na systém
- zákazník uvidí systém až na konci projektu; neskoré odhalenie nedostatkov môže vážne ohroziť úspech projektu
- malá prispôsobivosť zmenám po rozbehnutí procesu
- zvyšuje riziko pri vývoji nových druhov aplikácií

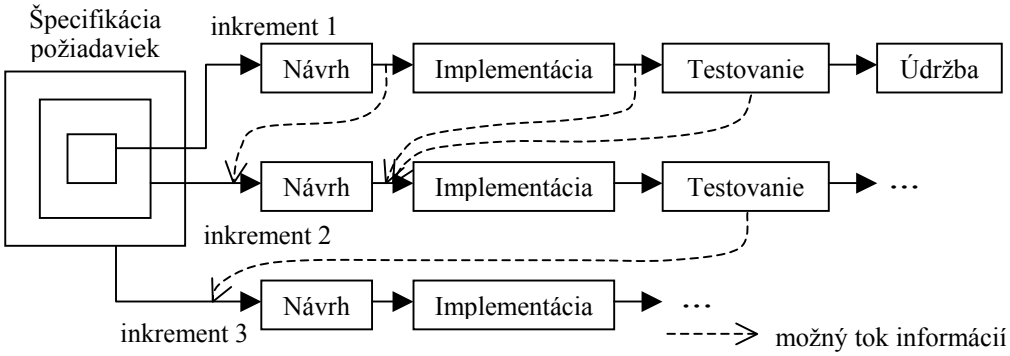


Výhody:

- znižuje riziko pri vývoji dobre zvládnutých aplikácií, kde sa používajú známe postupy riešenia
- dobrá viditeľnosť procesu vývoja, každá činnosť končí výstupom

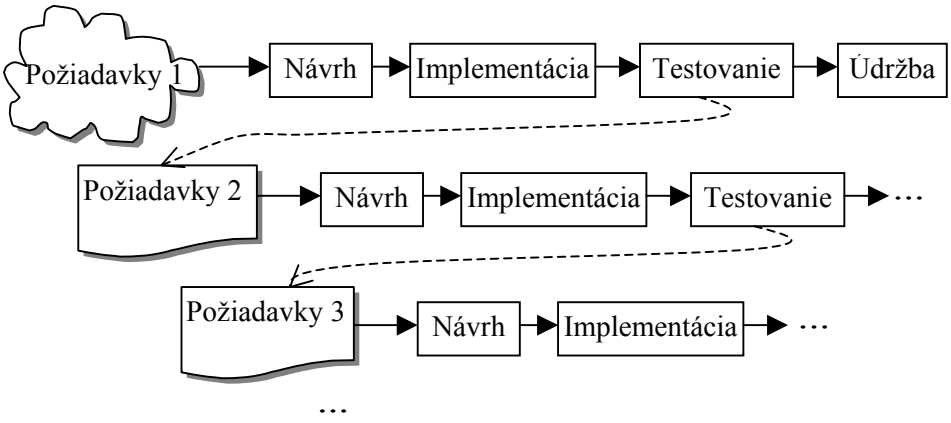
Inkrementálny model

- označuje sa aj prírastkový
- najskôr sa definujú všetky požiadavky
- systém sa vytvára a odovzdáva používateľovi po častiach
- proces je dobre viditeľný, pre každý inkrement treba vytvoriť plán a príslušnú dokumentáciu
- problémom môže byť návrh architektúry v prípade, ak je špecifikácia požiadaviek neúplná
- určitý netechnický problém môže spôsobiť aj uzatváranie zmluvy so zákazníkom



Evolučný model

- požiadavky sa všetky nedefinujú na začiatku (náčrt špecifikácie)
- súbežne prebiehajúce činnosti: návrh, implementácia, testovanie
- výstupy:
  - začiatočná verzia
  - priebežné verzie
  - záverečná verzia.



