Objektové modely a znalostní ontologie – podobnosti a rozdíly

Vojtěch Svátek, Martin Labský

Katedra informačního a znalostního inženýrství, Vysoká škola ekonomická v Praze,
nám. W. Churchilla 4, 130 67, Praha 3
{svatek|labsky}@vse.cz

Abstrakt. Objektové modely a znalostní ontologie jsou vyvíjeny relativně oddělenými odbornými komunitami. Jejich vnitřní struktura je však podobná, a často jsou vyvíjeny za stejným účelem: zabezpečit adekvátní strukturu a obsah dat. Ontologie ve srovnání s objektovými modely (UML) kladou větší důraz na přesnější postižení sémantiky reálného světa, a naopak menší na efektivitu zpracování dat. Jejich strojové využívání má charakter odvozování formulí v přesně vymezeném logickém kalkulu. Pro objektově-orientovaného vývojáře je tvorba ontologií spojena s překvapivými rysy. Článek se vedle obecného porovnání obou typů modelů zaměřuje na oblast sémantického webu, v jehož koncepci hrají ontologie klíčovou roli, a pokouší se vymezit vzájemné postavení obou typů modelů v tomto kontextu.

Klíčová slova: ontologie, UML, sémantický web

1 Úvod

Přibližně od 70. let začaly v širokém prostoru informatiky krystalizovat dvě komunity vycházející z téže koncepce – zachycení struktury reálného světa pomocí tříd uspořádaným do hierarchie, kterým pak lze přiřazovat objekty neboli instance. Pro první z nich je charakteristické používání terminologie odvozené od výrazu *objekt*, ve druhé se spíše hovořilo o *rámcích* (frames).

Objektový přístup se rychle vymezil v rámci hlavního proudu softwarového inženýrství, směřujícího k vývoji informačních systémů pro běžné použití, a získal v něm významné (v některých aspektech dominantní) postavení. Jeho rozvoj až do současnosti lze charakterizovat jako plynulý, bez převratných změn. Postupně se objektový přístup rozčlenil na tři volně propojené součásti: objektovou analýzu/vývoj, objektové programování, a (prozatím nejméně rozšířenou) práci s objektovými databázemi. Snaha o standardizaci v 90. letech vyústila ve vznik jednotného zastřešujícího jazyka pro oblast analýzy – grafického UML doplněného jazykem OCL pro specifikaci integritních omezení [3].

Systémy založené na rámcích byly naopak doménou akademického výzkumu, jako jeden z přístupů k reprezentaci znalostí v tzv. umělé inteligenci. Od počátku sloužily pro vývoj speciálních aplikací vyžadujících vedle běžných výpočtů také

logické odvozování, do jisté míry simulující lidský úsudek. Jejich rozšíření do praxe bylo minimální vzhledem ke složitosti jejich návrhu a údržby a malé přívětivosti používaných vývojových jazyků (zejména LISP). V době útlumu zájmu o „umělointeligentní“ aplikace v 80. letech se v oblasti rámců rozvíjel zejména teoretický směr, usilující o jejich podepření propracovanými logickými kalkuly. Těmi se stala na jedné straně rodina *rámcových kalkulů*, z nichž největší popularity doznal *F–logic* [14], na druhé straně pak rozsáhlý soubor podtypů *deskripční logiky* (DL) [5]. Zatímco rámcové kalkuly přímo reflektují podobu rámcových systémů (hierarchie tříd s atributy, doplněná o odvozovací pravidla), deskripční logika jde svou vlastní cestou: logickou teorii vymezuje jako soubor axiomů přiřazujících pojmenovaným třídám logické výrazy, kterým jsou tyto třídy ekvivalentní nebo jsou v nich zahrnuty. Hierarchie tříd proto není v DL uvedena přímo, ale odvozením subsumpce těchto logických výrazů („class expressions“).

