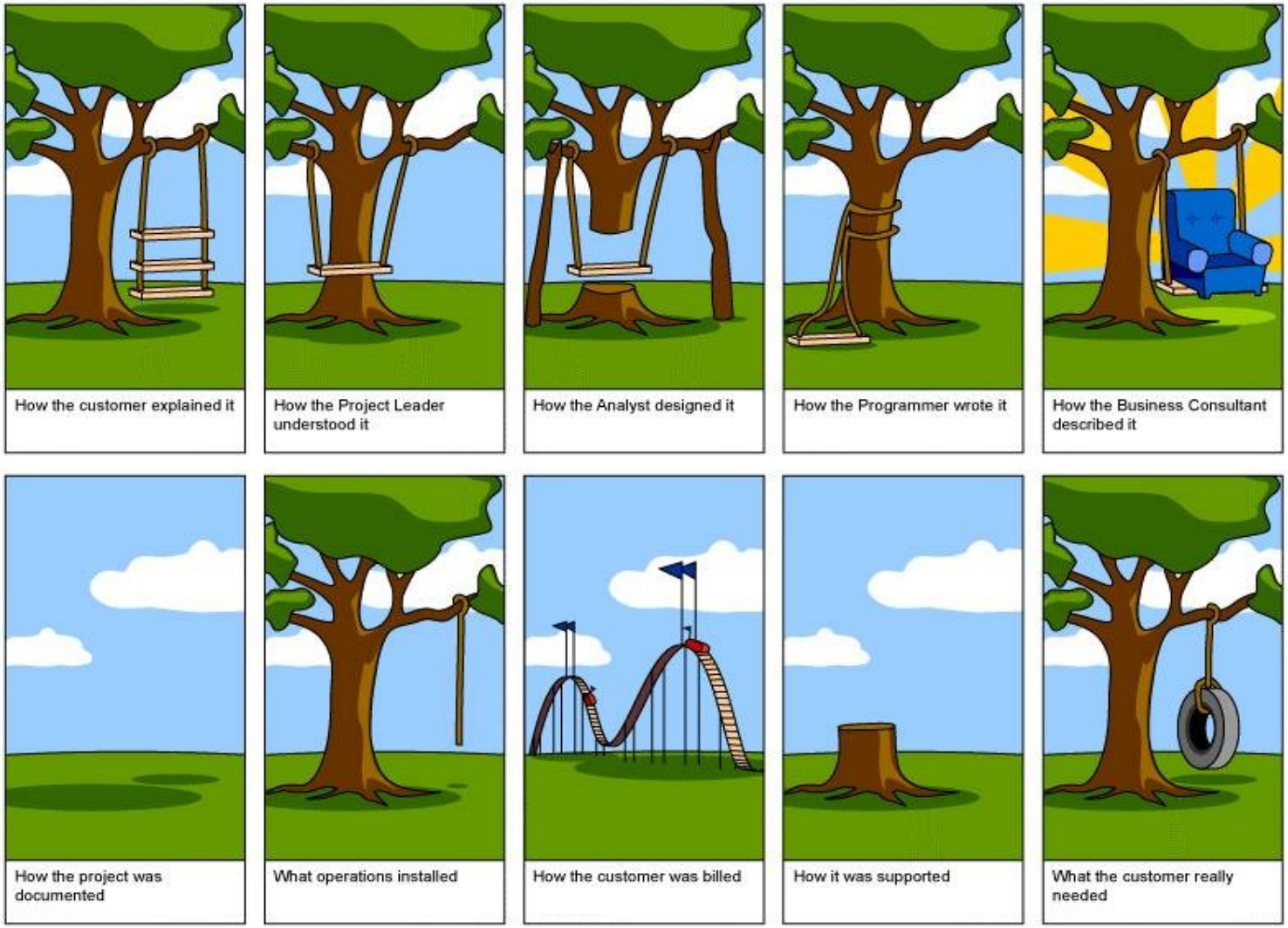
Objektové metody návrhu IS



SA vs. OOA

aneb

Opakování matka moudrosti

© 2008 Radek Ošlejšek
FI MU, Brno

oslejsek@fi.muni.cz
http://www.fi.muni.cz/~oslejsek/PA103

Struktura výuky

1. Úvod, strukturní vs. objektově orientovaná analýza, objektové principy
2.-5. UML, OCL
6.-8. Objektové metodiky, (R)UP
9.-11. Návrhové vzory, analytické vzory, refaktorizace, metriky ...

Doporučený předmět:
PV167: Projekt z objektového návrhu informačních systémů.

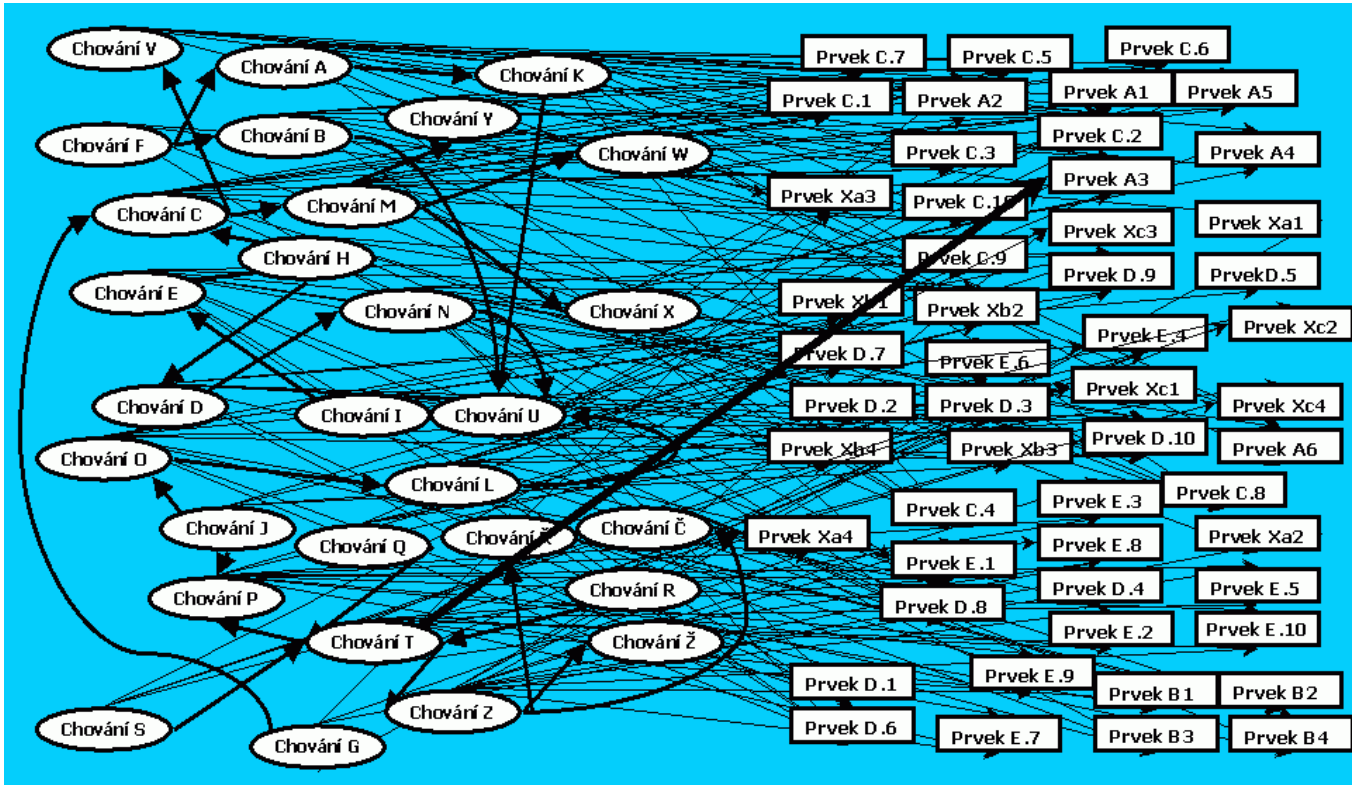
Zkouška:
Písemná, 90 minut. Tři otázky teoretické, jedna praktická. Hodnocení:

A: 40-34 B: 33-29 C: 28–24 D: 23-20 E: 19-16 F: 15-0

Průzkum:
Kdo absolvoval ANANAS?
Kdo umí programovat v nějakém *objektovém* jazyce?

