Architektura softwarových systémů

Ing. Jiří Mlejnek

Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

jiri.mlejnek@fit.cvut.cz
Softwarové inženýrství BI-SI1
ZS 2015/ Před. 5

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

Obsah

- Přechod od analýzy k návrhu
- Návrhová rozhodnutí
- Architektury
- Třívrstvá architektura

Přechod od analýzy k návrhu

- Analýza
  - Co?
  - Analytik
  - Analytická dokumentace
- Návrh (Design)
  - Jak?
  - Solution Architekt
  - Návrhová dokumentace (prováděcí)

Přechod od analýzy k návrhu

- Pracovní postupy
  - Volba a popis architektury
  - Analýza a popis vazeb na okolní komponenty
    - Způsob integrace
    - Předávané informace
    - Scénáře spolupráce
  - Volba a popis způsobu uložení dat
  - Vytvoření návrhového modelu tříd
    - Upřesnění atributů – datové typy
    - Přiřazení zodpovědností softwarovým třídám

Přechod od analýzy k návrhu

- Mapování případů užití na komunikaci softwarových tříd (příp. okolních komponent)
  - Sekvenční diagramy
  - Diagramy spolupráce
- Mapování uživatelského rozhraní
  - Na úložiště dat
    - Lokálně ukládaná data
  - Rozhraní jiných služeb
    - Data ukládaná v jiných komponentách

Přechod od analýzy k návrhu

- Návrh
  - Architektura
  - Způsob uložení dat
  - Návrhový model tříd
  - Model komunikace
- Analýza
  - Vize, obecné požadavky
  - Analytický doménový model
  - Analytický doménový model
  - Model případů užití

Přechod od analýzy k návrhu

Dotazy?

Návrhová rozhodnutí

- Technologie - implementační jazyk
- Způsob ukládání dat
- Frameworky / připravená řešení
- Architektura

Návrhová rozhodnutí

- Volba implementačního jazyka
  - Procedurální
    - C
    - Skriptovací jazyky
  - Objektově orientované
    - Java
    - C#
    - C++

Návrhová rozhodnutí

- Volba implementačního jazyka
  - Procedurální
    - Vhodné pro nízkoúrovňové programování
    - Pro rozsáhlé IS nevhodné
  - Objektově orientované jazyky
    - Většina dnes vyvíjených systémů
    - Dále se budeme zabývat pouze OO přístupem

Návrhová rozhodnutí

- Volba implementačního jazyka
  - Znalosti vývojového týmu – dostupnost na trhu
  - Dostupnost potřebných frameworků
  - Požadavky na přenositelnost (Java, Groovy)
  - Zaměření na OS Windows (.NET)
  - Nízké nároky na webhosting (PHP)

Návrhová rozhodnutí

- Volba způsobu uložení dat
  - Relační databáze
    - MySQL
    - PostgreSQL
    - OracleSQL
  - Objektová databáze
    - Gemstone
    - Caché
    - Versant
  - Soubory

Návrhová rozhodnutí

- Volba způsobu uložení dat
  - Relační databáze
    - Výhody
      - Nejrozšířenější způsob
      - Mnoho výrobců
      - Standard SQL
    - Nevýhody
      - Nutnost mapování objektů do relační databáze

Návrhová rozhodnutí

- Volba způsobu uložení dat
  - Objektové databáze
    - Odpadá problematika mapování objektů do relací
    - Prozatím málo rozšířené
  - Objektově relační
    - Relační databáze rozšířená o objektové rysy
      - Uživatelsky definované typy
      - Kolekce

Návrhová rozhodnutí

- Podklady pro rozhodování
  - Obecné (nefunkční) požadavky
  - Možný budoucí rozvoj systému
  - Znalosti dostupných lidských zdrojů
  - Rozpočet (licence, robustnost řešení)

