

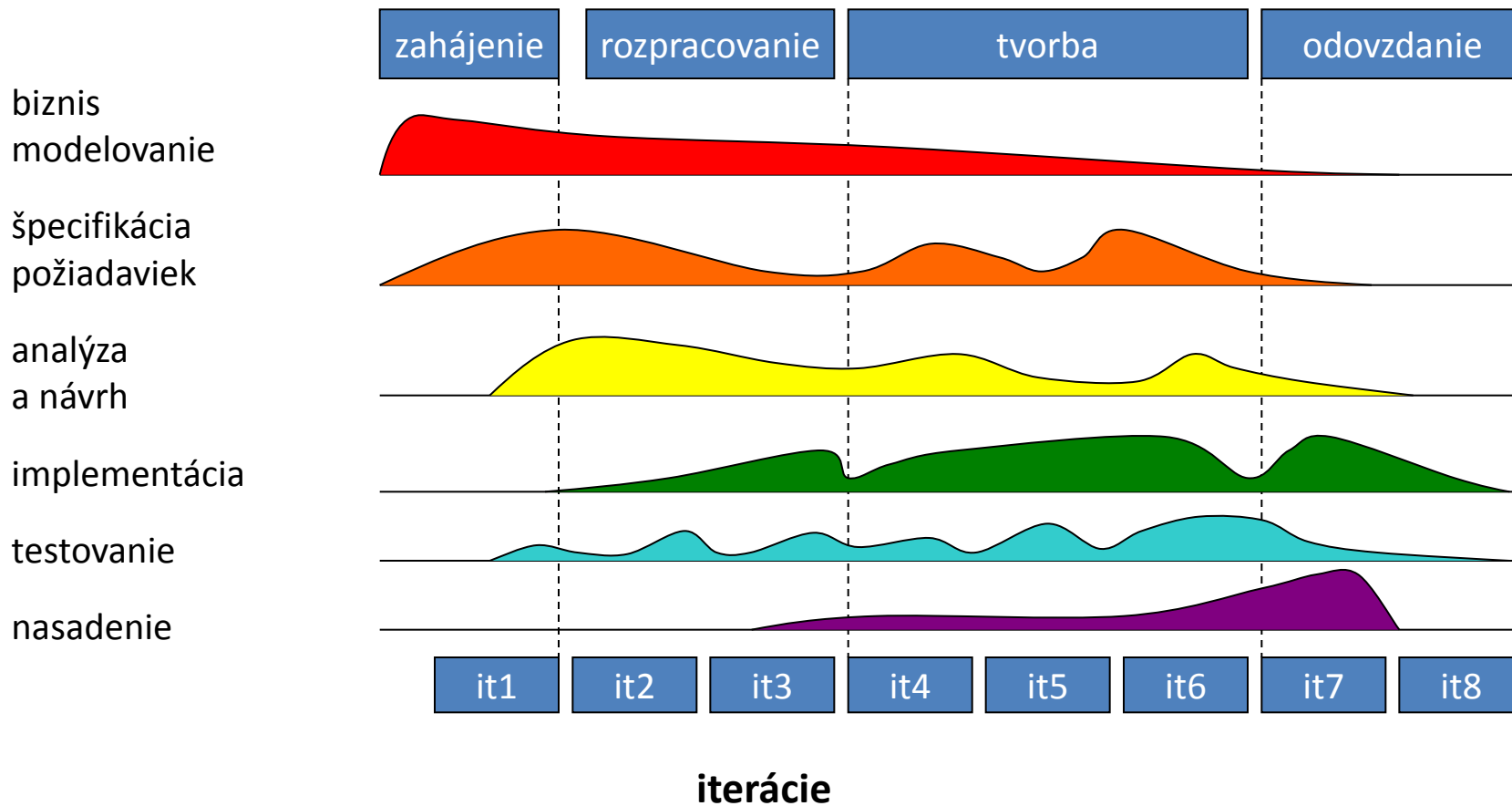
# 8

## Návrh

# RUP – schéma (obsah x čas)

## tok činností

## fázy



- Analýza:
  - Logický model tvoreného systému
  - Analýza požiadaviek z pohľadu problémovej domény
- Návrh
  - Presná špecifikácia spôsobov ako to implementovať
  - Zlúčenie technických riešení
    - Perzistencia objektov
    - Ich distribúcia
    - Architektúra
    - GUI
  - Založený na analytickom modeli

# Aspekty návrhu

- Kompatibilita
- Rozšiřovateľnosť
- Udržateľnosť
- Modulárnosť
- Odolnosť voči chybám
- Znovupoužitelnosť
- Robustnosť
- Bezpečnosť
- Použitelnosť

# Činnosti návrhu

- Návrh architektúry systému
  - Rozdeľuje systém do podsystémov alebo komponentov



Podsystém je množina elementov, ktorá je sama systémom a komponentom väčšieho systému

- Podrobný návrh systému
  - Každá časť systému je popísaná podrobne, aby to bolo dostatočné pre následné kódovanie
  - Časť systému – podsystém (subsystem)
- Dôraz na rozhrania

# Návrhové modely

- Návrhových podsystémov
- Návrhových tried
- Rozhraní
- Návrhových realizácií prípadov použitia
- Diagramov nasadenia

# Návrh architektúry systému

## Vstupy

- Model požiadaviek
- Model prípadov použitia
- Model analýzy
- Popis architektúry

## Výstupy

- Podsystem (načrtnutý)
- Rozhrania (načrtnuté)
- Návrhové triedy (načrtnuté)
- Model nasadenia (načrtnutý)
- Popis architektúry

# Návrh pod systému

## Vstupy

- Model požiadaviek
- Podsystem (načrtnutý)
- Rozhrania (načrtnuté)

## Výstupy


- Podsystem (úplný)
- Rozhrania (úplný)



# Návrh architektúry systému

- Architektúra SW systémov – vysokoúrovňový dizajn SW
  - Rámec pre podrobnejší návrh rozsiahleho systému
  - Popisuje organizáciu systému do podsystémov a alokáciu podsystémov na HW a SW komponenty
- Kroky:
  - Rozdelenie systému do podsystémov
  - Rozdelenie do vrstiev a oddielov
  - Návrh topológie systému
  - Identifikácia paralelizmu, alokácia na uzly a voľba komunikácie
  - Voľba spôsobu riadenia, a pod.

