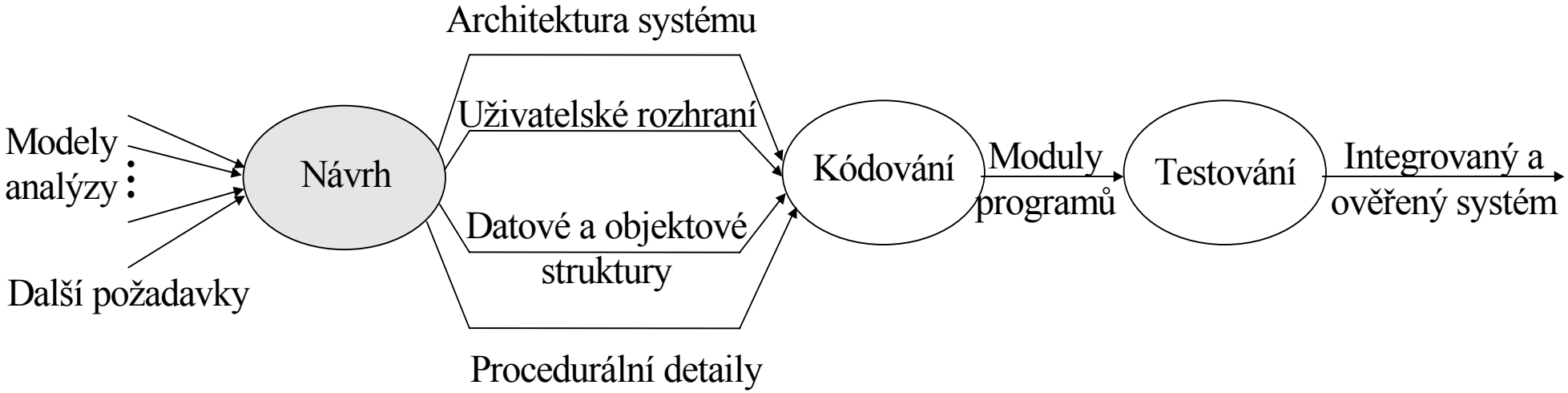
11 Návrh programového vybavení

- technické jádro procesu vývoje programového systému, existuje u všech modelů životního cyklu
- Jackson: „Začínající moudrost programátora (softwarového inženýra) spočívá v rozeznání rozdílu mezi uděláním programu, aby pracoval, a uděláním ho správně“.
- zásadní význam pro kvalitu (odhalení chyb, testovatelnost, udržovatelnost, použitelnost)