Návrhová rozhodnutí

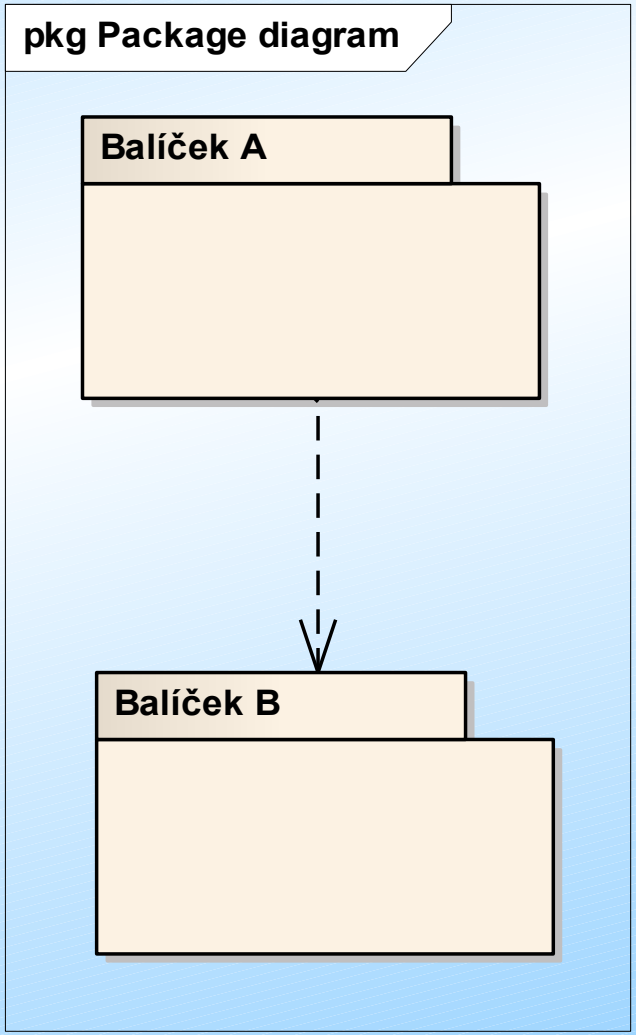
Dotazy?

Architektury

- Logické
  - Organizace softwarových tříd do balíčků, jmenných prostorů
  - Uspořádání balíčků do vrstev, podsystémů
- Fyzické (nasazení)
  - Rozložení komponent na výpočetní uzly
    - Tenký klient
    - Tlustý klient
    - Aplikační cluster
  - Bude probíráno později

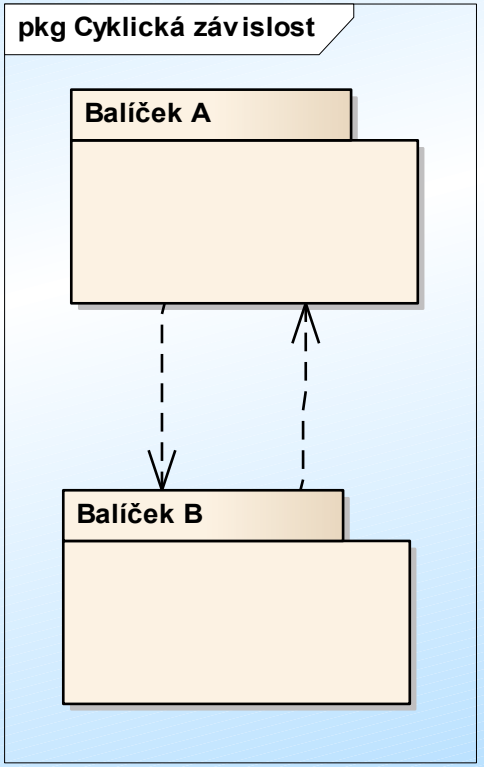
Architektury

- Notace – diagram balíčků UML
  - Balíček
  - Závislost
- EA: UML Structural - Package