# Rozdelenie systému do podsystémov

- Podsystem obsahuje aspekty systému s podobnými vlastnosťami (max 20)
  - Príklad PC obsahuje podsystémy – správa pamäte, systém súborov, plánovanie procesov, a pod.
- Podsystem identifikujeme podľa služieb, ktoré poskytuje
  -  Služba – množina funkcií, ktoré majú rovnaký základný účel
- Hranice podsystému sa zvolia tak aby väčšina komunikácie prebiehala vo vnútri podsystému

# Rozdelenie systému do podsystémov

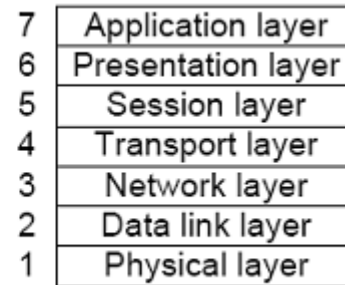
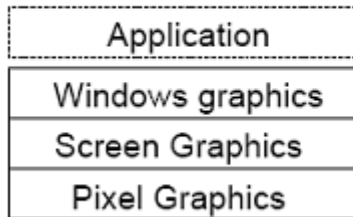
- Vzťah medzi dvoma podsystémami
  - Klient – poskytovateľ
  - Peer – to – peer
- Dekompozícia systému do podsystému – základné rozdelenie do:
  - horizontálnych vrstiev
  - alebo vertikálnych oddielov

# Rozdelenie do vrstiev

- Vrstvené systémy - usporiadaná množina virtuálnych svetov
- Každý svet je postavený z prvkov nižšieho sveta a poskytuje stavebné prvky vyššiemu svetu
- Medzi vrstvami je vzťah klient – poskytovateľ
- Znalosť je jednosmerná
- Vrstvené architektúry
  - Uzavreté
    - Vrstva je implementovaná iba pomocou prostriedkov najbližšej nižšej vrstvy
    - Obmedzuje závislosť medzi vrstvami - modularita
    - Ľahšie zmeny v rozhraní
    - Príklad - sieťový model ISO/OSI model
  - Otvorené
    - Môže používať prostriedky ktorejkoľvek nižšej vrstvy
    - Ťažká údržba – zmena podsystému môže ovplyvniť ľubovoľnú vyššiu vrstvu
    - Tvoreba efektívneho a kompaktnejšieho kódu

# Príklad

- Interaktívny grafický systém
  - Aplikácia pracuje s oknami
  - Okná sú implementované pomocou grafických operácií
  - Grafické operácie sú implementované pomocou operácií nad jednotlivými pixelmi



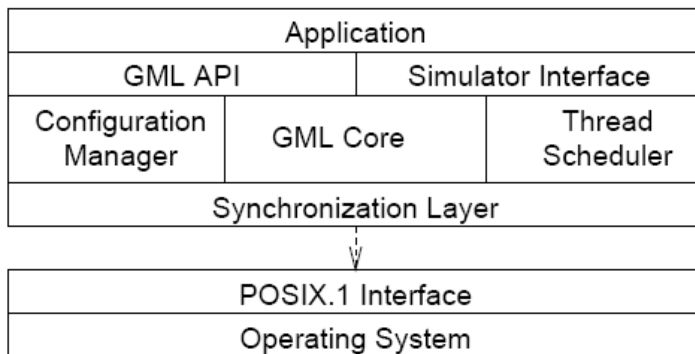
- ISO/OSI model

# Rozdelenie do vrstiev

- Špecifikácia systému obvykle definuje iba vrchnú vrstvu
- Spodná vrstva je daná dostupnými zdrojmi (HW, OS, knižnice)
- Pre malé systémy – cca 3 vrstvy
- Pre veľké systémy – cca 5-7 vrstiev
- Aj najzložitejšie systémy – max. 10 vrstiev
- Poznámka (odporúčanie RUP):
  - 0 – 10 tried                      vrstvy nie sú potrebné
  - 10 – 50 tried                    2 vrstvy
  - 25 – 150 tried                  3 vrstvy
  - 100 – 1000 tried                4 vrstvy

# Rozdelenie do oddielov

- Oddiely (partície)
  - rozdeľujú systém vertikálne na nezávislé alebo slabo zviazané podsystémy
  - Každý z nich poskytuje iný typ služieb
- Podsystémy môžu navzájom o sebe vedieť, ale táto znalosť nie je veľká, preto nevznikajú podstatné závislosti medzi oddielmi
- Systém môže byť postupne dekomponovaný do podsystémov pomocou vrstiev a oddielov
  - Väčšina veľkých systémov – zmes vrstiev a oddielov



Príklad hybridnej dekompozície



Rozdelenie systému do partícií

# Topológia systému

- Po identifikácii základných podsystémov – určenie tokov dát medzi nimi
  - Niekedy tečú dáta medzi všetkými podsystémami, v praxi len zriedka
  - Vo väčšine prípadov jednoduchá topológia
    - Jednoduchá sekvencia – napr. prekladač
    - Hviezda – napr. hlavný systém, ktorý riadi podriadené systémy



# Identifikácia paralelizmu

- Úloha – identifikácia podsystémov, ktoré musia a ktoré nesmú pracovať paralelne
- Paralelné podsystémy môžu byť implementované rôznymi HW jednotkami
- Podsystémy bez možnosti paralelného behu, môžu byť súčasťou rovnakého procesu
- Identifikovanie inherentného (prirodzeného) paralelizmu
  - Dva objekty sú inherentné ak dokážu prijímať udalosti v rovnakom čase bez vzájomnej komunikácie
  - Nemôžu existovať na jednom vlákne riadenia
  - Vlákno riadenia

# Alokácia podsystémov

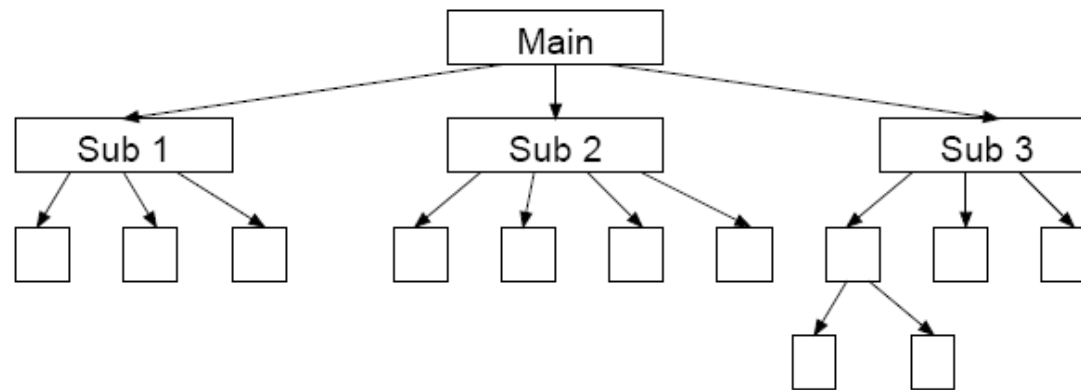
- Odhad požiadaviek na HW zdroje
  - Hrubý odhad výpočtovej sily na základe požadovaného počtu transakcií za sekundu a doby spracovania jednej transakcie a pod.
- Rozhodnutie o HW alebo SW implementácii
- Alokácia úloh na fyzické jednotky (PC alebo CPU)
  - Úloha vyžaduje vysoký výkon – viac CPU
  - Podsystémy, ktoré často komunikujú – umiestnené v jednej jednotke
- Určenie prepojenia fyzických jednotiek
  - Výber topológie
  - Určenie požiadaviek na mechanizmy a komunikačné protokoly