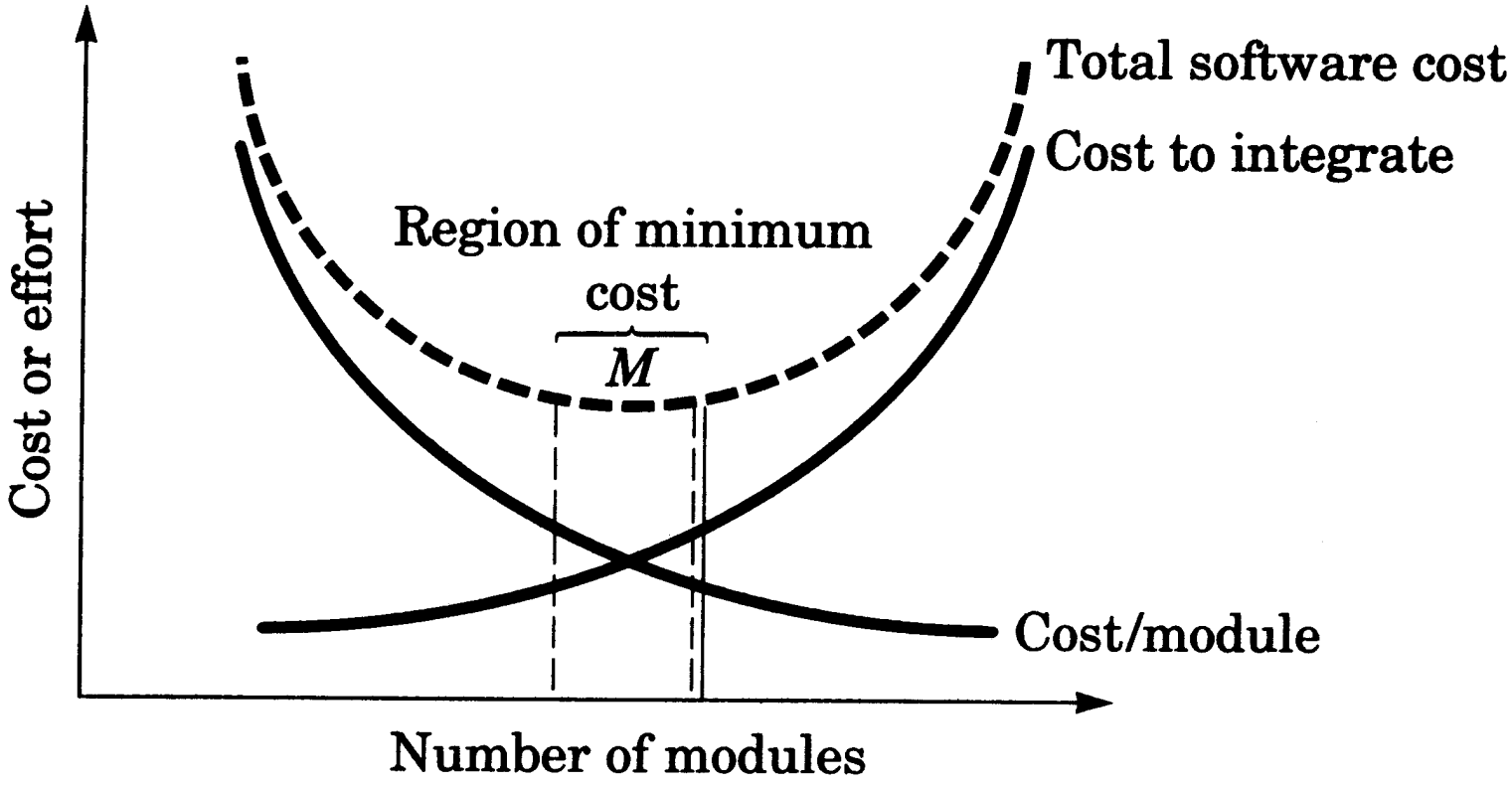


- typicky dva kroky:
  - předběžný (architektonický) návrh
  - detailní návrh

- historie:
  1. etapa (začátek 70.let) - kritéria pro modulární programy, návrh architektury programu shora dolů, strukturované programování.
  2. etapa (~1975) - metody transformace DFD a struktury dat.
  3. etapa (od poloviny 80.let) - objektově orientovaný přístup.
- přístupy k návrhu:
  a) Návrh orientovaný na funkce (strukturovaný návrh)
  b) Návrh orientovaný na struktury dat
  c) Objektově-orientovaný návrh

