Meta-modelování

(aneb ontologický statut – operační systém budoucnosti?)

Petr Hřebejk ✌, Václav Řepa ✍
✌ NetBeans a.s., Praha
✍ Katedra informačních technologií Vysoké školy ekonomické, Praha

Móda nebo nutnost?

Meta-data, meta-modely, meta-modelování. To jsou pojmy, s nimiž se lze poslední dobou v počítačové terminologii setkat stále častěji (rozhodně však nikoliv čistěji). Nejen to, někdy lze slyšet dokonce i slova jako meta-meta-modelování, či meta-meta-data. Co si pod podobnými termíny představovat? Odpověď na tuto otázku není ani zdaleka jednoznačná. Předpona "meta" obecně znamená pohled shora, na vyšší úrovni, výskok (vymanění se) ze systému, "za", "přes" (čili přesahující), to, co určuje atd., atp. V oblasti modelování zpravidla vyjadřuje skutečnost, že nějaký systém, model popisuje (na vyšší úrovni) jiné systémy, jiné modely. Jiný model, meta-model meta-modelu, popisuje obecné meta-modely. K čemu ale může být takové vršení meta-modelů dobré? Na tuto otázku se pokusí alespoň stručně odpovědět tento příspěvek, přičemž si neklade za cíl ani úplnost ani dokonalost.
V první řadě je třeba si uvědomit, že vytváření nových, obecnějších úrovní je pro oblast software a počítačů velmi charakteristické. První programy byly psány striktně pro jeden druh výpočtu, veškeré parametry byly obsaženy přímo ve zdrojovém kódu. Proč ne? První počítače měly tak omezené technické prostředky a jejich strojový čas byl tak drahý, že tento přístup byl optimální. Jak se ale začal zvyšovat počet úloh, pro něž se počítače používaly, bylo třeba jednotlivé úlohy zobecňovat a parametrizovat. Náklady na napsání nového programu pomalu převyšovaly náklady na technické vybavení. Podobný trend ostatně dal vůbec vzniknout počítačům v jejich dnešní podobě univerzálních strojů pro výpočty různého druhu. V pionýrských dobách strojových výpočtů nebyl neobvyklým hardware určený výhradně pro jeden druh výpočtu (ostatně, co jiného je dnes 3D grafický akcelerátor?[1]). Tento trend se ale nezastavil. Postupně začala být příliš drahá i manuální parametrizace programů. Software bylo nutné produkovat stále více a doby vývojových cyklů se neustále zkracovaly - a zkracují se doposud. A nejen to, stále větší tlak je vyvíjen na to, aby se nově příchozí pracovníci co nejrychleji seznámili s provozovaným - vyvíjeným systémem a byli co nejrychleji schopni se zapojit do jeho správy nebo vývoje. Stejně charakteristický, jako tento trend, je i způsob řešení problémů, které z něho vyplývají. Je jím zobecňování, generalizace, abstrakce. Pokud se na počátku programovalo v assembleru, později přišlo jeho zobecnění do vyšších jazyků. Nebylo třeba se zabývat instrukcemi a registry. Pokud programátor chtěl opakování podobných akcí, napsal cyklus a nestaral se příliš o jeho konkrétní binární podobu (jeho škoda - sem tam).
Není třeba podrobně popisovat vývoj k dnešnímu stavu věcí, stačí popsat současný stav sám. Programátor potřebuje tabulkový procesor? Prostě použije komponentu spreadsheet a nestará se příliš o jeho implementaci, podobně je to s databázemi, komunikačními protokoly a v některých