Proč modelujeme IS

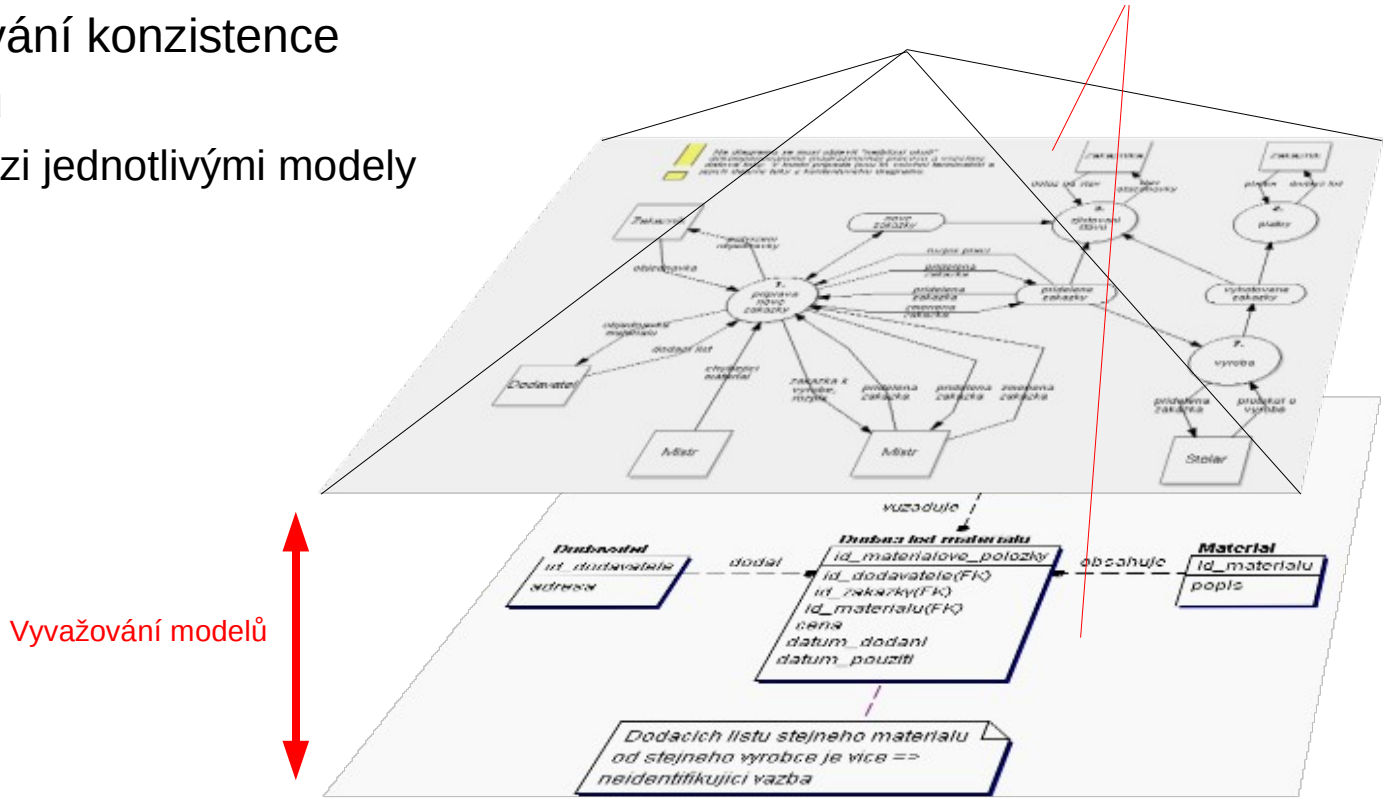
- Informační systémy jsou tvořeny **daty** a **operacemi**, které data zpracovávají a prezentují uživatelům
- Mnoho vazeb => složitý systém nelze zvládnout jako jeden celek
- Modelování = zvládnutí složitosti pomocí principu „rozděl a panuj“



Převzato z : objekty.vse.cz/Objekty

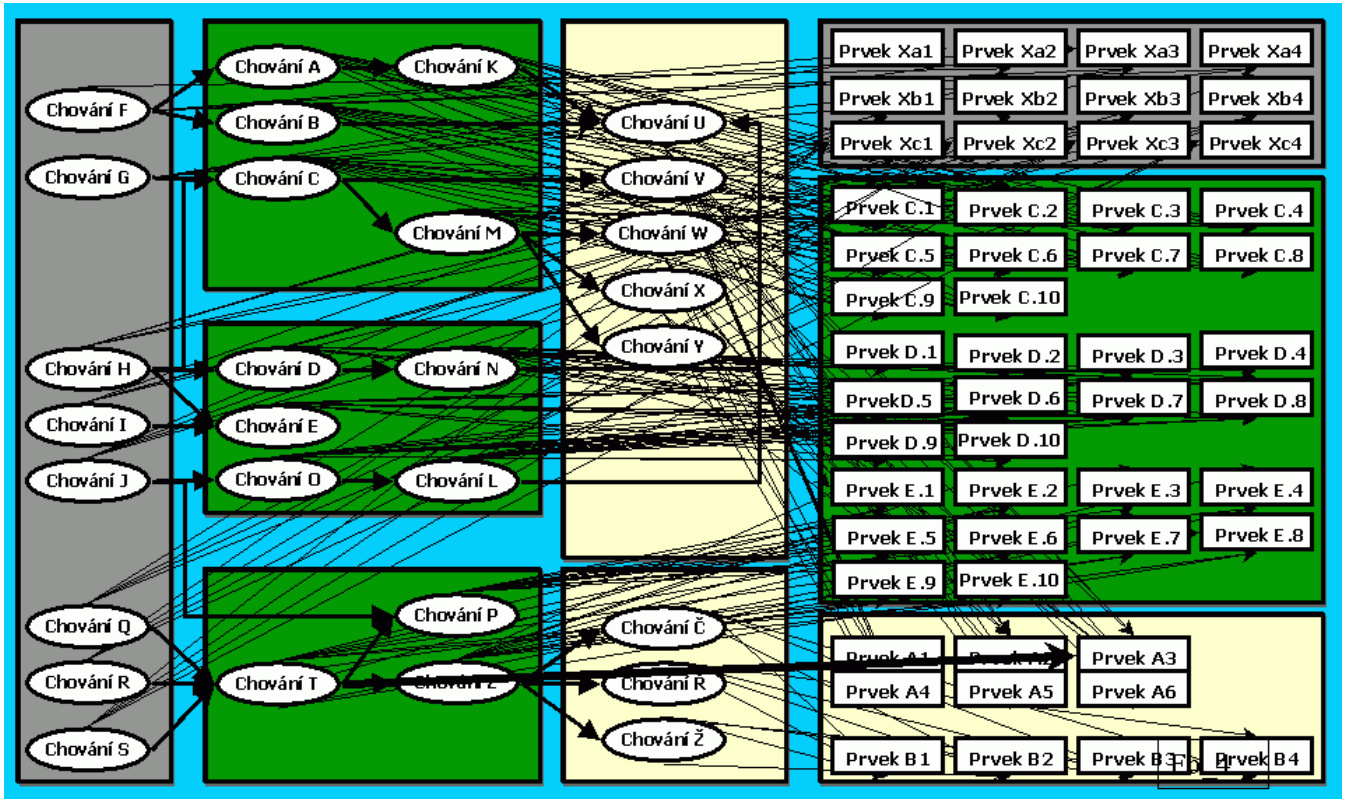
Strukturované modelování

- Oddělené funkční a datové modely
  - Kontextový diagram, DFD, události, ...
  - ERD, datový slovník, ...
- Postupné zpřesňování modelů
- Snaha o zachování konzistence
  - uvnitř modelu
  - vzájemně mezi jednotlivými modely



Strukturované modelování (II)

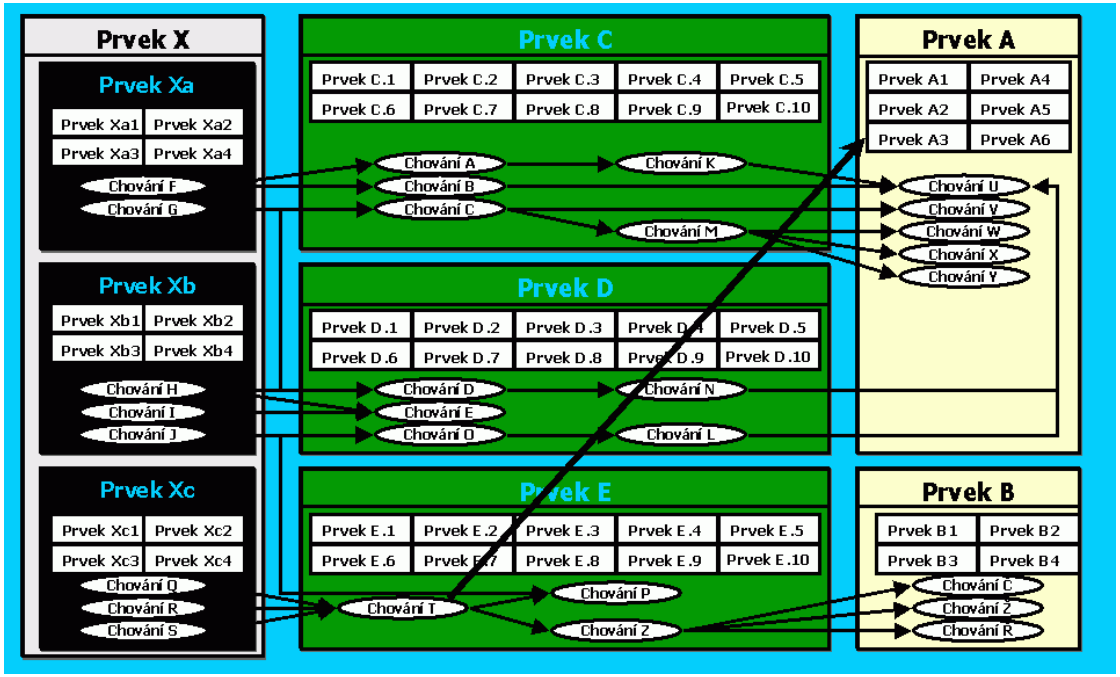
- Uspořádáním funkcí hierarchicky (funkční model) a dat (datový model)
- Usnadňuje orientaci ve funkčním a datovém modelu
- Stále velká složitost vztahů zejména mezi funkčním a datovým modelem