V 90. letech se zájem v oblasti rámcových systémů posunul od technologie odvozování k problematice sdílení a znovupoužívání znalostí napříč systémy a aplikacemi. To však nebylo možné, dokud systémy používaly každý svůj vlastní pojmový systém (konceptualizaci). Vyvstala proto potřeba zabezpečení jednotné sémantiky pojmů sdíleným modelem, pro který byl T. Gruberem zaveden pojem *ontologie*, vypůjčený z filozofie. Gruberův jazyk *Ontolingua* [10], definovaný nad obecným predikátovým jazykem KIF, se pak stal prvním a dlouho nejrozšířenějším jazykem pro tvorbu a výměnu takových modelů. Nově vzniklý obor *ontologické inženýrství* se po několik let rozvíjel v rámci akademického znalostního inženýrství, stále ještě mimo dohled praxe. Až na konci 90. let zájem o ontologie výrazně stoupá, a diskuse o nich začíná pronikat i na stránky prakticky orientovaných magazínů a do oddělení IT velkých firem. Je to v souvislosti s formulací ambiciózního projektu *sémantického webu*, u jehož kolébky stojí sám zakladatel WWW a ředitel konsorcia W3C[1], Tim Berners–Lee. Hlavním nástrojem, který má „novému“ webu zabezpečit sémantiku (chápanou jako využitelnost webových informací pro strojové odvozování) jsou právě ontologie. Ontologické jazyky v té době již překonaly závislost na specificky „umělointeligentní“ syntaxi (LISP) a migrovaly k syntaxi XML včetně využívání jmenných prostorů. Oproti ad hoc modelům je jejich předností formální základ, kterým je v současnosti zejména deskripční logika. Navzdory optimismu v akademických kruzích ovšem není úspěch ontologií a sémantického webu jako takového zaručen. Podle názoru (nejen) autora tohoto článku je jedním z významných faktorů dalšího rozvoje právě vymezení podstaty a role ontologií oproti výrazně rozšířenějším objektovým modelům, minimalizace chyb plynoucích z nepochopení odlišností obou typů modelů, a zabezpečení jejich vzájemné komplementarity při vývoji webových aplikací.

Struktura článku je následující. V kap. 2 je popsána koncepce sémantického webu s důrazem na ontologický jazyk OWL. V kap. 3 jsou vymezeny podobné a odlišné rysy ontologií a objektových modelů. Kap. 4 se věnuje aktuálně zkoumaným možnostem synergie ontologického a objektového přístupu. Některé další směry do budoucna jsou pak zmíněny v závěrečné kap. 5.

[1] http://www.w3.org.

2 Ontologie a sémantický web

V průběhu 90. let se začalo stávat zřejmým, že WWW se stane zdrojem informací mimořádného rozsahu. Současně však vyvstávaly problémy spojené s nestrukturovaností a nespolehlivostí informací na něm umístěných. To bylo vítanou příležitostí pro znalostní inženýrství, do té doby tvořící pouze jeden z mnoha rozmanitých směrů umělé inteligence, opět svůj obor přiblížit k reálným aplikacím. V rozmezí let (přibližně) 1996–2002 proto vznikla celá plejáda částečně na sebe navazujících ontologických jazyků, usilujících o doplnění formální sémantiky k webovým stránkám, ev. i aplikacím s webovým rozhraním. V tomto stručném přehledu úvodní pokusy přeskočíme (případné zájemce odkážeme např. na [19]) a budeme se věnovat až nejnovějšímu jazyku, který v sobě zahrnuje zkušenosti z podstatné části předchozího výzkumu.

2.1 OWL – jazyk pro webové ontologie

Jazyk OWL (s trochou dobré vůle lze zkratku vyložit jako „Web Ontology Language“) je výsledkem práce specializované skupiny při W3C [4]; současná verze je uvedena jako doporučení tohoto konsorcia. Existuje ve třech dialektech, z nichž každý následující má vyšší vyjadřovací sílu, avšak je současně náročnější na odvozovací prostředky: OWL Lite, OWL DL a OWL Full. Ontologie v jazyce OWL se skládá z hlavičky (obsahující např. informace o verzi nebo odkaz na ontologie, které jsou v ní implicitně obsaženy) a vlastního obsahu. Tím je sada axiomů, které definují třídy, individua, vlastnosti, a vazby mezi nimi.

Třídy. OWL umožňuje pracovat jak s pojmenovanými, tak i s tzv. anonymními třídami. *Pojmenovaná třída* je identifikována svým názvem, kterým je (podobně jako u dalších konstrukcí) URI – Uniform Resource Identifier [7] – to je nezbytnou podmínkou korektního využívání ontologií v otevřeném prostředí webu. *Anonymní třída* odpovídá logickému výrazu nad pojmenovanými nebo i jinými anonymními třídami. Pro libovolnou dvojici tříd lze vyslovit následující typy axiomů: subsumpce, ekvivalence a disjunktnosti. V praxi se nejčastěji používá subsumpce dvou pojmenovaných tříd, která odpovídá vztahu speciálnější a obecnější třídy v běžném chápání, a subsumpce (ev. ekvivalence) pojmenované třídy vůči anonymní třídě vymezené pomocí hodnoty určité vlastnosti (tj. pomocí tzv. „property restriction“). Příkladem anonymní třídy může být třída objektů, které mají vlastnost „být vlastněn“ vzhledem k alespoň jedné instanci třídy *osoba*. Pojmenovanou třídu *byt v osobním vlastnictví* můžeme pak definovat jako ekvivalent průniku třídy *byt* a výše zmíněné anonymní třídy.