# Dátové úložiská

- Interné a externé úložiská dát majú dobre definované rozhranie – slúžia ako hranice oddeľujúce jednotlivé podsystémy
- Typy úložísk:
  - Súbory
    - Lacné, jednoduché a permanentné, s nízkou úrovňou abstrakcie – nutný ďalší kód na prácu s nimi
    - Vhodné pre objemné a ťažko štruktúrovateľné dáta a dáta s malou informačnou hustotou s krátkou dobou životnosti
  - Databázy
    - Spoločné rozhrania pre množinu aplikácií pomocou jazyka SQL
    - Vhodné pre dáta ku ktorým pristupujú viacerí užívatelia
    - Nevýhody
      - vyššia réžia,
      - nedostatočná podpora pre zložitejšie dátové štruktúry
      - nemožnosť čistej integrácie s jazykom SQL

# Výber mechanizmu riadenia

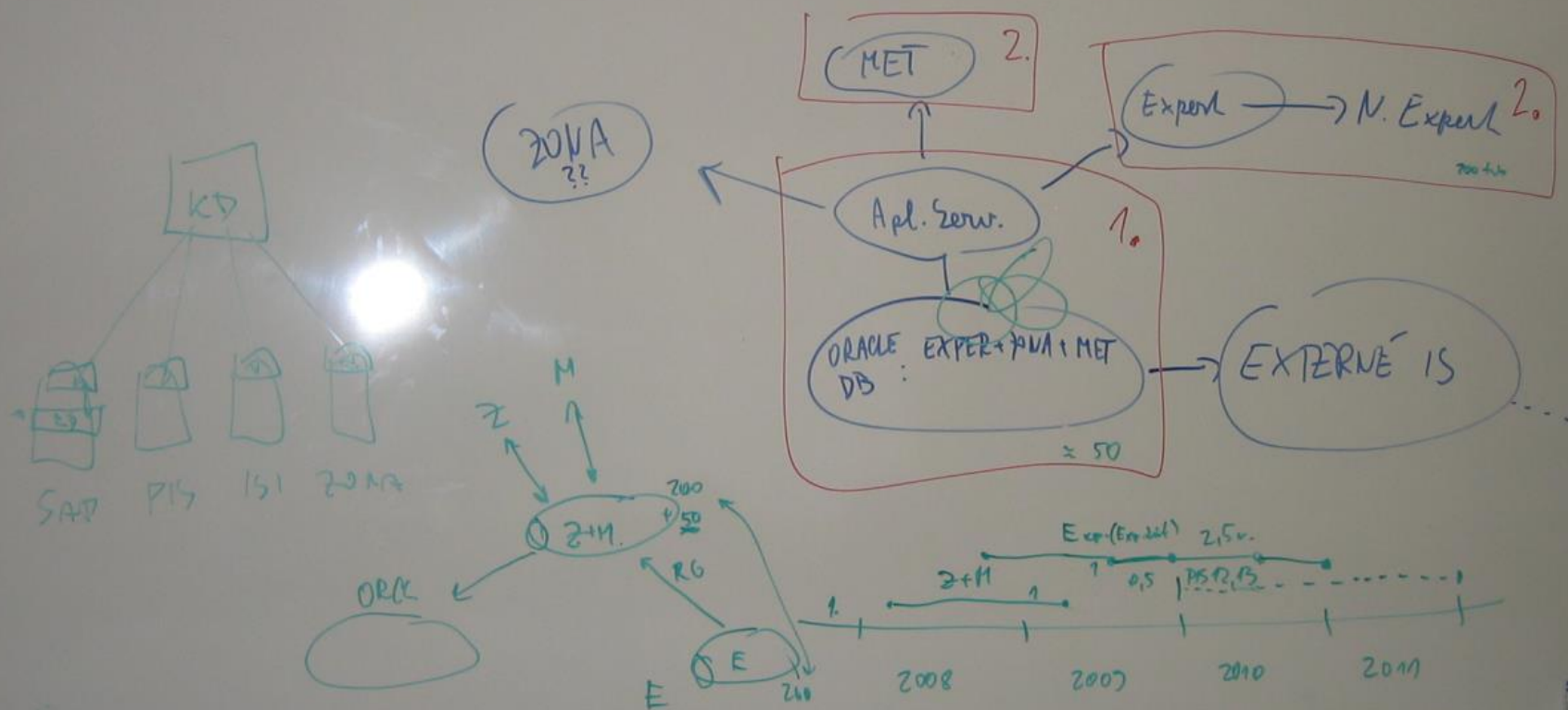
- V súvislosti s externými udalosťami existujú tri mechanizmy riadenia:
  - Sekvenčný systém riadený procedurálne
  - Sekvenčný systém riadený udalosťami
  - Paralelný systém



# Mechanizmy riadenia

- Systémy riadené procedurálne
  - Beh systému je riadený programovým kódom
  - Výhoda – jednoduchá implementácia
  - Nevýhoda – ťažké spracovanie asynchrónnych udalostí
- Systémy riadené udalosťami
  - Beh systému riadi dispečer, predstavovaný podsystémom, programovacím jazykom alebo OS
  - S jednotlivými udalosťami sú zviazané procedúry aplikácie
  - Procedúra po skončení činnosti vracia riadenie dispečerovi
  - Výhoda – jednoduchá obsluha nových typov udalostí
  - Nevýhoda – zložitá implementácia
- Paralelné systémy
  - Riadenie niekoľkých nezávisle bežiacich objektov
  - Udalosti prichádzajú k objektom ako správy
  - Objekt môže čakať na vstup, zatiaľ čo ostatné pokračujú v činnosti

# Návrh architektúry



# Návrh tried

## Vstupy

- Realizácia prípadov použitia
- Návrhová trieda (načrtnutá)
- Rozhrania (načrtnuté)
- Analytická trieda

## Výstupy

- Návrhová trieda (úplná)
- Rozhrania (úplne)

