# KOMPONENTOVÉ MODELY A ARCHITEKTURY

Lukáš Holý

# Přehled

- Komponenty obecně
  - Rozhraní, verze, kontrakty, kompozice, mimofunkční požadavky, certifikace
- Komponentové modely a frameworky

- Architecture Description Languages
- Inversion of Control a Dependency Injection
- ENT Meta-Model

# Komponenty

- Výpočetní jednotky
- Slouží ke kompozici ve větší celky
- Kompozice třetími stranami
- Měly by být „black-box“ (nevidíme dovnitř)
  - Navenek zveřejňují „různě kvalitní“ rozhraní
- Jsou nezávisle nasaditelné

# Výhody, nevýhody

- Výhody - převažují
  - Měly by urychlit vývoj aplikace
  - Zlevnit cenu softwaru, např. díky znovupoužití existujících, otestovaných,… komponent (třetích stran)
  - Zlepšit předvídatelnost chování, …
- Nevýhody, problémy
  - Režie
  - Mnoho modelů, frameworků - brzdí rozvoj většího trhu
  - Systém je často těžké sestavit, udržovat.
  - Komponenty se vyvíjejí, musíme sledovat jejich vývoj.

# Rozhraní a verze

- Verze rozhraní je třeba rozlišovat
  - Nejlépe i třídy nebo metody
- Podpora více verzí (kompatibilita)
  - Každá verze komponenty může mít jiné rozhraní.
- Kontrola verzí
  - Akceptuji, odmítám, „něco mezi“
- Abstraktní rozhraní/rozhraní vázaná s implementací
- Součástí rozhraní by mělo být i ověření správnosti volání, které by komponenta sama provedla.

# Kontrakty

- Kontrakty komponent a interakce
- Měly by zajistit dobré spojení rozraní, určit „práva a povinnosti“ obou (všech) stran

- Změna vnitřního chování stále může narušit dobrý běh
- Kontrakty se mohou stranami vyjednávat, teoreticky se mohou měnit za běhu

- Vypršení kontraktu

# Kontrakty

- Úrovně:
  - Syntaktická
    - typy, pole, metody…
  - Sémantická
    - Rozsahy, pre- a postconditions a invarianty
  - Chování (průběh)
    - omezení kompozice, (vnitřní chování)

- Ověřitelnost oproti specifikaci kontraktem
  - V době překladu, načtení, za běhu.
  - Není možné obecně úplně ověřit.

# Kontrakty a přístupy

- Je obtížné se úplně vyvarovat nepopsaným závislostem
  - Kompromis mezi přesností kontraktu a omezeními
- Klient by neměl spoléhat na nepopsané vlastnosti
- Kontrola přístupu, meta-programming interface
- Někdy možnost kombinovat prg. jazyky
- Vyvarování se platformě závislých jednotek a datových typů

# Kompozice

- Formy
  - Komponenty mezi sebou
    - „normální“ spojení komponent
    - Vnoření komponent - hierarchie
  - Framework-komponenta
    - Nasazení komponenty do frameworku
    - Plug-In – rozšíření vnořeného frameworku komponentou
  - Frameworky mezi sebou
    - Vnoření frameworků
    - Spojení komponent z různých frameworků

# Kompozice

- Volání metod (funkcí, procedur), rozhraní, třídy (poskytují a požadují)
- Komponenty, porty
- Konektory - nepřímá kompozice, funkce navíc
- Glue code – adaptér spojující různé typy
- Switch – druh glue code s možností přepnutí spojení
- Adaptery- JavaBeans, delegace metod

- Umět určit vlastnosti celku z kompozice komponent
  - Certifikované vlatnosti

# Certifikace

- Prohlášení o certifikovaném subjektu
- Někdo důvěryhodný stojí za prohlášeními
- Certifikovat je možné :
  - Frameworky
  - Komponenty
  - Systémy – složení frameworků a komponent
  - Procesy – kvalitní proces -> kvalitní produkt (?)
- Důležitá predikce výsledku kompozice – bez ní je hodnota certifikace zvláště částí málo hodnotná
- Specifikace kontraktů musí být dostatečně popsány
  - Zachycení všech vlastností
  - Určení výsledných vlastností systému