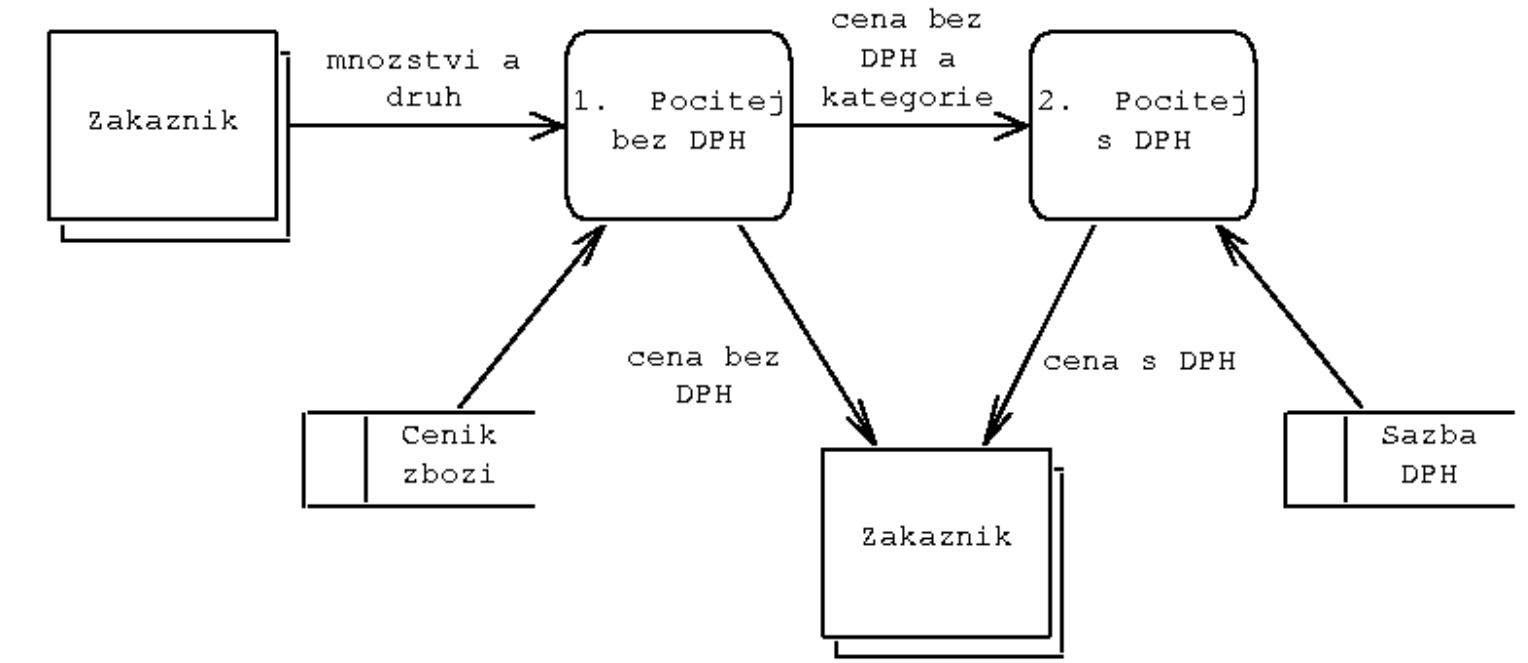
11.1 Strukturovaný návrh

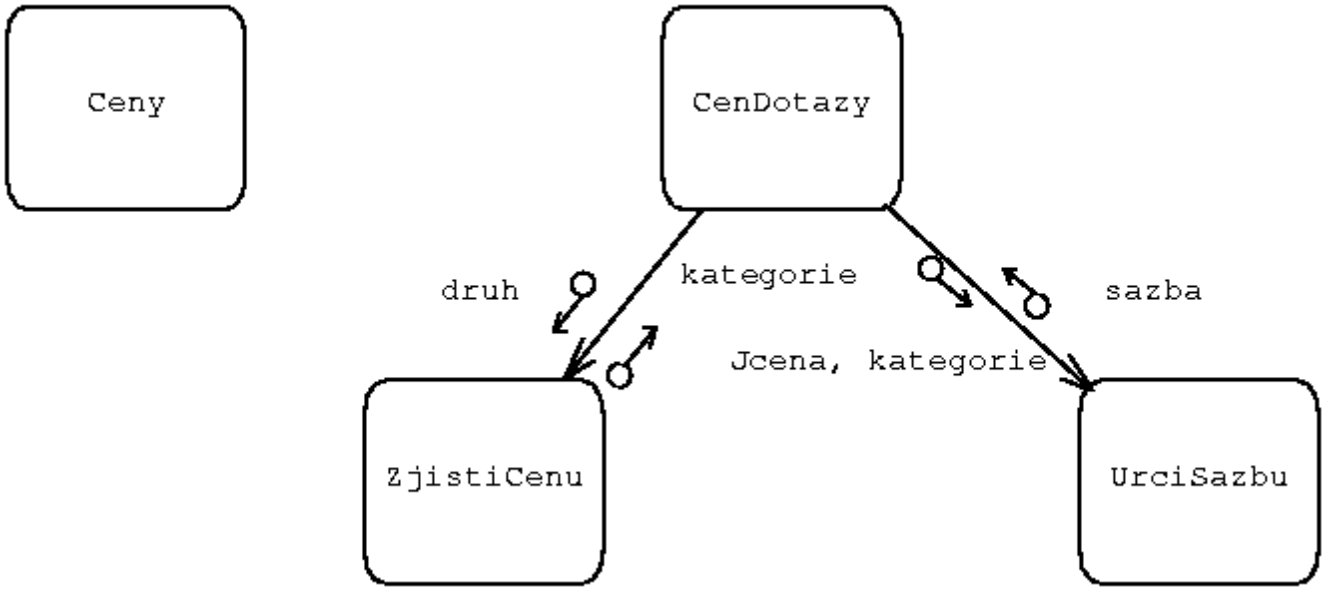
Techniky:
- abstrakce
- zpřesňování
- modularita - rozdělení na pojmenované, adresovatelné komponenty (moduly), řešení problému složitosti