Hodí sa

- ak nevieme dopredu vyjadriť špecifikáciu (systémy umelej inteligencie, znalostné systémy)
- pre systémy s „krátkym“ životom

Výhoda: znižuje riziko pre nové aplikácie, pretože špecifikácia a implementácia sú v súlade

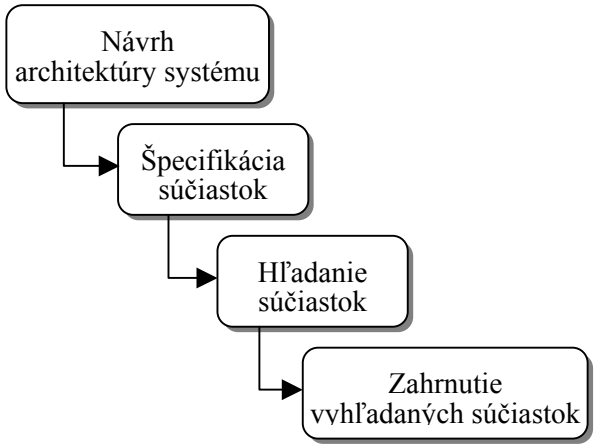
Nevýhody:

- proces nie je viditeľný
- systémy sú často slabo štruktúrované
- vyvíjanie prototypov si vyžaduje zvláštne schopnosti (napr. jazyky).

Ďalšie modely životného cyklu softvéru

Formálna transformácia

- formálna matematická špecifikácia systému a jej transformácia do programu
- zatiaľ úspešný postup iba pre malé a jednoduché systémy
- dobrá viditeľnosť procesu vývoja, priamo sa vyžaduje vytvorenie dokumentov v každej etape

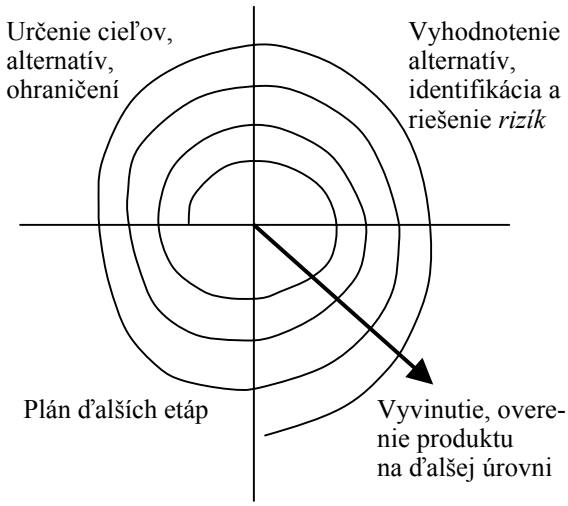


Zostavovanie systému zo znovupoužiteľných súčiastok

- vývoj založený na znovupoužití softvérových súčiastok
- časti systému (softvérové súčiastky) už existujú
- vývoj systému = vyhľadanie a integrácia existujúcich súčiastok
- znovupoužitie môže ovplyvniť aj návrh, vtedy sa súčiastky hľadajú hneď po náčrte požiadaviek, na základe vyhľadaných súčiastok sa prípadne modifikujú požiadavky a navrhne architektúra systému
- znižujú sa celkové náklady na vývoj softvéru, riziko neúspechu projektu a aj potrebný čas
- zvyšuje sa spoľahlivosť systému
- možnosti zakomponovania štandardov do súčiastok

Špirálový model (Boehm, 1988)

- každé kolo špirály = etapa procesu tvorby softvéru
- dobrá viditeľnosť procesu vývoja, každý segment špirály v každom kole má vyústiť do dokumentu
- riziko: niečo čo môže spôsobovať problémy
- vyhodnotenie alternatív → prototyp, simulácia, modelovanie



– riešenie rizík, t.j. rozhodnutie o spôsobe zníženia rizík: prototyp, simulovanie, modely, benchmark

Výhody špirálového modelu:

– upriamuje pozornosť na možnosti znovupoužitia
– upriamuje pozornosť na skoré odstraňovanie chýb
– zvýrazňuje požiadavky na akosť
– integruje vývoj a údržbu
– pokrýva aj súčasný vývoj softvéru/hardvéru

