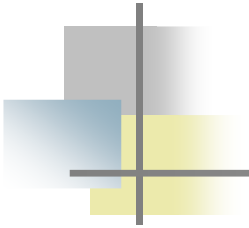
KIV/ASWI 2007/2008

Návrh: třídy a struktury pro efektivní implementaci

Obsah a cíl této části

- Jak rozpracovat výsledky analýzy
  - » tak, aby šly jednoznačně implementovat v konkrétním programovacím jazyce a provozní platformě
  - » tak aby implementaci mohlo provádět více lidí, aby byla efektivní (co do výkonu i tvorby) a výsledek byl snadno udržovatelný
- Dát návod jak poznat rozumný návrh od chybného



Aspekty návrhu

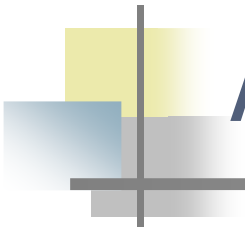
- Postup návrhu tříd
- Architektura, konvence
- Vzory řešení typických situací
- Jazykové idiomy

Klíčová rozhodnutí před detailním návrhem

- (jestli – **vize**)
- (co – **požadavky**)

- **Architektura**
  - technologie
  - struktura
  - pravidla, konvence

- (jak – návrh)

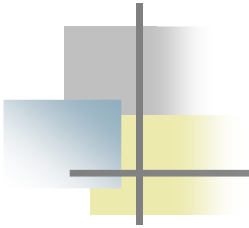


Architektura

Architektura systému

the highest-level concept of a system in its environment

- Klíčové aspekty organizace
  - » jak systém členit
  - hrubé členění na subsystémy a balíky
  - rozhraní a závislosti mezi nimi
  - přiřazení tříd do balíků/subsystémů
- Aspekty prolínající celou implementací (konvence)
  - » jak systém navrhovat
  - jednotný přístup k tvorbě objektů
  - lokalizace

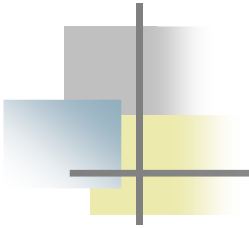


Výběr technologie

- Postup tvorby návrhu i jeho konečnou podobu ovlivňují/omezují vlastnosti prostředí

- Programovací jazyk, databáze, knihovny
- Použití frameworků a hotových komponent

- Ovlivňující faktory
  - stávající systém
  - efektivita vývoje
  - marketing



Logické členění – balíky

- Motivace: logický objektový model velký
- Členění pro získání
  - přehledu o systému
  - rozdělení implementace mezi členy týmu
- Co je balík
  - skupina souvisejících tříd, tvoří organizační celek
  - mapování do jazyka (balík vytváří jmenný prostor)
  - hierarchické vnořování
- Třídy balíku
  - funkčně příbuzné, v jedné vrstvě aplikace nebo kdekoli

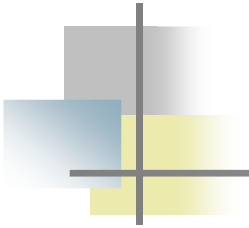
Komunikace mezi balíky

- ROZHRANÍ
- ROZHRANÍ
- ROZHRANÍ

Funkční členění – subsystémy

- Motivace: monolitická aplikace nepraktická
- Subsystém = skupina souvisejících balíků a/nebo tříd tvořících funkční celek
  - vícenásobně použitelných, volně vázaných částí
  - celků vhodných pro vývoj a údržbu
- Třídy subsystému
  - funkčně soudržné (lokalita změn)
  - často vázané na jednoho aktéra
- Třídy mimo
  - stejně silná vazba na více subsystémů
  - zakreslené na úrovni subsystémů

horizontální vrstvy vs. vertikální domény (finance, personalistika, řízení, utility, …)



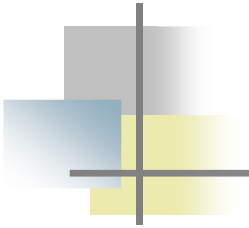
Konvence a politiky

- „Architectural policies“
  - obecná pravidla pro návrh v libovolné části aplikace
  - musí dodržovat všichni vývojáři

- Správa paměti
- Synchronizace, transakce
- Defenzivní programování
- Lokalizace (L10N, I18N)