# Návrhová trieda

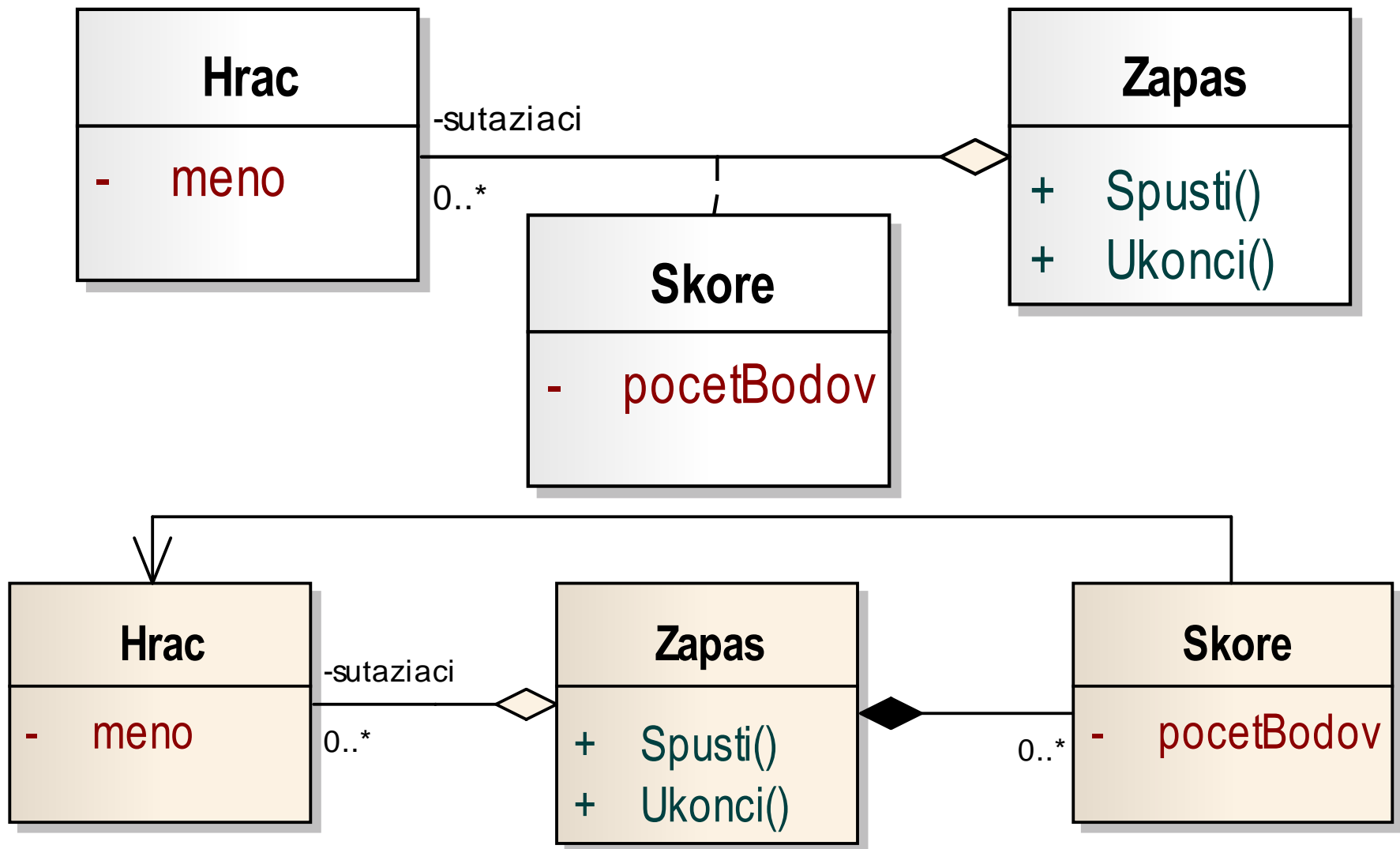
- Úplná a dostačujúca
- Jednoduchá
- Vysoko súdržná
- Bez tesných väzieb



# Objektovo orientovaný návrh

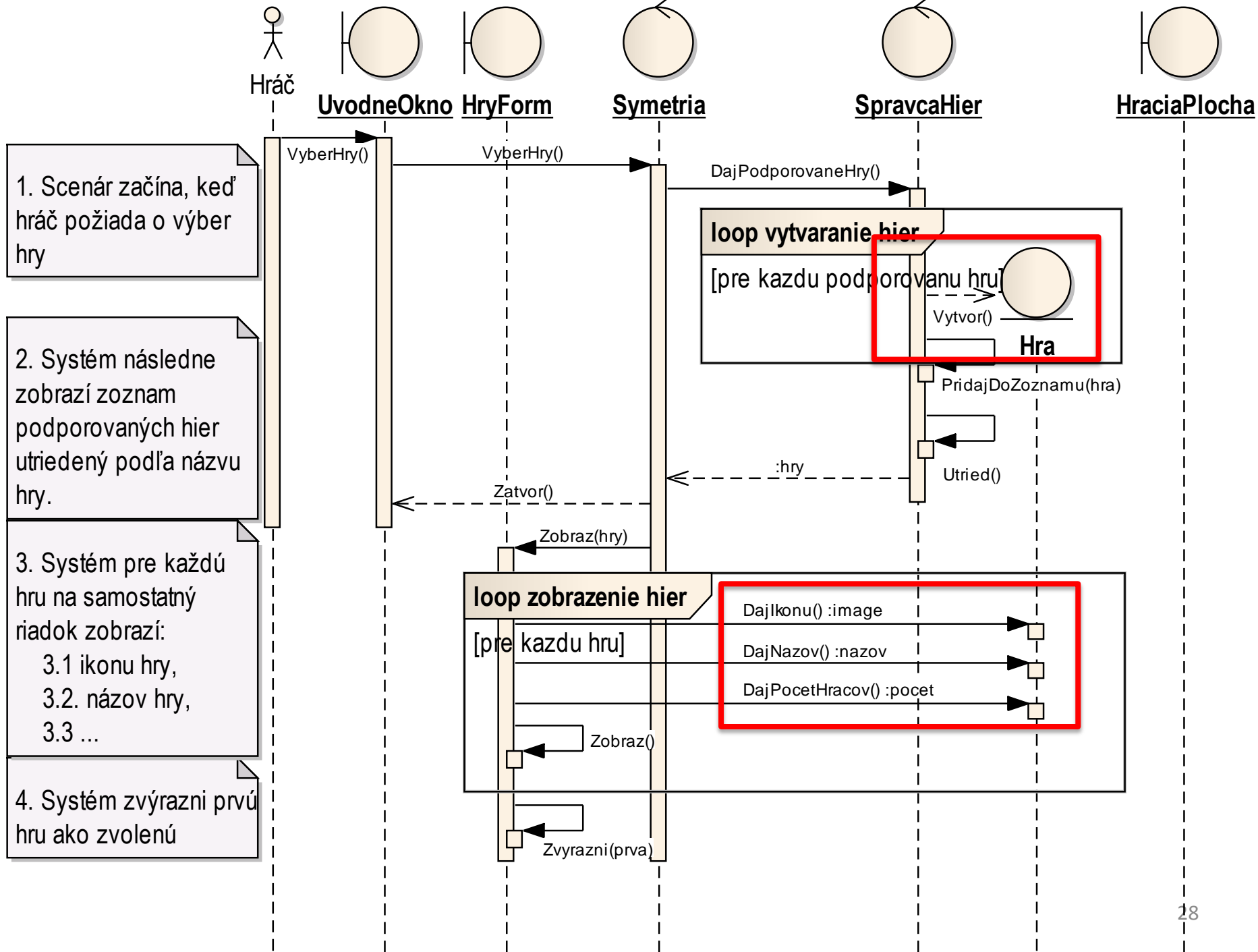
- Vstup: analytické triedy
- Analytická trieda sa môže stať:
  - jedinou triedou,
  - časťou triedy,
  - agregovanou triedou
  - skupinou spriaznených tried,
  - asociáciou a pod.
- Vytvorenie návrhových tried
- Definícia operácií, atribútov
- Definícia asociácií, agregácií a kompozícií

