DATOVÉ MODELOVÁNÍ A TYPOVÁNÍ

František Huňka
Ostravská univerzita v Ostravě, frantisek.hunka@osu.cz

Ferdinand Mácha
Charonware, s.r.o. Ostrava, f.macha@seznam.cz

ABSTRAKT:
Datové modelování poskytuje celou řadu mechanismů a nástrojů pro tvorbu konceptuálních modelů. Datové modely stále častěji využívají k pružným a snadno adaptovatelným řešením kromě vlastního modelu také meta-model. Jednou ze sémantických abstrakcí, která se využívá na tvorbu meta-modelu je typování. Příspěvek se zabývá oběma formami abstrakce datových modelů a to generalizací a typováním. Uvádí společné a rozdílné vlastnosti. Dále uvádí použití vzoru responsibility jak v hierarchické struktuře dat, tak v kompozici datových entit.

ABSTRACT:
Data modeling provides a number of mechanisms and tools for conceptual models creation. Data models use often frequently apart from their own model also a meta-model for flexible and easy adaptable solutions. One of the semantic abstractions, which is utilized for the meta-model creation, is typing. Paper deals with both forms of data model abstractions that are generalization and typing. It describes common and different properties. Furthermore it introduces using of the Responsibility pattern both in hierarchical structure and in the composition of data entities.

KLÍČOVÁ SLOVA:
datové modelování, datové abstrakce, konceptuální model, typování

1 Úvod

Konceptuální modely mají své stálé místo při tvorbě podnikových informačních systémů. Jejich původní záměr a cíl se poněkud rozšířil s vývojem dalších programových paradigmat zejména s objektově orientovanou perspektivou. To vyústilo v zavedení třívrstvé architektury pro tvorbu informačních systémů, kde právě prostřední vrstva je tvořena doménově závislým konceptuálním modelem. Je snaha, aby vytvořené konceptuální modely byly pružné ke změnám a tedy bez velkých problémů akceptovaly změny v modelované doméně. Typování podle [3] *power types* představuje důležitý mechanismus, který patří k sémantickým abstrakcím vrstvy meta-modelu. Existují dobře popsané návrhové vzory viz [4], které popisují konkrétní použití těchto vzorů při návrhu podnikových informačních systémů. Navíc konceptuální modely se staly výborným prostředkem při procesu reengineeringu.

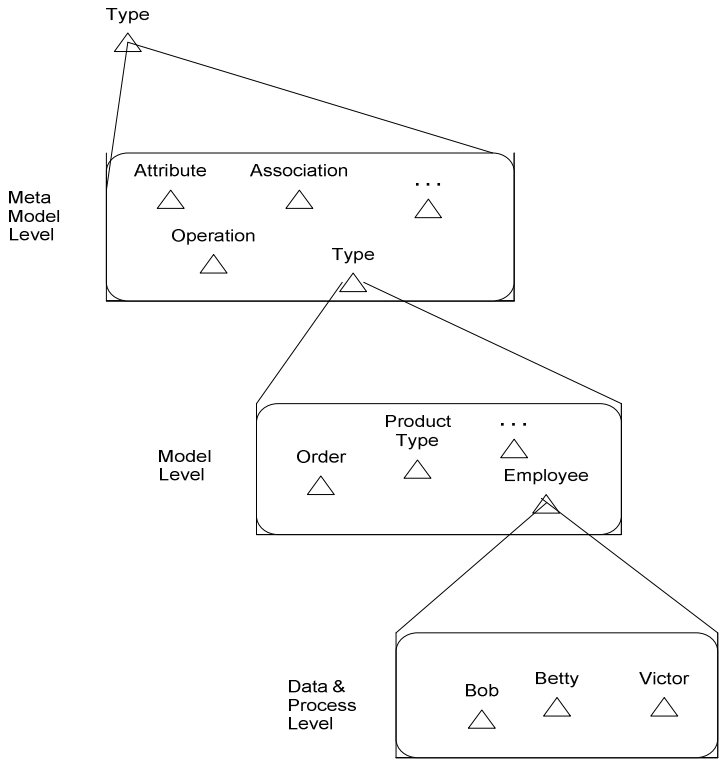
2 Konceptuální datové modely

Konceptuální modely podniků mohou být využity jak pro modelování podnikových procesů, tak pro aktuální programový návrh informačních systémů. Historicky byly konceptuální modely primárně použity k získání požadavků na informační systémy a k hladkému převodu těchto požadavků na různé databázové systémy resp. souborově orientované informační systémy. Konceptuální modely začínají specifikací požadavků a ústí do konceptuálního

