DSL, Prototypování, MDE

Tvorba programových systémů Y36SI3

Ondřej Macek
macekond@fel.cvut.cz

Dnešní přednáška

- Prototypování
- DSL
- Generování kódu
- MDE

Prototypování a DSL - motivace

Problém:

- Zákazník nerozumí modelům
- Ne vše jde modelem přesně zachytit

Řešení

- Prototypování
- DSL

PROTOTYPOVÁNÍ

Prototypování

- Předvádění prototypů aplikace zákazníkům
- Rychlá aktualizace prototypu podle nových požadavků
- Zahazování nepotřebných/špatných prototypů

Přístupy k prototypování

- Rapid prototyping
  - velké množství malých prototypů – mnohé z nich zahozeny
- Evolutionary prototyping
  - vytvoří se masivní základ, který se postupně upravuje
- Incremental protyping
  - víc prototypů, které se později spojí

Prototypování – klady a zápory

+	-
Jasný dorozumívací prostředek	Nedostatečná analýza/návrh
Snižuje náklady a čas	Fixace na prototyp
Zákazník je zapojen do vývoje	Neodhadnutí ceny prototypů

Omezit se na doménu

DOMAIN SPECIFIC LANGUAGES (DSL)

Domain Specific Languages

- Language Oriented Programming

- **Doménově specifický jazyk na vysoké úrovni abstrakce**
  - abstraktnější než Java, C#

    (tzv. general purpose languages)
  - konkrétnější než UML

    (tzv. general modeling language)

Příklady DSL

- HTML

- Apache httpd.conf

```
ServerRoot  "/var/www"
```

- OCL

```
Circle.allInstances->iterate(  c : Circle ;   variable sum : Real = 0      |
                sum + 3.14 * c.radius * c.radius
               )
```

- QVT

.....

Příklad: Validace kreditních karet

```
<validation>
    <validate country="CZE">
        <accept />
    </validate>
    <validate country="USA">
        <securitycode match="exact" />
    </validate>
    <validate country="TKL">
        <fail />
    </validate>
    <validate country="*">
        <securitycode match="exact" />
        <addressmatch match="matchZipStreetNumber" />
    </validate>
</validation>
```

Příklad: XAML

```
<Window ….>
    <StackPanel>

       <Button Content="Drop Shadow Under Me" Width="200" Margin="10">

            <Button.Effect>
                <BlurEffect Radius="10" />
            </Button.Effect>

        </Button>

    </StackPanel>
</Window>
```

Realizace DSL

- Interní vs. Externí DSL
- Interpreter DSL
  - Vlastní parser DSL
  - DSL je přímo spustitelné
- Kompilátor DSL
  - DSL převedeno do nějakého konkrétního jazyka
  - Může se využít funkcí tohoto jazyka
  - Generování kódu

Výhody DSL

- Popis způsobem blízkým doméně
- Snadný popis problému v doméně
- Validace už na doménové úrovni
- Kvalita, produktivita, spolehlivost, udržovatelnost, znovupoužitelnost

Nevýhody DSL

- Nutnost naučit se DSL
- Nutnost udržovat DSL a potřebné prostředí
- Možná ztráta výkonu oproti general purpose languages
- Složitější problém => složitější DSL
- Rychlý nárůst jazyků

GENEROVÁNÍ KÓDU

Generátory kódu

- Pasivní generátory
  - jednorázově na začátku vývoje (např. z modelu)
  - průvodci vytvořením základu aplikace (wizards)
  - šablony kódu v IDE

- Aktivní generátory
  - při každé kompilaci projektu
  - model -> generátor kódu -> .java -> kompilátor

Generování kódu za běhu

- Aktivní generátor
  - Vytváření kódu za běhu
- Eval v PHP
- Reflection (Java, C#)
- Ken Thompson (UNIX)
  - z regulárních výrazů generoval za běhu strojový kód který se hned spustil
  - opravdový programátor

Příklady generátorů kódu

- Hibernate/NHibernate
- Ruby On Rails
  - convention over configuration
- PHP frameworky
  - CakePHP
- Axis2
  - Java
  - generování Javy z WSDL a WSDL z Javy
- NBusiness
  - C#
- ADO.NET Entity Framework
  - Microsoft .NET

Pro a proti generátorů

+	-
Zrychlení práce	Svazují
Zlepšení kódu	Možná závislost na SW (generátoru) třetí strany
	Kód „vypadá“ jinak

Generátory kódu

- Napsat jednoúčelový generátor není těžké
  - práce s textem
  - procházení textového souboru/XML stromu, rozhodování na základě atributů
  - popis modelu
  - šablony pro různé typy výstupu

MODEL DRIVEN ENGINEERING

Model driven engineering (MDE)

