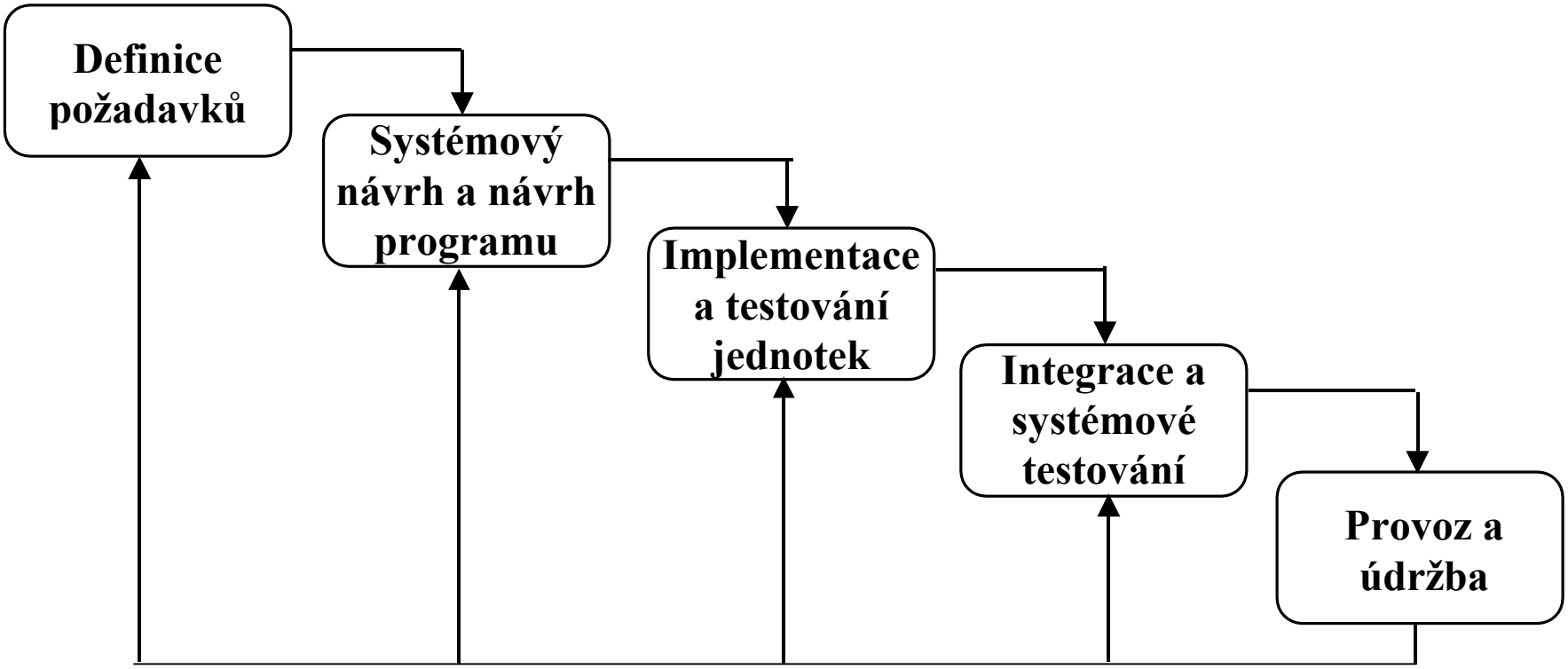
2 Životní cyklus programového díla

- **Typické etapy:**
  1. **Specifikace požadavků**
     - **specifikace problému**
     - **analýza požadavků**
  2. **Vývoj programu**
     - **návrh**
     - **kódování (programování)**
  3. **Verifikace a validace**
  4. **Provoz a údržba + další činnosti (posuzování, dokumentace, ...)**
- ***Model životního cyklu* - ukazuje etapy (především specifikace a vývoje) a jejich návaznosti**
  - ♦ **základní**
    - ◊ **klasický (vodopád)**
    - ◊ **evoluční vývoj**
      - **specifikační prototypování**
      - **evoluční prototypování**
      - **inkrementální vývoj**
    - ◊ **formální transformace**
    - ◊ **sestavování z komponent**
  - ♦ **kombinované**