schéma (konkrétního databázového systému). Později bylo použití konceptuálních modelů rozšířeno na konstrukce a modely libovolného typu domény, které používá množinu konstruktů teoretického modelování, které zachycují způsob existence reality. V oblasti podnikového prostředí se tato realita kromě dalších věcí hlavně skládá z podnikových procesů a podnikových rolí. Další oblast, ve které konceptuální modely našly bohaté uplatnění je oblast business process reengineeringu. V klasické třívrstvé architektuře tvoří konceptuální model prostřední vrstvu a je nezávislý na vrstvě uložení. Takto konceptuální modely blíže vystihují modelovanou doménu a jsou nezávislé na uložení dat.

3 Struktura datového modelu

Meta-model je model, který se používá k popisu různých druhů (typů) modelů. Např. meta-model entitě relačního diagramu a data flow diagramu bude definovat instance typů jako např. typ entity, typ relace, typ atributu, typ procesu, datové úložiště a datový tok. V objektově orientovaných modelech, bude meta-model obsahovat typ asociace, typ atributu, typ metody a typ operace. Tyto typy meta-modelu vytváří pak horní vrstvu modelu a diktují způsob, jakým bude spodnější vrstva vyjádřena. Např. typ operace bude umožňovat na úrovni modelu pouze konkrétní operace. Stručně můžeme říci, že meta-model obsahuje typy, jehož instance jsou zase typy.



Obr. 1 Úroveň meta-modelu, modelu, dat a procesů

Úroveň meta-modelu nabývá na významu se zvyšující se potřebou rychlé adaptace programového vybavení novým podmínkám. Existuje i návrhový vzor reflexe, který se zabývají uvedenou problematikou. Např. v programovacím jazyce Java pak k dispozici celá knihovna *java.util.reflection*, která poskytuje řadu tříd a metod pro získání informací o meta-

modelu. Na podobném principu jsou také založeny tzv. anotace, které se objevují od verze Java 5 a výše.

Uvedená metaúroveň má ale také velký význam při praktickém datovém modelování. Na této úrovni mohou být uložena pravidla, která se pak mohou používat za běhu vlastního programu. Také tato úroveň vede k návrhu pružnějších a změnám odolných datových modelů. Nejčastěji se pro tuto činnost využívá sémantická abstrakce *typování* v literatuře [3] označena jako *power types*.

Typování bývá často opomíjeno jednak proto, že konkrétní programovací prostředek nemusí tento typ abstrakce podporovat a jednak také proto, že častěji před typováním bývá upřednostněna generalizace. Proto si nejdříve popíšeme základní vlastnosti obou konceptů a jejich výhody a nevýhody.

4 Generalizace a typování

Abstrakce představuje nejdůležitější princip pro opětovnou použitelnost. Konkrétně se jedná o nahrazení konkrétní datové entity, abstraktnější datovou entitou. Uvedená abstrakce nabývá většinou jedné ze dvou následujících forem:

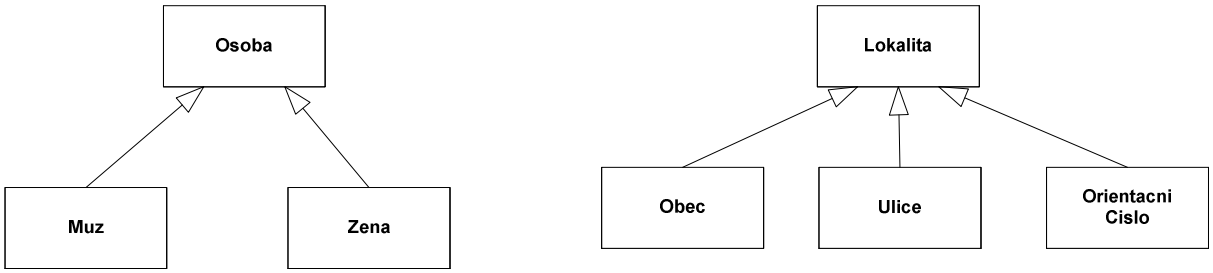
- generalizace,
- typování.