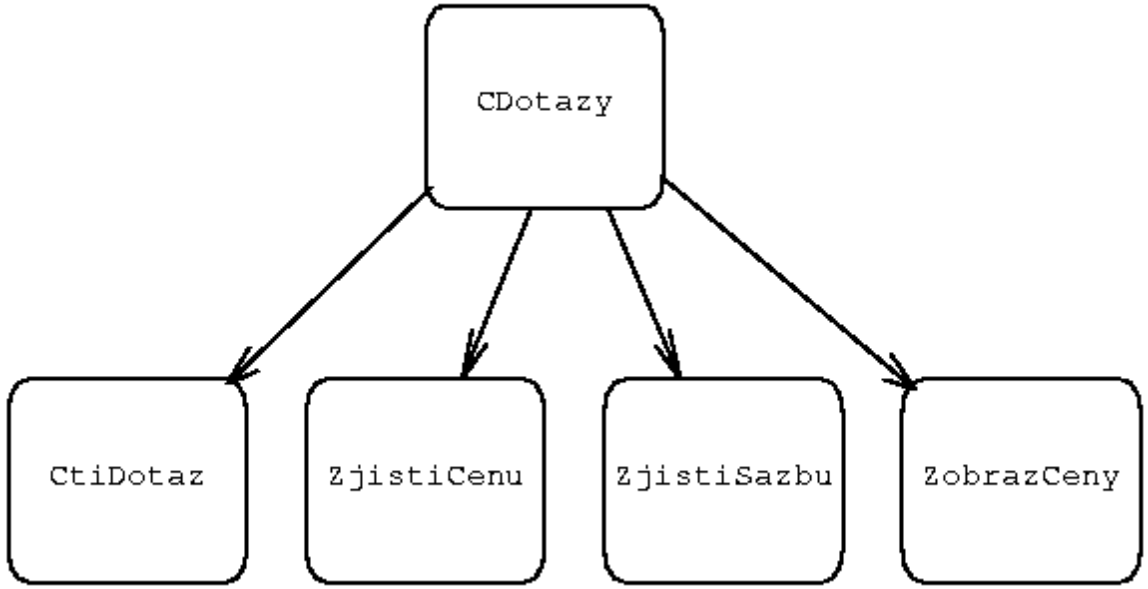


Modul **- jednotka zapouzdřující datové typy, funkce, proměnné.**

Cíl: Nalezení struktury modulů s vysokou *funkční nezávislostí* (snadnější vývoj, údržba, redukce šíření chyb, opakované použití).


Zakaznik
mnozstvi a druh
1. Pocitej bez DPH
cena bez DPH a kategorie
2. Pocitej s DPH
Cenik zbozi
cena bez DPH
cena s DPH
Sazba DPH
Zakaznik


Ceny
CenDotazy
druh
kategorie
sazba
Jcena, kategorie
ZjistiCenu
UrciSazbu



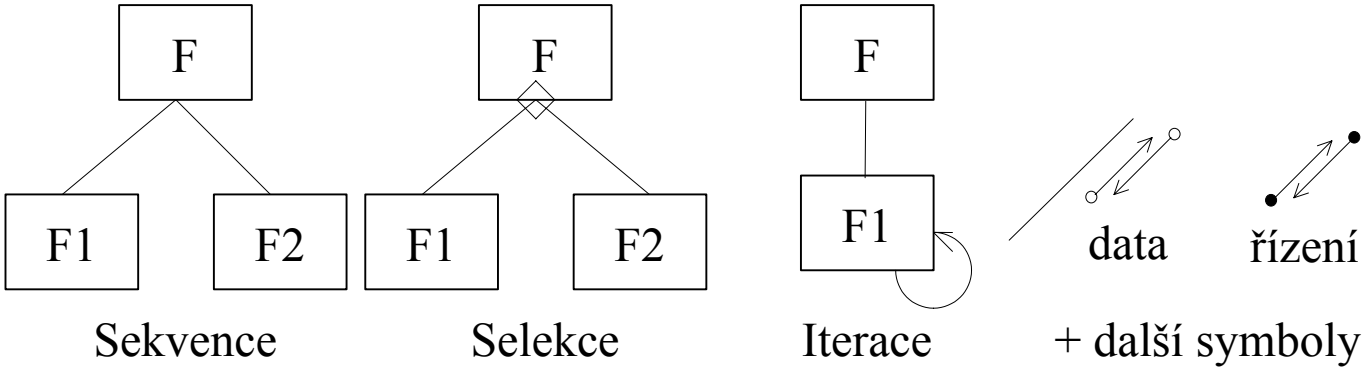
Míry nezávislosti:

- **koheze (cohesion) - náhodná, logická, časová, sekvenční, ...**
- **propojení (coupling) - žádné, datové, struktury, řízení, globální, ...**

Nástroje:

a) vyjádření struktury (architektury programu)