2.1 Klasický životní cyklus („vodopád“)



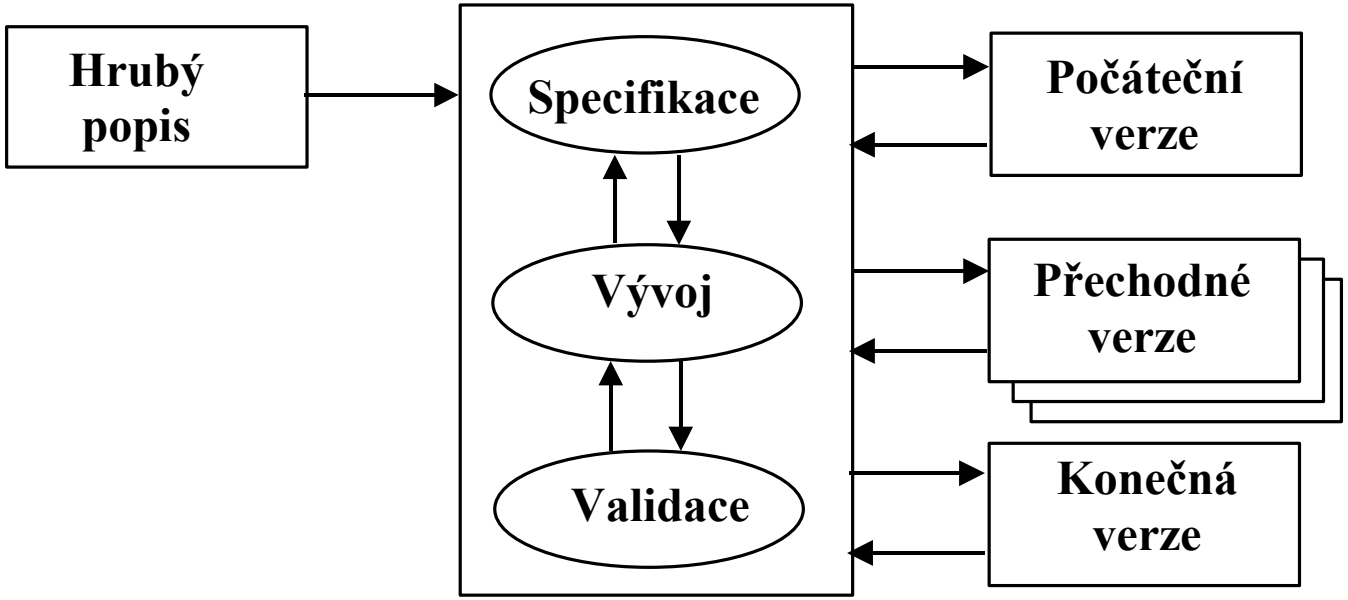
Výhody: - pokrok projektu je viditelný (výhodné pro manažery).

Nevýhody: - etapy se překrývají (problémy s požadavky odhaleny při návrhu) → iterace zhoršující podmínky řízení projektu → zpravidla odložení dalších změn → nesplnění požadavků nebo zhoršení strukturovanosti programu.