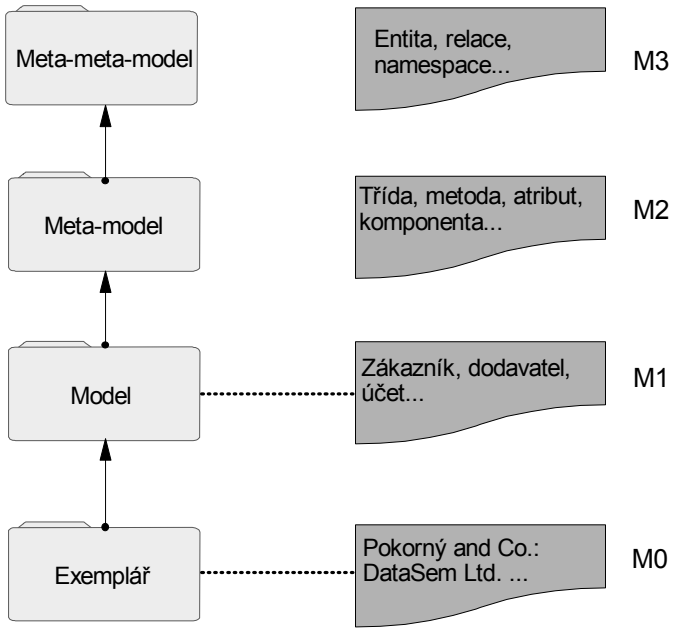
[1] Hle jak krásně se nám zde vývojová spirála dobrala po vývoji negací opět původního bodu. Ovšem, na vyšší úrovni poznání.

oblastech i s celými oborovými řešeními. Potřebuje-li vývojář databázi, komunikační zázemí, transakční processing, **zajímá jej daleko více jejich obecná podoba, nežli konkrétní implementace**. Touto podobou míníme samozřejmě **model**. Návrh rozsáhlého IS nemůže probíhat jinak, než analýzou oblasti, návrhem architektury a rozpracováváním jednotlivých dílčích modelů. Úroveň, na níž lze přestat a zbytek práce přenechat stroji, záleží víceméně na zvolených nástrojích. Použití lidské síly se posouvá stále na vyšší a vyšší úroveň. Dnešní stav je takový, že největší potíže začíná působit integrace jednotlivých druhů modelů - skloubení jednotlivých dimenzí IS (Voříšek J. (1998)). Jak nejvhodněji provázat funkční a datovou stránku IS, nebo jak skloubit dvě části IS tam, kde v jedné je použit strukturovaný přístup a v druhé objektový, jak zajistit propojení a "inteligentní" (nikoliv technickou) výměnu dat mezi nástroji pro modelování procesů v podniku a pro datové modelování? Podobné otázky jsou nejen teoreticky zajímavé, ale jejich řešení může přinést i nezanedbatelný finanční efekt. Řešení je prosté: tak jako již mnohokrát v historii IS, je opět třeba použít zobecnění a posunout se na další metu. Bez nároku na pojmovou přesnost lze tvrdit, že tato meta je meta-meta. Aplikujíce stejné omezení lze tvrdit, že právě řešení těchto otázek je náplní meta-meta-modelování.

Současný stav

Nyní je namístě učinit text poněkud exaktnějším. Obecně je předpona "meta" správně chápana jako relativní. Pohybujeme-li se na nějaké úrovni, je meta-úroveň o jeden stupeň výše. Tento přístup může ovšem být v některých případech poněkud matoucí. V oblasti modelování IS se dnes běžně používá i absolutního významu předpony meta. Je přesně dáno na jaké úrovni se pohybujeme podle toho, kolik předpon meta užijeme. Tento přístup užívá i dnes obecně přijímané rozdělení modelů do čtyř úrovní v tzv. čtyřúrovňové architektuře. čtyřúrovňová architektura tvoří základ všech tří hlavních standardů pro definici mete-meta-modelů (MOF, CDIF, STEP EXPRES[2]).Charakteristika rozdělení modelů do čtyř úrovní je zřejmá z obrázku 1, konkrétní aplikace v MOF je uvedena v tabulce 1, viz též Přílohu 1.

[2] Vhodným zdrojem pro seznámení s jednotlivými subjekty činnými v oblasti meta-meta-modelování je http://www.metamodel.com



obrázek 1 - čtyřúrovňová architektura metamodelů (volně dle Bézivin J. (1998))

Úroveň	**Popis**	**Příklad**
meta-metamodel úroveň MOF	Infrastruktura pro arichtektury metamodelů. Definice jazyka pro tvorbu metamodelů	*MetaClass, MetaAttribute, MetaOperation Mof::Class, Mof::Attribute, Mof::Operation*
Metamodel úroveň UML	Instance meta-metamodelu. Definuje jazyk pro specifikaci modelů	*Class, Attribute, Operation, Component*
Model model uživatelských objektů	Instance metamodelu. Definuje jazyk pro popis informační domény	StockShare, askPrice, sellLimitOrder, *StockQuoteServer*
uživatelské objekty (uživatelská data)	Instance modelu. Definuje specifickou informační doménu.	*<Acme_Software_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>*