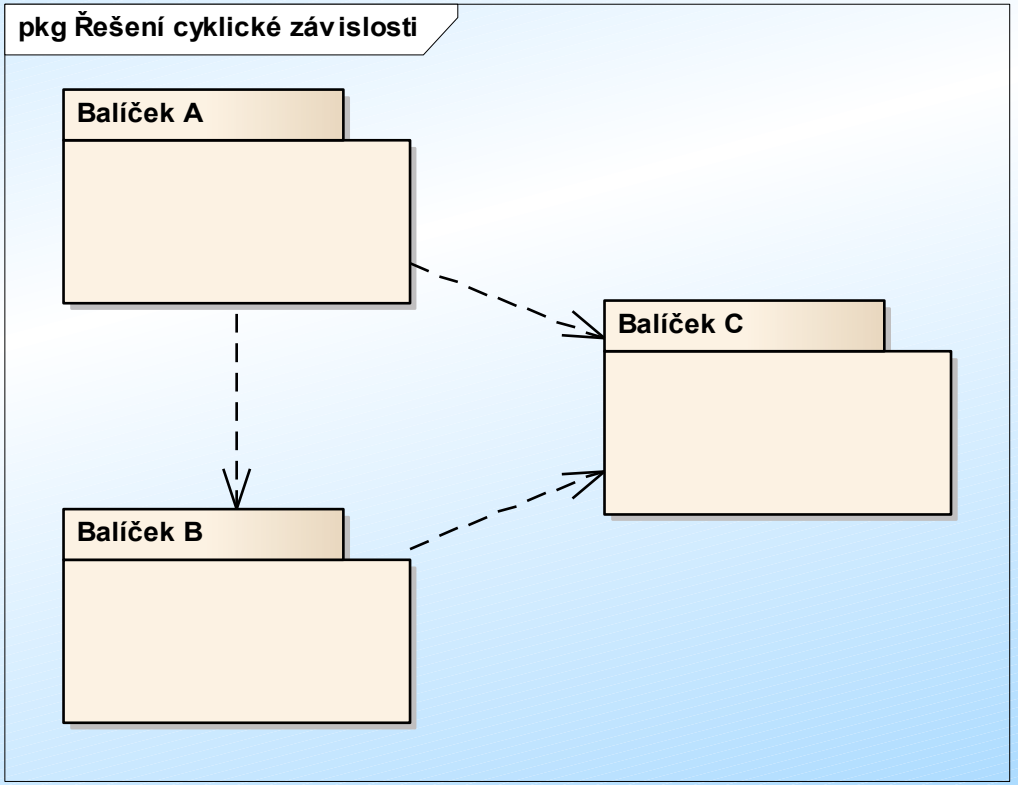
Architektury

- Cyklická závislost mezi balíčky



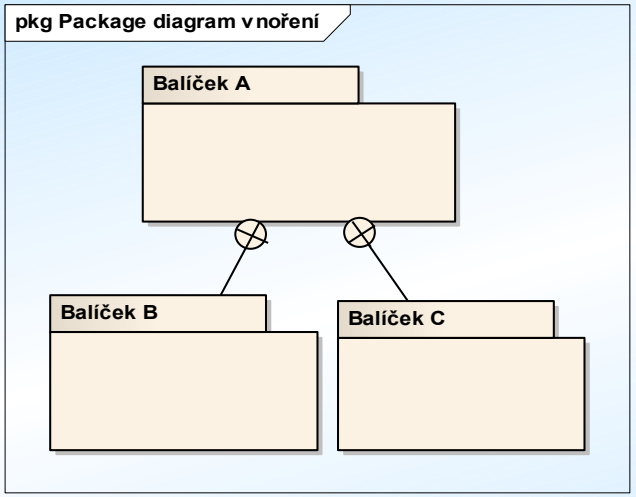
Architektury

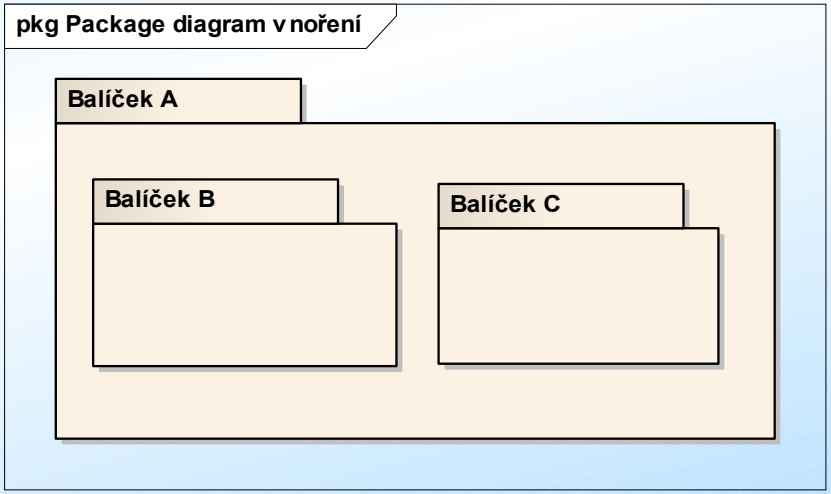
- Řešení cyklických závislostí
  - Přesun třídy
  - Vyčlenění tříd do nového balíčku
  - Vytvořením rozhraní (Dependency inversion)

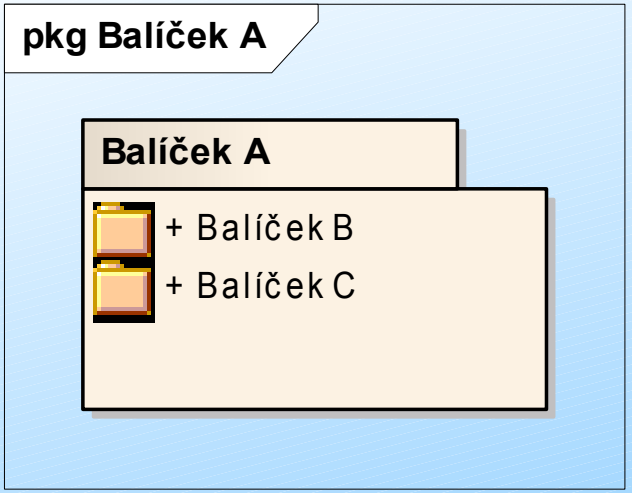


Architektury

- Diagram balíčků
  - způsoby zobrazení vnoření







Architektury

- Roury a filtry (Pipes and filters)
- Servisně orientovaná architektura (SOA)
- Vícevrstvé
  - Monolitická aplikace
  - Dvouvrstvá
  - Třívrstvá
  - Vícevrstvá