- **diagram struktury (strukturogram)**



- Jacksonův diagram

b) popis algoritmu - procedurální popis - viz minispecifikace

11.2 Objektově-orientovaný návrh

- **Možný postup (metodika OMT):**

A) Systémový návrh

A1) Rozčlenění na podsystémy

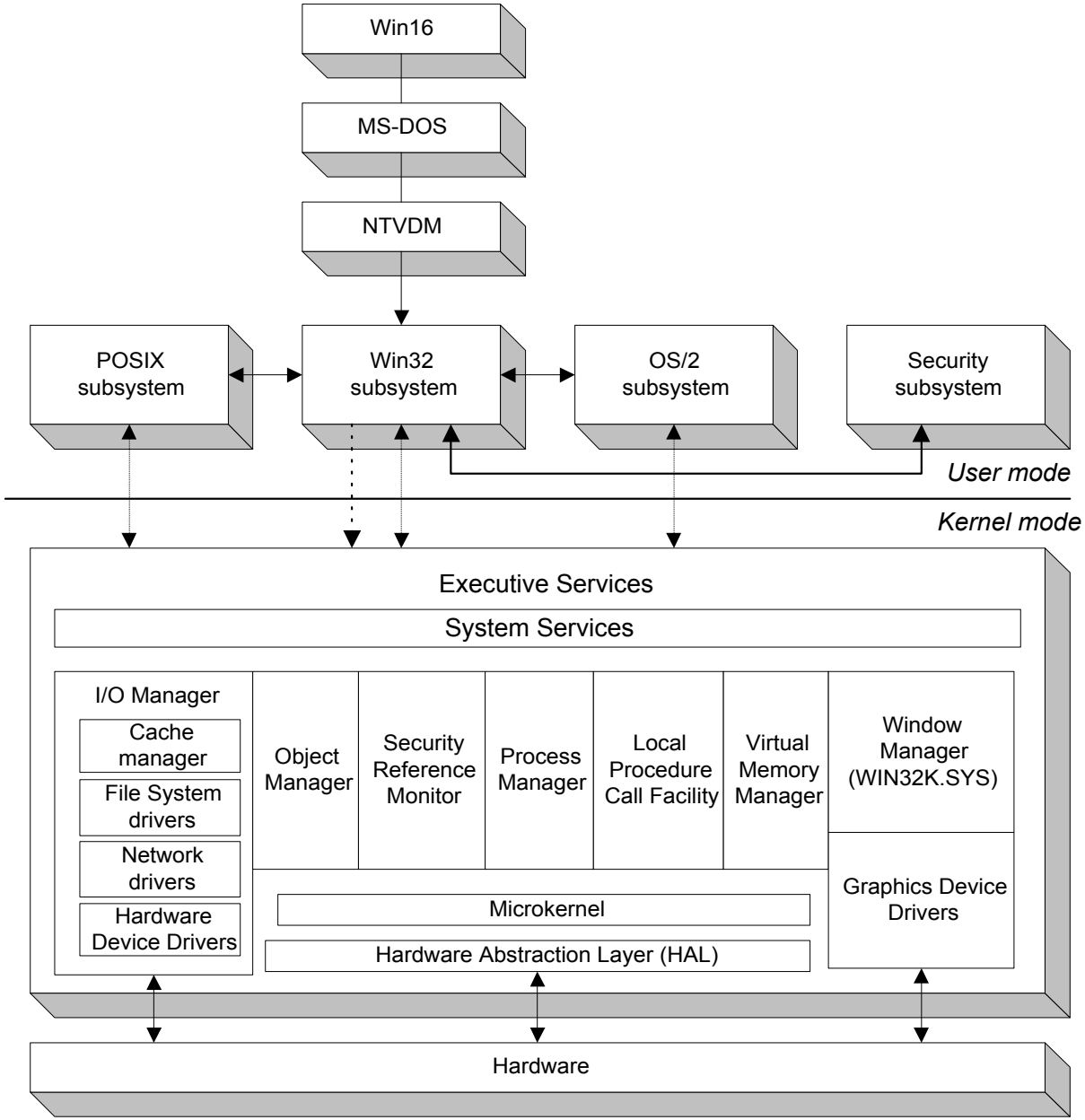
- podsystém - kolekce tříd, asociací, operací, omezení, atd. s dobře definovaným rozhraním k jiným podsystémům (poskytuje služby)
- většina interakcí uvnitř (koheze, propojení)
- v rozčleňování lze pokračovat
- vztahy podsystémů: - klient - server (poskytovatel)
- sobě rovný (peer-to-peer)
- podsystémy mohou být organizovány jako vrstvy nebo sekce (partition)
- architektura s vrstvami - uzavřená
- otevřená
- v zadání vždy specifikována nejvyšší vrstva a často i nejnižší
- přenositelnost přepisem vrstvy
- sekce - nezávislé nebo slabě vázané podsystémy
- vrstvy a sekce lze kombinovat
- v UML podsystémy (stereotyp <<subsystem>> symbolu seskupení) a komponenty

Př)

<table>
<tr><td colspan="3">aplikace</td></tr>
<tr><td rowspan="3">řízení uživatelského rozhraní</td><td>grafika okna</td><td rowspan="3">simulační knihovna</td></tr>
<tr><td>grafika obrazovky</td></tr>
<tr><td>grafika bodu</td></tr>
<tr><td colspan="3">operační systém</td></tr>
<tr><td colspan="3">HW</td></tr>
</table>

<table>
<tr><td>Aplikační</td></tr>
<tr><td>Prezentační</td></tr>
<tr><td>Relační</td></tr>
<tr><td>Transportní</td></tr>
<tr><td>Síťová</td></tr>
<tr><td>Linková</td></tr>
<tr><td>Fyzická</td></tr>
</table>

Sedmivrstvý model ISO-OSI



Architektura Windows NT

A2) Identifikace souběžnosti

- v reálném světě objekty souběžné, při implementaci omezení počtem procesorů, resp.procesů → určení současně aktivních objektů
- souběžné podsystémy nemusí být nutně implementovány na samostatných procesorech (přerušení, řízení úloh, procesů, ...)
- vlákno řízení (thread) - zahrnuje vzájemně závislé objekty, jsou implementována jako úlohy
- v UML diagramy aktivit