Individua. *Individuum* (též instance či objekt) je vždy identifikováno pomocí URI. Jeho existence je nezávislá na jakékoli třídě; naopak může být přiřazeno nejen k jedné, ale i k více třídám zároveň. Axiomy umožňují stanovit odlišnost nebo naopak shodu dvou individuí (uvedených pod různými názvy). Individua nejsou běžnou součástí ontologií, zařazují se zejména tehdy, když je nutné s jejich

pomocí definovat určitou třídu. Např. pokud chceme třídu *převod nemovitosti* vymezit jako množinu transakcí provedených podle jistého zákona sbírky, musíme tento zákon v ontologii definovat jako individuum (instanci třídy *zákon*). Někdy se vedle instancí tříd také hovoří o instancích (tj. prvcích) *relací*; ty však zásadně nejsou součástí ontologií, nýbrž s nimi spojených bází faktů.

Vlastnosti. Také *vlastnosti*[2] jsou identifikovány pomocí URI. Z logického hlediska jde o binární[3] relace nad množinou individuí. Ani vlastnosti tedy nejsou definovány v rámci určité třídy. Axiomy, které se týkají přímo vlastností, mohou vymezovat jejich definiční obor a obor hodnot, obecné matematické charakteristiky – zda se jedná o tranzitivní, symetrické a/nebo funkční relace, dále pak vztah subsumpce[4], ekvivalence a/nebo inverzity dvojice vlastností.

Dědičnost. Z logické podstaty ontologií vyplývá, že omezení hodnot vlastností se dědí z obecnější třídy na speciálnější; obdobně, matematické charakteristiky i obory argumentů obecnějších vlastností se dědí na vlastnosti speciálnější.

2.2 OWL v šatu RDF

RDF a RDF Schema. Základní syntaxí OWL je XML, nikoliv ovšem „nativní" – mezi stromovou strukturu XML a výše popsané struktury OWL je totiž v současné verzi vložena ještě úroveň jazyka *RDF* (Resource Description Framework) [11] včetně jeho vlastní sémantické nadstavby *RDF Schema* (RDFS)[5]. RDF je jednoduchý jazyk umožňující definovat tvrzení ve tvaru *Zdroj–Vlastnost–Hodnota* (neboli *Subjekt–Predikát–Objekt*), přičemž hodnotou může být buď jiný zdroj nebo prostá textová hodnota (literál). Tvrzení představují navzájem nezávislá fakta, a je možné je umístit přímo na webové stránky jako jejich „znalostní anotace", nebo do specializovaných veřejných databází. Zdroje (a to i vlastnosti jako zvláštní druh zdrojů) jsou identifikovány pomocí URI. Jako příklad tvrzení v RDF můžeme uvést trojici z Tab. 1. Vidíme, že tvrzení RDF jsou právě příkladem dat, kterým je možné pomocí ontologií dodat potřebnou sémantiku: můžeme předpokládat, že druhý a třetí prvek tvrzení odkazují na vlastnost resp. třídu z ontologie umístěné na adrese http://www.reality.cz. Vztah mezi OWL a RDF je tedy dvojí: na jedné straně OWL sémanticky rozšiřuje RDF, na straně druhé je RDF syntaktickým prostředkem pro zápis OWL.

Ačkoliv je RDF jazykem velmi jednoduchým po strukturní stránce, totéž nelze říci o jeho formálně–logických vlastnostech. Zasloužila se o to zejména možnost chápat *celé tvrzení* rovněž jako zdroj – *reifikovat* ho – a přiřazovat mu hodnoty vlastností (můžeme pak např. konstatovat, že výše uvedené tvrzení o zprostředkovaném bytě je prohlášeno příslušnou realitní kanceláří – predikátem

[2] V rámcových systémech se používá termín *slot*, v DL termín *role*.
[3] Řada ontologických jazyků ovšem povoluje i relace s vyšším počtem argumentů.
[4] Pro vlastnosti můžeme tudíž definovat hierarchii obdobně jako pro třídy.
[5] Spojení obou jazyků se zpravidla označuje jako RDF/S.

Tabulka 1. Příklad tvrzení v RDF

Subjekt	http://www.realitniXY.cz/zprostredkovani/5678
Predikát	http://www.reality.cz/onto#TypNemovitosti
Objekt	http://www.reality.cz/onto#Byt_v_osobnim_vlastnictvi

tohoto „meta–tvrzení“ pak bude např. „prohlašuje“). RDF, jehož vyjadřovací síla nedosahuje síly predikátového kalkulu 1. řádu, tak překvapivě povoluje konstrukce (dokonce libovolně) vyšších řádů[6].

RDF Schema pak umožňuje zavést nad zdroji a vlastnostmi hierarchickou strukturu tříd. Jde vlastně o ontologický jazyk, který se od OWL liší chudším vyjadřovacím aparátem (vedle hierarchií umožňuje již jen vymezit definiční obor a obor hodnot vlastností) a nedokonalou formální sémantikou.