Tab. 1 – Čtyři úrovně (meta)modelů v MOF (OMG/MOF (1997))

Důkazem praktického významu výše uvedených teoretických úvah je aktivita ISO v této oblasti. ISO (ISO/IEC JTC1/SC7 N2112 (1999), viz též Přílohu 2) se snaží porovnat současné standardy v oblasti meta-meta-modelování a posoudit možnosti jejich standardizace. Lze tvrdit, že existence obdobného standardu bude v blízké budoucnosti velice žádoucí. Už nyní byly demonstrovány výhody použití MOF a souvisejícího standardu XMI pro přenos meta-dat mezi různými systémy - konkrétně ORACLE - IBM a UNISYS (OMG (1999)). Obecně lze říci, že formalizace jednotlivých druhů modelů (ERD, DFD, diagramy UML, OLAP, jednotlivé programovací jazyky atd.) by měla umožnit snažší sdílení dat mezi těmito modely a tím i urychlit vývoj a snížit závislost na konkrétních produktech a standardech, což jsou obecné požadavky, kladené na informační systémy.

Jiný pohled

Až doposud byl celý výklad velice idylický. Existuje-li tlak na zvýšení výkonnosti (lidí) a nezávislost výsledných produktů, použije se prostě obecnější model a veškerá manuální, nezajímavá a opakující se práce se deleguje na stále výkonnější a levnější hardware. Co více si lze přát? Tento přístup v sobě ale skrývá i nemalá úskalí a nebezpečí. Je třeba si uvědomit, že delegování nepříjemných činností na hardware probíhá vždy pomocí vytvoření nového modelu, v němž se začneme pohybovat. To zároveň znamená, že necháme "pod sebou" všechny ostatní modely. Každý model v sobě ovšem nese charakteristické prvky toho, proč vznikal. Model je obecně **zjednodušení za účelem zkoumání nějakého rysu skutečnosti** (nebo modelu, ležícího pod ním). Co se stane, pokud použijeme několik vrstev modelů, u nichž není účel zobecňování konzistentní? Tato skutečnost v praxi nastává.

Někdy se dokonce účel proč meta-model vzniká, nebo význam, který je jeho vzniku přikládán, liší od člověka k člověku. Přechod assemblerů k vyšším programovacím jazykům je pro někoho zejména příjemná změna v efektivnosti psaní programů, ale najdou se i jiní, a dlužno říci že všeobecně uznávaní, kteří v přechodu ke strukturovaným jazykům najdou všeobecné principy, aplikují je i na datovou stránku věci atd. (Jackson, M.A. (1975)). Jsou ERD obecným modelovacím nástrojem pro vztahy různých druhů, nebo jen a jen cesta k navržení maximálně odpovídající a výkonné relační databáze? Proč někdo používá diagramy STD pro řízení v reálném čase a někdo pro modelování velmi vágních podnikových procesů[3]? Každý z modelů v sobě nese určitou sémantiku.

Podle našeho názoru jsou odlišnosti v chápání různých druhů modelů dobře patrné na přístupu k objektově orientovaným technologiím. Pro někoho je OO cestou ke znovupoužitelnému kódu, pro někoho je to univerzální modelovací nástroj. Zhruba lze říci, že první přístup vede směrem ke komponentovým technologiím, zatímco druhý směřuje k obecným objektovým modelům a CASE systémům. S ryze technickým přístupem k OO se lze setkat například v (Kraval 1998). Dozvíme se zde, že základním a původně jediným charakteristickým rysem OO je enkapsulace (zapouzdření), objekt je tedy prosté shrnutí dat a funkcí do jednoho balíku. Kde je ona známá dědičnost? Ta přichází později jako důsledek programátorské lenosti. Vidíme-li, že objekty jsou si podobné, vytvoříme nový syntaktický konstrukt pro „dědění“. Polymorfismus je zákonitým pokračováním tohoto trendu. Interface je v tomto pojetí pouhým prostředkem pro zajištění kompatibility různých komponent systému. Věříme, že se nyní leckterý objektový guru začne bouřit. Ano, objektům je možno přikládat i obecnější významy, je ale třeba si uvědomit, že jednotlivé prvky OO skutečně vznikaly tímto technickým způsobem, z čehož vyplývají i významné rozdíly mezi tím jak jsou např. pojmy třída, vlastnost, metoda, či generalizace používány v OO a tím, jak jsou používány v jiných oborech. S trochou nadsázky lze říci že dědičnost (generalizace) v OO má k myšlenkovému postupu, nazývanému generalizace, stejný vztah, jako příkaz GOSUB v BASICU. V jiných oborech vytváříme třídy podle jedné nebo několika vlastností objektů. Hierarchií tříd může být více, to znamená, že jeden objekt může být instancí různých tříd[4]. Z náležitosti ke třídě lze usuzovat na různé vlastnosti objektu, přičemž ovšem lze definovat výjimky.[5] Není náhodou, že některé z těchto nedostatků se snaží odstraňovat modely, používané v expertních systémech a v systémech pro