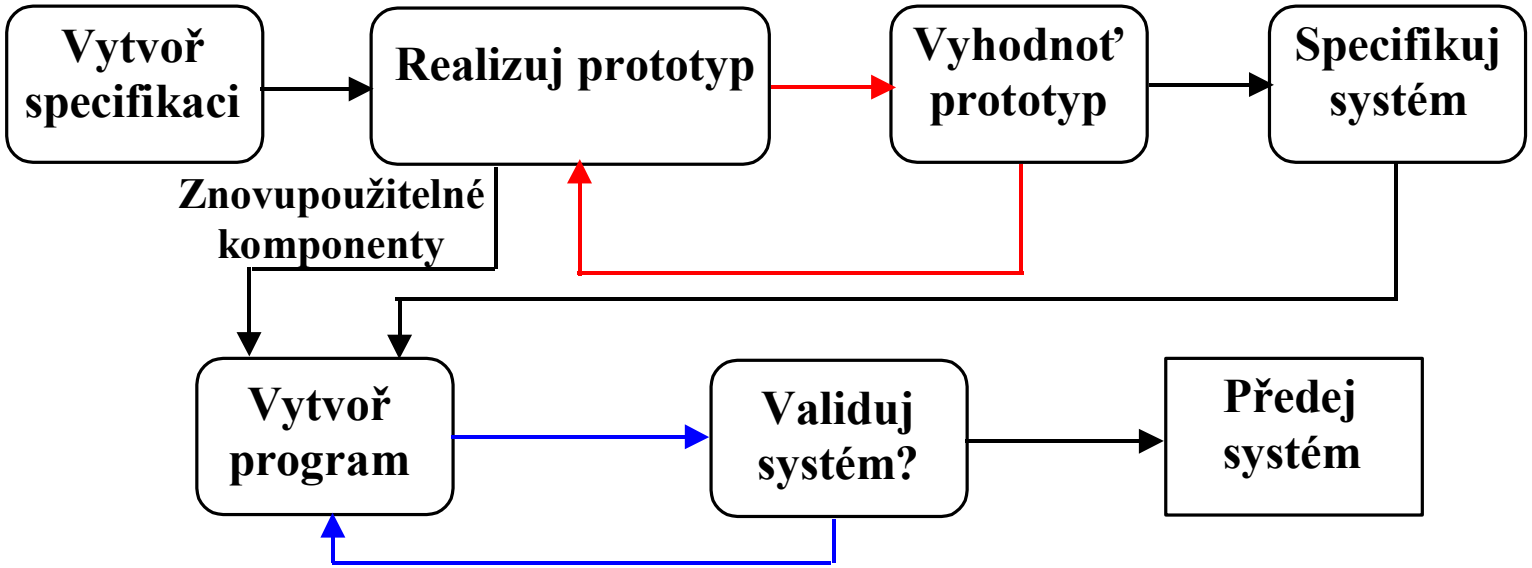
Použití: jasnost požadavků na začátku řešení.

2.2 Evoluční vývoj

- vhodný při nejasnostech v požadavcích, resp. nutnosti vyvíjet systém postupně

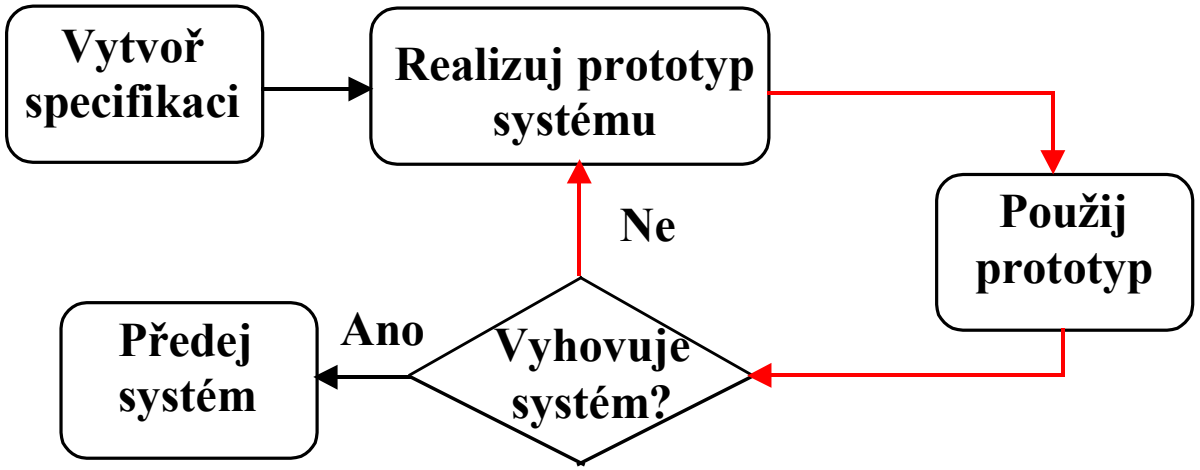


- **Specifikační prototypování (throw-away)**



Výhody: - postupné zpřesnění požadavků
Nevýhody: - některé funkce může být obtížné prototypovat
- nefunkční požadavky (spolehlivost, robustnost) nelze testovat na prototypu
- nebezpečí dopracování prototypu