Jak najít balíky, subsystémy

- Rozdělení dopředu zřejmé
  - jednoduché a/nebo standardní aplikace
  - vodítko: použití architektonických stylů
- Na základě objektového modelu
  - nutno vidět všechny třídy a vztahy
  - shluk těsně vázaných tříd = kandidát
- Na základě případů použití
  - potlačit nepodstatné případy použití
  - pro zbylé zkusit vyjmout klíčovou třídu $\rightarrow$ závislé třídy
  - napomohou řídící objekty



Kdy začít členit do subsystémů

Členění ovlivňuje plánování a postup prací

- Standardní projekty
  - možno odhadnout nebo stanovit předem
- Menší systémy
  - rozdělení na základě analýzy
- Velké projekty (analýza trvá dlouho)
  - nahrubo předem $\Rightarrow$ rozfázování prací včetně analýzy
  - vodítko: aktéři, znalost struktury fyzického systému, možnosti reuse, vývojové kapacity
  - přesně až během návrhu architektury

Základní architektonické styly

[BUS96] defines an architectural pattern as: "An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them."

- Klient-server
  - tlustý klient
- 3vrstvé a vícevrstvé
  - oddělení prezentace, business logiky a datové části
  - dnes standard
- Vrstvená
  - delegování na podřízené
  - prezentace / řízení / doména / business služby / technická infrastruktura / knihovní třídy / systémové

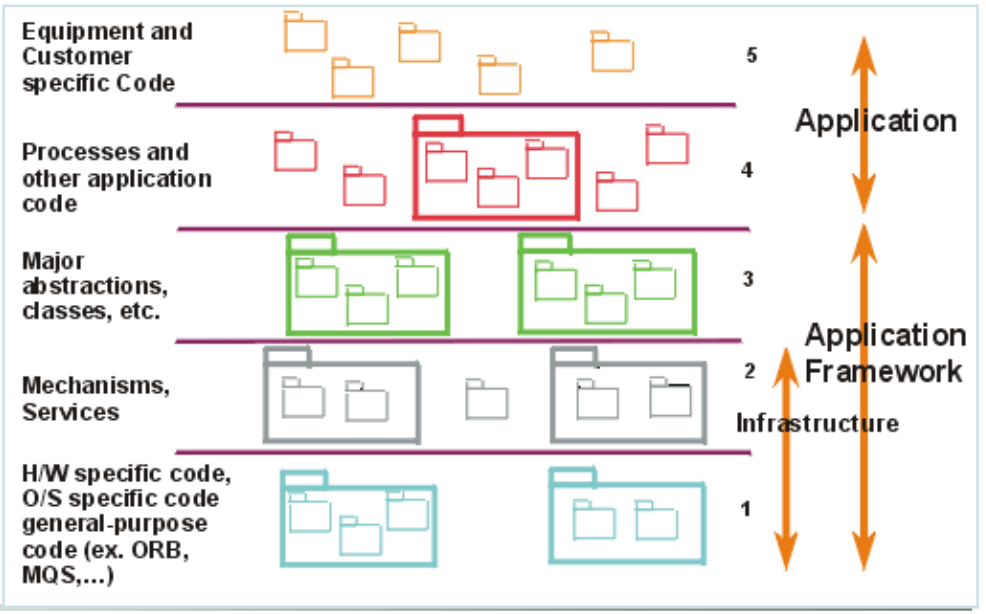


Image from IBM RUP 2005 Concept: Architecture

Další architektonické styly

- SOA (Service-oriented architecture)
- Pipes and filters („kolona“)
- Blackboard
- Broker