[3] A co s těmi, kdož chtějí k modelování podnikových procesů používat Petriho sítě?!?

[4] Město z hlediska úřednického a požárního bude modelováno dvěma hierarchiemi tříd i když jde stále o jedno a totéž město – význam obecných souvislostí mezi modely je zřejmý

[5] Patří objekt do třídy pták? Pak létá, nejde-li o tučňáka či slepici, ovšem.

reprezentaci poznatků, např. sémantické modely (Delobel C., Lécluse Ch., Richard P. (1995)). V těchto systémech se také začínají objevovat odkazy na souvislosti s ontologiemi, o nichž bude řeč v následujícím odstavci.
De facto lze říci, že dnes se běžně vyskytují pokusy skloubit meta-meta-modelování s tzv. „ontologiemi". Ontologie v tomto pojetí samozřejmě nemůže být chápána striktně jako nauka o bytí. Ale nelze ani tvrdit, že ontologie ve filozofii a ontologie v oblasti modelování jsou prostě homonyma. Úroveň, na níž postoupilo modelování v oblasti SW, dává vzniknout domněnkám, že by již bylo možno tyto modely na vyšších úrovních propojovat s celkovými představami o světě, resp. o jednotlivých oblastech.
Bézivin (Bézivin (1998)) míní, že pro oblast software engineering je třeba zavést vlastní verzi definice pojmu ontologie. Vychází ze zkušenosti s nejrůznějšími, zpravidla oborově zatíženými definicemi, z nichž jako příklady uvádíme následující:

- logikova měkká definice trojčlenkou praví: "ontologie se má k možnosti jako se má logika k jistotě" (Guarino (1997));
- uměle inteligentní definice upřesňuje: "explicitní reprezentace pojmotvornosti" (Gruber T.R. (1992));
- a pragmatická znalostně-bazální definice již tvrdě útočí: "systém primitivních slovníkových pojmů k použití při budování umělých systémů" (Mizoguchi (1993)).

Bézivin tvrdí, že tento pojem je třeba definovat v kontextu jiných, důležitých pojmů:

a) *slovník* je jazykově závislá množina vysvětlených slov. Je používán v lokálním kontextu a jako takovému je mu cizí universálnost, jakož i formálnost;
b) *slovník dat (data dictionary)* je technickou formou *slovníku*, použitou ve specifickém kontextu vývoje informačního systému
c) *schema databáze* definuje vlastnosti databáze, jako atributy, domény, parametry atd.
d) *taxonomie* je hierarchií pojmů, kde běžnou základní relaci *isA* doporučuje Bézivin ještě doplnit relací *partOf*[6]
e) *mereologie (mereotopologie)* je popisem částí a celků v dané doméně
f) a konečně ***ontologie*** je v tomto kontextu precizní definicí pojmů a relací, existujících v konkrétní doméně (například v dané organizaci, oboru, aplikační oblasti apod.). V tomto smyslu jsou tedy různé úrovně a druhy ontologií. Cílem je zde pracovat se základními pojmy a oprostit se přitom matoucí zátěže problémů, vyplývajících z používání přirozeného jazyka (homonyma, metonyma, polysema...) (Kayser D. (1998)).