4.1 Generalizace

Generalizací se rozumí zobecnění specializované datové entity do obecnější datové entity; mezi těmito entitami se stanoví relace generalizace. Generalizaci lze charakterizovat těmito specifickými vlastnostmi:

- existuje jen mezi datovými entitami (třídami), nepřenáší se na instance,
- způsobuje dědění datových atributů a relací z obecnější do specializovanější datové entity,
- každá instance od obecnější entity dědí pouze strukturu datových atributů, nikoli však obsah,
- je to relace typu nebo – existuje-li k jedné obecnější datové entitě více specializovanějších entit, daná instance se vytvoří právě k jedné z nich.

Na následujícím obrázku jsou pomocí diagramu tříd UML uvedeny dva ilustrativní případy generalizace. Abstraktnější třídy Osoba a Lokalita deklarují datové atributy a metody, jejichž strukturu zdědí specializovanější podtřídy.



Obr. 2 Příklady použití generalizace

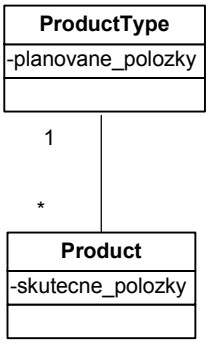
Např. třída *Osoba* deklaruje následující atributy: jméno, rok narození.
Třída *Muž* deklaruje specializovanější atributy: počet manželek a zdědí atributy jméno, rok narození. Třída *Žena* deklaruje specializovanější atributy: počet narozených dětí a zdědí atributy jméno, rok narození.

4.2 Typování

Při typování se místo obecnější datové entity definuje datová entita, která reprezentuje typ této datové entity. Tím, že se při typování obecnější datová entita specializuje a to ne datovou entitou jako při generalizaci, ale instancemi typové datové entity, vytvořená instance dědí nejen strukturu datových atributů obecnější datové entity, ale zároveň i jejich obsah. Na obrázku 3 jsou pomocí diagramu tříd UML uvedeny stejné příklady jako na předchozím obrázku, ale s použitím typování. Jedná se o typování osob a typování lokalit. Uvedené datové entity jsou spojeny asociací.

Obr. 3 Příklady použití typování

TypOsoby jsou instance *Muž* (jméno, rok narození, počet manželek) a *Žena* (jméno, rok narození, počet narozených dětí) a instance od třídy Osoba pak patří buď pod *Muže* nebo *Ženu*. Hodnoty datových atributů zůstávají stejné pro všechny specializované instance.
Obdobně *TypLokality* představuje konkrétní kategorii lokality, kde instance lokalita představuje konkrétní specializaci. Jinými slovy, instance *TypOsoby* a *TypLokality* obsahují obecné informace, které jsou pak přístupné všem konkrétním instancím třídy *Osoba* resp. *Lokalita*.
Typování se dále hlavně používá v případech kategorizace entit např. *ProductType* a *Produkt* viz obr. 4. *ProductType* (Objednávka) obsahuje všechny plánované položky (plánovaná váha, plánované složení příměsí). Instance třídy *ProductType* pak představují různé objednávky lišící se různou hodnotou plánovaných položek. Instance třídy *Product* jsou vždy svázány s konkrétní instancí třídy *ProductType* a liší se v hodnotách skutečných položek (skutečná váha, skutečné složení příměsí).



Obr. 4 Využití typování pro plánované a skutečné položky.