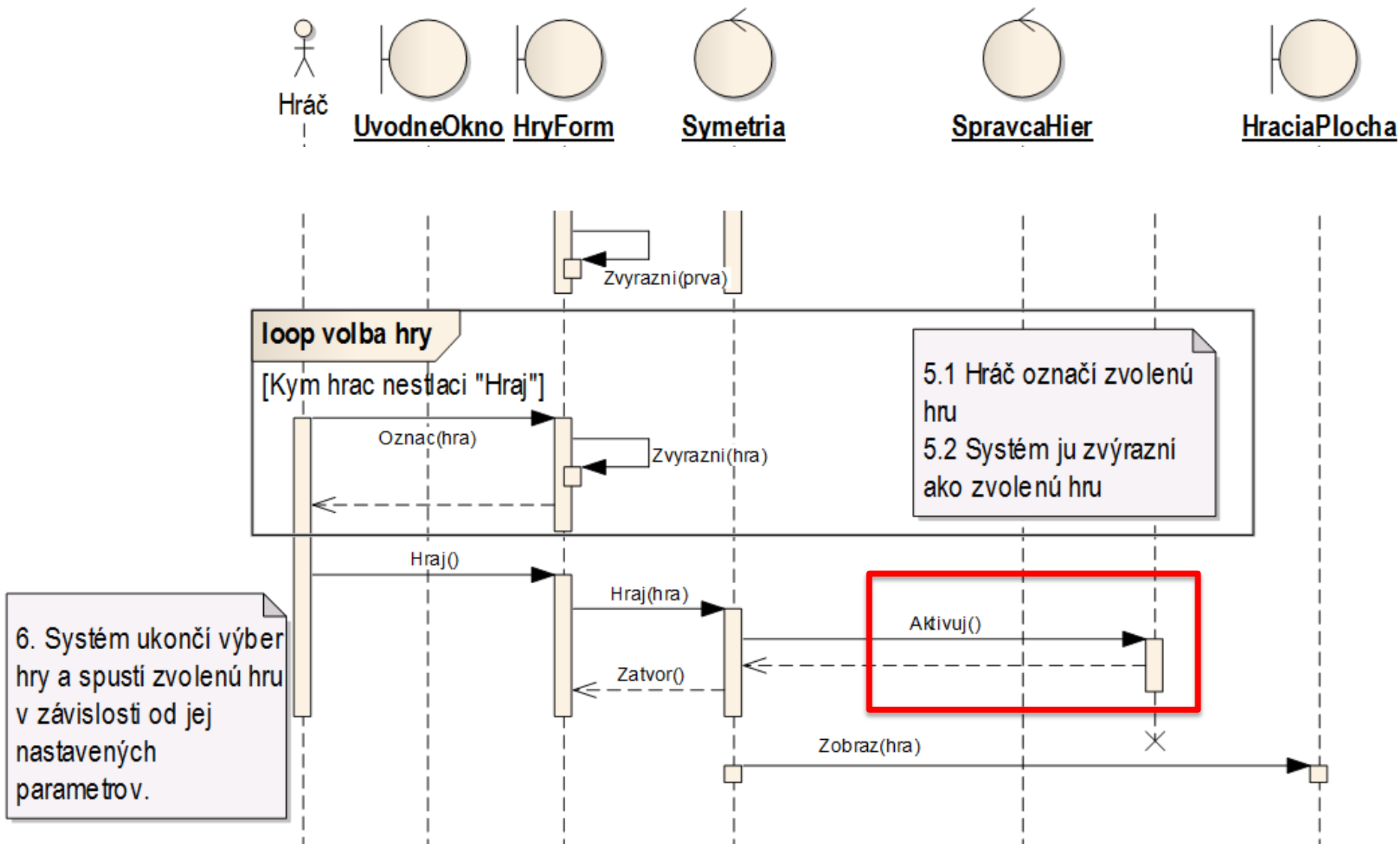
# Symetria – zmeny tried



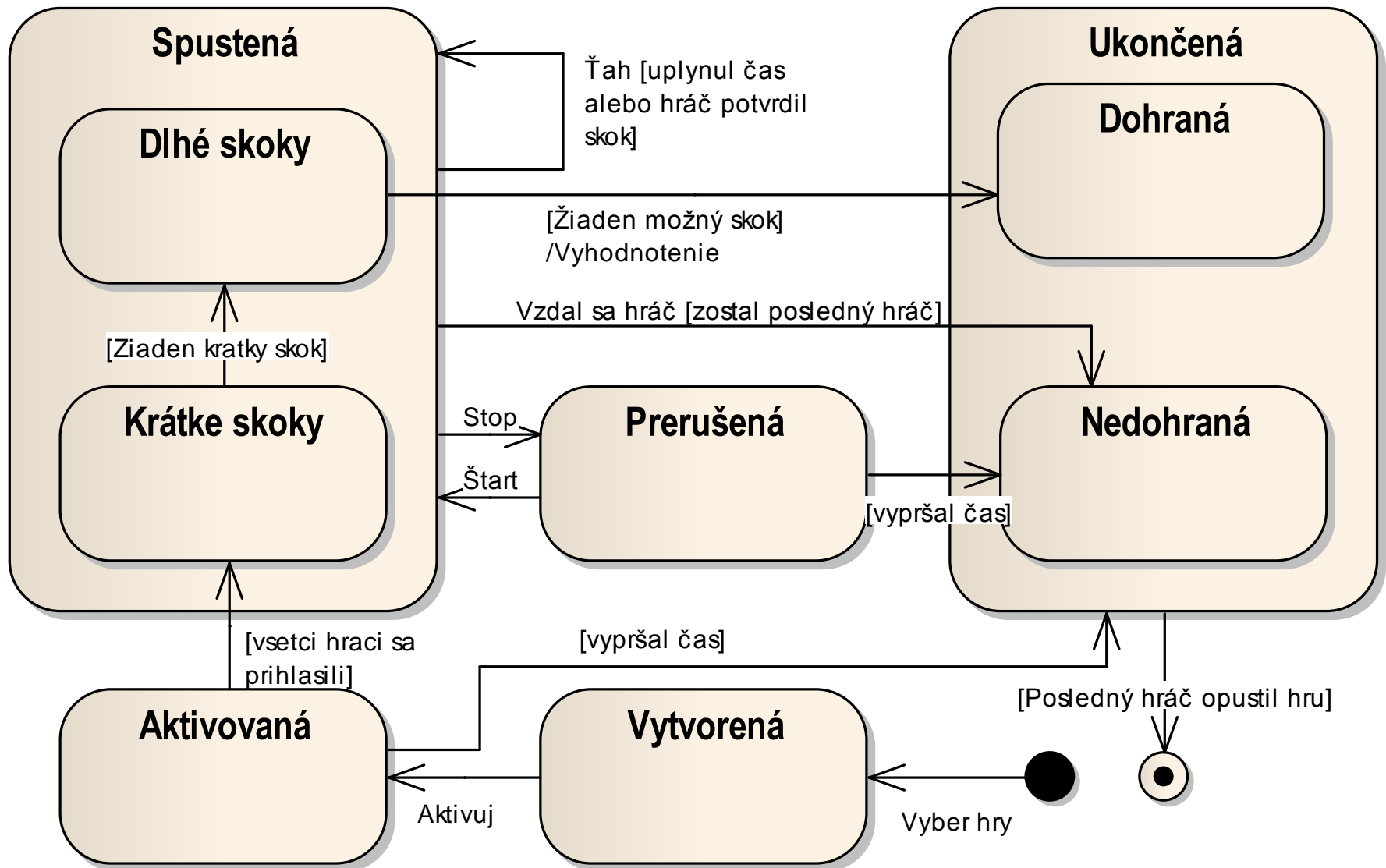
# Definícia operácií

- Operácie -zoznam slovies (jednoduchý spôsob)
- Z popisu interakcií medzi objektmi
  - Nakreslenie diagramov spolupráce alebo sekvenčných diagramov
  - Zistenie stimulov, ktoré dokáže objekt prijať –operácie
- Ďalšie možnosti operácií
  - Inicializácia novo vytvorenej inštancie spolu s prepojením s asociovanými objektmi
  - Vytvorenie kópie inštancie
  - Test ekvivalencie inštancií, a pod.
- Operácie popíšeme: názov, parametre, návratová hodnota, krátky popis, viditeľnosť