Bézivin dále hovoří o třech druzích informací, které by měly být v ontologii obsaženy:

- *terminologická informace*, čili množina pojmů a relací (tzv. *definiční úroveň ontologie*)
- *axiomatická informace*, čili množina výroků nad definovanými pojmy a relacemi (tzv. *axiomatická úroveň ontologie*), popisující základní zákonitosti
- *pragmatická informace*, čili množina „provozních" informací, důležitých pro použití ontologie v jím daném kontextu. Patří sem například způsoby grafické reprezentace jednotlivých meta-prvků modelů atd. (nzv. *vercajková úroveň ontologie*)

Základními vlastnostmi ontologie jsou *sdílení* a *filtrování*. *Sdílením* se myslí shoda názorů mezi různými agenty, založená na přijetí společné ontologie – stejné chápání základních pojmů.

[6] tato myšlénka úzce souvisí s koncepcí dvou základních typů hierarchických abstrakcí, uvedenou v tomto textu kousek níže

Filtrování není ničím jiným, nežli použitím principu abstrakce (viz např. Řepa V. a kolektiv (1999)). „Modely reality“, které vytváříme, zahrnují do svého zájmu vždy jistou účelovou podmnožinu reality, danou zpravidla objektem modelování (zajímá nás jen něco) a též metodikou činností, jejichž součástí tvorba modelu je (metodika určuje co musíme vzít v úvahu, co nemusíme, či dokonce co nesmíme). Proto je nutné odfiltrovat přebytečný zbytek reality, abychom se mohli zaměřit jen na to podstatné, co nás zajímá. Úlohou ontologie je zde obecně určit co nás má z reality zajímat a co nikoliv. V oblasti metodik vývoje IS je dobrým příkladem tohoto použití abstrakce „Princip tří architektur vývoje IS“, ve smyslu účelovosti modelů podrobněji rozebíraný například v (Řepa V. (1999)).
V oblasti metodik vývoje IS se ustálil pohled na věc z trochu jiného úhlu. Je zde zvykem uvažovat o hierarchických abstrakcích v jejich dvou základních druzích (Řepa V. (1999)). Hierarchické abstrakce jsou zde prostředkem rozkladu prvků vyvíjeného systému do detailní úrovně pohledu. Pojmy vyšší úrovně sestávají z pojmů nižší úrovně. Na každé úrovni podrobnosti jsou popsány jednotlivé její prvky (pojmy) a vazby mezi nimi. Prvky vyšších (neelementárních) úrovní popisu jsou *abstraktními* prvky a mohou být popsány na nižší úrovni pomocí prvků, z nichž každý jeden může být jak abstraktním, tak konkrétním (elementárním . dále již nerozložitelným) prvkem (pojmem).
Obecně existují dva základní typy hierarchické abstrakce:

- abstrakce *část - celek (kolektivizace, agregace)*. Tato abstrakce se typicky používá ve funkčním modelu, kde se dělí systém na subsystémy, části subsystémů atd. Pro agregaci je typická principiální neomezenost dělení. Vyšší celek je zcela definován jako souhrn svých částí (nemá jiný význam).
- abstrakce *specifický podtyp - obecný typ (generalizace).* Tato abstrakce se typicky používá v datovém modelu (a potažmo i v objektovém modelu), kde je možné uvažovat o jednotlivých specifických variantách nadřízeného pojmu (entity, objektu). Na rozdíl od agregace není nadřízený celek definován jako souhrn podřízených částí, ale jako nositel jejich společných vlastností (atributů).

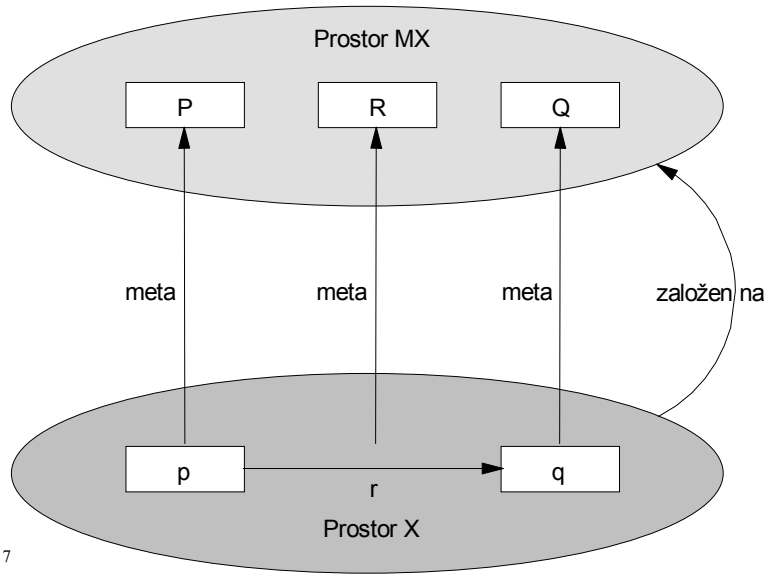
Je důležité, že tyto dva základní typy abstrakce jsou vzájemně neslučitelné (a tím tvoří jádro základního rozporu mezi funkčním a datových modelem - části datového a funkčního modelu jsou tak vzájemně těžko slučitelné).