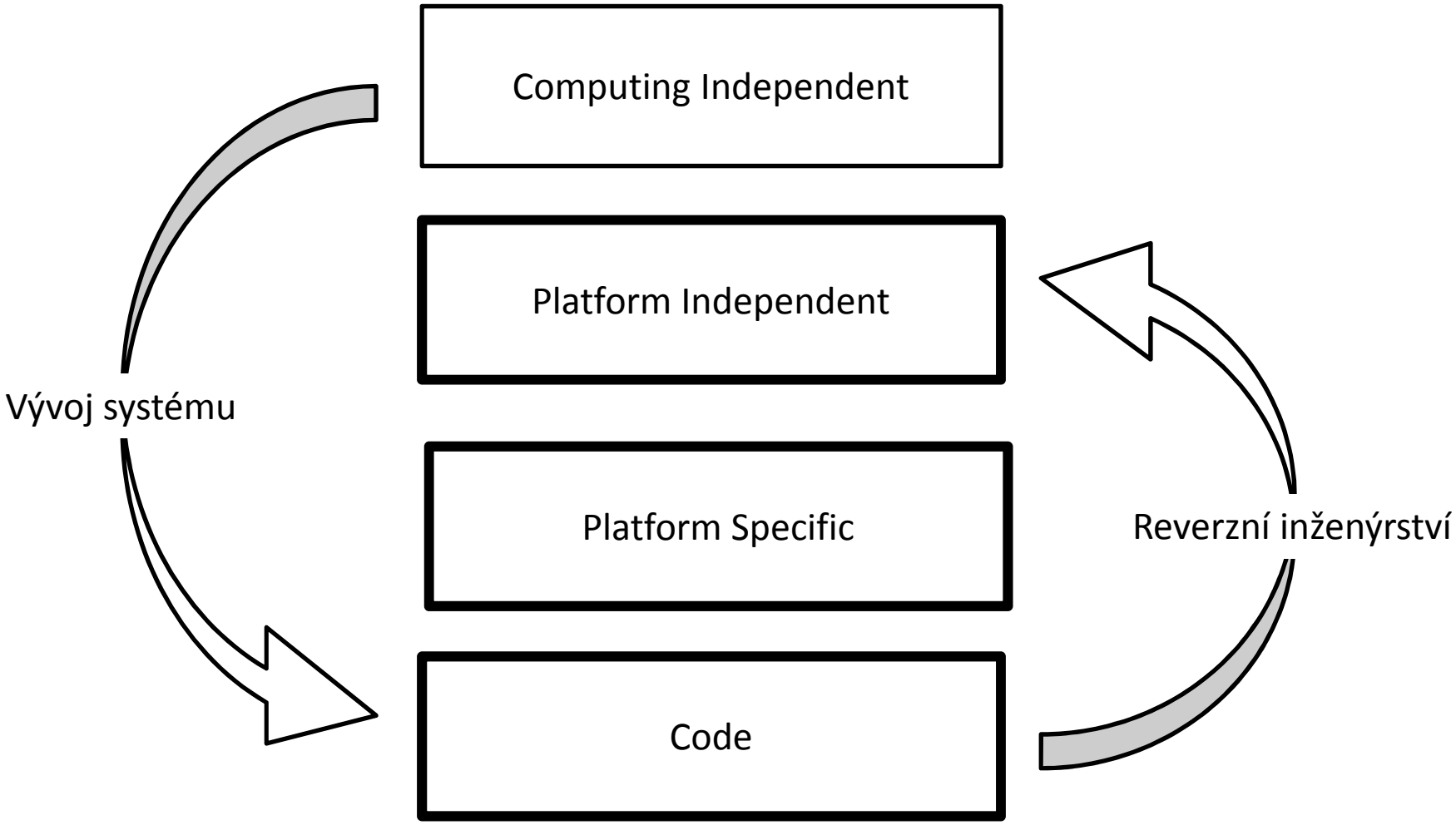
MDE

Model driven architecture (MDA)

Model driven development (MDD)

Model má vždy pravdu

Tři (čtyři) základní úrovně modelů



Přístup k MDE

- Spustitelný model

- Generování kódu

Doporučená literatura

DOMAIN-SPECIFIC LANGUAGE. Wikipedia, the free encyklopedia

http://en.wikipedia.org/wiki/Domain-specific_programming_language

[cit . 2010-10-14]

Freeze J.: CREATING DSLS WITH RUBY. Artima Developer – Best practices in enterprise software development

http://www.artima.com/rubycs/articles/ruby_as_dsl.html

[cit. 2010-10-21]

HAAN Johan. ***DSL DEVELOPMENT: 7 RECOMMENDATIONS FOR DOMAIN SPECIFIC LANGUAGE DESIGN BASED ON DOMAIN-DRIVEN DESIGN*** . The Enterprise Architect – Building an agile enterprise

http://www.theenterprisearchitect.eu/archive/2009/05/06/dsl-development-7-recommendations-for-domain-specific-language-design-based-on-domain-driven-design

[cit . 2010-10-14]