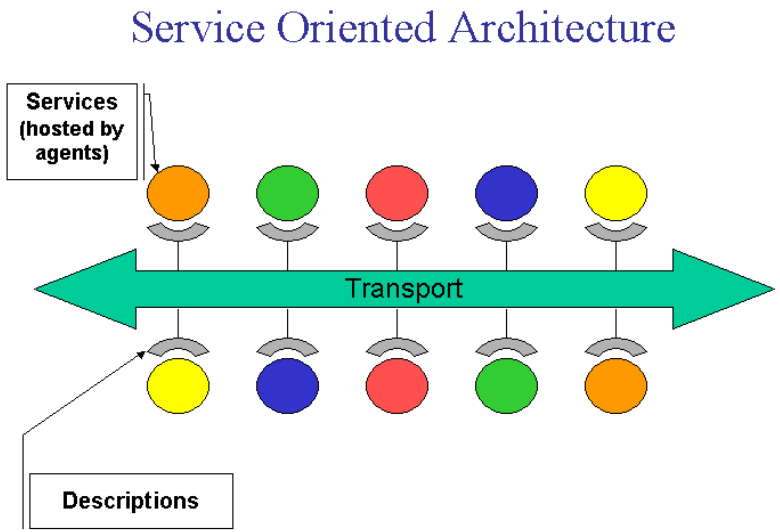


Image from http://lists.w3.org/Archives/Public/www-ws-arch/2003Feb/0030.html

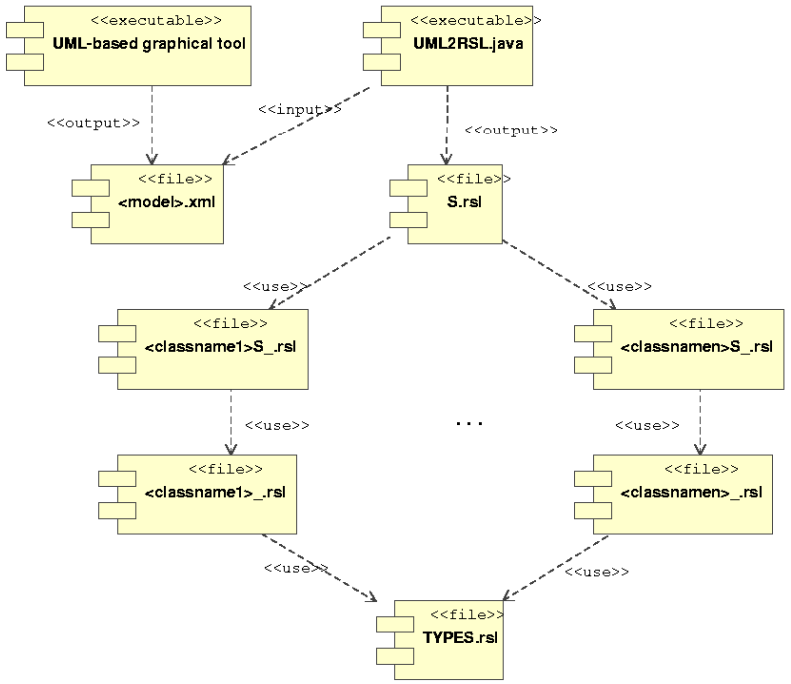
Komponentový přístup

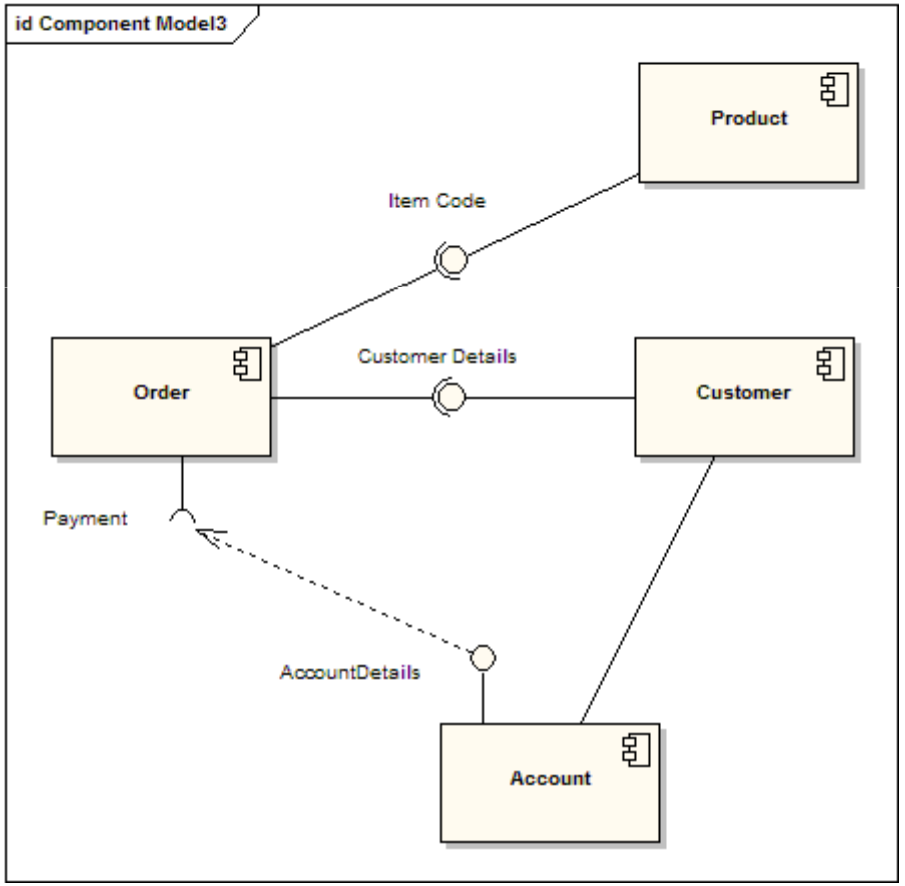
- Dotažení zapouzdření a rozhraní do konce
  - aplikace jako Lego skládačka (teoreticky)
- Komponenta
  - black-box – implementace „nedůležitá“ a „nedosažitelná“
  - rozhraní rozlišena na poskytovaná a vyžadovaná ⇒ vazby
    - » provides/exports, requires/imports/depends
  - explicitní specifikace rozhraní a vlastností
    - » manifest, deployment descriptor
- Technologie
  - CORBA, EJB (částečně), portlety, OSGi
  - IoC containers: Spring, PicoContainer