A3) Alokace podsystémů procesorům a úlohám

- odhad požadavků na HW zdroje
- HW - SW kompromisy
- alokace úloh procesorům
- určení fyzického propojení
- v UML diagramy nasazení (viz dále)

A4) Volba způsobu správy perzistentních dat

- využití správy souborů OS - laciné, přídavný kód v aplikacích, použití: komplikované struktury, archívy, dočasné ukládání
- využití systému řízení báze dat (SŘBD) – použití: souběžný přístup řady uživatelů, efektivní správa příkazy SŘBD, přenositelnost, rozšiřitelnost

A5) Volba implementace řízení v programu

- varianty externího řízení:
- sekvenční řízení kódem - vstupy jen na základě naprogramovaného vyžádání, nelze implementovat asynchronní vstupy
- sekvenční řízení událostmi - dispečer událostí (monitor) jazyka, operačního systému nebo podsystému, pružné rozhraní

A7) Ošetření mezních podmínek

- inicializace, ukončení, chybové stavy

A8) Stanovení priorit kompromisů

B) Objektový návrh

B1) Specifikace všech tříd a jejich interakce

- v UML diagram tříd pro specifikační pohled, diagramy interakce, stavový diagram

B2) Návrh algoritmů pro implementaci složitých operací

B3) Optimalizace přístupu k datům

- přidání redundantních asociací, uložení odvozených atributů

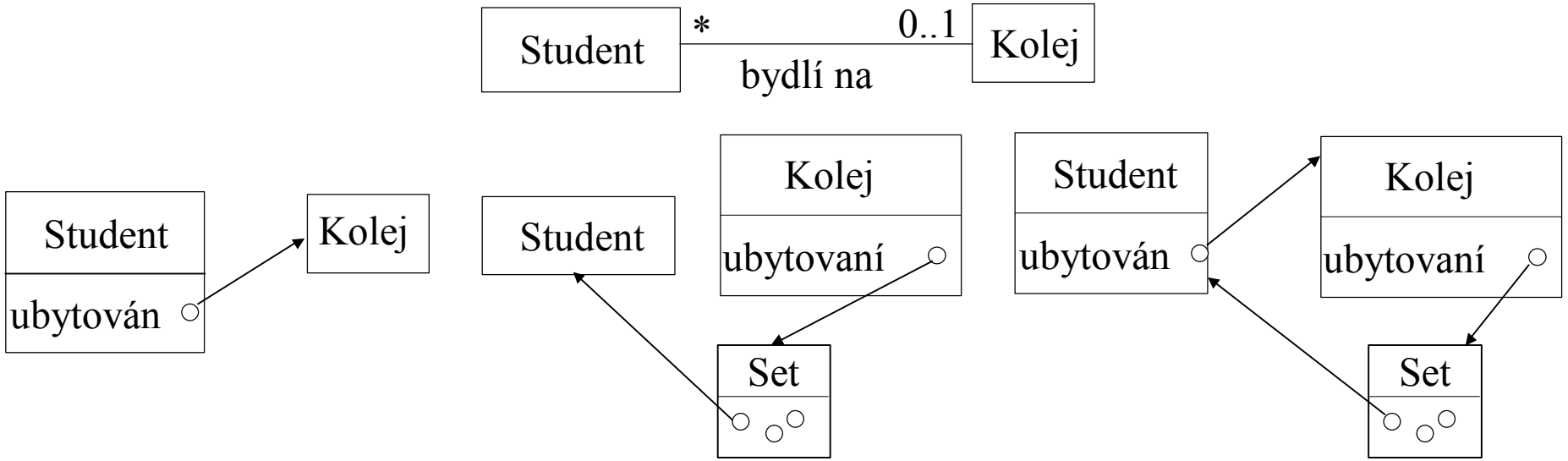
B4) Implementace řízení vnějších interakcí

B5) Přizpůsobení struktury tříd pro zvýšení dědičnosti

- restrukturalizace, abstrakce společného chování

B6) Návrh implementace asociací

Př)



B7) Určení reprezentace objektů

- datové typy pro atributy, samostatné třídy (úsečka, bod)

- v UML diagram tříd pro implementační pohled

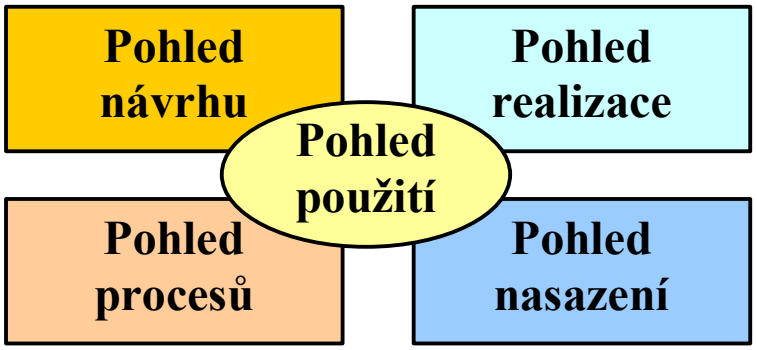
B8) Vytvoření modulů **(pokud nebyly vytvořeny dříve)**