Ach ontologie! Jde jistě o velmi lákavou představu, je ale třeba ji podrobit hlubšímu zkoumání. Modelování, při svých krocích mezi úrovněmi, je vždy vedeno účelem. Zjednodušuje za účelem zkoumání jednoho charakteristického rysu skutečnosti, abstrahuje od jiných. To kontrastuje s přístupem filozofickým, který se snaží zobecňovat jinak, snaží se vždy zahrnout více a více skutečností pod jednu obecnou tezi. Něco takového dělají i informační systémy. Stále více informací je ukládáno v počítačích, stále více oblastí je transformováno do dat a algoritmů. Ale co ve skutečnosti dělají modely? Pravý opak.

Příklad: Máme skutečnost - více instancí. Za účelem uložení dat o instancích vytvoříme datový model a do něj ukládáme data. To je první zúžení věci. Co uděláme dál - máme víc datových modelů, vymyslíme tedy jejich meta-model, to, jak obecně je budu zapisovat. Jsme tak nadšeni

z toho co se nám povedlo, že si myslíme, že jde o nějakou obecnou závislost, která je platná pro skutečnost. "And this is obviously wrong"[7].

Co dělá filozofie a lidské myšlení obecně? Hledá analogie, dělá agregace podobných skutečností z různých oblastí. Jejich společné rysy má pak za obecně platné principy. Nezatížena pragmatickým účelem, směřuje zpříma k *vše*obecně platným a *vše*zahrnujícím představám. Ontologií pak přirozeně míní tu skutečnou podstatu věcí, nikoliv její projev (či spíše projevy) v dané (ých) oblasti (ech), jak tomu je u meta-modelářů. Pstružina K. (1999) v tomto smyslu doporučuje spíše, nežli „ontologie", používat pojmu „**ontologický statut**". Tento pojem dostatečně zdůrazní, že nejde o tu „pravou a jedinou" ontologii (podstatu všeho bytí), nýbrž o její průmět v dané rovině (oblasti zájmu, oboru, smyslu…). Přesto však nepůjde jen o ledasjaký „model" čehokoliv (resp. jakékoliv části, či varianty čehokoliv), ontologická povaha mu zůstane: základním rysem je zde striktní **požadavek úplnosti** (ten je dán významem pojmu ontologie v aristotelovském smyslu, jako „nauka o bytí jako bytí samo a *vše*, co jeho jest"). Chceme-li tedy meta-model nazývat ontologií (přesněji ontologickým statutem), musí se jednat o model, *zahrnující „veškerý svět" příslušného statutu*. Tento požadavek považujeme za klíčovou charakteristiku meta-meta-modelu, jednoznačně jej odlišující od „pouhého" leckterého meta-modelu, či modelu. Jde současně i o vysvětlení jeho jedinečné role a důležitý krok k rozlousknutí základní otázky absolutnosti, či relativnosti meta-meta modelu.



7 (Bézivin, J. (1998)) - volně citujeme: "*Máme zde dva prostory, představující model X a meta-model MX. Pro každou entitu z X existuje odpovídající meta-entita v MX. V X je definována relace r(p,q), zatímco v MX jsou definovány pojmy P a Q a relace R. Platí vztahy meta(p,P) a meta(q,Q). Model MX je ontologií modelu X, což odpovídá vztahu založen_na(X,MX). Klasickou chybou je zde předpokládat, že platí vztah meta(X,MX), což je"obviously wrong".*". Dodejme ještě, že Bézivin tvrdí, že z analýzy mnoha návrhů meta-meta-modelů (rozuměj spíše jazyků, či nástrojů popisu těchto) mu vyplynulo, že všechny sestávají výhradně ze třech základních složek: **pojmu, relace a prostoru**. Jako problém většiny těchto jazyků vidí, že na této úrovni přidávají zbytné entity, čímž porušují, na této úrovni zcela esenciální, pravidlo minimálnosti jazyka. Přitom tvrdí, že minimálnosti by asi bylo lze dosáhnout za použití kombinace toliko dvou relací: **založen_na** a **rozšiřuje**.