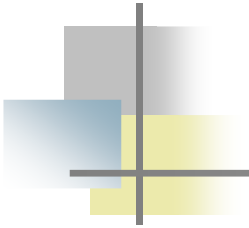
Přístup Inversion of Control (IoC) a Dependency Injection (DI)

UML – reprezentace komponent

- Rozdíl UML1 a UML2



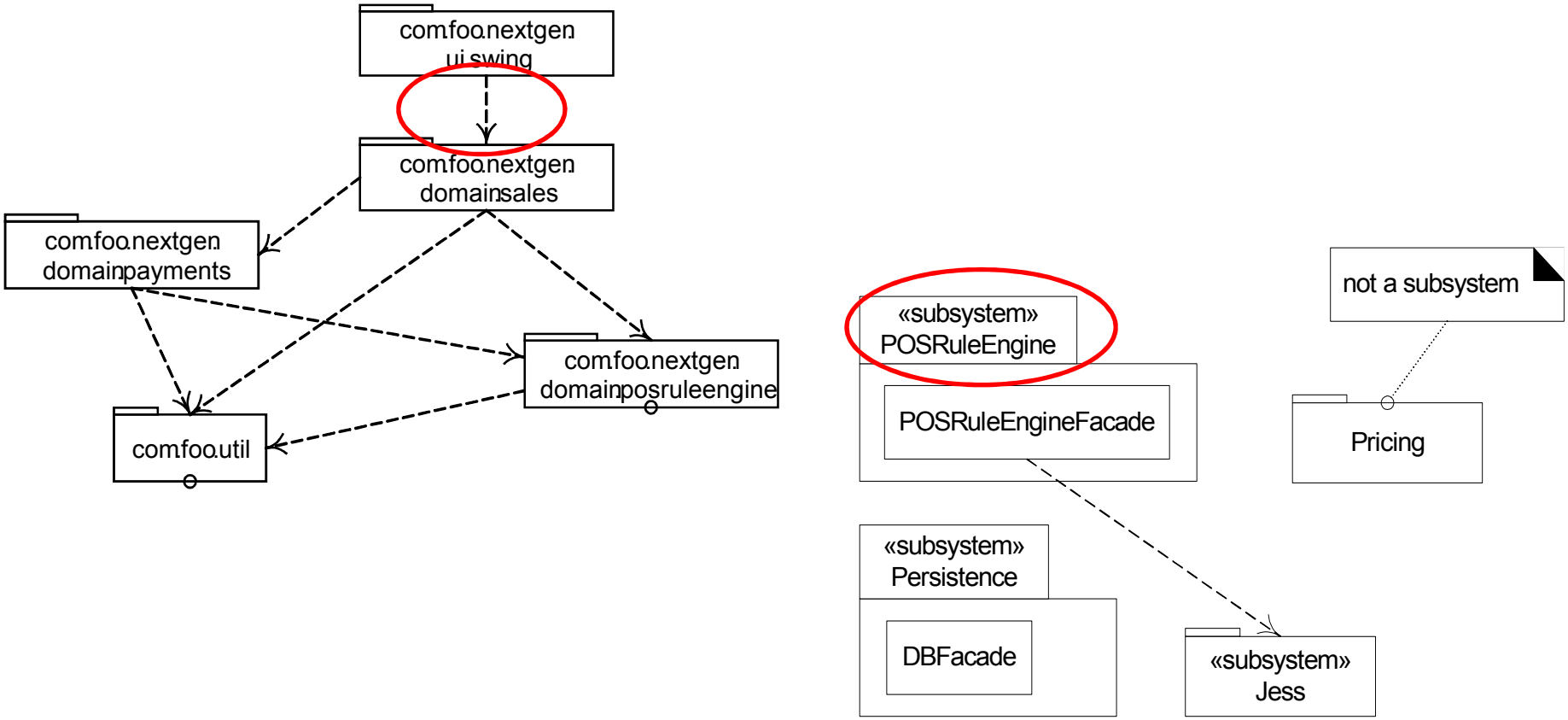




ORM

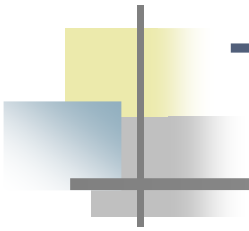
- Objektově relační mapování
  - aplikace chce objekty, perzistentní data chceme v relační DB
- Ručně: data access objects + relační model
  - třídy (atributy) označené «persistent» → tabulky (sloupce)
  - object ID → primární klíč
  - 1:1 asociace → FK vazba; agregace → obrácená FK vazba
  - generalizace → sada tabulek s FK, *join* pro získání potomka
- Knihovny / rámce pro persistenci
  - Java – JDBC, Hibernate
  - PHP – PEAR MDB, ADOdb, Scrubs, Metastorage

UML – struktura architektury



Dokumentace architektury

- Referenční architektura
  - dokument
  - kostra aplikace
- Dokument
  - RUP Artifact: Software Architecture Document
- Modely
  - UML
    - » „implementation view“ (komponenty), „logical view“ (třídy, balíky), „process view“ (interakce, stavový model)
  - ad-hoc diagramy



Třívrstvý model

MVC struktura aplikace

- 3vrstvá architektura „v malém“
  - oddělení vrstev, které se nejčastěji mění
  - typicky uživatelské rozhraní

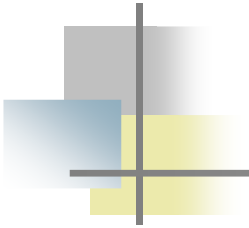
- Základ: návrhový vzor Model-View-Controller
- Realizace pomocí tříd/objektů
  - *hraniční* – rozhraní