Příklad. Uveďme si, jak by mohl v syntaxi XML/RDF vypadat fragment ontologie definující výše zmíněný *byt v osobním vlastnictví*.

```
<owl:Class rdf:ID="Byt_v_osobnim_vlastnictvi">
  <owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="\#Byt" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="\#veVlastnictvi" />
        <owl:someValuesFrom rdf:resource="\Osoba" />
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:equivalentClass>
...
</owl:Class>
```

Všimněme se, že definice této pojmenované třídy je založena na *ekvivalenci* vůči anonymní třídě, a ta je vymezena *subsumpcí* vůči dvěma dalším třídám; dvojice subsumpčních axiomů se implicitně interpretuje jako konjunkce, a z množinového hlediska jde tedy o průnik. Již v takto nevelké definici se odkazuje na tři jmenné prostory – RDF (atributy `ID` a `resource` umožňující odkazovat na zdroj), RDFS (subsumpce tříd) a OWL. Výhodou takto vrstveného přístupu je skutečnost, že i aplikace, která nezná OWL ale jen RDF/S, může z ontologie získat určitou informaci. Ze syntaktického hlediska je každá ontologie v OWL korektním souborem tvrzení RDF. Trojice *Zdroj–Vlastnost–Hodnota* jsou ovšem v XML syntaxi RDF nepříliš dobře viditelné, a v případě OWL někdy ani nemají samostatně význam. Naštěstí je uživatel při tvorbě ontologie od syntaxe zpravidla odstíněn, a používá některý z existujících grafických nástrojů.

[6] Základní varianty OWL – OWL Lite a OWL DL – proto reifikaci nepovolují, aby byly zajištěny výhodné výpočetní vlastnosti jazyka.

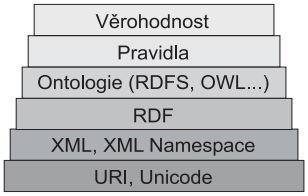
2.3 Příklad aplikace

Ontologie s výše naznačeným zaměřením může sloužit k popisu nabídky realitních kanceláří formálními anotacemi. Inteligentní vyhledávací služba pak např. pro požadavek nalezení nabídek pronájmu bytů od soukromých vlastníků najde nejen záznamy, které vyhovují zadání explicitně (tj. dané individuum je v anotaci přiřazeno třídě `Byt_v_osobnim_vlastnictvi`), ale také anotace, kde je řeč o osobě, která je vlastníkem dané nemovitosti, a zároveň jde o byt. Realitní kanceláře proto nemusí používat jednotnou strukturu informací (předdefinovanou např. pomocí DTD nebo XML Schema), ani není třeba pro každý z jejich katalogů sestavit obalový program (wrapper), který by strukturu konvertoval. Stačí, když budou informace zaznamenávat ve věcné shodě se společnou ontologií.

Ještě sofistikovanějším využitím možností formálního jazyka, jako je OWL, je integrace různých ontologií pro stejnou věcnou problematiku. Odlišnosti definic tříd mohou upozornit na skutečnost, že např. dvě realitní kanceláře používají stejný pojem s odlišným významem. Při omezení na prosté taxonomie (případně DTD nebo XML Schema) je riziko chybného mapování větší.

2.4 Souhrnný pohled na sémantický web

Předmětem tohoto článku jsou především ontologie, vzhledem k jejich souvislosti s objektovými modely; sémantický web je ovšem koncipován jako souhrn několika na sebe navazujících vrstev. Jedno z možných schémat je na obr. 1. Adresování je založeno na *URI*, a univerzální syntaktický standard je *XML*. O úroveň výše stojí *RDF* jako primární datová struktura pro uchovávání faktů. Ještě výše jsou situovány *ontologie*, které faktům dodávají sémantiku. Ontologie mohou samy sloužit jako nástroj pro odvozování, jejich možnosti jsou však omezené. Poměrně dobře lze implementovat algoritmy pro testování jejich konzistence (tj. splnitelnosti tříd) a příslušnosti individuí ke třídám. Z ontologií však nelze vyvozovat nová fakta kromě těch, která pro individua přímo vyplývají z jejich příslušnosti ke třídě. Proto se pracuje na vývoji další, *pravidlové* vrstvy, která bude zřejmě vycházet ze zkušeností jednak s deduktivními databázemi, jednak s věcnými pravidly v informačních systémech, aktivovanými pomocí událostí. Jako poslední úkol k řešení je pak chápáno zabezpečení *věrohodnosti* nalézaných i odvozovaných informací.



Obr. 1. Schéma vrstev sémantického webu

3 Podobnosti a rozdíly objektových modelů a ontologií

Ontologie do jisté míry sjednocují role, které v objektovém světě představují tři relativně odlišné okruhy nástrojů:

- zachycují realitu na konceptuální úrovni, relativně nezávisle na technologických omezeních, proto se podobají konceptuálním modelům v UML
- v prostředí sémantického webu představují jakýsi druh „databázového“ schématu pro báze faktů (RDF)
- jsou zpracovány odvozovacími nástroji v implementované aplikaci, tj. tvoří přímou součást procedurální implementace, jako objektové programy.