Architektury

- Roury a filtry
  - Rozdělení na nezávislé komponenty
  - Každá komponenta provádí specifickou transformaci dat
  - Příklad - unixové filtry

Architektury

- Servisně orientovaná architektura
  - Orientována na služby
  - Integrace nezávislých systémů
  - Bude popsáno později

Architektury

- Vícevrstvé
  - Monolitická (jednovrstvá)
    - Prototypy
    - Rychlý počáteční rozvoj
    - Složitá udržovatelnost

Architektury

- Vícevrstvé - monolitická (jednovrstvá)

```
<html>
  <body>
    <div>
      <?
       $link = mysql_connect("localhost","root","")
       $result = mysql_query("select * from  Zprava,$link);
       while ($row = mysql_fetch_array($result)) {
          echo ("<h4>$row["Nadpis"]</h4>");
          echo ("<div> $row["Text"]<br></div>");
       }
       mysql_close($link);
       ?>
     </div>
  </body>
</html>
```

Architektury

- Vícevrstvé - dvouvrstvá architektura
  - Vhodná pro CRUD aplikace
  - Vrstvy
    - Prezentační
    - Datová

Architektury

- Dvouvrstvá architektura – prezentační vrstva

```
<body>
  <div>
    <?
      $zpravaDAO = new ZpravyDAO();
      $zpravy = $zpravaDAO->vratZpravy();
      foreach ($zpravy as $zprava){
        echo "<h1>".$zprava->vratNadpis()."</h4>";
        echo "<div>".$zprava->vratText()."</div>";
      }
    ?>
   </div>
  </body>
```

Architektury

- Dvouvrstvá architektura – datová vrstva

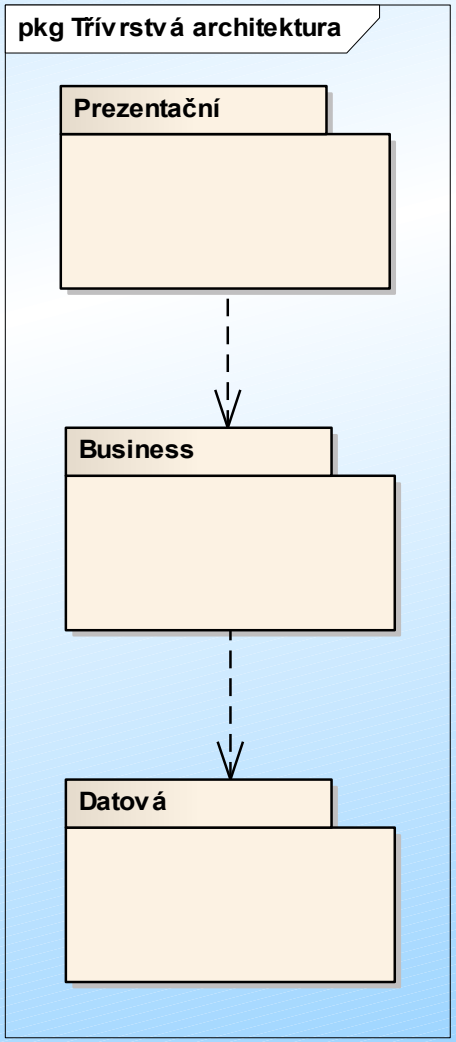
```
class ZpravaDAO {
  public function vratZpravy(){
    $result = mysql_query("select * from Zprava",getLink());
      $i=0;
      while ($row = mysql_fetch_array($result)) {
         $zpravy[$i]=new Zprava($row["Nadpis"],$row["Text"]);
         $i++;
      }
      return $zpravy;
   }
}
```

Architektury

- Třívrstvá architektura
  - Vhodná pro složitější (enterprise) aplikace
  - Vrstvy
    - Prezentační
    - Business (doménová)
    - Datová (technické služby)