- soubor (C, C++) nebo konstrukce jazyka

- ukrývání informace, koheze

- v UML diagram modulů a komponent

- **Modelování architektury systému v UML**



Pohled	Statické aspekty	Dynamické aspekty
použití	diagramy použití	diagramy interakce, stavové diagramy, diagramy aktivit
návrhu	diagramy tříd, diagramy objektů	diagramy interakce, stavové diagramy, diagramy aktivit
procesů	diagramy tříd, diagramy objektů	diagramy interakce, stavové diagramy, diagramy aktivit
realizace	diagramy komponent	diagramy interakce, stavové diagramy, diagramy aktivit
nasazení	diagramy rozmístění	diagramy interakce, stavové diagramy, diagramy aktivit

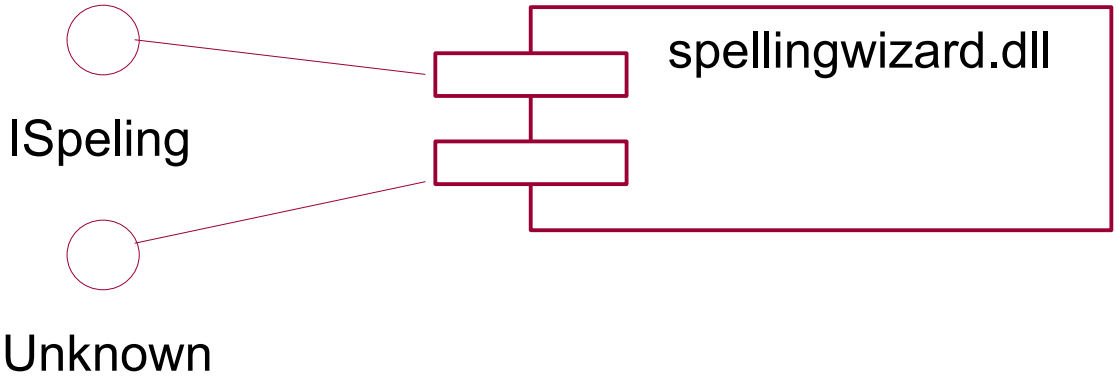
◊ Diagramy komponent

Komponenta **– fyzická, vyměnitelná část systému, která souhlasí s a realizuje určitou množinu *rozhraní*.**

- všechny nejznámější prostředky pro systémy založené na komponentách (COM+, CORBA, Enterprise Java Beans – EJB) používají rozhraní jako pojítko komponent

- komponenta a modul (package) jsou často totožné, ale mohou být různé (např. třída *string* v modulu java.lang)

Př)



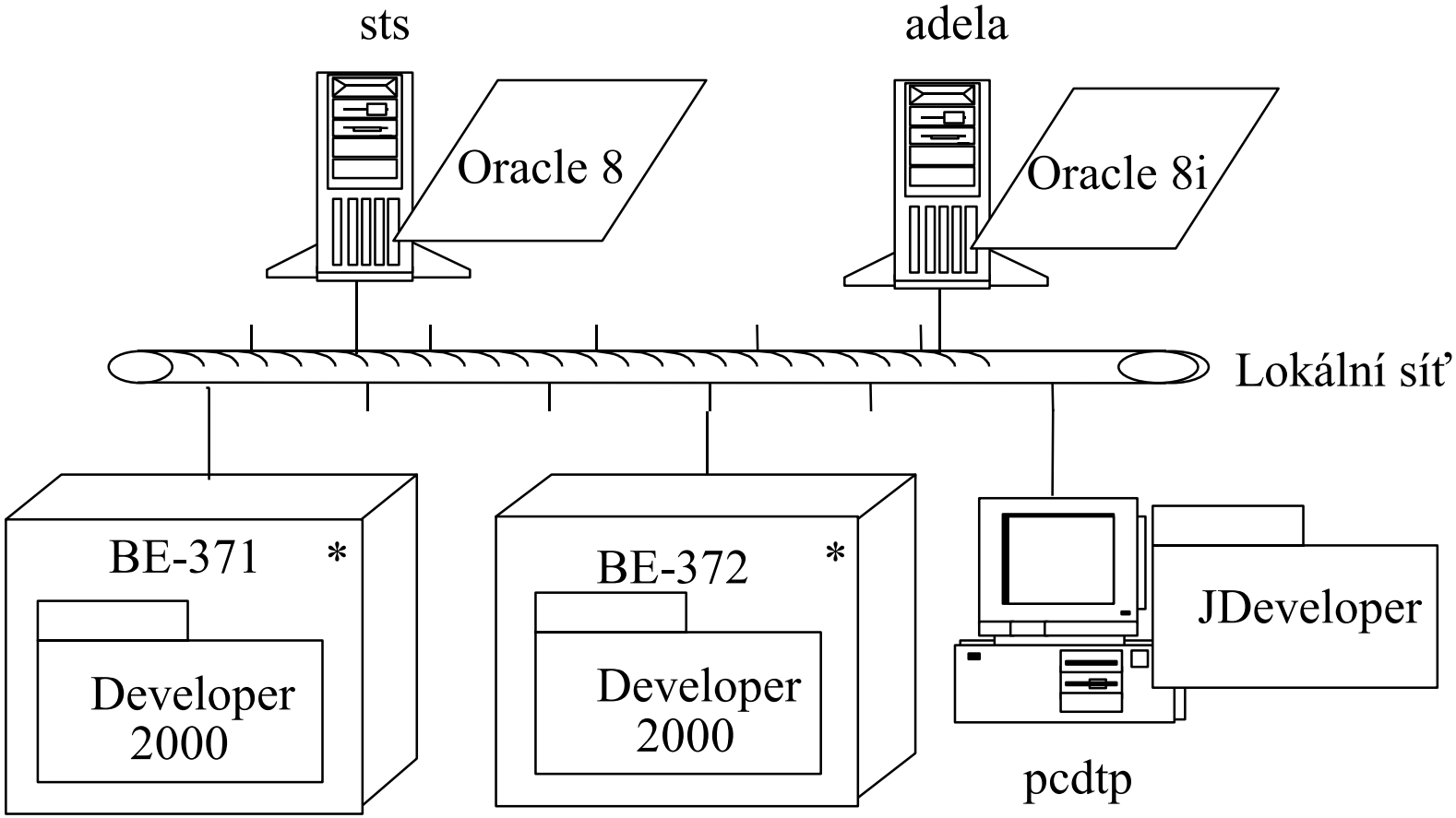
- diagram komponent ukazuje především rozhraní a komponenty a vztahy mezi nimi (závislost, realizace atd.)