Převzato z : objekty.vse.cz/Objekty

Objektové modelování

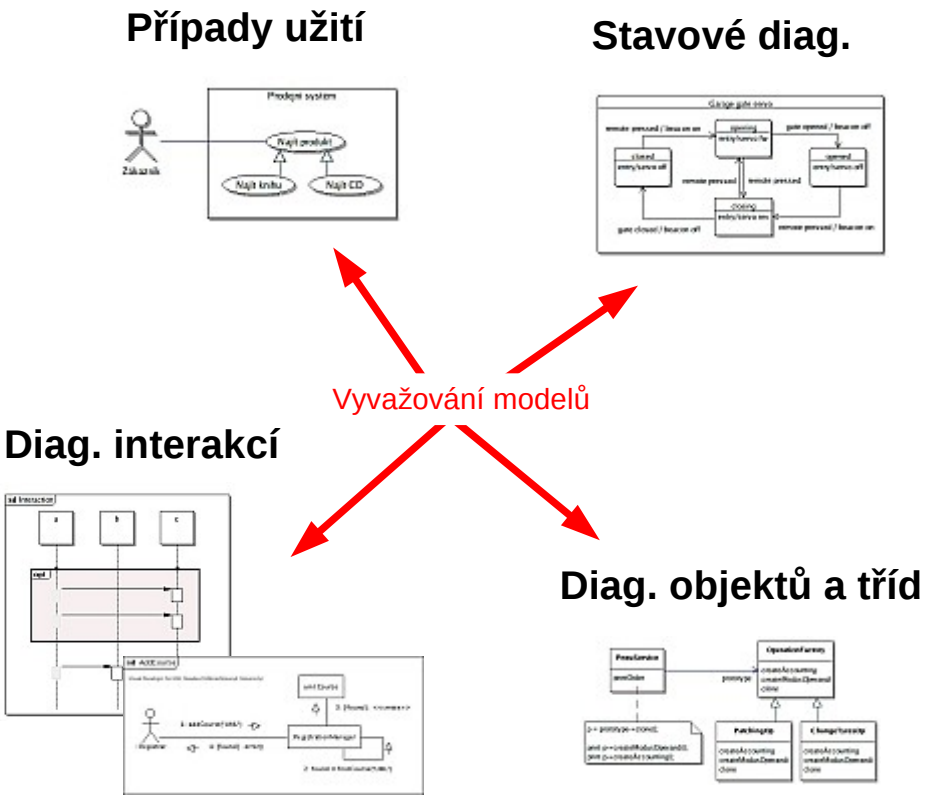
- Rozdělení systému a elementy (objekty), které v sobě zahrnují jak data, tak operace => závislosti mezi souvisejícími funkcemi a daty jsou vnitřní záležitostí objektů
- Vztahy mezi objekty jsou mnohem jednodušší a jednoznačnější
- Uspořádání objektů do hierarchií ještě více zpřehledňuje systém
- OO modelování = rozdělení dat a funkci do objektů a hierarchií



Převzato z : objekty.vse.cz/Objekty

Objektové modelování (II)

- Více modelů než u SA
- Ne všechny modely se ale vždy používají a ne ve všech etapách nebo všech částech systému
- Postupné zpřesňování modelů
- Snaha o zachování konzistence
  - uvnitř modelu
  - vzájemně mezi jednotlivými modely
- Hlavní cílový digram je diagram tříd
- Ostatní diagramy slouží převážně na nalezení správného diagramu tříd
- Inkrementální vývoj



Objektové paradigma

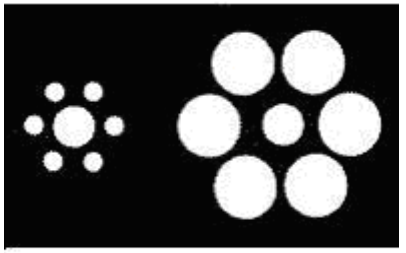
OO koncepty: krátká historie

- používají se od 40.let dvacátého století
- poprvé vyjádřeny při vývoji Simula-67 (Dahl, Nygaard, 1966)
- přijato na základě
  - rostoucí výpočetní síly
  - praktické AI aplikace, které prokázaly užitečnost OO konceptů při reprezentaci znalostí

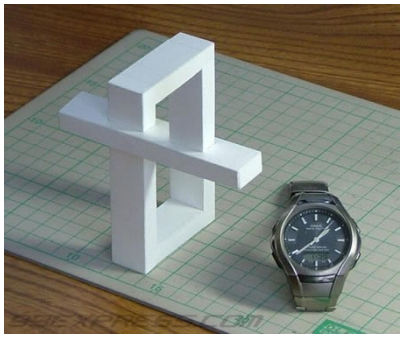
Myšlení a metody organizace

„Při vnímání reálného světa lidé stále používají 3 metody organizace, které prostupují veškerým lidským myšlením, a které nám pomáhají zvládnout složitost světa:

(1) rozlišení poznatků o ***určitých objektech a jejich vlastnostech***, např. když rozlišují mezi stromem a jeho velikostí a prostorovými vztahy k ostatním objektům



(2) rozlišení mezi ***celými objekty*** a jejich ***jednotlivými částmi***, např. když porovnávají strom s jeho jednotlivými větvemi;



(3) vytváření a rozlišení mezi různými ***třídami objektů***, např. když tvoří třídu všech stromů a třídu všech kamenů a rozlišují mezi nimi.“

Klasifikační teorie, ***Encyclopedia Britanica,* 1986**

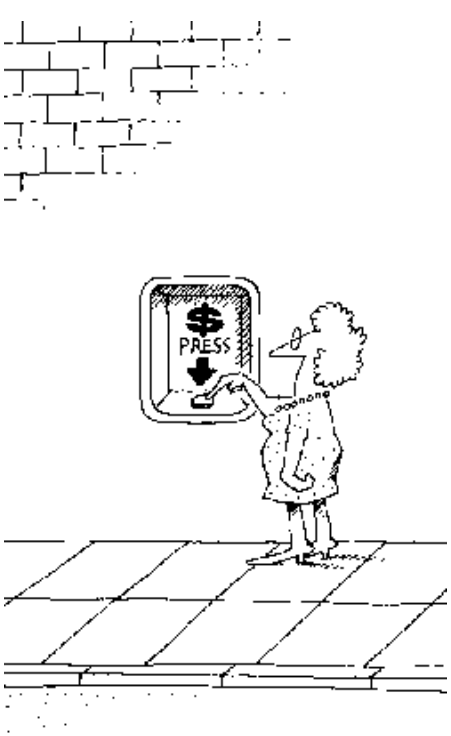
Objektově-orientovaná analýza

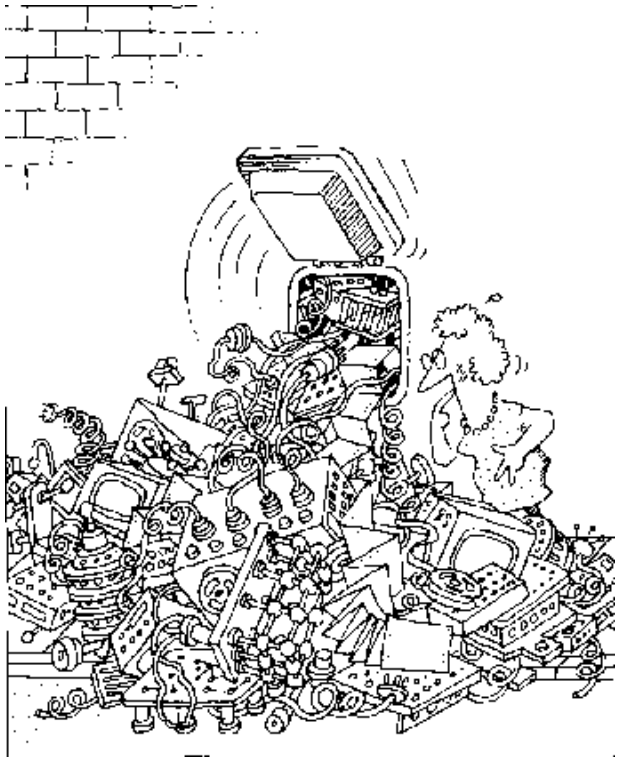
- Modeluje požadavky na systém prostřednictvím objektů a služeb, které poskytují
- Má svůj původ v objektově-orientovaném programování a návrhu
  - oproti strukturovaným přístupům **člení problém jiným způsobem**
  - od strukturované analýzy se špatně přechází k OO návrhu
- Je (tvrdí se to) „přirozenější“
  - během vývoje systému mají funkce (procesy) tendenci se měnit, zatímco objekty mají tendenci zůstat beze změn …

  …a tak model podle strukturované analýzy přestane časem platit, zatímco objektově orientovaný model nezestárne