# Mimofunkční požadavky

- Dimenze nakládání s mimofun. požadavky (MP)
  - Vnější / vnitřní řízení MP
  - Rozsah řízení MP – v celém systému / v komponentách zvlášť (+spolupráce mezi nimi)
  - Minimálně 4 možnosti nakládání s MP
- Klasifikace (různé)
  - Chování – pre- a postconditions apod.
  - Synchronizace – těžká specifikace
  - QoS – spolehlivost, použitelnost, …
- Podpora u komponent
  - Zda podpora vůbec existuje
  - Pokud ano, jak se dají specifikovat (typy, prostředky)
  - Dají se MP v komp. modelu skládat (které a jak)

# Rozšiřitelnost a životní cyklus

- Rozšiřitelnost umožní lepší přizpůsobení potřebám
- Nezávisle rozšiřitelný systém
  - Pokud rozšíření nezávislých stran lze kombinovat
- Dimenze rozšiřitelnosti
  - každá rozšiřitelná vlastnost u nezávisle rozšiřit. systému
  - Některé dimenze se mohou překrývat -> problémy
- Životní cyklus:
  - Modelování, implementace, balení
  - Nasazení, nastavení, spuštění

# Komponentové modely

- Mnoho modelů, mnoho rozdělení a kategorií
- Odlišné typy komponent, interakce a skládání
- Kategorie založené na kompozici
  - Kompozice je možná „ručně“, žádné úložiště.
    - UML 2.0
  - Kompozity lze uložit, ale není možné je zpětně načítat.
    - Web Services
  - Kompozity nelze uložit, ale komp. lze skládat při nasazení.
    - JavaBeans
  - Kompozity lze ukládat i zpětně načítat a dále skládat.
    - SOFA

# Komponentové modely

- Kategorie založené na sémantice
  - Třídy - JavaBeans
  - Objekty - .NET
  - Architektonické jednotky – UML2.0, SOFA
- Kategorie založené na syntaxi
  - OOP jazyky - JavaBeans
  - Jazyky s mapováním na IDL - .NET
  - Jazyky s popisem architektury (ADL) – UML2.0, SOFA
    - program musí být vytvořen v nějakém progr. jazyce

# Komponentové frameworky

- Framework implementuje specifikaci modelu
- Řídí zdroje, poskytuje podpůrné služby, …

- Frameworků je mnoho, možnými řešeními alespoň částečného sjednocení jsou:
  - Podpora přizpůsobení frameworků různým požadavkům
  - Přizpůsobení komponent frameworkům

# Architecture Description Languages

- Slouží k popisu architektur
- K vytvoření přehledu nad celkem a o jeho správnosti
- Podobná situace jako u modelů a frameworků – mnoho jazyků, různé vlastnosti a možnosti
- Důležité jsou :
  - Vlastnosti a možnosti jazyka samotného z pohledu:
    - komponent
    - konektorů
    - **utváření architektury**
  - Podpora nástroji

# Architecture Description Languages

- Pro prvky kompozice jsou důležité:
  - Rozhraní, typy
  - Sémantika, omezení, vývoj (změna)
  - Mimofunkční požadavky
- Pro utváření architektury jsou důležité:
  - Kvalita popisu – pochopitelnost,trasovatelnost, různorodost
  - Kvalita popisovaného systému – škálovatelnost, vývoj, dynamika
  - Vlastnosti popisovaného systému - mimofunkční požadavky, omezení
- Pro nástroje je důležité
  - Aktivní specifikace, množství pohledů, možnosti analýzy, generování kódu

# Možnosti sestavení

- Inversion of Control
- Druhy Dependency Injection
  - Využítím konstruktoru
  - Využitím metod set…()
  - Využitím rozhraní
- Vyhledání služby  (Service locator)
- DI je jednodušší analyzovat z pohledu vazeb komponent
- Cílem je oddělení konfigurace služeb od jejich použití aplikací

# ENT Meta-Model

- ENT - Meta-model rozhraní komponent schopný reprezentovat libovolný komponentový model
  - Component model level
  - Application level
- Facets (8) – dimenze představující pohled na komponentu
- Elements – navenek viditelné „části“ komponent (atributy, metody, …)
- Traits – slouží k seskupení elementů, samotné nic neříkají
- Tags – další informace, které nemohou být popsány traity
- Reprezentace jednotlivých modelů – EJB, CORBA, OSGi, SOFA
- Z ENT meta-modelu je vytvořen MOF M3 model
- Úprava modelu a generování tříd - EMF

Děkuji za pozornost !