Uzávěr

Je zajisté užitečné a zajímavé přibližovat obecné myšlenkové postupy a koncepce používané v meta-modelování. V textu byly zmiňovány čtyři hlaví oblasti související s meta-modelováním:

- objektově orientovaný přístup
- ontologie ve smyslu znalostně bázových a expertních systémů
- metodiky vývoje IS
- a konečně ontologie ve smyslu filozofickém a ve smyslu lidského myšlení obecně.

První tři oblasti lze, jak bylo řečeno výše, přibližně nazvat ontologickými statuty. Čtvrtá oblast je ontologie sama. Všechny čtyři oblasti spojují určité společné koncepty, či principy. Jsou jimi již zmiňované hierarchické abstrakce, jejich dva základní typy: typ celek-část a typ generalizace-specializace. Hierarchické abstrakce se používají v každé z oblastí a liší se mírou formalizace. Lze říci, že v oblasti OO jsou formalizovány nejvíce a oblast ontologií v tradičním významu tohoto slova je formalizuje nejméně. Při meta-modelování je třeba dbát na to, jak formalizované hierarchické abstrakce užíváme. Je také třeba uvědomovat si rozdíly mezi formálními abstrakcemi, užívanými v ontologických statutech a abstrakcemi jako myšlenkovými postupy, zmiňovanými již v prvním odstavci. Nelze je zbrkle prohlásit za ekvivalentní. Prosté zaměňovaní méně formalizovaných hierarchických abstrakcí za více formalizované a zejména pak záměna ontologie a ontologického statutu může vést k chybné interpretaci vztahu prostoru a meta-prostoru (viz pozn. 7 a související text). Při meta-modelování je nutné vnímat nejen „zužující se“ hierarchii meta-modelů, ale také stejnou měrou se rozšiřující hierarchii meta-prostorů, která vzniká požadavkem na úplnost ontologického statutu v němž se pohybujeme a z něhož hierarchické abstrakce používáme.

Nesmíme se tedy těšit na postupné zjednodušování modelů tak, že např. na úrovni meta-meta-modelu bude tak málo prvků, že vše bude jednoduché a prosté. Zaprvé bude na tuto úroveň zaneseno mnoho implicitní (a potenciálně nekonzistentní) sémantiky z modelů nižších úrovní. A za druhé: čím je model obecnější, tím více případů pod sebe zahrnuje, kterážto komplexita pouze čeká v ústraní na to, aby meta-modeláře zaskočila v ten nejméně vhodný okamžik Na druhou stranu lze tvrdit, že další přibližování nástrojů na všech úrovních meta-modelování k obecným myšlenkovým postupům, hojně užívaným právě v ontologiích, je velmi pravděpodobné.

Literatura

Bézivin J. (1998): Who's Afraid of Ontologies?, Proceedings of OOPSLA'98 Workshop No.25 - CDIF, Vancouver, BC

Delobel C., Lécluse Ch., Richard P. (1995): Databases: From Relational to Object-Oriented Systems International Thomson Publishing London

Hřebejk P. (1998): Metainformační systém, diplomová práce, VŠE, Praha

Gruber T.R. (1992): Towards Principles for the Design of Ontologies Used for Knowledge Sharing, Workshop of Formal Ontology, Padua

Guarino N. (1997): Understanding, Building and Using Ontologies. International Journal of Human and Computer Studies vol. 46 n. 2/3, 1997, pp. 293-310

ISO/IEC JTC1/SC7 N2112 (1999): Study Report on the Feasibility of Mapping Modeling Languages for Analysis and Design Models

Jackson, M.A. (1975): Principles of Program Design, Academic Press, London

Kayser D. (1998): Ontologicaly Yours, invited talk at the ICCS'98 conference, Montpellier, France

Kraval Ilja, Ivachiv Pavel (1998): Základy komponentní technologie COM s příklady ve Visual Basicu 5.0, Computer Press

Lemesle R. (1998): Meta-modeling and Modularity, Proceedings of OOPSLA'98 Workshop No.25 - CDIF, Vancouver, BC

Mizoguchi R.(1993): Expert Systems and Knowledge Base Technology, International Journal of Computer and Engineering Management, Vol.1, No. 2., pp.24-39

OMG (1999): http://www.omg.org/cgi-bin/doc?omg/99-04-04 [Document omg/99-04-04 (Presentation given at the April Meta Data Conference) XML Metadata Interchange (OMG XMI) Distributed Metadata Interchange for the WEB Generation]

OMG/MOF (1997): Meta Object Fasility (MOF) Specification, AD/97-08-05, Object Management Group, Framingham, Mass.