  …a proto jsou objektově-orientované systémy lépe udržovatelné

Iluze jednoduchosti





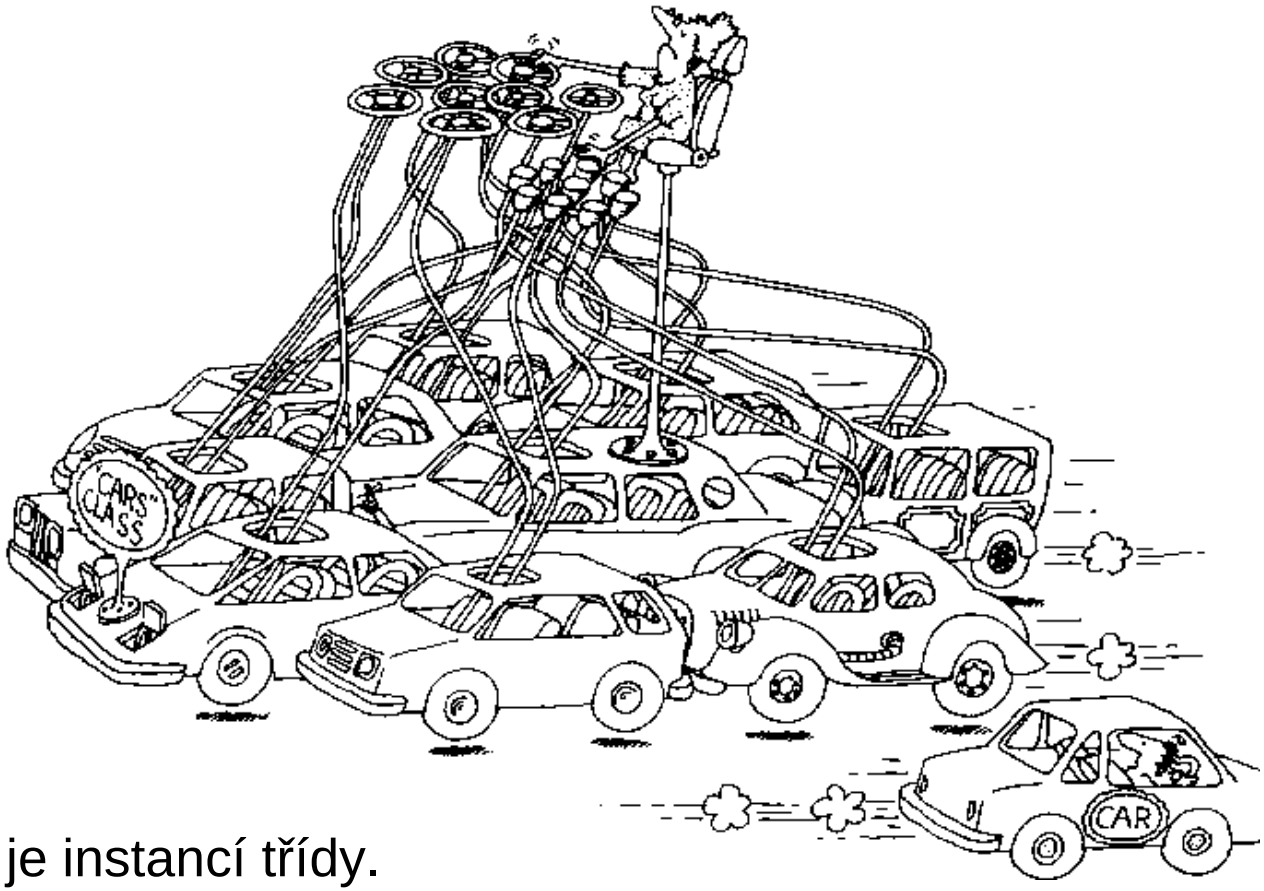
Modelovací primitiva a OO koncepty

- Základní modelovací primitiva
  - Objekty: entity, které mají atributy a služby
  - Třídy: seskupování objektů s podobnými atributy nebo službami
  - Atributy: společně reprezentují stav objektu
  - Metody (služby, funkce): operace, které mohou provádět všechny objekty ve třídě

- OO koncepty
  - zapouzdření = ukrývání (zakrývání) informace
  - abstrakce (procedurální a datová)
  - dědičnost
  - polymorfismus
  - vazba, propojení objektů
  - předávání zpráv

Třída

G.Booch: ***Třída*** je množina objektů, které sdílejí společnou strukturu a shodné chování.



Objekt je instancí třídy.

Objekt: stav, chování, identita

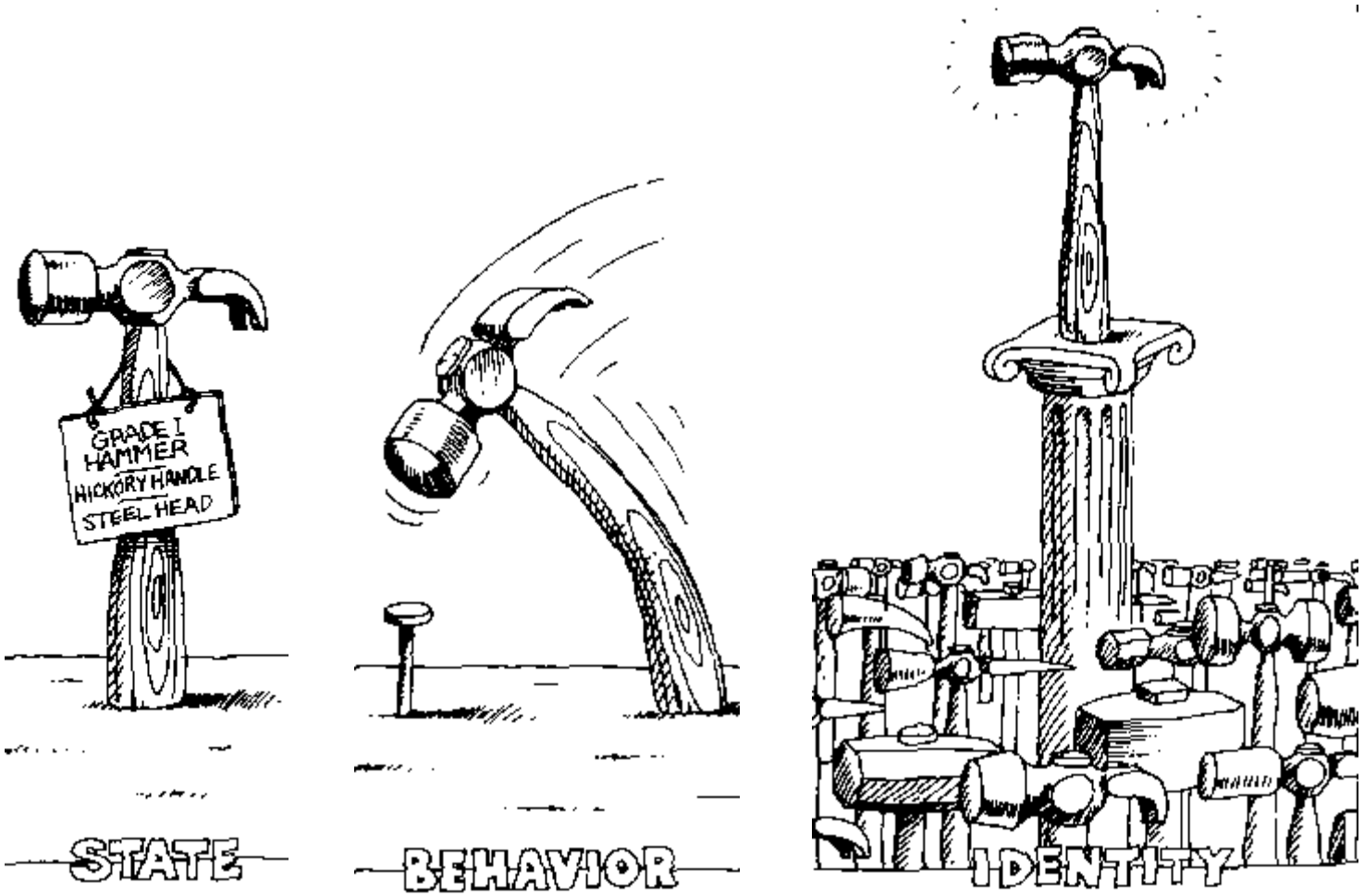
G. Booch: „Object is something you can do things to.“

Objekt má ***stav***, ***chování*** a ***identitu***; struktura a chování podobných objektů jsou definovány v jejich společné ***třídě***, pojmy ***instance*** a ***objekt*** jsou vzájemně zaměnitelné.

Stav objektu zahrnuje všechny (obvykle statické) vlastnosti objektu plus současné (obvykle dynamické) hodnoty těchto vlastností.

Chování vyjadřuje, jak objekt koná a reaguje, v pojmech změn stavu a předávání zpráv.

Objekt: stav, chování, identita



Identita - příklad

- V internetovém obchodě jsou třídy *Product* (nabízené zboží) a *Size* (rozměry zboží – výška, šířka a hloubka v cm)
- Jak mohou být identifikovány produkty?
  - Adresa v paměti
  - Výrobní číslo – dva produkty se stejným výrobním číslem jsou považovány za identický produkt (za stejnou instanci).
- Jak mohou být identifikovány objekty třídy *Size*?
  - Adresa v paměti
  - Rozměry – dva objekty třídy *Size* se stejnými hodnotami výšky, šířky a hloubky jsou považovány za identický objekt

Metody (služby)

Základní služby

- Vytvoření - (konstruktor objektu)
- Propojení - připojení (rozpojení) k jinému objektu
- Přístup - získání a nastavení atributů objektu
- Uvolnění - odpojení a zrušení objektu

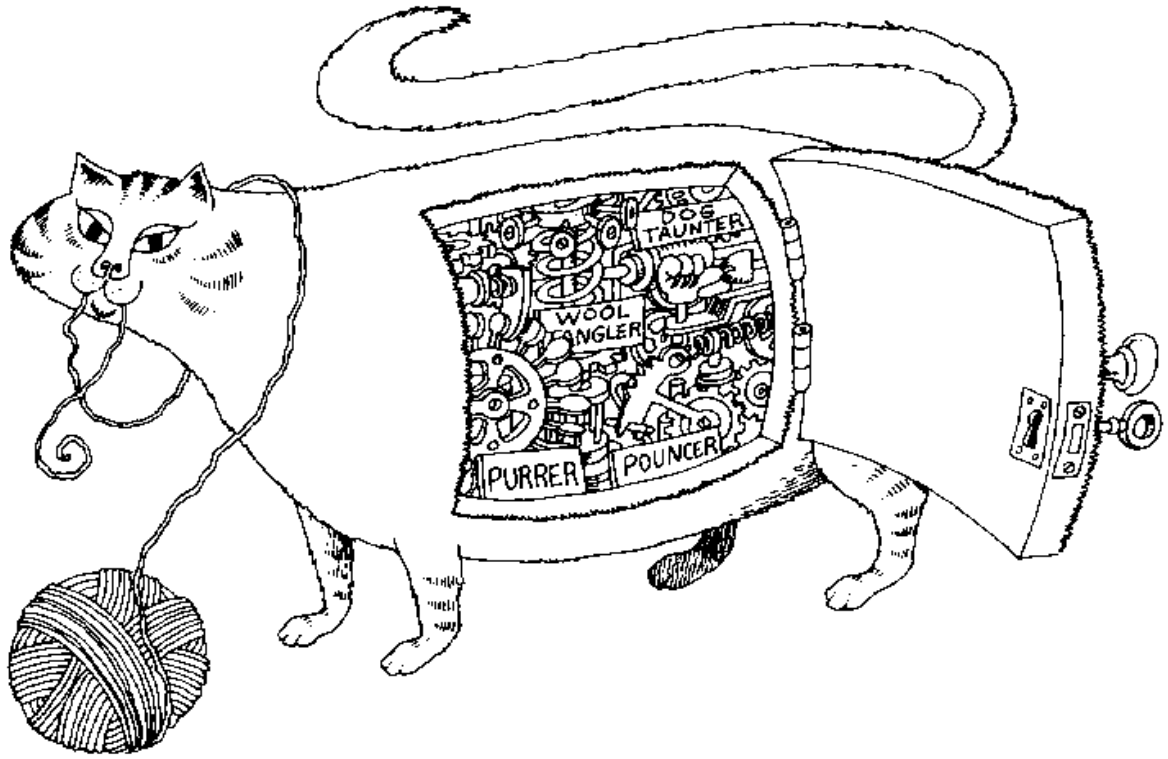
Výpočetní služby

- Zpracování atributů objektu a sestavení výsledku (výsledek se může měnit podle aktuálního stavu objektu)