Z hlediska „životního cyklu aplikace“ se podstata ontologií (dokonce ani fyzická) nemění při přechodu z fáze analýzy přes projekci až k implementaci. Jsou sice zpravidla vytvářeny v graficky orientovaných nástrojích, ale tyto nástroje je již od první chvíle ukládají ve stejném formátu, který bude používán pro „výpočty v implementované aplikaci“, tj. pro logická odvozování.

Zde se primárně zaměříme na porovnání *modelů*, tj. ontologických jazyků (zejména OWL) a UML. Zajímá nás, co ontologie nabízejí jiného pro lidského uživatele oproti UML, a nakolik může být pro objektové návrháře obtížné si na tento alternativní přístup zvyknout. Databázového pohledu se dotkneme v závěru části 4. Konečně, podobnost s objektovými programy platí pouze na povrchové úrovni, protože cíl a podstata prováděných výpočtů jsou odlišné.

3.1 Porovnání na úrovni konstruktů

Zatímco každá ontologie představuje jeden uniformní model, projekt v UML zpravidla zahrnuje celou řadu komplementárních modelů. S výjimkou diagramu tříd (doplněného o OCL) ovšem vesměs leží mimo záběr problematiky pokryté ontologiemi v jejich současné podobě[7].

Podobnosti. Na první pohled vykazují ontologie a diagram tříd v UML značnou míru shody. V obou se hovoří o třídách, instancích a dědění[8]. V obou případech také existuje možnost definovat např. omezení na kardinalitu – v případě ontologií jde o jejich generickou součást, v případě UML o doplněk vyjádřený v samostatném jazyce OCL. Objektové resp. datotypové vlastnosti v OWL (viz níže) lze také chápat jako analogii asociací a atributů v UML.

Odlišnosti. Při podrobném zkoumání se odlišností vynoří celá řada.

[7] Lze předpokládat, že další „generace“ ontologických jazyků bude umožňovat modelování vývoje v čase, a bude tak porovnatelná i s diagramy stavů ev. aktivit.

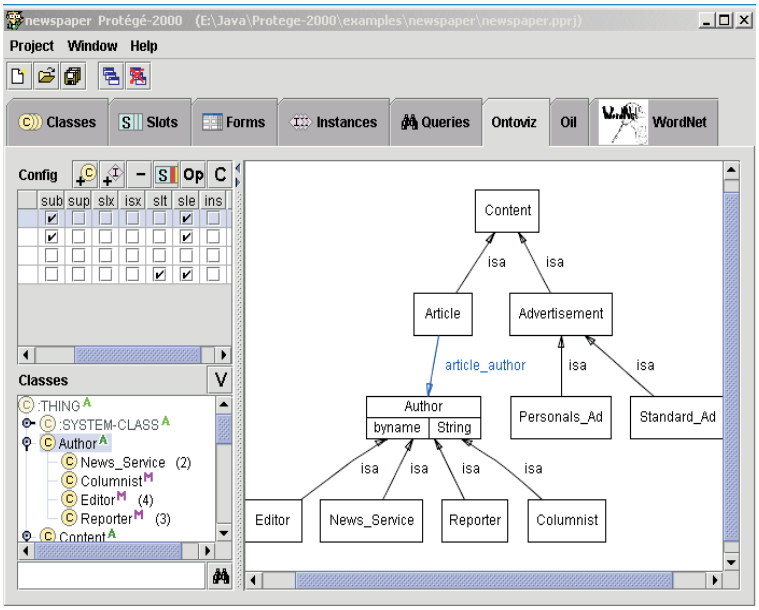
[8] V UML se ovšem dědí atributy, zatímco u ontologií omezení na hodnoty vlastnosti (u tříd), a matematické charakteristiky a obory argumentů (u vlastností).