- **Evoluční prototypování (exploratory programming)**

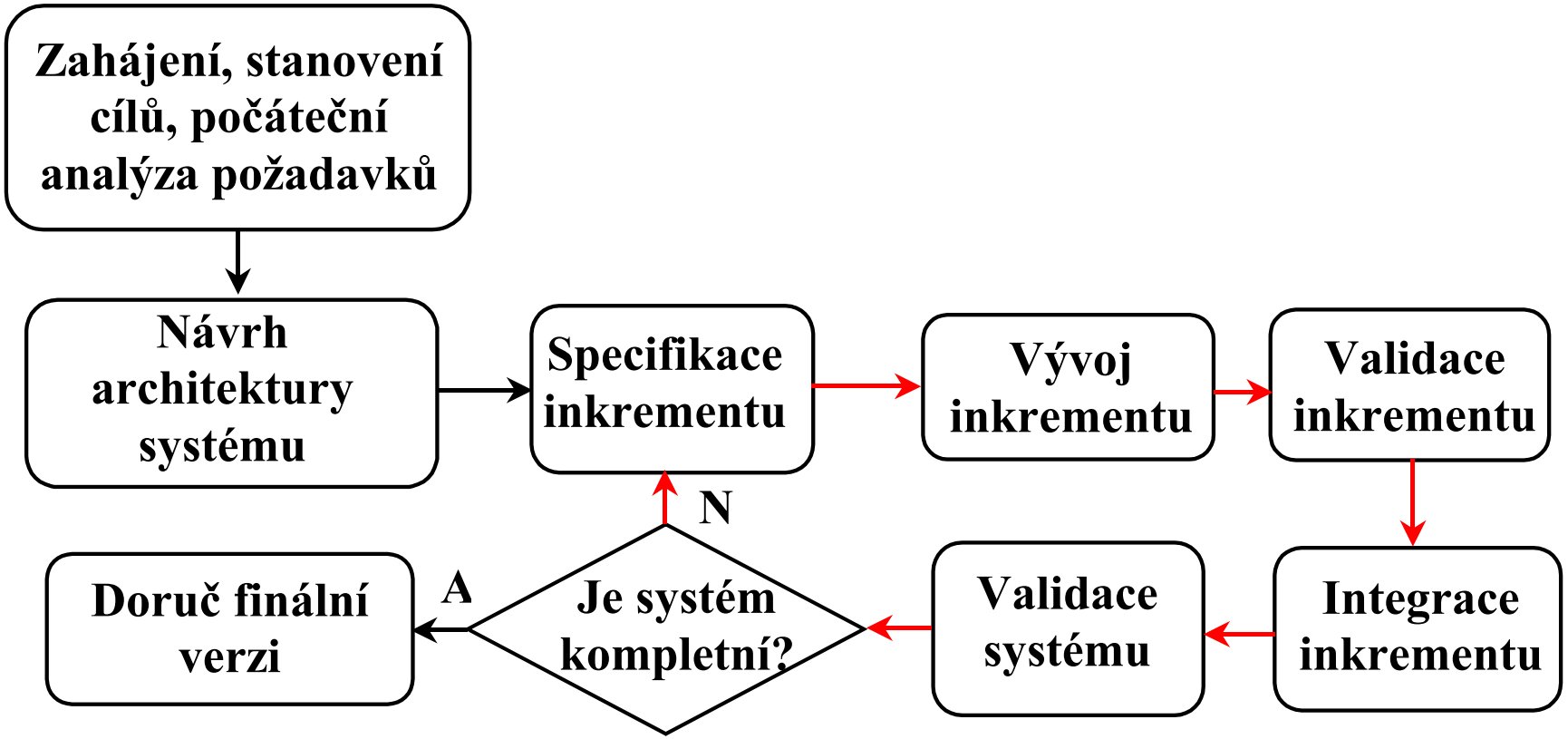


Výhody: - není nutná přesná specifikace požadavků

Nevýhody:
- jen validace na základě posouzení zákazníka
- produkt bývá špatně dokumentovaný
- produkt bývá špatně strukturovaný → obtížná údržba
- není jasná požadovaná specializace vývojářů → zpravidla malé týmy univerzálních, zkušených