Monitorovací služby

- Sledování vnějších systémů, zpracování vnějších vstupů (zahájení a ukončení monitorovací činnosti)

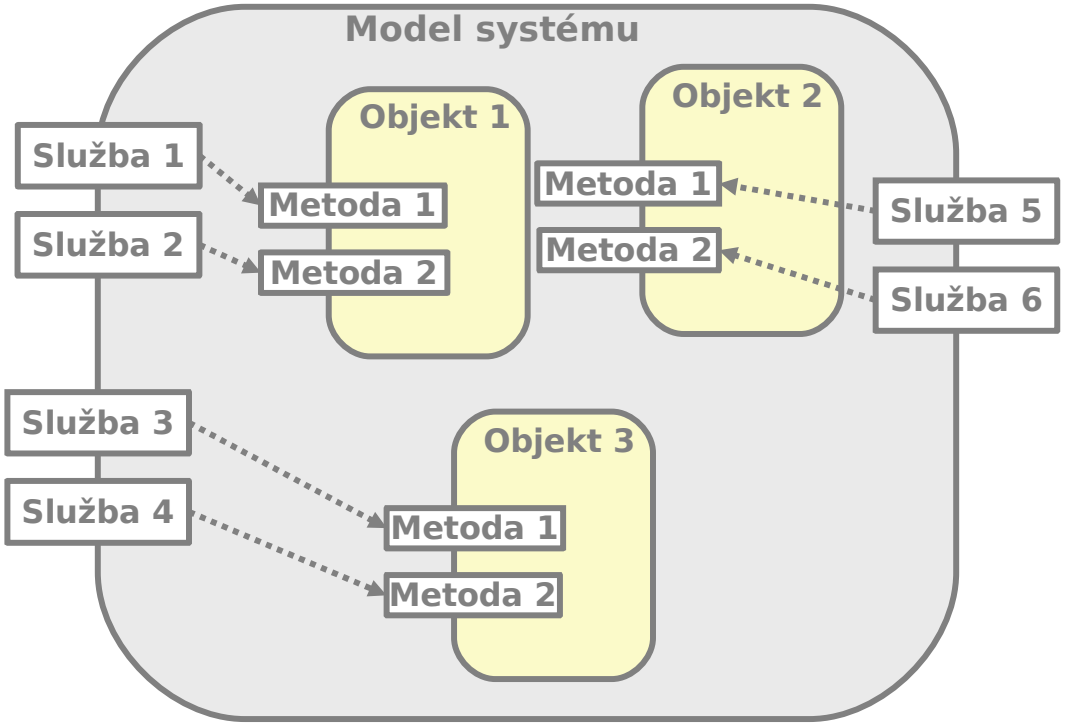
Zapouzdření v OOA



Princip použitý při vývoji celé programové struktury. Spočívá v tom, že každá komponenta programu by měla zapouzdřit a ***ukrýt jediné návrhové rozhodnutí***. Rozhraní každého modulu je definováno tak, aby odhalilo co nejméně o vnitřním dění v modulu.

Zapouzdření

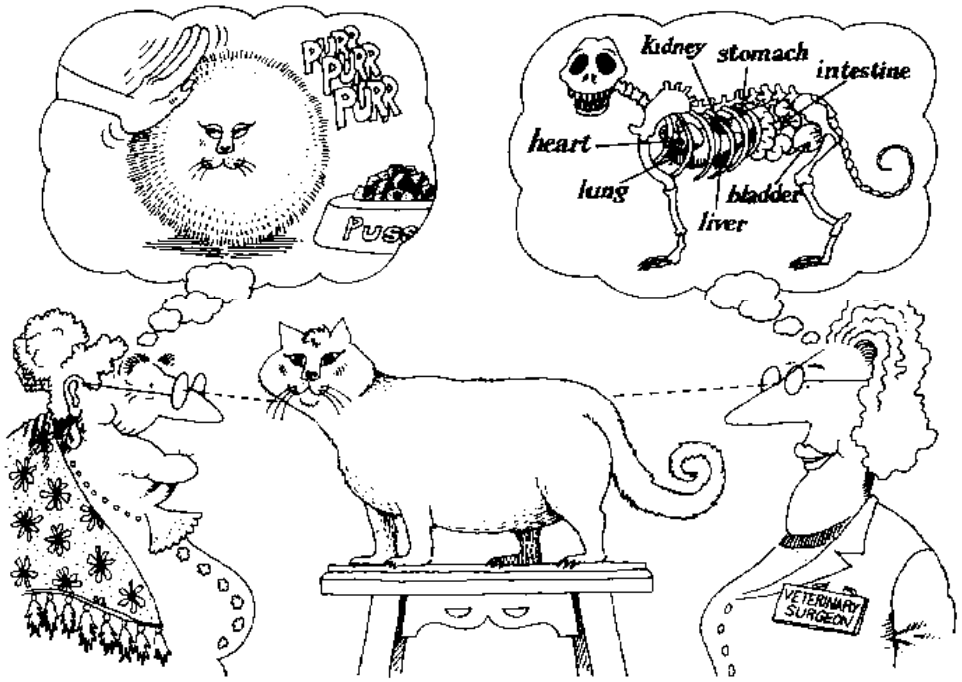
- Rozhraní třídy zachycuje externí pohled
- Implementační detaily skryté před externími entitami
- Objekty mohou obsahovat jiné objekty



Přínosy zapouzdření

- Redukuje šíření postranních efektů, pokud nastane změna
- Seskupení dat a operací podporuje znovupoužití komponenty
- Rozhraní mezi zapouzdřenými objekty jsou zjednodušena (zprávy) - šířka propojení je zúžena

Abstrakce: různé pohledy na objekt



Abstrakce vymezuje podstatné znaky objektu, které jej odlišují od ostatních druhů objektů, a tak poskytuje ostře definované konceptuální hranice ***relativně podle perspektivy pozorovatele***.

Abstrakce

- Obvykle spojena s předchozím principem ukrývání informace
- Označuje esenciální charakteristiky objektu - co dělá z objektu příslušníka třídy?
- Umožňuje, aby návrhář reprezentoval datový objekt na různých úrovních detailu.
- Specifikace datového objektu v kontextu operací, které na něj mohou být aplikovány.
- Principem abstrakce je ignorovat ty aspekty subjektu, které nejsou významné pro současný účel, abychom se mohli více soustředit na ty, které významné jsou.

Abstrakce procedurální a datová

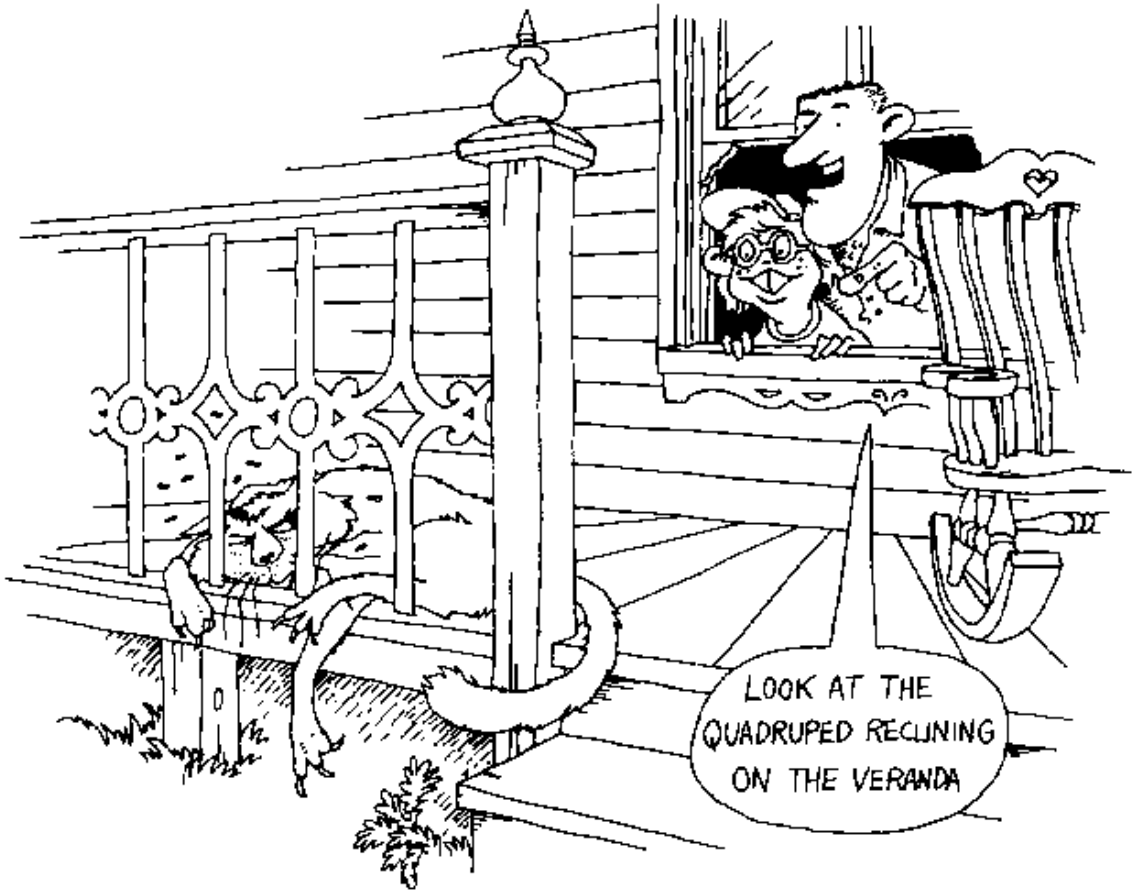
Procedurální abstrakce:

Každá operace, která docílí jasně definovaný výsledek, může být považována za a používána jako jednoduchá entita, i když je ve skutečnosti realizována nějakou sekvencí operací nižší úrovně.