  - *řídící* – mechanismy
  - *datové* – perzistence údajů
  - *+ knihovní, systémové*

Hraniční třídy

- Obsahují funkčnost přímo závislou na okolí
  - dialog s uživateli
  - komunikace s externími systémy
    - aktéři přistupují k systému pouze přes hraniční objekty
- Úlohy hraničních objektů
  - prezentace / čtení informací
  - zapouzdření externích prvků
    - předávání dat od / k interním objektům
  - zpracování informací, jejich uchovávání
- Podle toho základní atributy a metody

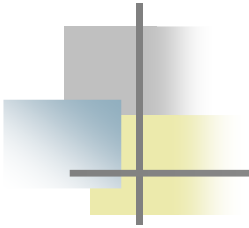
Hledání hraničních tříd

- Objekty zřejmé z popisu rozhraní v PU
  - specifikace požadavků, znalost systému
  - odraz v prototypu (možno přímo převzít)
- Alespoň jeden H objekt pro každého aktéra
  - nemusí platit pro abstraktní aktéry
- Rozbor případů použití
  - vyznačit místa s rozhraním na systém
  - H objekt obsahuje primárně funkčnost závislou na podobě rozhraní (ne zpracování)
- Obvykle kombinace $\Rightarrow$ konsolidace výsledků



Datové (entitní) třídy

- Zapouzdřují informace uchovávané delší dobu
  - typicky přežívají mezi případy použití
- Atributy
  - uchovávaná informace, jednoduché i strukturované
    - » samostatně zpracovávaná data ⇒ separátní objekty
- Metody
  - funkčnost svázaná s přístupem k informacím
  - životní cyklus objektu, přístup k datům, zpracování dat, složitější přímo související analýzy apod.