# Stavový diagram triedy SkokovaHra

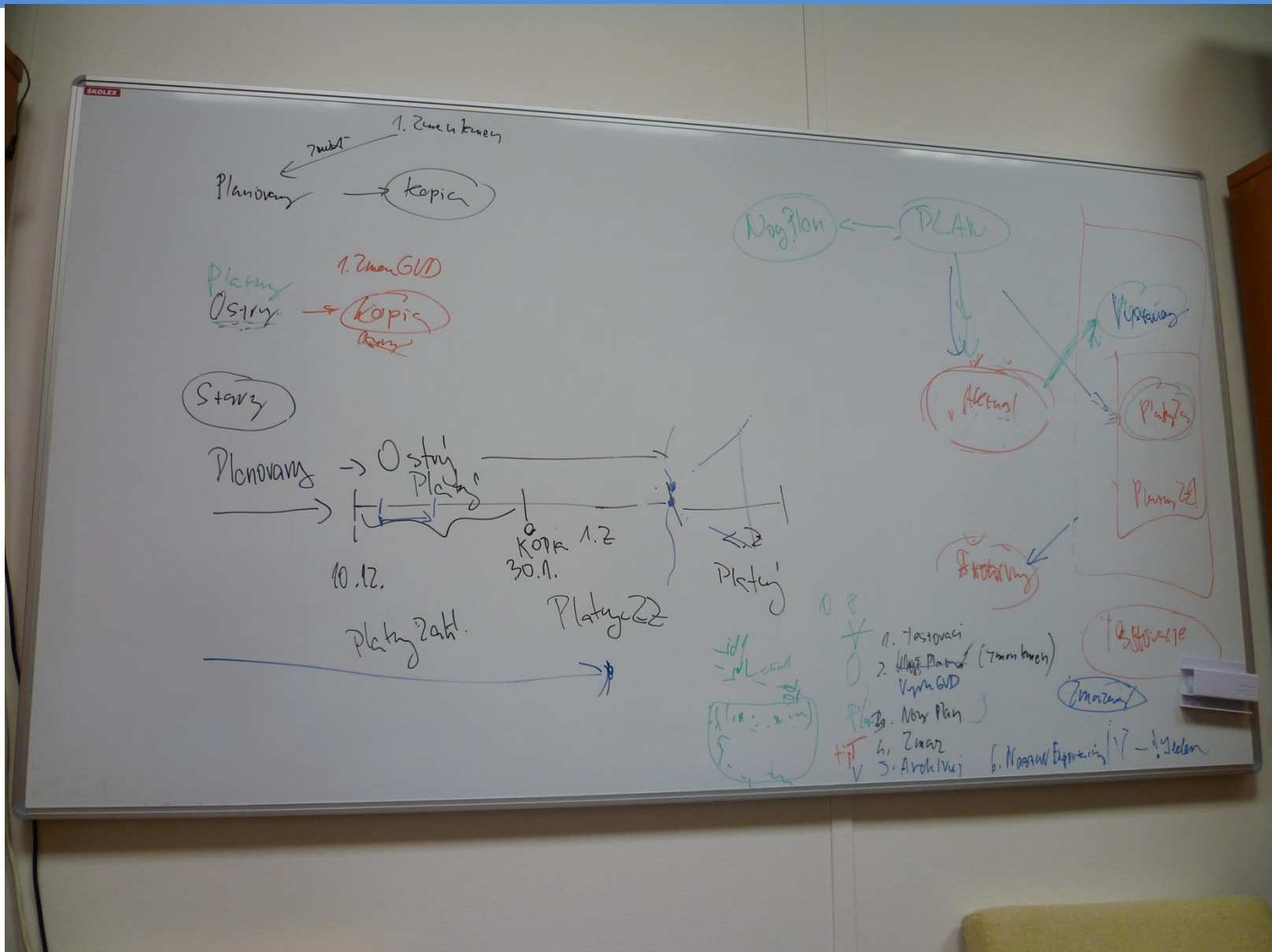


# Doplnenie operácií

## SkokovaHra

- + Aktivuj(pocetHracov :byte) : bool
- + DajKonu() : Image
- + DajLimitHracov(max :byte\*, min :byte\*) : void
- + DajNazov() : String
- DajPocetHracov() : byte
- + HracOdstupil() : bool
- + JeKoniec() : bool
- NastavDlheSkoky() : void
- NastavPocetHracov(pocet :byte) : void
- OverPravidla(typ :TypPravidiel) : bool
- + Pokracuj() : bool
- + SkokovaHra()
- + Spusti() : bool
- + Stop() : bool
- + Ukonci(dohrana :bool) : bool
- + UrobnyTah(tah :Tah) : int
- VypocitajBody(tah :Tah) : int

# Stavy grafikonu





# Definícia atribútov

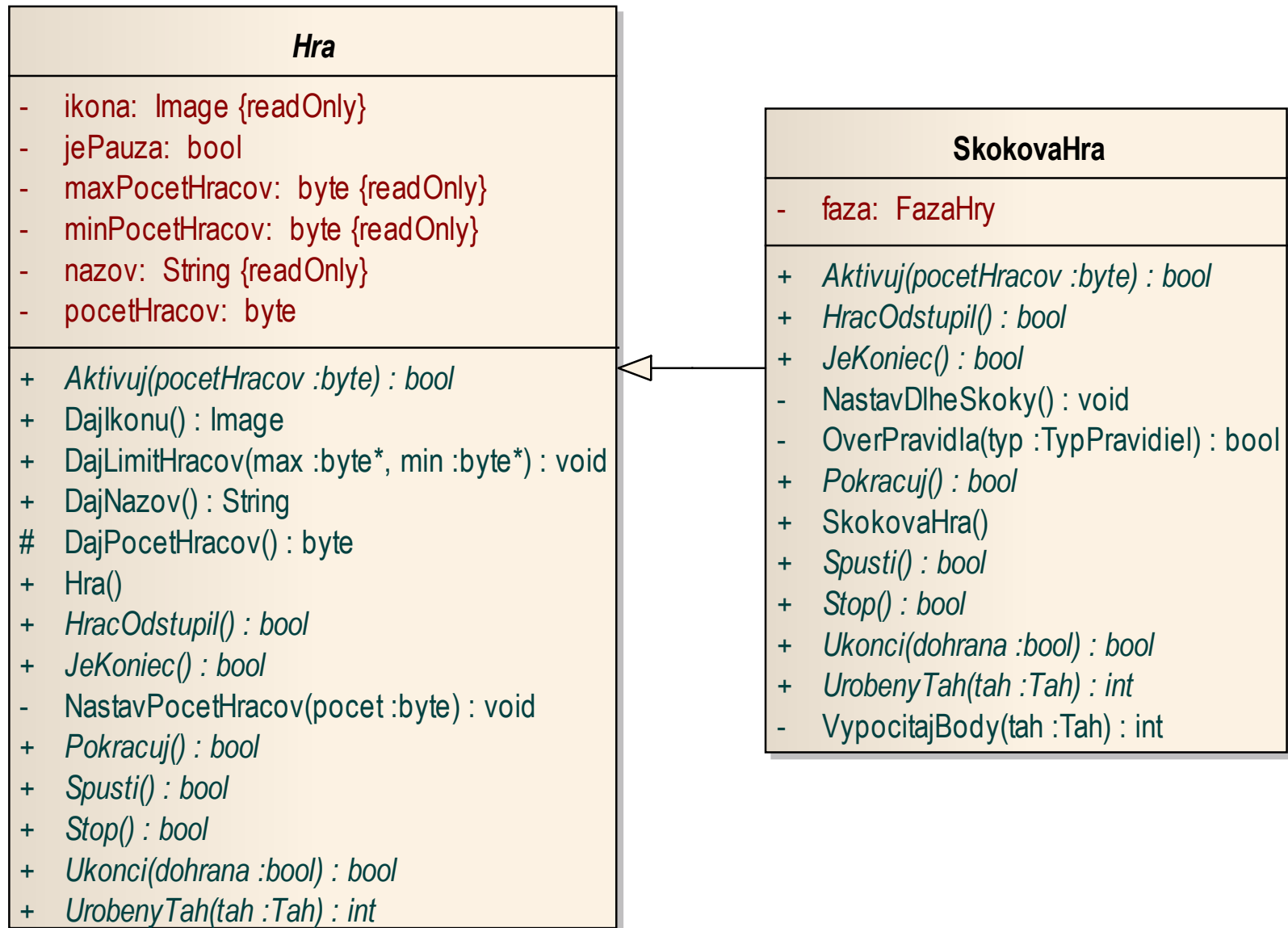
- Varianty:
  - Vychádzame z logických atribútov objektu (čo je potrebné pre zachovanie stavu objektu ?)
  - Aké atribúty sú potrebné pre implementáciu operácií
- Atribúty v návrhu musia byť jednoduché (int, boolean, a pod) alebo musia vyjadrovať hodnotu (string) –inak to budú asociácie
- Atribúty sa popíšu:
  - meno, typ počiatočná hodnota, viditeľnosť