- existují předdefinované stereotypy <<executable>>, <<library>>, <<document>>…

◊ **Diagramy rozmístění (deployment)**

***Uzel* – fyzický prvek existující v době výpočtu,který reprezentuje výpočetní zdroj.**

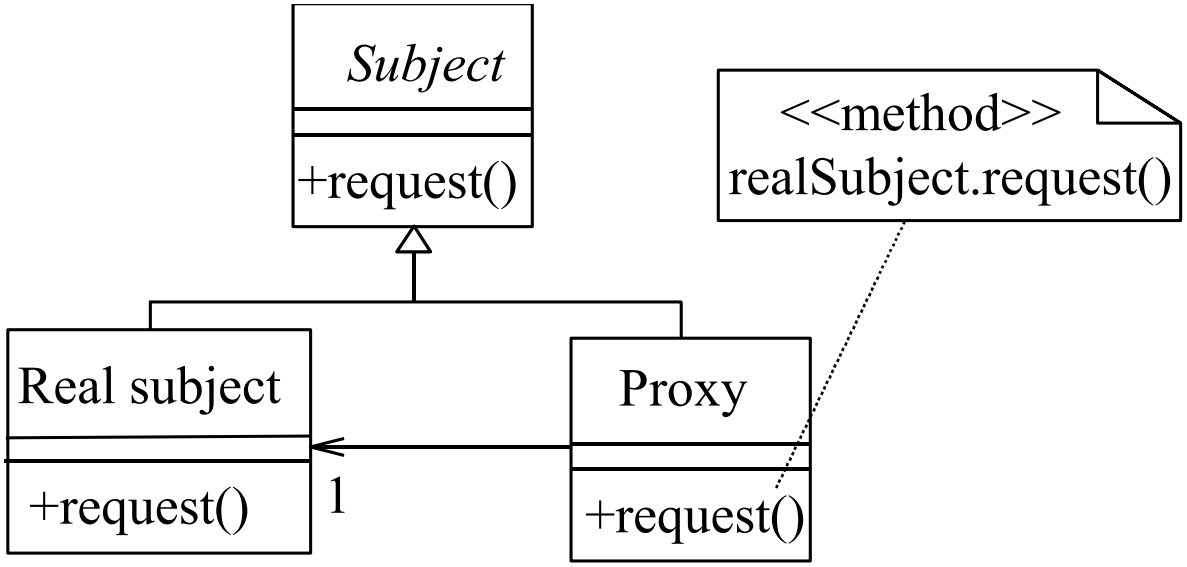
Diagram rozmístění – ukazuje spojení uzlů a rozmístění komponent v uzlech.



- **Návrhové vzory**

  - vzorová řešení opakujících se témat

  Př) komunikace objektů v jednom procesu s objekty jiném (možná vzdáleném) bez nutnosti hledat objekty v síti



  - bližší informace např. http://www.hillside.net/patterns