- **Techniky prototypování (RAD - rapid application development)**
  - generátory aplikací a jazyky čtvrté generace,
  - skládání z opakovaně použitelných komponent,
  - jazyky pro rychlé prototypování (Smalltalk, CASE nástroje)

- **Inkrementální vývoj**



- odstraňuje nevýhody evolučního prototypování (neustálé změny, problémy s řízením projektu) a zachovává výhodu zpětné vazby zákazníka

Př) Rational Unified Process (viz později)

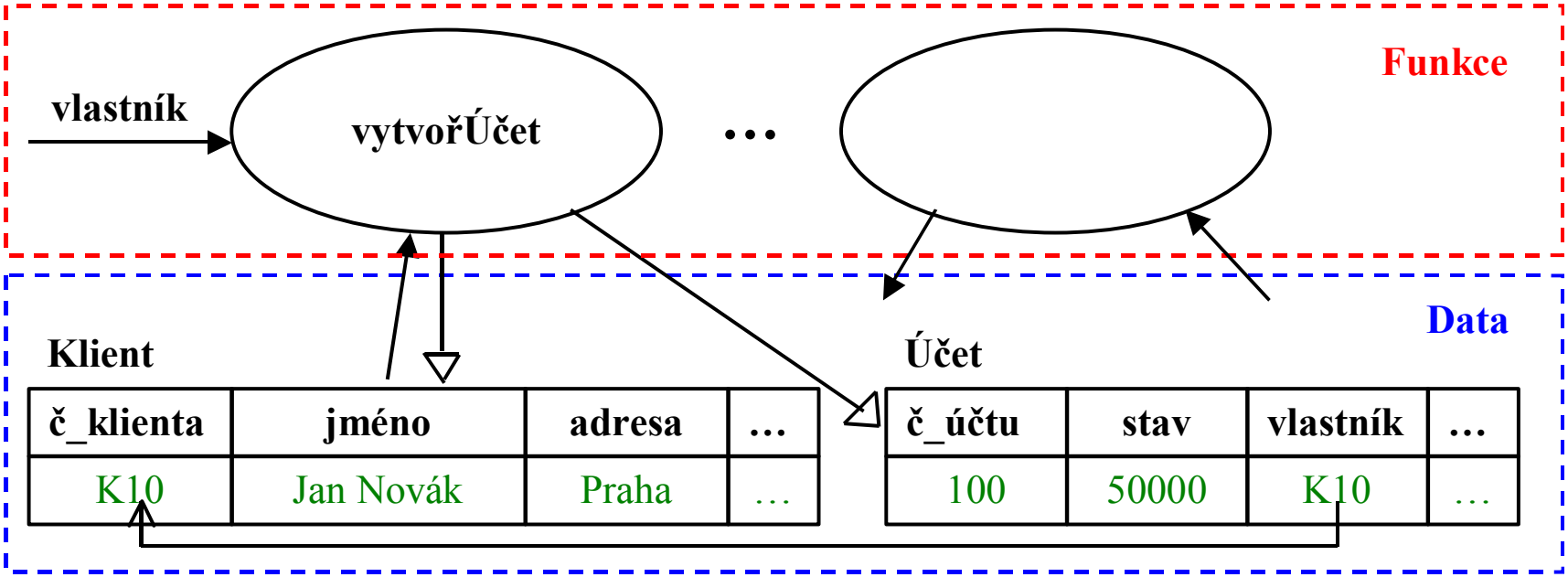
2.3 Přístup k vývoji programových systémů

2 základní přístupy:

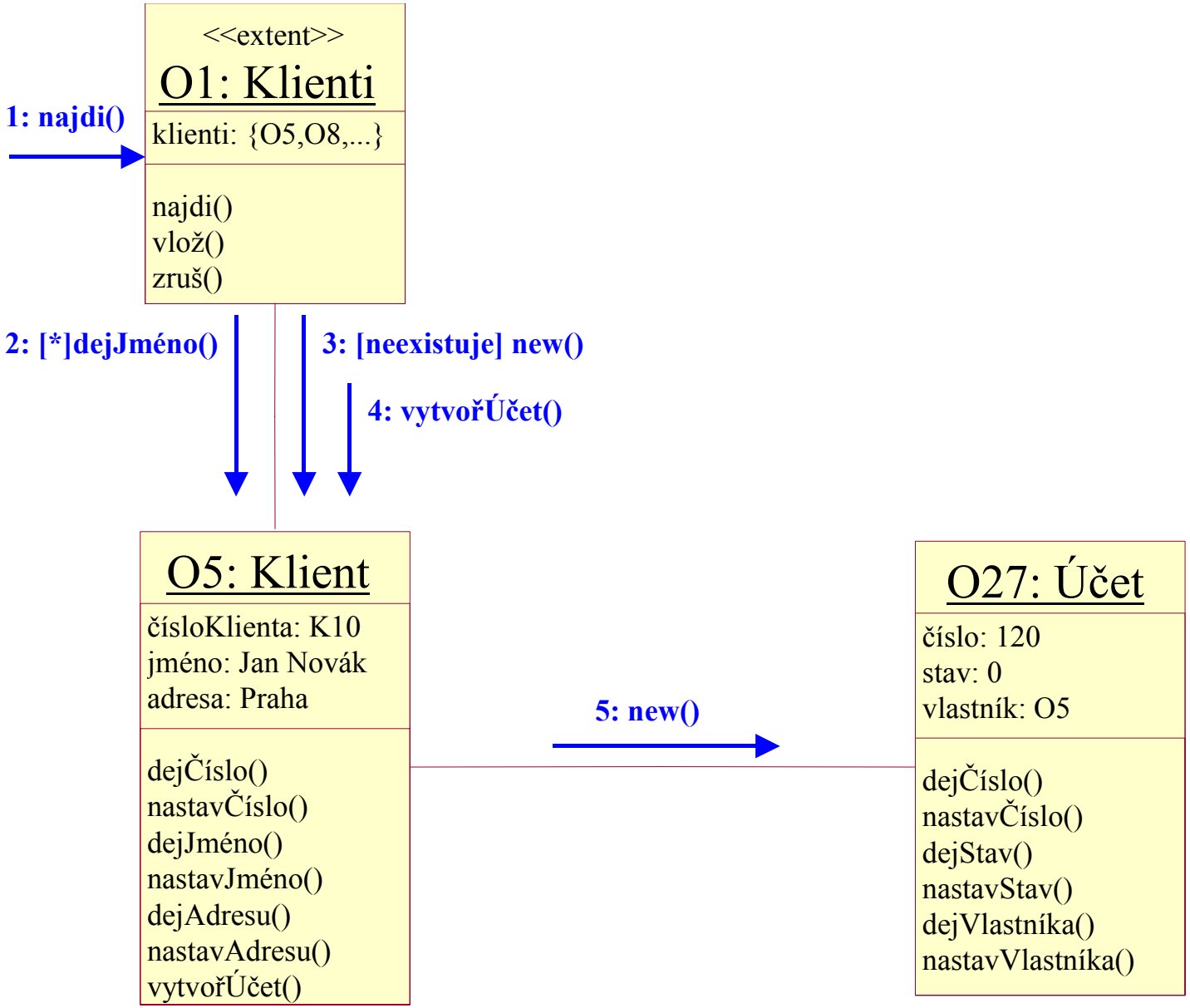
a) **klasický (strukturovaný) - analýza a návrh prováděny na základě funkční dekompozice a analýzy a návrhu datových struktur, datové struktury a funkce odděleny.**