Datová abstrakce:

Princip definice datového typu pomocí operací, které jsou aplikovány na objekty příslušného typu s tím omezením, že hodnoty těchto objektů mohou být modifikovány a pozorovány pouze pomocí operací.

Vhodná úroveň abstrakce



Hierarchie

Hierarchie znamená hodnostní zařazení nebo uspořádání abstrakcí.

2 nejvýznamnější hierarchie

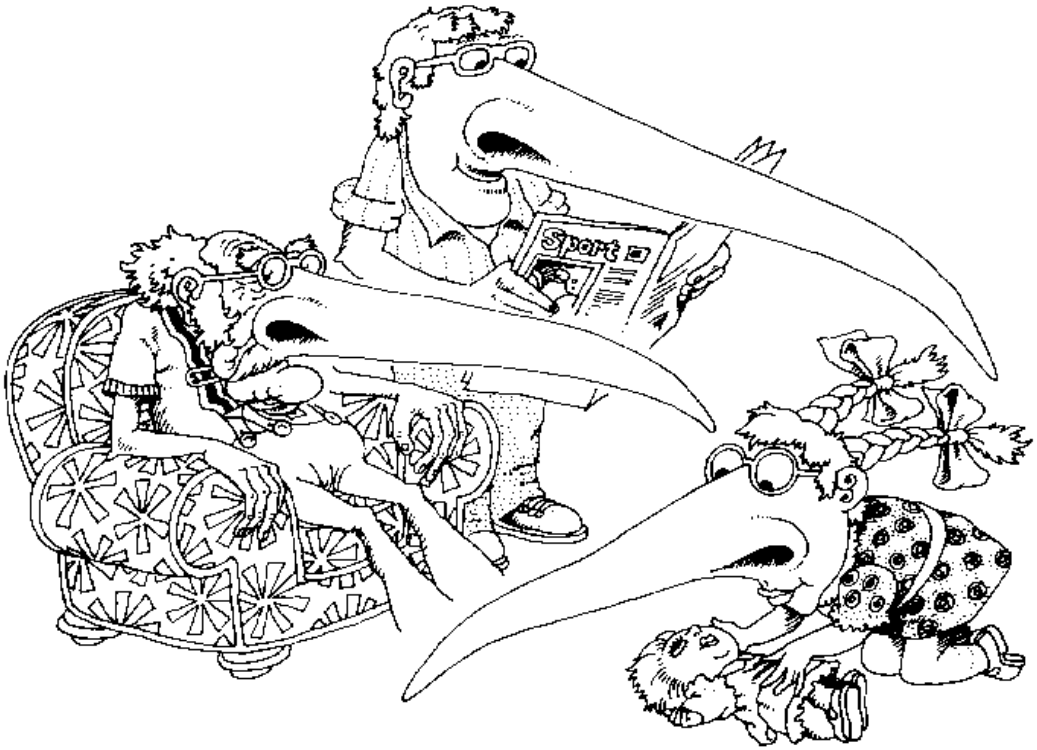
- **„is-a“** hierarchie, „je něčím“
  - *dědičnost*
  (medvěd je savec, „quick sort“ je třídící algoritmus)

- **„part of“** hierarchie, „(sou)část něčeho“
  - *sestava, vnitřní struktura objektu*
  (převodovka je částí auta, vyšetření zraku je součástí komplexního v.)

Dědičnost

- „druh“ hierarchie abstrakcí
- definuje vztah mezi třídami, kde jedna třída (podtřída) sdílí strukturu nebo chování definované v jedné nebo více třídách (nadtřídách):
  - jednoduchá - jedna nadtřída
  - násobná - více než jedna nadtřída
- podtřída může *rozšířit* nebo *změnit* existující strukturu/chování nadtříd(y)

Dědičnost



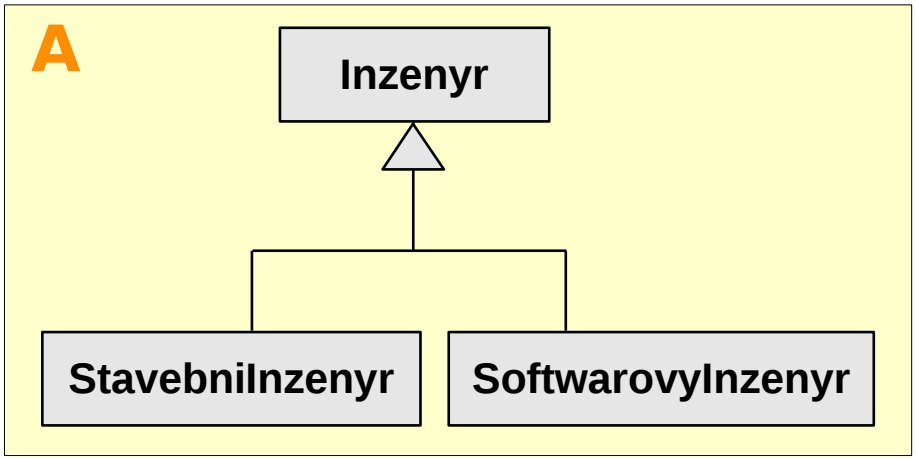
Mechanismus, který vyjadřuje ***podobnost mezi třídami***, zjednodušuje definici *třídy* pomocí dříve definované(ných) třídy(tříd). Vyjadřuje generalizaci a specializaci tím, že v hierarchii tříd explicitně určuje společné *atributy* a *služby.*

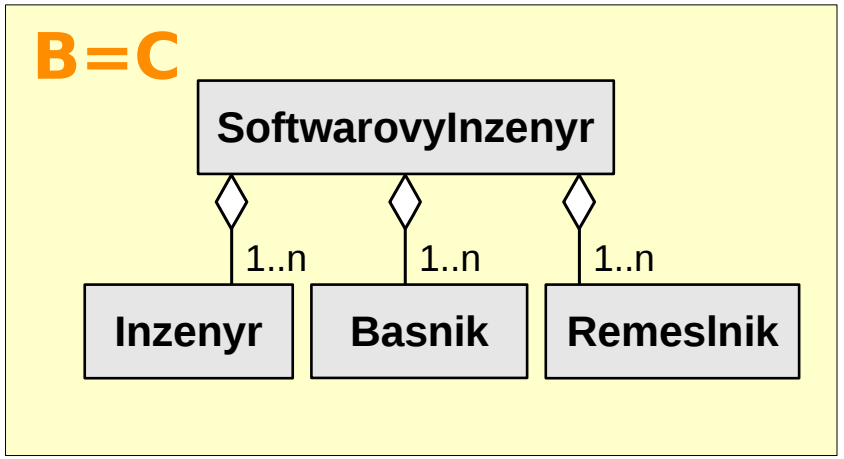
„Would you rather buy or inherit ?“

A. Každý softwarový inženýr je inženýr.

B. V každém softwarovém inženýrovi je inženýr.

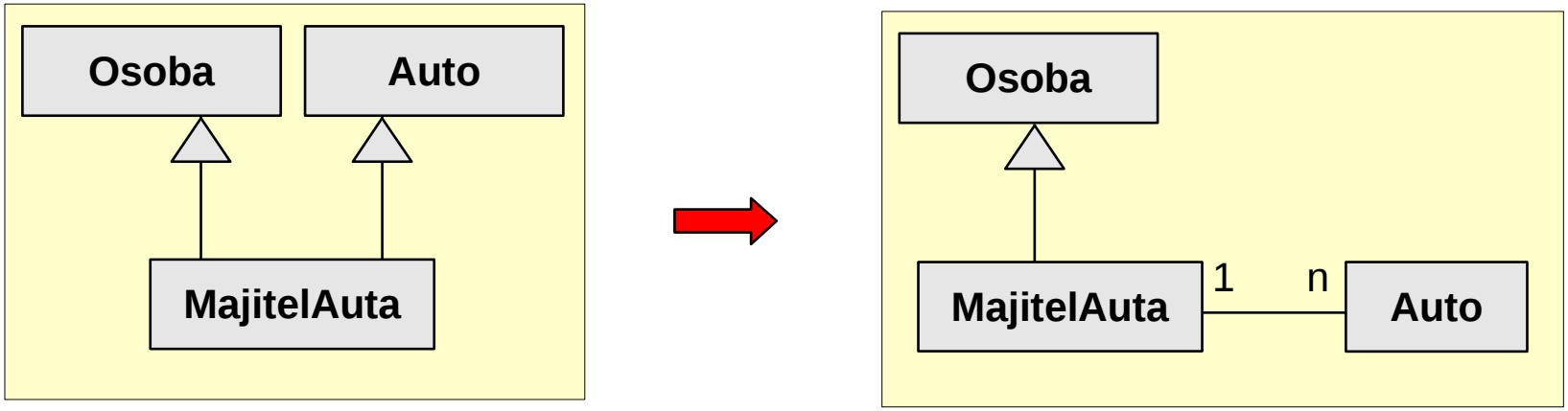
C. Každý softwarový inženýr má inženýrskou komponentu.





Nepoužívejte dědičnost pro popis vnímaného vztahu „je něčím“, pokud odpovídající objektové komponenty se **mohou změnit během provozu** (běhu).