– Podstatným rozdílem je samostatné postavení *individuí* a *vlastností* v ontologiích[9], zatímco v UML jsou instance, atributy, a de facto i asociace odvozené od tříd. Navíc, zatímco v UML představují atributy a asociace podstatně odlišné konstrukce, v OWL jsou tzv. objektové a datotypové vlastnosti zastřešeny jednotným pojmem vlastnosti. Oborem hodnot *objektové vlastnosti* je pojmenovaná nebo anonymní třída, zatímco oborem hodnot *datotypové vlastnosti* je datový typ definovaný pomocí XML Schematu. Pokud však není datotypová vlastnost definována jako funkční, může nabývat zároveň více hodnot: jakožto binární relace totiž může mít více instancí (tj. dvojic, které jsou jejími prvky) shodujících se v prvním argumentu.
– Třídy v ontologiích jsou často *intenzionálně definovány* pomocí omezení na hodnoty vlastností. Díky tomu lze dobře testovat *konzistenci* ontologie, chápanou jako logickou splnitelnost všech tříd v ní uvedených. Zdá se, že model tříd v UML může být v tomto smyslu nekonzistentní jen v případě vzájemně rozporných (vlastních či zděděných) integritních omezení OCL u téže třídy; testování konzistence v UML ovšem probíhá zejména napříč modely.
– OCL neumožňuje práci s *anonymními třídami*, chybí mu proto aparát potřebný pro jejich konstrukci. Na druhé straně ovšem některá omezení nad hodnotami atributů (tj. „datotypových vlastností“) zavedené v OCL nejsou ve shodě s typem deskripční logiky používaným v OWL. Probíhající výzkum nyní usiluje o formulaci typu DL zachovávajícího výhodné vlastnosti pro odvozování, a přitom umožňujícího definovat potřebná omezení na *datotypových vlastnostech* [17], např. „*datum narození* - *letopočet* > 18“.
– Ontologie jsou *statické* – neumožňují specifikovat změny v čase, zatímco v UML lze (pomocí stavového modelu) popsat *životní cyklus objektů*, který se projevuje ve smyslu změn atributů a vazeb. Existují snahy zabudovat do jazyků typu OWL jistý druh *temporální logiky*. Z koncepce ontologií ovšem vyplývá, že životní cyklus individua bude v takovém případě, vedle změn jeho účasti v instancích relací (zhruba odpovídajících objektovým vazbám) zahrnovat také změny jeho příslušnosti ke třídám.
– V ontologiích se za žádných okolností nevyskytují *procedurální metody*.

3.2 Porovnání infrastruktury pro vývoj a uchovávání modelů

Přestože nástroje pro vývoj a údržbu ontologií zaznamenaly v posledních letech velký pokrok (od textových editorů ke graficky orientovaným rozhraním), poskytovaný komfort nedosahuje úrovně dosažené v oblasti objektového modelování, a to zejména v oblasti *uživatelského rozhraní*. Vedle pozdějšího vzniku disciplíny hraje zřejmě roli i komplikovaná podstata logických modelů – rozsáhlé axiomy lze totiž jen obtížně převést do srozumitelné grafické podoby, a je nutné je editovat (vzhledem k prohibitivní nepřívětivosti syntaxe RDF/XML) v jakémsi textovém pseudokódu. Plně grafickou podobu má tudíž jen práce s hierarchiemi tříd a vlastností. Pro ilustraci uvádíme ukázku prostředí editoru *Protégé* [2], který patří mezi ontologickými editory k nejpoužívanějším (obr. 2).

[9] Zde máme na mysli zejména OWL a další jazyky založené na DL. Rámcové jazyky, jako je F-logic, mají namísto vlastností atributy pevně spjaté s třídami.



Obr. 2. Prostředí editoru *Protégé*

Dalším důležitým aspektem je zabezpečení *persistence* modelů. V tomto směru jsou nevýhody ontologií méně patrné: nástroje pro serializaci ontologií do relačních databází, navržené např. v projektech *KAON* [16] nebo *Sesame* [8], jsou obdobou relační serializace používané pro UML. Je jen otázkou (zřejmě nepříliš dlouhého) času, kdy se i v ukládání ontologií do databáze dosáhne podobné standardizace, jako se to již podařilo pro serializaci v RDF/XML.

4 Sbližování objektů a ontologického inženýrství

Je zřejmé, že větší kontakty mezi oběma komunitami by byly oboustranně prospěšné. Míra obeznámenosti s druhou z problémových oblastí je vyšší u ontologického inženýrství, protože informace o objektovém přístupu tvoří jednu ze standardních složek vzdělání každého informatika. Jako rozhodující se proto jeví rozšíření znalostí potřebných pro tvorbu ontologií mezi objektové vývojáře, tak, aby byli v případě potřeby schopni tvorbu ontologií (zejména v prostředí podnikové praxe) zastat. Rozebereme dvě z možností, jak toho dosáhnout.

4.1 Přímá tvorba ontologií objektovými vývojáři

Na pohled nejjednodušší se zdá být varianta naučit objektově–orientované informatiky tvorbě ontologií se vším, co s tím souvisí. To je možné, pokud dotyčný

projevuje o oblast aktivní zájem[10]. Většinou však nebude motivace (zejména podnikových) informatiků natolik velká, aby studiu ontologického inženýrství věnovali potřebné množství času. Nástroje pro práci s ontologiemi se vyznačují menším komfortem a stabilitou než běžně rozšířené nástroje s objektovou orientací. Navíc je přechod k „ontologickému pohledu“ spojen s nutností změny určitých *modelovacích návyků*. Upřesnění těchto změn je námětem na rozsáhlejší studii, něco však lze vyvodit již ze zkušeností získaných v letech 2002–2003 prvním z autorů článku při výuce předmětu „Modelování znalostí“ na katedře informačního a znalostního inženýrství Vysoká školy ekonomické v Praze.