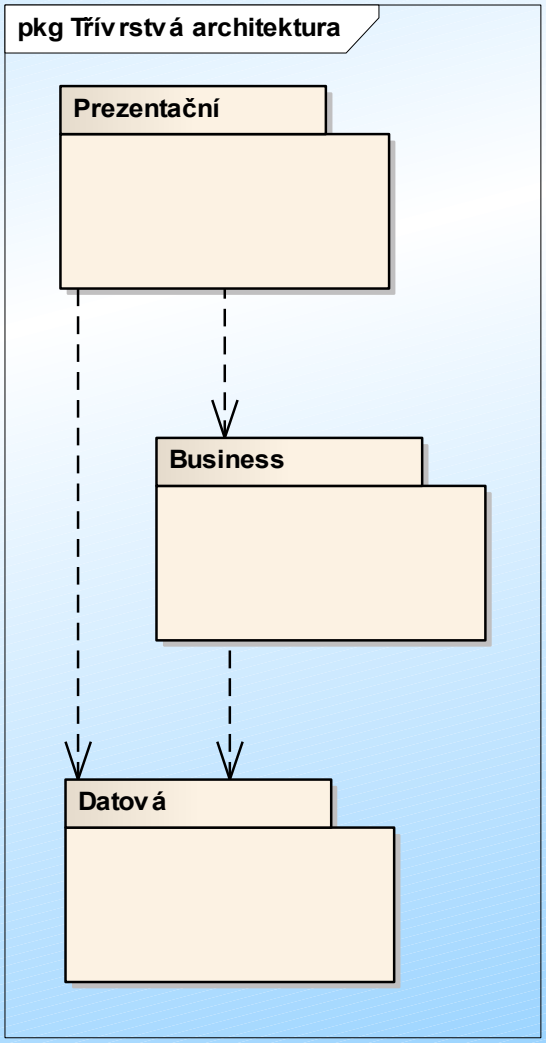
Architektury

- Třívrstvá architektura - striktní
  - Závislost vždy směrem dolů
  - Pouze o jednu úroveň



Architektury

- Třívrstvá architektura – relaxovaná
  - Závislost vždy směrem dolů
  - Přes libovolný počet úrovní



Třívrstvá architektura

- Datová vrstva
  - Řeší problémy
    - Persistence dat
    - Logování

Třívrstvá architektura

- Datová vrstva – persistence dat
  - Vzory
    - Row Data Gateway
    - Table Data Gateway
    - Active Record
    - Data Mapper
  - Existující řešení
    - Hibernate
    - Doctrine 2
    - ZendFramework

Třívrstvá architektura

- Datová vrstva - logování
  - Úroveň / priorita
    - DEBUG, INFO, WARN, ERROR, FATAL, apod.
  - Logger – třída umožňující programátorovi zapsat do logu
  - Appender/Handler – třída zodpovědná za záznam zpráv do logu
    - Umístění logů – do souboru, na konzoli, email, databáze, apod.
    - Formátování logů (datum, zpráva, úroveň, atd.)

Třívrstvá architektura

- Datová vrstva – logování
  - Existující řešení
    - Java Logging
    - LOG4J
    - a další

```
Logger log = LoggerFactory.getLogger(StavVolny.class);
log.info("Vytisk byl výpujčen");
```

Třívrstvá architektura

- Prezentační vrstva
  - Problém závislosti zdola nahoru
  - Řešení - MVC
    - Komponenty
      - Model
      - View
      - Controller
    - Varianty
      - Aktivní
      - Pasivní
    - Vztah k třívrstvé architektuře

Třívrstvá architektura

- Prezentační vrstva
  - Existující řešení
    - JSF 2 (RichFaces, MyFaces)
    - Swing, SWT, AWT
    - JQueryUI, AngularJS
    - GWT, SmartGWT
    - JavaFX
    - Silverlight
    - FLEX
    - ...

Třívrstvá architektura

- Business vrstva
  - Business logika
  - Procesy a validace
  - Nezávislá na prezentační a datové vrstvě

Třívrstvá architektura

- Výhody
  - Oddělení business logiky od prezentační vrstvy
  - Nezávislost business logiky na způsobu uložení
  - Snadná výměna jednotlivých vrstev
  - Jednoduché testování
  - Více různých prezentačních vrstev

Třívrstvá architektura

- Výhody
  - Znovupoužitelnost
    - Čím nižší vrstva tím je možné jí znovu použít na více projektech (služby pro logování, odesílání emailů, persistenci, apod...)
    - Čím vyšší vrstva tím je specifičtější pro konkrétní projekt (business pravidla, grafické rozhraní, apod...)

Třívrstvá architektura

Dotazy?

Děkuji za pozornost.