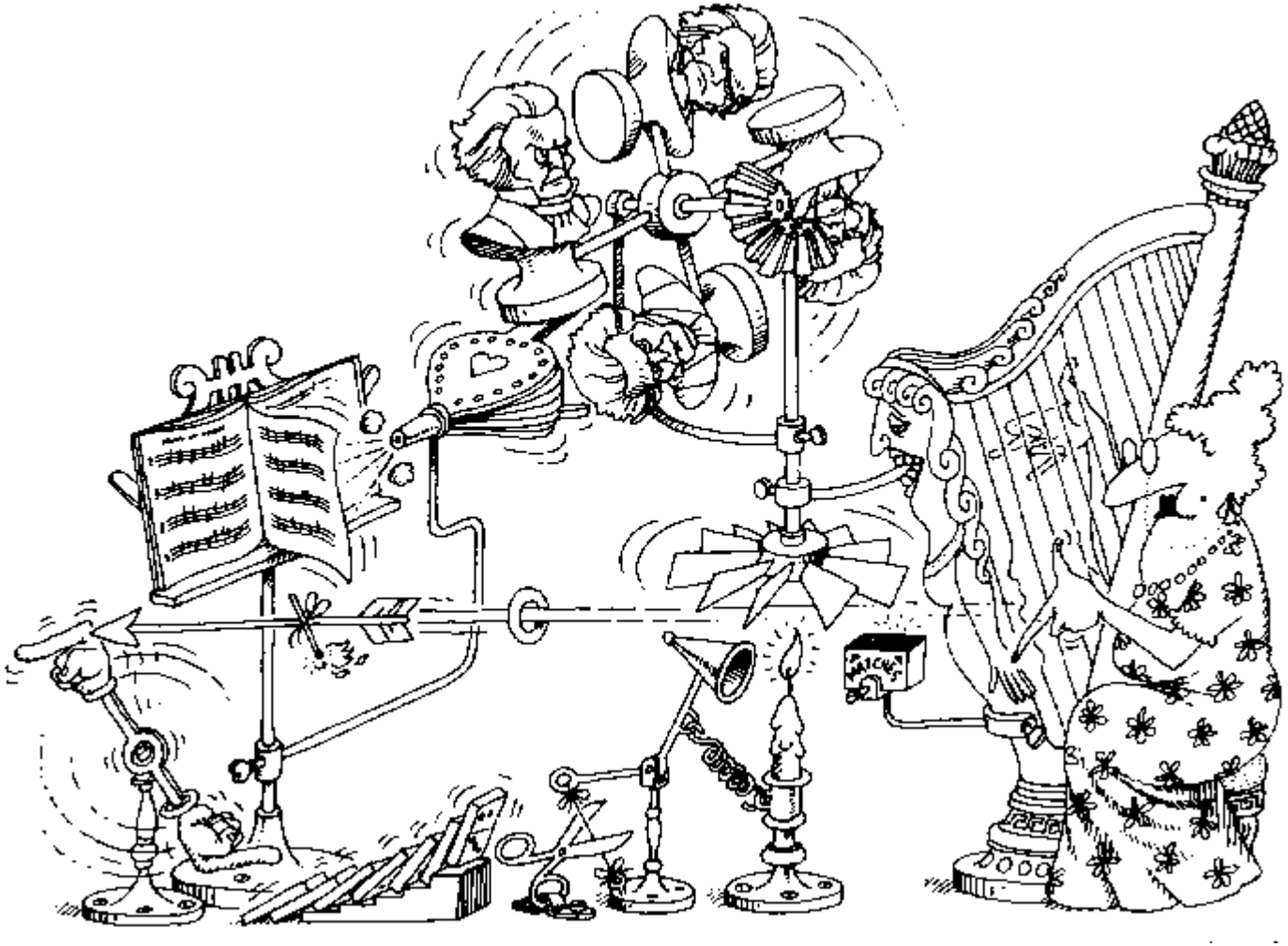
Násobná dědičnost



Pravidlo dědičnosti „Býti něčím“:

Nedovolte třídě *B* dědit od třídy *A*, pokud neprokážete, že na každou instanci *B* můžeme pohlížet také jako na instanci *A.*

Spolupráce objektů, předávání zpráv

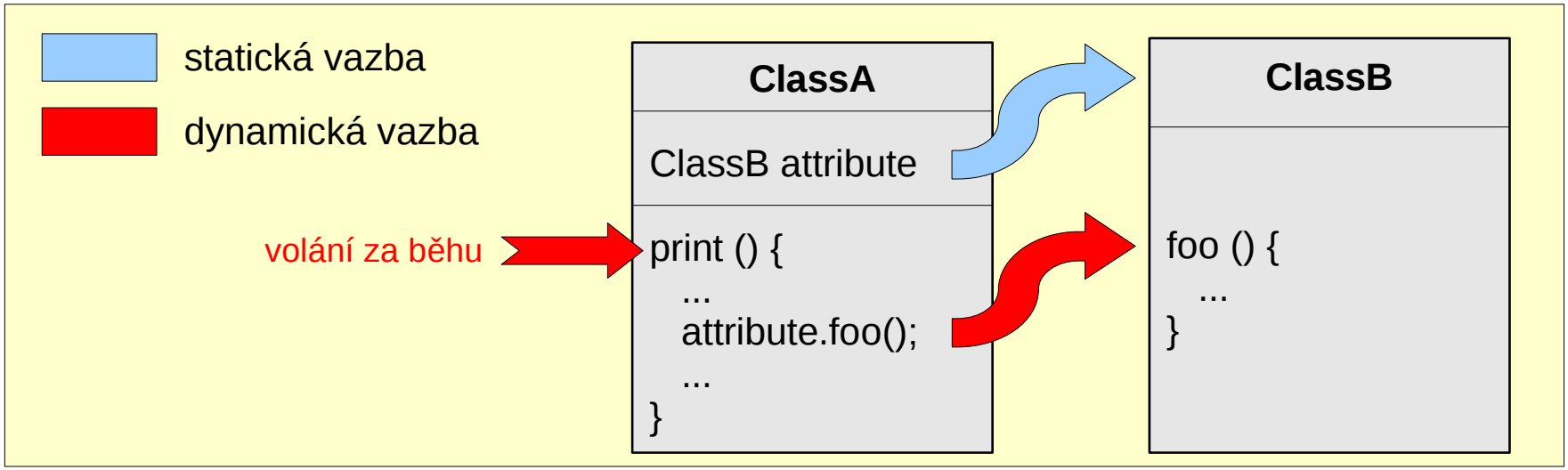


Propojení, vazba mezi objekty

Propojení (vazba) - fyzické nebo konceptuální spojení mezi objekty. Označuje specifickou asociaci, pomocí níž jeden objekt (klient) používá služby jiného objektu (poskytovatele, dodavatele), nebo pomocí níž může jeden objekt navigovat druhý objekt.

Kdy vazba nastane:

- v čase vytváření programu (vnoření) – „skutečně“ statická vazba
- v čase překladu (reference) – statická vazba
- při běhu (volání) – dynamická vazba

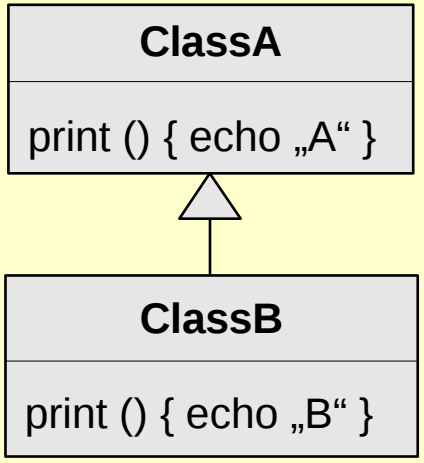


Polymorfismus

- Koncept z teorie typů, kdy jedno jméno může označovat různé věci:
  - „+“ znamená „stejnou věc“ pro real a integer
  - „+“ je implementováno odlišně pro real a integer
- Polymorfismus je důsledkem interakce mezi dědičností a dynamickou vazbou
  - (pod)třída dědí jméno operace
  - vazba implementované metody na toto jméno nastává až při provádění

Co vypíše následující program?

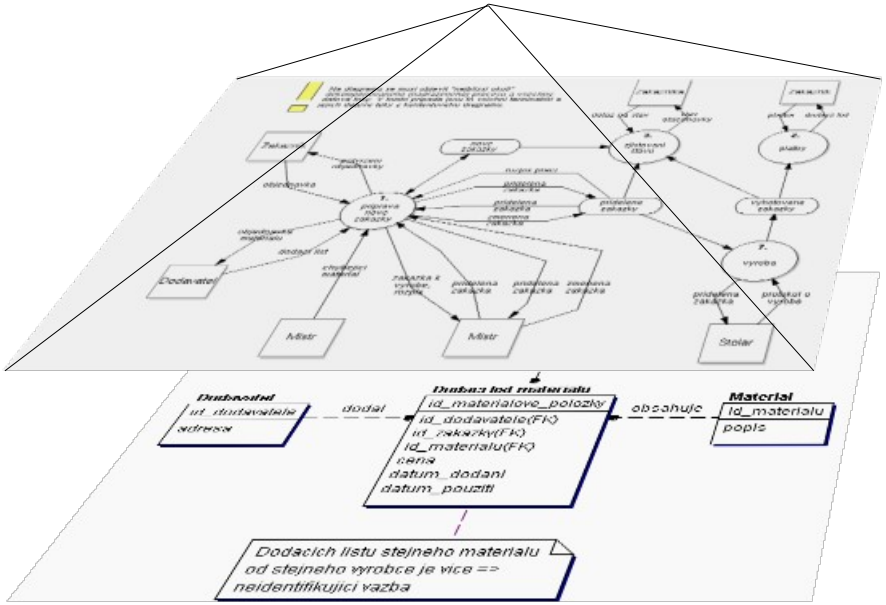
```
ClassA object;
object = new ClassB();
object.print();
```