Zkušenosti s výukou ontologického inženýrství na VŠE v Praze. Náplní předmětu je tvorba modelů v souvislosti s vývojem znalostních aplikací. Jedním ze dvou modelů, které mají studenti za úkol vytvořit v rámci semestrální práce, je právě ontologie zvolené problémové oblasti. Jako formální jazyk byl v r.2002 používán OIL a v r. 2003 DAML+OIL[11]; realizace probíhala v editoru *OilEd*, jehož uživatelské rozhraní je odvozeno ze zmíněného *Protégé*. Z celkového počtu cca 35 studentů jich většina studovala hlavní specializaci „Informační technologie“ garantovanou katedrou informačních technologií, jejich studijní profil měl proto podstatně blíže k běžné podnikové informatice (zahrnující i objektový přístup) než ke znalostním technologiím.

Ve studentských ontologiích se opakovaně objevily znaky nasvědčující, že přechod od objektového modelování k ontologiím vyžaduje *změnu návyků*:

- Do ontologie byly ve velkém množství zařazovány *datotypové vlastnosti* relevantní jen *pro určitou třídu*. Jednalo se jak o vlastnosti s *otevřeným oborem hodnot* – např. jméno, příjmení a rodné číslo (osoby) – tak o vlastnosti s *uzavřeným oborem hodnot* – např. booleovské: „zdanitelný“ (příjem) ANO/NE, nebo vícehodnotové: (řidičský průkaz) „skupiny“ A/B/C atd. Jde zjevně o napodobování *atributů* z objektových modelů.
  Studentům bylo třeba opakovaně zdůrazňovat, že silné stránky ontologií se uplatní zejména v souvislosti s *objektovými* vlastnostmi[12]. Datotypové vlastnosti prvního typu (s otevřeným oborem) jsou použitelné pouze pro uchování dat, které není hlavní motivací tvorby ontologií – tou je zajištění možnosti aktivního odvozování. Datotypové vlastnosti druhého typu (s uzavřeným oborem) je zase často možné transformovat na podtřídy (např. *zdanitelný příjem* a *nezdanitelný příjem*).
- *Individua* byla zaměňována s hodnotami *výčtového datového typu*. Příkladem je třída *vzdělání* s instancemi *základní*, *středoškolské*, atd. Taková reifikace datotypové vlastnosti je konceptuálně značně problematická, vhodnější by bylo (podle kontextu) zavést např. třídy typu *osoba se základním vzděláním*.

[10] Důkazem je například druhý z autorů tohoto článku.

[11] Šlo o předchůdce jazyka OWL, založené rovněž na deskripční logice. Rozdíly mezi těmito třemi jazyky jsou nevelké a nezasahují do podstaty modelovacího procesu

[12] Jak bylo zmíněno v části 3.1, existuje snaha rozšířit ontologické jazyky o možnost specifikace omezení nad datotypovými vlastnostmi; to se však týká zejména vlastností kvantitativního charakteru [17].

– Studenti používali téměř výhradně pojmenované třídy, a to i tam, kde by použití anonymní třídy bylo přirozené – jde–li o třídu, která se uplatní jen v jediném omezení a odpovídá výrazu složenému z relativně nezávislých složek. Příkladem může být třída *fyzická osoba – vlastník nemovitosti*, která se použije v rámci omezení na hodnoty vlastnosti (daň) „se ukládá“ (komu).

Ostatní časté chyby mají evidentně jinou příčinu, než návyky z objektového (ev. funkčního/datového) modelování. Jde zejména o záměnu ontologií se slovníky, glosáři či tezaury (tj. chápání konceptuálních tříd a vlastností jako pouhých jazykových výrazů), nebo narážení na meze vyjadřovacích schopností ontologického jazyka (např. pokusy o postižení relací vyšších řádů). Zejména problém vlivu *jazykové složky* modelu se naopak netýká jen ontologií, ale lze ho zřejmě zobecnit na konceptuální modely jako takové, včetně modelů objektových.

4.2 Objektové metody a nástroje ve službách sémantického webu

Jinou možností je do podoby ontologií dodatečně transformovat UML modely, ať už vzniklé přímo pro potřeby sémantického webu nebo nezávisle (např. rozsáhlé existující knihovny návrhových vzorů). V prvním případě je řešením *rozšíření* UML (na úrovni meta–modelu) o konstrukce zabezpečující sémantiku v ontologiích (vlastnosti, omezení) [6]. Falkovych [9] naopak navrhuje metodu pro automatickou *transformaci* modelu tříd UML do podoby ontologie.