Pstružina K. (1999): diskuse k pojmu „ontologie“, sauna VŠE, Praha

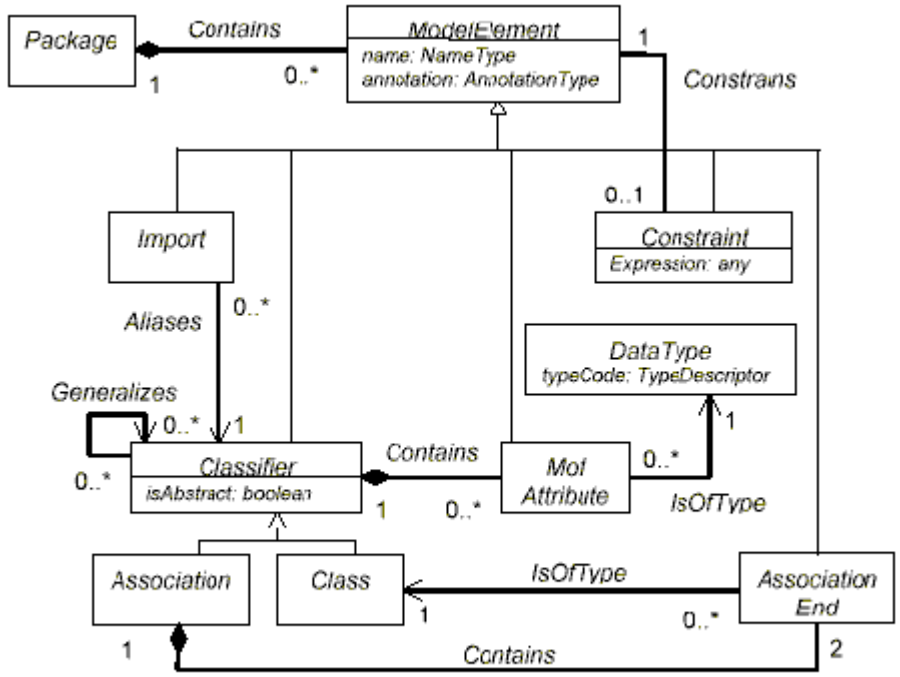
Řepa V. a kolektiv (1999): Metody a techniky vývoje informačního systému, Ekopress, Praha

Řepa V. (1999): Information Systems Development – the BPR Challenge, Proceedings of the ISD 99 International Conference, Kluver Academic Press, Boise, ID.

Voříšek J. (1998): Strategické řízení informačního systému a systémová integrace, Management Press, Praha

Přílohy

Příloha 1 - Model statického jádra MOF (OMG/MOF (1997))



Příloha 2 - Korespondence mezi MOF a CDIF (ISO/IEC JTC1/SC7 N2112 (1999))

Element statického jádra MOF	Element Meta-meta-modelu CDIF
(metameta)třídy	**Meta-meta-entity**
Package	*SubjectArea*
Class	*MetaEntity*
Association	*MetaRelationship*
MofAttribute	*MetaAttribute*
Classifier	*AttributableMetaObject*
(metameta)asociace	**Meta-meta-relace**
Package-Contains-ModelElement	*CollectibleMetaObject.DefinedIn.SubjectArea*
Classifier-Generalizes-Classifier	*AttributableMetaObject.HasSubtype.*
AttributableMetaObject	
Classifier-Contains-MofAttribute	*MetaAtttribute.IsLocalMetaAttributeOf.*
AttributableMetaObject	
(metameta)atributy	**Meta-meta-attributy**
Classifier.isAbstract	*AttributableMetaObject.IsAbstract*
ModelElement.name	*MetaObject.Name*
ModelElement.annotation	*MetaObject.Description*