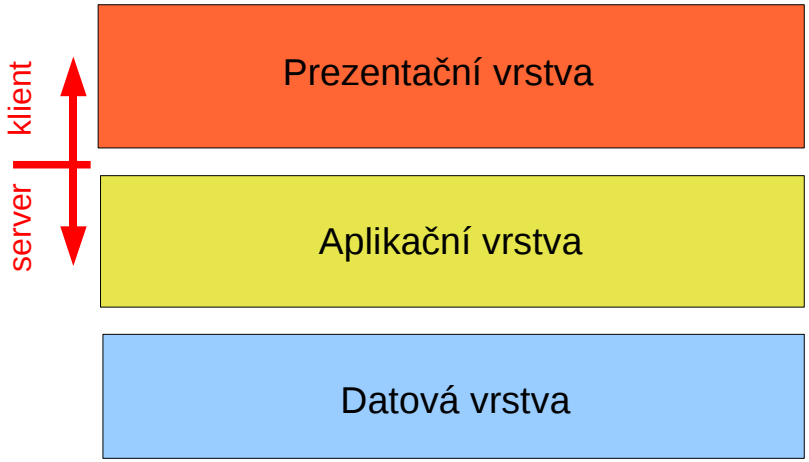


Modely Architektury

Dvouvrstvý/třívrstvý model

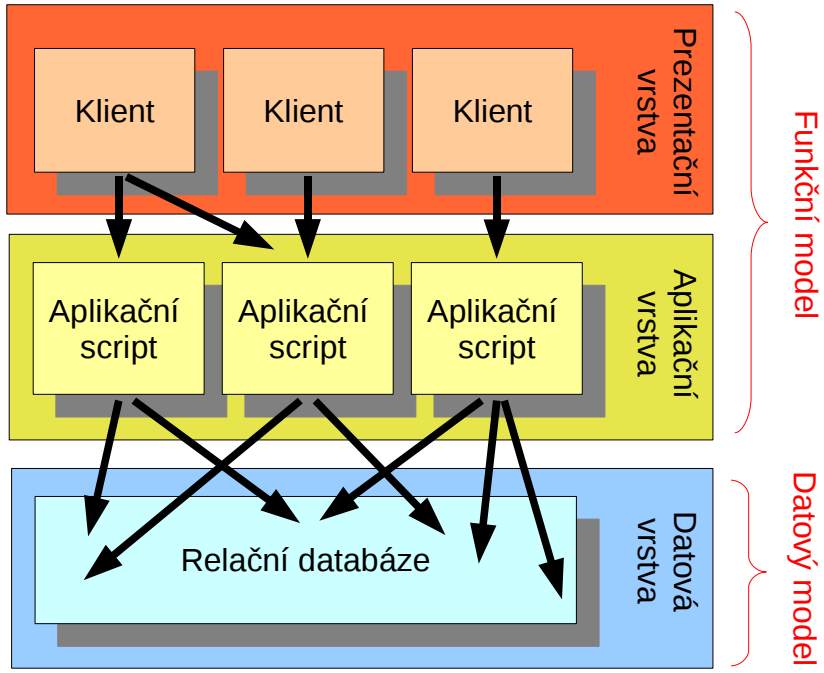
SA = funkční + datový model





Modely architektury: SA

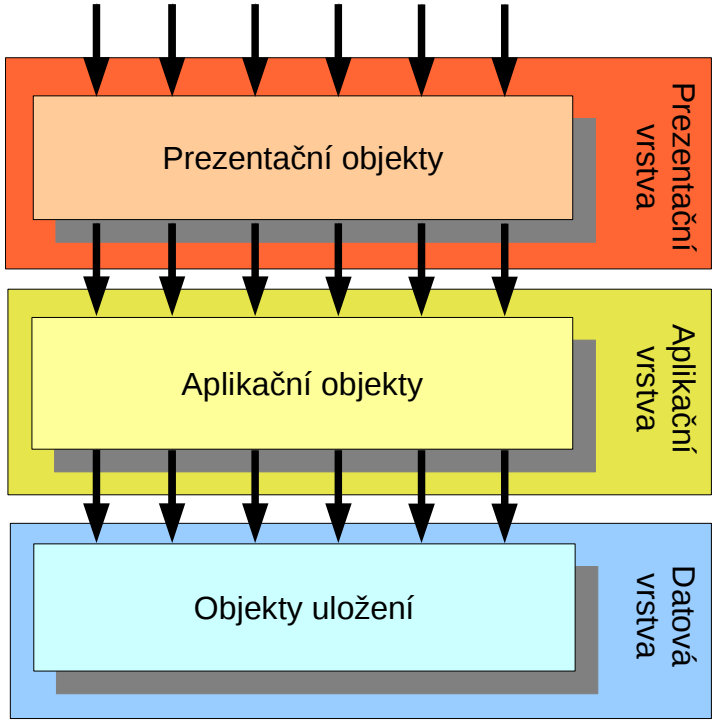
- Typické znaky:
  - Všechny závislosti v DB
  - Komunikace přes DB
  - Každý klient „hraje za sebe“
  - Komplikované SQL dotazy
- Realizace:
  - Formuláře (HTML, XML, CSS, ...)
  - Skriptování (PHP, ASP, ...)
  - Relační databáze
- Použití:
  - Dynamické webové stránky
  - Klient-server aplikace



Provázání přes datovou vrstvu:
+ rychlá implementace
+ použití naučených postupů
+ technologie připravena
- s jednou tabulkou pracují různé scripty
- složitá struktura databáze
- špatná škálovatelnost

OOA: Modely architektury

- Interakce
  - Horní vrstvy vystupují jako klienti vůči spodním vrstvám
  - Spodní vrstvy vystupují vůči horním vrstvám jako servery
  - **Objekty by neměly znát objekty vyšších vrstev**
- Př: Ve které vrstvě verifikovat formulář?
  1) Verifikuje prezentační vrstva, aplikační ho jen uloží
  2) Verifikuje aplikační vrstva, prezentační ho jen zobrazuje



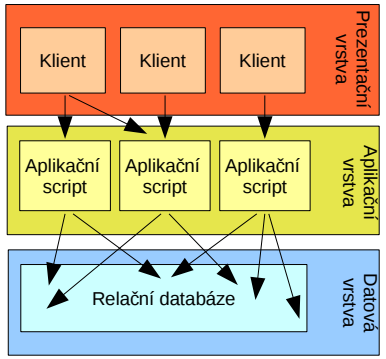
- Odpověď:
  1) Aplikační vrstva spoléhá na to, že dostane platná data => je závislá na horní vrstvě
  2) Správná odpověď :-)

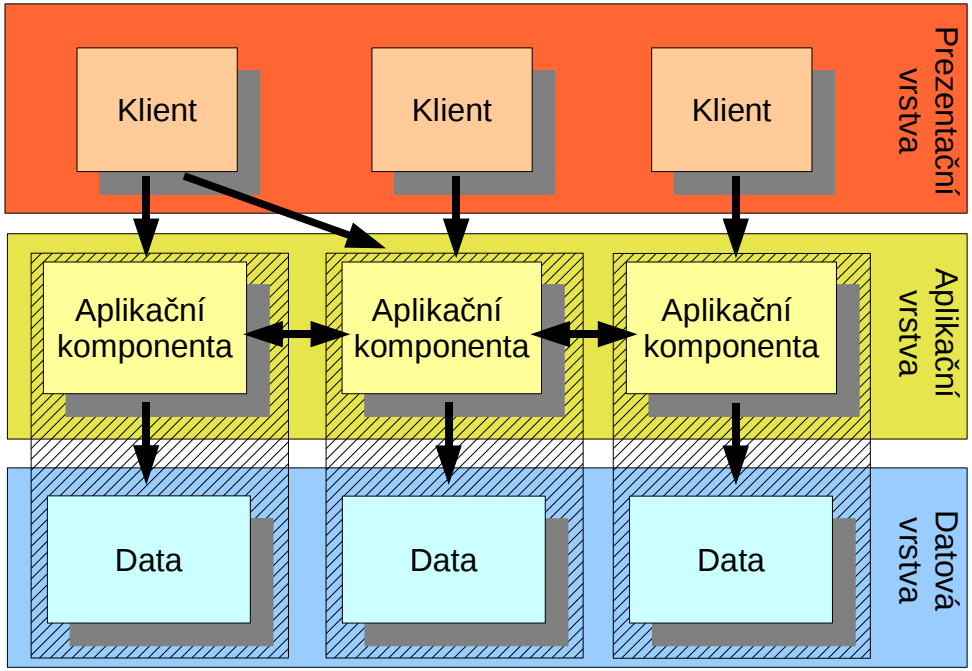
OOA: Modely architektury (II)

- Prezentační objekty
  - Vstup a výstup dat a zpracování
  - Jedna obrazovka nebo dialog (sekvence)
- Aplikační objekty
  - Formulují konceptuální strukturu systému
    - Nezávislost na presentaci
    - Nezávislost na uložení
- Objekty uložení
  - Formulují persistentní vrstvu systému
    - Uložení informací a jejich zpřístupnění
    - Zamykání
    - Integrita

OOA: Modely architektury (III)

- Použití komponentových technologií
  - Provázání přes aplikační vrstvu
  - Udržení složitosti provázaní pomocí komponent
- Typické znaky:
  - Dodržení separace v datech
  - Řízení kontextu v aplikační vrstvě
- Realizace:
  - CORBA, DCOM, SOAP/XML





Provázání přes aplikační vrstvu:
+ robustnost a škálovatelnost
+ snadná údržba a rozšiřování
+ paralelní vývoj
+ snadná integrace s dalšími systémy
- náročná aplikační vrstva
- nové přístupy
- nelze využít *pokročilých* vlastností DB

Realita: Objekty nad relační databází

- Široké nasazení relačních DB v IT
  - Objektové DB sice existují, ale nejsou ještě „zralé“
  - Relační technologie bude zřejmě dominantní i nadále
- Relační DB nepodporují přímo struktury objektového modelu
  - Je nutné provádět mapování objektového modelu na entitně-relační model
  - Způsoby mapování probereme u modelu tříd

Třívrstvý model: příklad

... aneb není objektový model jako objektový model

Prezentační, aplikační i datová logika v jedné třídě

Proč potřebujeme OOA ?

Motivace a přínosy:

1. zvládnutí složitějších problémových oblastí
2. zlepšení interakce mezi analytikem a expertem problémové oblasti
3. zvýšení vnitřní konzistence analytických výsledků
4. explicitní vyjádření společného
5. vytvoření modifikovatelných specifikací
6. opětné použití analytických výsledků
7. konzistentní nosná reprezentace pro analýzu a návrh

Témata diplomových prací

Dostupná nová témata diplomových prací pro semestr Jaro 2009

- Modul do systému Maven
- Modul do systému pro průběžnou integraci
- Systém pro zátěžové testy webových aplikací

Práce jsou vypisovány v rámci aktivit SPP (sdružení průmyslových partnerů) ve spolupráci se společností oXy Online, s.r.o.