4.3 Shrnutí

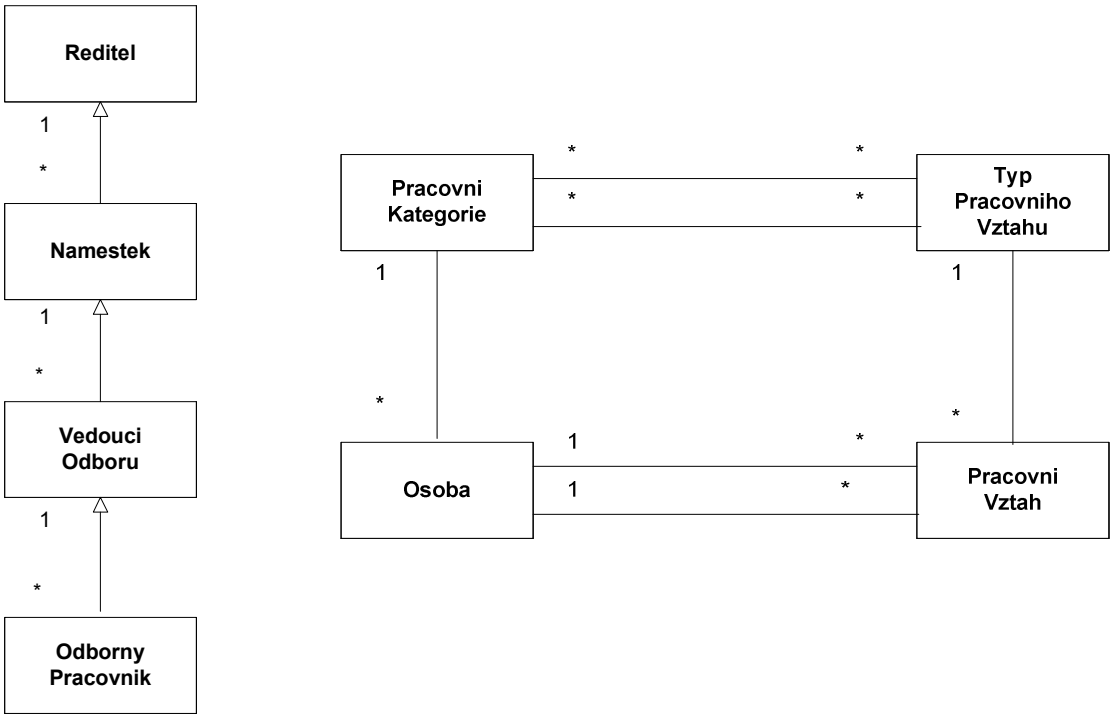
Typování je podobné generalizaci. V obou případech je využit princip obecnější datové entity. Při generalizaci se tato obecnější datová entita specializuje pomocí konkrétnějších datových entit. Při typování se obecnější datová entita specializuje instancemi typové datové entity.
Při typování tak instance typové datové entity odpovídají konkrétnějším datovým entitám u generalizace. Typování se používá v případě, že seznam typů se mění a je malá důležitost dědění. Generalizace se používá v případě, že seznam specializovanějších entit statický a je větší důležitost dědění.

Při generalizaci instance specializovanější datové entity dědí strukturu datových atributů od obecnější datové entity. Při typování specializovanější instance dědí od obecnější instance nejen strukturu datových atributů, ale zároveň i jejich obsah.

Datové entity typ se vzájemnými relacemi patří do metaúrovně (směrnice, obecné zákony). Standardní entity vyjadřují operační úroveň (každodenní skutečnost). Mezi operativní úrovní a metaúrovní existuje vzájemný vztah – operativní úroveň zachycuje konkrétní případy, které musí (měly by být) splňovat pravidla předepsaná metaúrovní.

5 Použití typování v datových modelech

V praxi se často setkáváme s problematikou modelování hierarchie rolí. Využití generalizace je možné, ale přináší celkově strnulé, ne příliš adaptivní řešení, vzhledem ke změnám. Máme tím na mysli změny obsazení dané role jinou osobou, resp. změnu rolí pro danou osobu. Řešením je, nejdříve odstranit danou instanci a pak vytvořit novou instanci s odpovídající rolí.