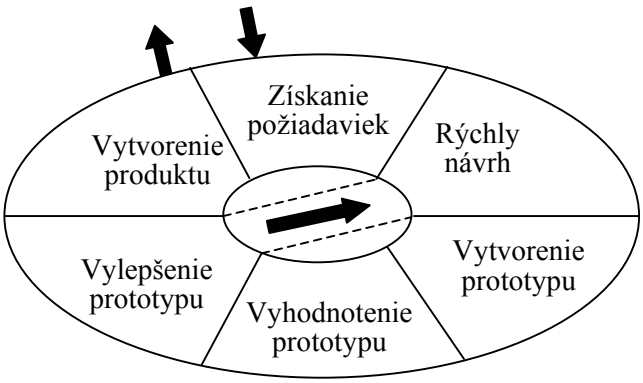
Nevýhody špirálového modelu:

– zmluva o vývoji softvérového systému často určuje vopred procesný model a výstupy
– zhodnocovanie rizík treba vedieť robiť, je všeobecný a treba ho pre daný projekt rozpracovať

Prototypovanie softvéru

PROTOTYP → „prvý z určitého druhu“

SOFTVÉROVÝ PROTOTYP → čiastočná implementácia systému, ktorej cieľom je dozvedieť sa niečo o riešenom probléme alebo o možnom riešení problému; zvyčajne sa orientuje na špecifikáciu požiadaviek (na rozdiel od prototypovania hardvéru, ktoré sa používa najmä na validáciu návrhu)



Prečo vytvárame softvérový prototyp?

– zníženie rizika (rozpočet, rozvrh, splnenie požiadaviek na výrobok)

Kedy vytvárame softvérový prototyp?

– Nemáte ani potuchy čo vlastne používateľ chce.
– Používateľ si myslí, že vie čo chce, ale vy *sa* (alebo *ich*) chcete presvedčiť, že viete tiež.
– Všetci sa zhodujú na funkcionalite, ale nikto v skutočnosti nevie ako by to malo „vyzerať“.
– Základnej funkcionalite rozumiete, ale máte podozrenie, že veľa ďalších požiadaviek je ešte neodhalených.

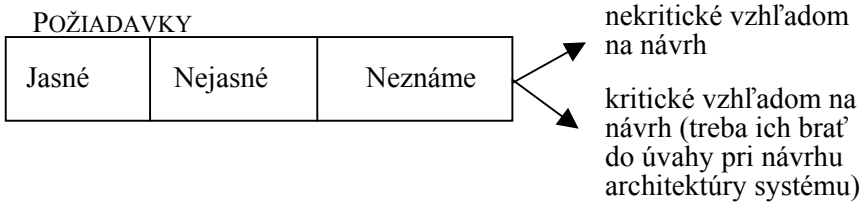
Prototypovanie sa používa najmä pri evolučnom modeli životného cyklu softvéru.

Klasifikácia prototypov

Prototypovanie na zahodenie – prieskumné prototypovanie

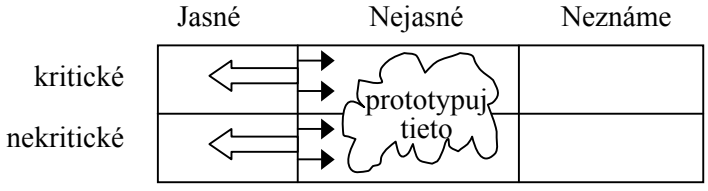
angl. throw-away – evolutionary Prototypovanie

– pomáha redukovať požiadavky *nejasné-kritické*
– pomáha znižovať pravdepodobnosť, že nejaké požiadavky ostali v oblasti *neznáme-kritické*



PROTOTYP NA ZAHODENIE →

– cieľom je dospieť k porozumeniu požiadaviek na systém
– prototypujú sa nejasné požiadavky, ktoré sú kritické vzhľadom na návrh (ak to vieme odlíšiť)
– zvyčajne horizontálny vzhľadom na implementované funkcie



PRIESKUMNÝ PROTOTYP →

– cieľom je vyvíjať systém v spolupráci so zákazníkom
– prototypujú sa jasné, dobre pochopené požiadavky