  - Snaha o skrývanie informácií –súkromné atribúty
- Overenie potreby nájdených atribútov

# Definícia atribútov

SkokovaHra
<ul style="list-style-type: none"><li>- faza: FazaHry</li><li>- image: Image {readOnly}</li><li>- jePauza: bool</li><li>- maxPocetHracov: byte {readOnly}</li><li>- minPocetHracov: byte {readOnly}</li><li>- nazov: String {readOnly}</li><li>- pocetHracov: byte</li></ul>
<ul style="list-style-type: none"><li>+ Aktivuj(pocetHracov :byte) : bool</li><li>+ DajIkonu() : Image</li><li>+ DajLimitHracov(max :byte*, min :byte*) : void</li><li>+ DajNazov() : String</li><li>- DajPocetHracov() : byte</li><li>+ HracOdstupil() : bool</li><li>+ JeKoniec() : bool</li><li>- NastavDlheSkoky() : void</li><li>- NastavPocetHracov(pocet :byte) : void</li><li>- OverPravidla(typ :TypPravidiel) : bool</li><li>+ Pokracuj() : bool</li><li>+ SkokovaHra()</li><li>+ Spusti() : bool</li><li>+ Stiahni() : bool</li></ul>

- Ak trieda obsahuje viac než 10 atribútov, 10 asociácií alebo 20 operácií
  - zle navrhnutá?
  - je nutné ju rozdeliť?