Poněkud jiným směrem vedou návrhy na přímé využití objektových modelů pro podporu sémantického webu [13]. Objektové databázové modely by tvořily nadstavbu nad RDF, tj. nahrazovaly by dnes obvyklou ontologickou vrstvu. To by ovšem znamenalo i změnu pohledu na dynamiku sémantického webu: namísto aktivního *odvozování* by v něm hlavní roli hrálo pouhé *vyhledávání* sémantických informací. Jakkoli taková představa není jistě větší části „znalostní“ komunity po chuti, nelze zcela vyloučit, že se v budoucnu v jisté podobě prosadí, a že „nový web“ bude spíše (distribuovanou) databází nežli znalostní bází.

5 Závěr

Donedávna „čistě akademická“ komunita znalostních inženýrů oproti minulosti více reflektuje vývoj v praktičtěji zaměřených oblastech, ať už se jedná o automatizaci *business procesů* (kombinace sémantického modelu DAML-S s BPEL4WS) [15], využívání *věcných pravidel* [18] nebo již zmíněnou standardizaci *datových typů* (XSD v OWL). Podobně i *rozšíření UML* ve směru k ontologiím a nástroje pro *transformaci z UML* do ontologických jazyků mohou výrazně usnadnit vývoj nových ontologií i znovupoužití rozsáhlých existujících zdrojů referenčních objektových modelů. Využití zkušeností z objektového modelování by však mělo být intenzivnější také v oblastech:

– *výuky modelování* (např. zkušenosti s častými chybami uživatelů)
– vývoje *uživatelsky orientovaných nástrojů*
– *spolehlivosti* uchovávání a aktualizace modelů (např. podpora verzí).

Přebírání těchto zkušeností samozřejmě nemůže být mechanické, nýbrž s přihlédnutím k odlišnostem obou koncepcí.

Na druhé straně lze očekávat, že některé myšlenky vzniklé v oblasti ontologií mohou být přínosné i pro objektovou komunitu. Může jít např. o úlohy *vyhledávání* v rozsáhlých knihovnách modelů nebo o *slučování* nezávisle vzniklých modelů, kde by obohacení o některé logické konstrukty používané v ontologiích (a aplikace odpovídajících formálních kalkulů) mohla vést ke zpřesnění výsledků.

Reference

1. OilEd homepage, http://oiled.man.ac.uk/
2. Protégé project, http://protege.stanford.edu/.
3. UML Resource Page, http://www.omg.org/uml/.
4. Web-Ontology Working Group, W3C, http://www.w3.org/2001/sw/WebOnt/.
5. Baader F., Calvanese D., McGuinness D. L., Nardi D., and Patel-Schneider P. F., eds. *The Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press, 2002.
6. Baclawski K., .Kokar M. K., Kogut P. A., Hart L., Smith J., Holmes W. S., Letkowski J., Aronson M. L. Extending UML to Support Ontology Engineering for the Semantic Web. In: *Fourth International Conference on UML*, Toronto (2001)
7. Berners-Lee T., Fielding R., Masinter L. Uniform Resource Identifiers (URI): Generic Syntax. IETF Draft Standard August 1998 (RFC 2396). http://www.ietf.org/rfc/rfc2396.txt.
8. Broekstra J., Kampman A., van Harmelen V.. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *Proc. 1st Int'l Semantic Web Conference*, 2002, LNCS, Springer Verlag.
9. Falkovych K., Sabou M., Stuckenschmidt H. UML for the Semantic Web: Transformation-Based Approaches. In: Omelayenko B., Klein M. (eds.), *Knowledge Transformation for the Semantic Web*, IOS Press, Amsterdam, 2003.
10. Gruber T.R.: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2) (1993).
11. Lassila O., Swick R. [eds]. Resource Description Framework (RDF) Model and Syntax Specification. World Wide Web Consortium Recommendation, 1999.
12. Brickley D., Guha R.V., eds. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 30 April 2002. http://www.w3.org/TR/rdf-schema.
13. Güttner J., Hruška T. Sématický Web jako flexibilní databáze. *Datakon* 2003.
14. Kifer M., Lausen G., Wu J. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741-843, 1995.
15. Mandell D. J., McIlraith S. A. Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. *Proceedings of the Second International Semantic Web Conference* (ISWC2003), Sanibel Island, Florida, 2003.
16. Motik B., Maedche A., Volz R. A Conceptual Modeling Approach for building semantics-driven enterprise applications. *In Proc. 1st International Conference on Ontologies, Dataases and Application of Semantics* (ODBASE-2002).
17. Pan J. Z., Horrocks I. Extending Datatype Support in Web Ontology Reasoning. In *Proc. of the 2002 International Conference on Ontologies, Databases and Applications of SEmantics* (ODBASE 2002), Oct, 2002.
18. Schroeder M., Wagner G. (eds.). *Rule Markup Languages for Business Rules on the Semantic Web*. Sardinia 2002.
19. Svátek V. Ontologie a WWW. *Sborník konference Datakon 2002*. Brno 2002.