Hledání datových objektů

- Zřejmé z výsledků analýzy
  - často doménové objekty
  - data vstupující do systému
- Rozbor případů použití
  - vyznačit data používaná při zpracování
  - vyhledat přímo svázané operace
  - zapouzdřit do objektů
- Úpravy
  - separovat společné vlastnosti $\Rightarrow$ dědičnost
- Realizace v SQL/RDBMS až později

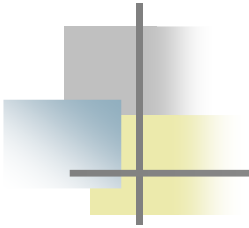
Řídící třídy

- Řízení aplikace
  - stavové přechody, mechanismy (business rules)
  - funkčnost, která se může změnit nezávisle na datech
  - některé nelze nebo není dobré svázat s hraničními / datovými třídami
    - komplexní kontroly nebo zpracování
    - koordinace více objektů

  $\Rightarrow$ jejich vyčlenění do specielních tříd
- Vlastnosti řídících tříd
  - obvykle jen pro daný případ použití
  - obvykle poměrně malé, těžiště v několika metodách

Hledání řídících objektů

- Typicky v rozborech případů použití
  - transakční operace
  - izolování hraničních a datových objektů
  - zajištění komunikace mezi objekty
- Technika
  - pro začátek jeden případ použití $\Rightarrow$ jeden řídící objekt
  - hledat typické chování, přiřadit řídícímu objektu
  - více různých $\rightarrow$ vyrobit potřebné další řídící objekty
  - typické chování nenalezeno $\rightarrow$ řídící objekt zbytečný

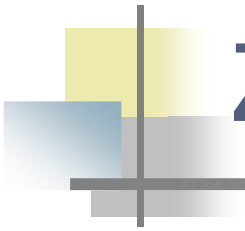


Systémové a knihovní třídy

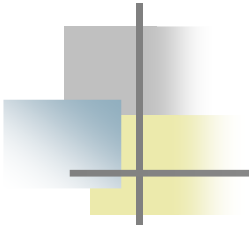
- Realizace nízkoúrovňových úloh
  - komunikace
  - soubory
  - zobrazování
  - …
- Cíl: reuse (využití již hotového)
  - znalost knihoven
  - namapování hraničních, případně datových tříd

Úvaha o strategiích alokace řízení

- Hraniční $\times$ datové $\times$ řídící objekty
  - každý může obsahovat řídící mechanismy
- Cíl: minimalizace dopadu, lokalita změn
  - uživatelské / systémové rozhraní
  - reprezentace a předávání informace
  - malá změna požadavků $\Rightarrow$ lokální změna implementace
- Druhy aplikací podle typu řízení
  - výpočetní a řídící systémy, dialogové, kombinované
  - vyvážené



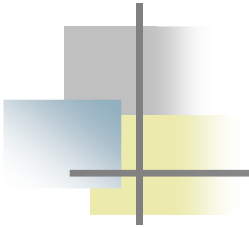
Získávání detailů pro návrh



Metody tříd

- Cíl: interní funkčnost jednoznačně odvozená od požadované vnější
  - trasovatelnost požadavků (údržba!)

- Rozbor případů použití
  - návrh, implementace rozvíjí logický objektový model
  - zodpovědnost → metoda
    - » jakmile je dost informací (může potřebovat 0..3 iterace)
    - » obvykle Z:M = 1:N



Mechanismy spolupráce tříd

- Mechanismy = spolupráce více tříd na realizaci funkčnosti
  - ideální stav: zobecnitelné
- Rámují jednotlivé zodpovědnosti
  - rozbor scénáře případu užití
  - správně identifikovat spouštěcí událost
  - návrh postupu předávání zpráv (zodpovědnosti)
  - využití návrhových vzorů (Listener, Strategy, …)

Vztah PU ↔ metody

Realizace případu užití

- Scénář jednoho PU
  - popisuje konkrétní interakci instancí → mechanismy
  - generuje metody jejich tříd
- Soubor scénářů
  - vytváří celkový souhrn chování třídy
  - při vhodném výběru její úlohu → rozhraní