# Doplnenie abstrakcie



# Jeden alebo dva modely?

Stratégia	Dôsledky
Spresnenie analytického modelu na návrhový	Jeden návrhový, ale žiaden analytický
Analytický model spresníme na návrhový a použijeme CASE nástroj na obnovu analytického	Jeden návrhový, ale obnovený analytický nemusí byť dostatočný
Ustálime analytický a jeho kópiu spresníme na návrhový	Dva nesynchronizované modely
Udržujeme 2 samostatné modely	Dva synchronizované modely – náročná údržba

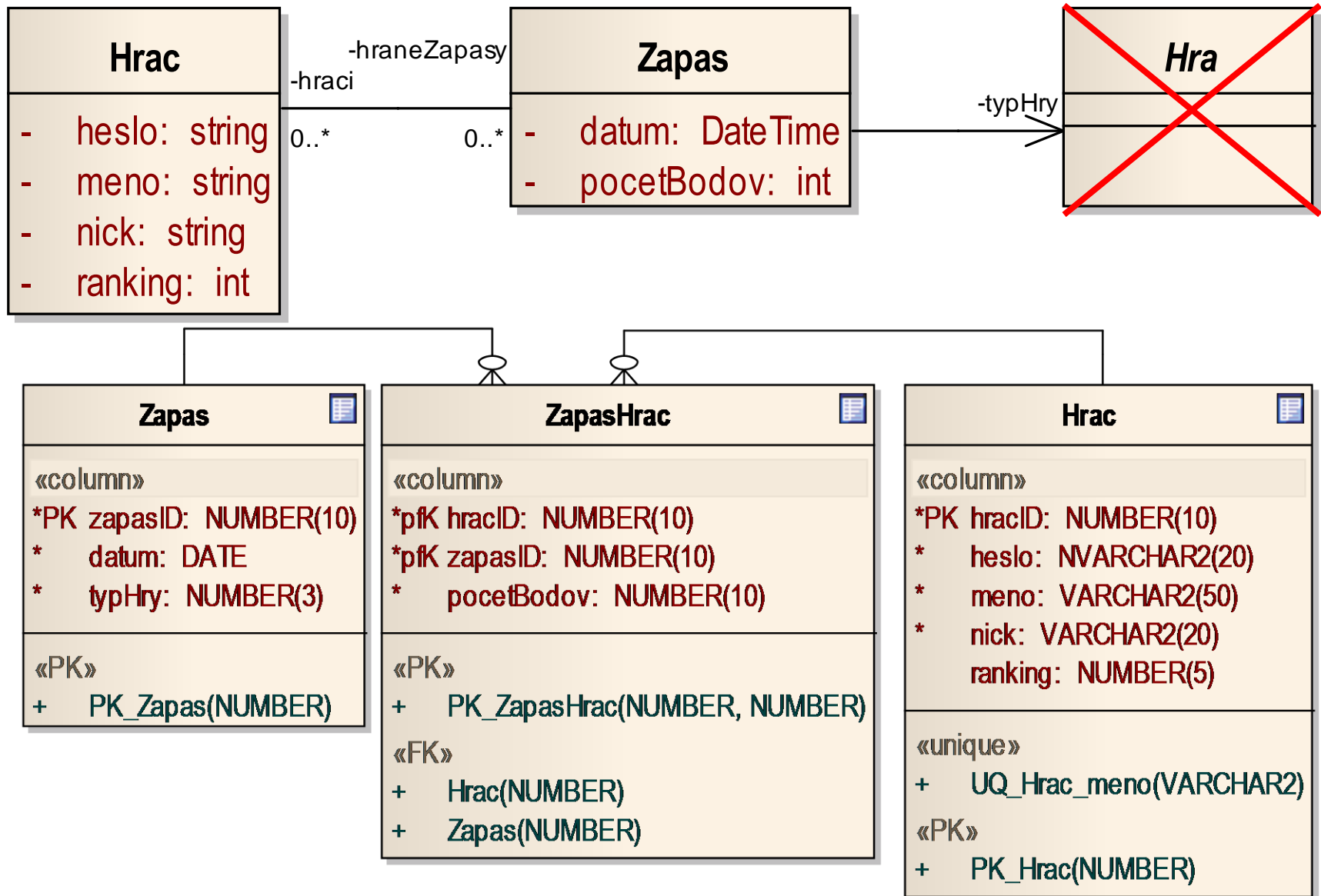
# Význam analytického modelu

- Nové osoby v projekte
- Porozumenie systému po dlhej dobe
- Pochopenie systému –uspokojovanie požiadaviek
- Sledovateľnosť požiadaviek
- Plánovanie údržby a rozširovania
- Pochopenie logickej architektúry

# Tvorba návrhových tried

- Hraničné triedy
  - Ak sú k dispozícii nástroje pre návrh GUI, potom jedna hraničná trieda = jedno okno alebo formulár
  - Jedna trieda = API alebo protokol
- Entitné (dátové) triedy
  - Často pasívne a perzistentné-implementácia v súbore alebo v relačných databázach
  - Ak nie sú perzistentné–implementácia v pamäti
- Riadiace triedy
  - Obsahujú aplikačnú logiku

# Vytvorenie dátového modelu



# Realizácia prípadov použitia - návrh

## Vstupy

- Model požiadaviek
- Model prípadov použitia
- Model analýzy
- Model návrhu
- Model nasadenia

## Výstupy

- Realizácia prípadov použitia - návrh
- Rozhrania (načrtnuté)
- Návrhové triedy (načrtnuté)
- Podsystem (načrtnutý)



- Namiesto analytických tried –návrhové, rozhrania, komponenty
- Odhaľovanie nových nefunkčných požiadaviek a tried
- Identifikácia návrhových vzorov

# Modely tried projektu

- Doménový model tried
  - Výsledok biznis modelovania
- Konceptuálny model tried
  - Výsledok analýzy
- Implementačný model tried
  - Výsledok návrhu (UML) a implementácie (kód)

Ďakujem za pozornosť.

- Kontrola modelu
  - Overenie realizácie prípadov použitia
  - V návrhu nesmú chýbať správanie potrebné pre niektorý z prípadov použitia