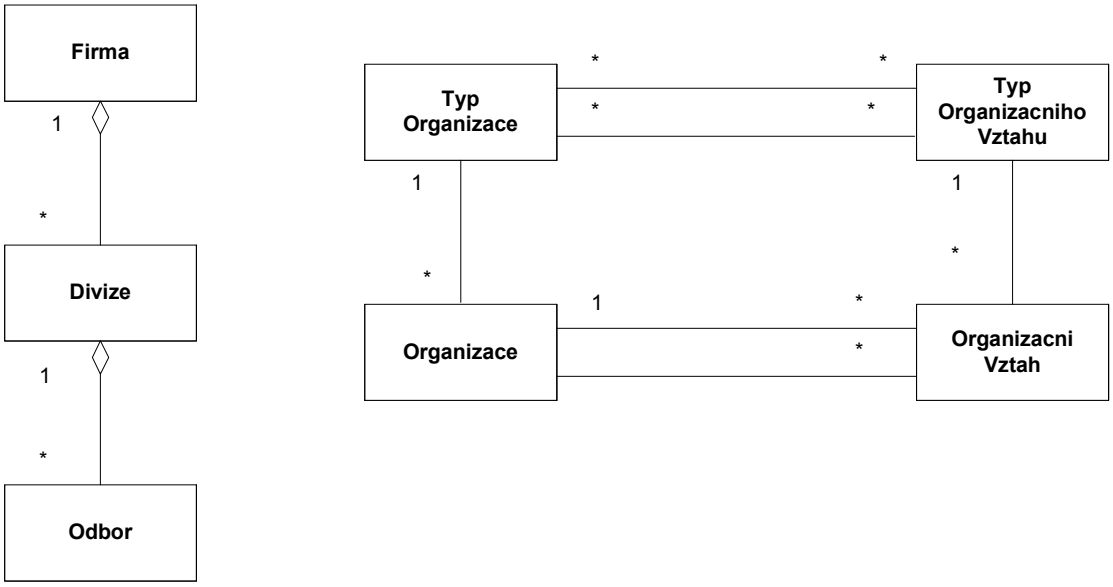


Obr. 5 Řešení pomocí generalizace a typování problematiky hierarchie rolí

Použití typování vede k pružnějším strukturám. Entity (typy), v našem případě *PracovníKategorie* a *TypPracovnihoVztahu*, představují úroveň tzv. metamodelu. Entity *Osoba* a *PracovniVztah*, pak představují úroveň model. Použití typování v tomto případě vychází z návrhového vzoru responsibility uvedeného v [3].

Pracovní kategorie představují konkrétní role, tedy ředitel, náměstek, vedoucí odboru, odborný pracovník. Instance Osob pak sdílí odpovídající pracovní kategorii. Vzor responsibility je, jak je z obrázku zřejmé, ještě doplněn o typ pracovního vztahu a pracovní vztah. Typem pracovního vztahu může být podřízený, nadřízený a tento typ je pak dále charakterizován konkrétní instancí pracovního vztahu. Typ pracovního vztahu a pracovní vztah je nutným doplňkem uvedené struktury.

Následující obrázek popisuje organizační strukturu, která využívá skládání. Použití agregace, v levé části obrázku, vede k nepružné struktuře. Naopak použitím typování umožňuje pružnou strukturu, která snadno akceptuje změny organizační struktury. Obdobně jako na předchozím obrázku, *typOrganizace* a *TypOrganizačníhoVztahu* představuje úroveň metamodelu a entity *Organizace* a *OrganizačníVztah* pak reprezentují konkrétní úroveň modelu.



Obr. 6 Využití generalizace a typování v případě skládání

Využití typování umožňuje také zavedení řady pravidel do meta-modelu, které pomáhají při vlastní realizaci. Atributy typů entit obsahují normativní údaje, které se využívají při implementaci pravidel.

Závěr

Cílem příspěvku bylo ukázat na důležitost typování jako jedné ze sémantických abstrakcí při návrhu konceptuálních modelů. Jeho použití nejenže vytváří pružné změnám přizpůsobivé modely, ale také umožňuje do vrstvy meta-modelu zavést pravidla, která zlepší strukturování a robustnost celého modelu a zlepší také jeho integritu. Typování jako sémantická abstrakce se využívá také v ontologiích.

LITERATURA

[1] Geerts GL, McCarthy (2006) Policy-Level Specification in REA Enterprise Information Systems. Journal of Information Systems. Vol 20, No. 2 pp. 37-63.

[2] Hruby P (2006) Model-Driven Design Using Business Patterns. Springer-Verlag Berlin Heidelberg

[3] Martin J, Odell JJ (1998) Object-Oriented Methods: a Foundation. Prentice Hall PTR Upper Saddle River, New Jersey

[4] Fowler M (1997) Analysis Patterns: Reusable Object Model. Addison-Wesley

[5] Šašera L, Močovský A, Červeň J (2001) Datové modelování v příkladech. Grada

[6] Hučka M., Huňka F., Kašík J., Vymětal D. (2008) Modelování podnikových procesů na bázi vlastnických vztahů a jejich směny (systém REA). Systémova integrace ročník 15, číslo 4, 2008 ISSN: 1210-9479

[7] Vymětal D., Hučka M., Huňka F., Kašík J. (2008) Production Planning Model Using REA Ontology. E + M Ekonomie a Management ročník: XI, 4 / 2008. ISSN: 1212-3609

Příspěvek vznikl za podpory projektu GAČR 402/08/0277 Modelování podnikových procesů na bázi vlastnických vztahů a jejich směny (systém REA).