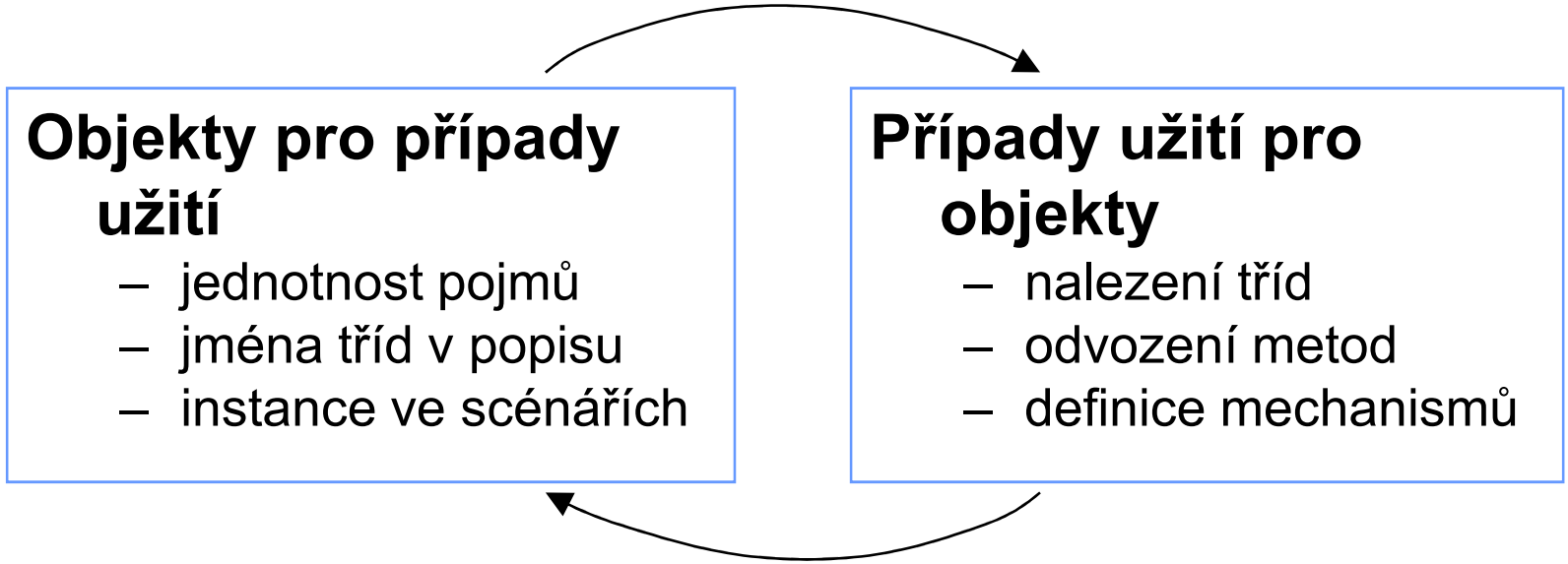
- Výsledek: tvorba/doplňování objektového modelu
  - algoritmy, třídy, metody, parametry

Stavové modely objektů

- Přechody mezi stavy
  - na základě volání metod
  - podmíněné částmi interního stavu
- Objekty řízené podněty
  - nezávislost na stavu: volání metod v libovolném pořadí
  - časté pro datové objekty
- Objekty řízené stavem
  - stav určuje možná volání: pořadí metod tvoří protokol
  - přechody mezi stavy chráněné podmínkami
  - časté pro řídící objekty

Souvislost analýzy struktury a dynamických aspektů

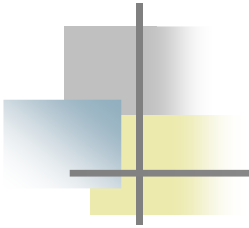
- Iterativnost = základní vlastnost objektové analýzy a návrhu



Nutnost konsolidovat model

- Z analýzy scénářů
  - pohled na jednu roli třídy $\Rightarrow$ některé metody třídy
  - ze souboru scénářů všechny metody třídy potřebné pro realizaci jednotlivých případů použití

- Výsledek: nekonzistence rozhraní
  - názvy metod, syntaxe volání
  - duplikáty, podobnosti, rozpory v sémantice
  - příčiny: různé scénáře, různí autoři



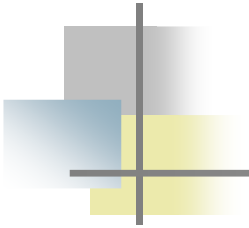
Úpravy vnitřku tříd

- Konsolidace syntaxe a sémantiky tříd
  - komplementarita metod (Create – Destroy)
  - znalost domény a implementace
    $\Rightarrow$ další atributy a metody
  - snaha o minimalitu, reuse
  - viditelnost vlastností (public, private)