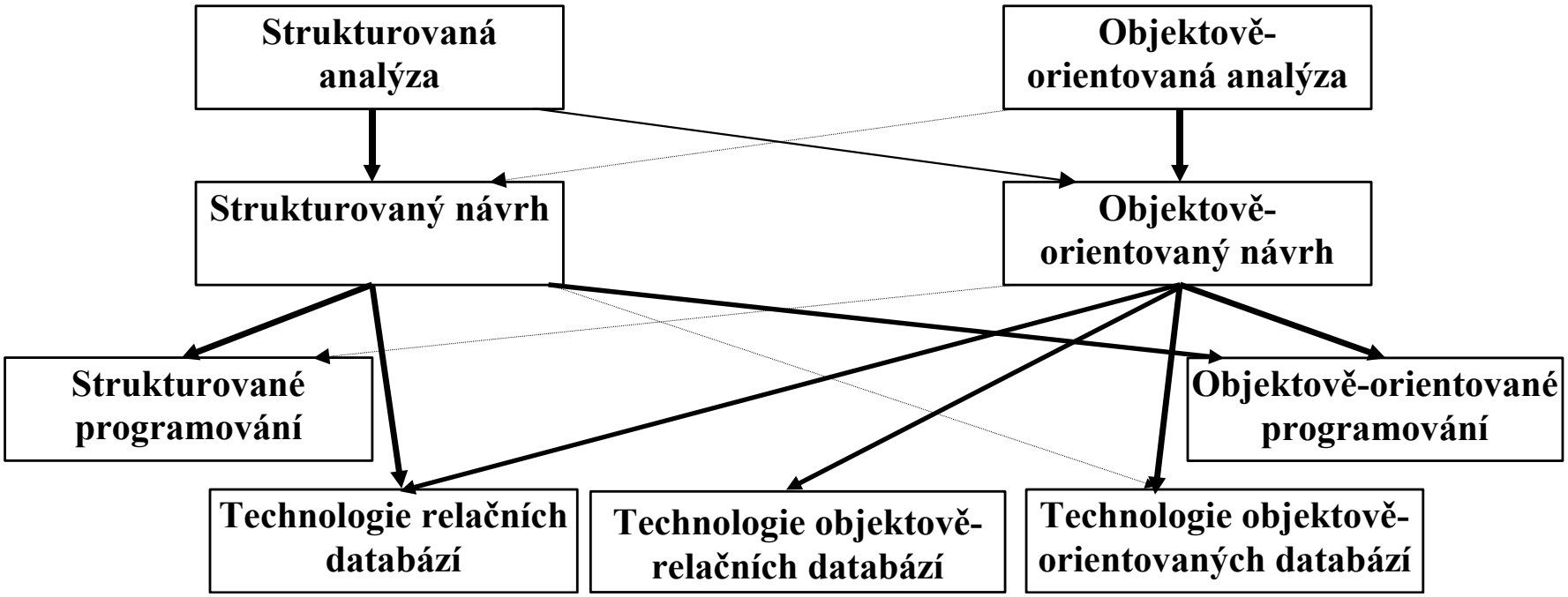
b) **objektově-orientovaný - analýza a návrh prováděny na základě identifikace objektů a jejich vzájemných vztahů, datové struktury a funkce pohromadě**

- **Podstata klasického přístupu**



- **Podstata objektově orientovaného přístupu**




Strukturovaná analýza
Objektově-orientovaná analýza
Strukturovaný návrh
Objektově-orientovaný návrh
Strukturované programování
Objektově-orientované programování
Technologie relačních databází
Technologie objektově-relačních databází
Technologie objektově-orientovaných databází

2. 4 Úloha modelování při vývoji systému

Model **- abstrakce něčeho (systému, jeho části) pro účely pochopení, experimentování apod. před vlastním vytvořením.**

- **použití modelu:**
  - ♦ **náhrada fyzického zařízení pro experimentování (simulace),**
  - ♦ **komunikace se zákazníkem,**
  - ♦ **vizualizace,**
  - ♦ **redukce složitosti (abstrakce)**
  - ♦ **dokumentace.**
- **zpravidla několik typů modelů pro různý pohled na systém (statický/dynamický, logický/fyzický, vnější/vnitřní, …)**