$\Rightarrow$ Následné zpětné úpravy scénářů

Úpravy celkového chování třídy

- Role třídy (zodpovědnosti, mechanismy)
  - ⇒ sady metod, rozhraní
    - diagram tříd, komponent, kompozitů
- Postup implementačního mechanismu
  - ⇒ pořadí volání metod, protokol
    - entity s životní cyklem, řídící objekty, akční členy, ...
    - obvykle stavové modely
- Dohromady kontrakt třídy
  - důležitá součást specifikace návrhu



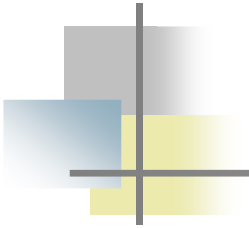
Úpravy vztahů a modelu

- Úpravy objektového modelu
  - vyhledání opakujících se prvků $\Rightarrow$ rodičovské třídy
  - vyjasnění vztahů (asociace $\rightarrow$ agregace, c-s, ...)
  - nové řídící objekty, nové vztahy mezi třídami

Vzory řešení standardních problémů

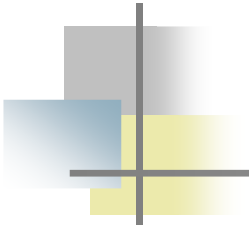
Návrhové vzory pro řešení standardních problémů

- Cíl:
  - rozvinout esenciální model tříd z analýzy
  - systematický přístup k řešení
- Základní prostředky
  - návrhářská / programátorská zkušenost
  - návrhové vzory
    - GoF patterns, J2EE patterns, …
  - jazykové idiomy, jmenné konvence



Multiobjekty, kompozity

- Cíl:
  - implementovat násobné vazby
  - pracovat se složenými objekty
- Kolekce
  - flexibilní implementace 1:N vazby
  - přistupovat přes interface, implementační třída dle výkonnosti
- Composite
  - objekt s (rekurzivní) heterogenní strukturou
  - práce s kompozitem stejná jako s jeho prvky



Tvorba instancí

- Cíl
  - řízené vytváření instancí (oproti *new* operátoru)
  - flexibilita
- Factory
  - vytváří instance jiných tříd
  - obecné rozhraní, konkrétní implementace → polymorfní *new*
- Factory method
  - *getInstance()* na třídě
  - zabraňuje vytvářet objekty kýmkoli

Komunikace mezi subsystémy

- Cíl:
  - zapouzdřit implementaci, poskytnout rozhraní
- Fasáda
  - „jedna" třída / rozhraní pro subsystém
  - deleguje na implementační objekty
- Singleton (Jedináček)
  - „globální objekt" v čistě objektové implementaci
  - sdílená data, nastavení

Přenos dat mezi MVC objekty

- Cíl: odstínit vrstvy od implementačně závislých věcí
- Hraniční – řídící
  - asociativní pole
  - value object, Transfer Object, JavaBean
    - jazykový / jmenný idiom
  - Front Controller, Application Controller
    - separování správy událostí a řízení aplikace
  - Session Facade
    - sloučení business logiky do větších celků pro klienty

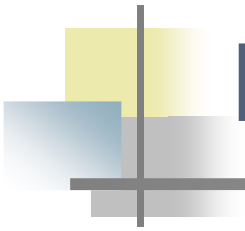
Přenos dat mezi MVC objekty (2)

- Řídící – datové
  - Business Object
    - reprezentace doménové třídy v implementaci
  - Data Access Object (+ Transfer Object)
    - Factory pro value object
    - odstínění aplikace od databázové vrstvy
  - vzor Observer / Listener
    - reakce na změny dat v jiných částech aplikace

Přístup k externím prvkům

- Cíl:
  - odstínit business logiku od nízkoúrovňových protokolů
  - zamezit propagaci změn v prostředí
- Metoda: zapouzdřit do rozhraní

- Adapter
  - přizpůsobení rozhraní prostředí pro aplikaci
  - mapování dat
- Proxy
  - přístup ke vzdáleným zařízením / objektům
  - lokální cache, pozdní zápis



Rekapitulace

Návrh: rekapitulace

- Známe (skoro) všechny detaily implementace
  - rozpracovaná funkčnost
  - vazba na konkrétní technologie
  - vyřešený způsob komunikace na okolní prostředí
    - » zejména způsob uložení persistentních dat
  - struktura aplikace a rozhraní mezi částmi
  - používané konvence a návrhové vzory
- Co dál
  - naprogramovat co zbývá
  - otestovat, opravit