UML – NĚKOLIK KRITICKÝCH POZNÁMEK

Martin Molhanec

ČVUT-FEL, Technická 2, 166 27 PRAHA 6, Dejvice, Česká republika, tel.: ++420 (2) 2435 2118, email: molhanec@fel.cvut.cz, web: http://martin.feld.cvut.cz/~mmm

Abstrakt

UML patří v současné době mezi nejrozšířenější nástroj určený pro modelování. Nicméně při kritickém pohledu do učebnic toho nástroje zjistíme, že aspoň pro oblast analýzy, je způsob, jakým je mnoho pojmů vysvětlováno, přinejmenším pochybným.

1. Úvod

Tento příspěvek vznikl jako kritická reakce na způsob, jakým jsou UML a jeho pojmy vysvětlovány v některých učebnicích tohoto nástroje. Samotný UML, jak je obecně známo, vznikl jako ambiciózní modelovací nástroj, za jehož vznikem stojí tři největší postavy z oblasti softwarového inženýrství – *James Rumbaugh*, *Grady Booch* a *Ivar Jacobson*. Cílem UML bylo a je nahradit starší metodologie výše zmíněných autorů jedinou, která v sobě absorbuje to nejlepší z nich. Navíc si UML dalo do vínku mnoho dalších cílů – například přesnou syntaxi, rozšiřitelnost, a mnohé další.

2. Příklady chybného vysvětlení

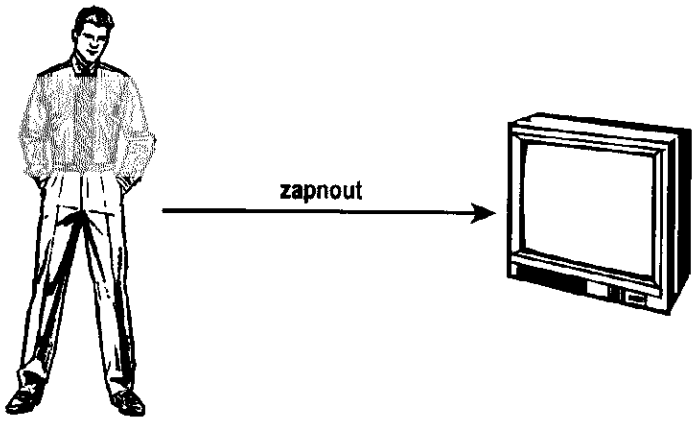
O to víc mne překvapilo, že při četbě některých učebnic UML jsem se setkal s vysvětlováním některých pojmů tak, že z hlediska užití UML pro analýzu systémů, je dle mého názoru mylné až přímo škodlivé. Soustředím se pouze na oblast diagramů tříd, protože tento dílčí nástroj UML přímo koresponduje s diagramem ERD používaném v klasickém datovém modelování.

2.1 Myslíme v jazyku UML

Jako první příklad uvedu některé dle mého názoru chybné vysvětlovaní pojmů z knihy *Myslíme v jazyku UML*, autora *Jaseph Schmuller*, kterou vydala *GRADA Publishing* v roce 2001 v edici *myslíme v ...* s podtitulkem *knihovna programátora*[1].

2.1.1 Směr vztahu

První věc, která mne zarazila a se kterou nesouhlasím se vyskytuje na str. 33. Jedná se o tvrzení že vztah (v knize *asociace*) má směr. Celé vysvětlení je doprovázeno následujícím obrázkem.



Obrázek 2.8:
Objekty jsou často nějak vzájemně spojeny.
Když zapnete televizi, jste s ní ve vztahu jednosměrné asociace.

Vysvětlení směru je doprovázeno následujícím textem.

> Asociace „*zapnout*“ je jednosměrná. Znamená to, že vy zapínáte televizi. Ať už sledujete televizi rádi nebo ne, televize vám vaší náklonnost neoplácí. Jiné asociace, jako např. „*je ženatý/vdaná*“ jsou obousměrné.

Nemohu si pomoci, ale domnívám se, že jde o hrubý omyl. Vztah je *souvislost* mezi dvěma *objekty*. Rozumný vztah mezi *osobou* a *televizí* je například vztah *vlastnictví*. Vzhledem ke gramatice konkrétního jazyka lze takový vztah číst buď jako *osoba vlastní televizi* nebo *televize je vlastněna osobou*. Spíše než o příklad na vysvětlení *vztahu* se jedná o příklad na zasílání *zpráv*! Koneckonců, vysvětlení pojmu *zasílání zpráv* v této knize bezprostředně předchází vysvětlení pojmu *asociace*.

Další nebezpečně matoucí myšlenky se objevují na straně 48 a následující. Nejprve tu máme jeden nevinný obrázek.

Obrázek 4.1:
Asociace mezi hráčem a mužstvem.

Ano, na tomto obrázku není nic škodlivého. Až na skutečnost, že autor opět manipuluje s pojmem *směr vztahu*! Směr vztahu se podle autora označuje malým vyplněným trojúhelníčkem za názvem vztahu. Ve skutečnosti, a stačí se například podívat do knihy *The Unified Modeling Language Reference Manual* [3] autorů *James Rumbaugh, Ivar Jacobson a Grady Booch*, se ovšem jedná o pouhé označení pořadí, jak číst název vztahu! Cituji z uvedené reference: „*Intuitively, the name arrow shows which way to „read“ the name“*.
Nicméně špatný výklad dále pokračuje. Nejprve autor zavede pojem role.

Obrázek 4.2:
V asociaci hraje každá třída určitou roli.
Tyto role můžete znázornit v diagramu.

Aby poté vysvětlil, že *asociace* mohou fungovat také opačným směrem, což doloží následujícím obrázkem.

Obrázek 4.3:
V jednom diagramu se mohou objevit dvě asociace mezi třídami.

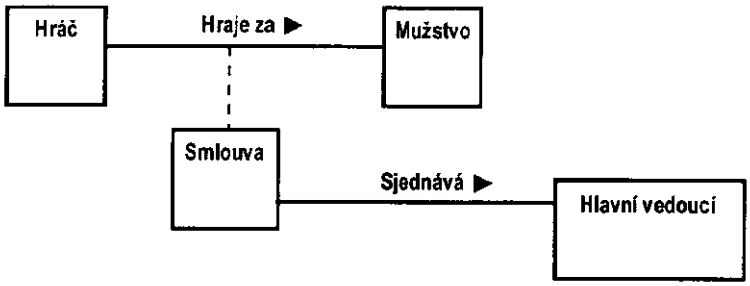
Který vysvětlí následujícím textem.

```
Asociace mohou fungovat také opačným směrem: mužstvo
zaměstnává hráče. Obě asociace můžete znázornit ve stejném
diagramu, směr asociace určuje vybarvený trojúhelník.
```

Ano – je to tak! Autor nás učí, že si vlastně každý vztah můžeme nakreslit v modelu dvakrát – pro každý směr čtení vztahu jednou!

2.1.2 Další možné nesprávnosti

I další části kritizované publikace obsahují některé, dle mého názoru, špatně zvolené příklady. Například obrázek, vysvětlující *vztahovou třídu*, není podle autorů UML správný. Podívejme se na něj.

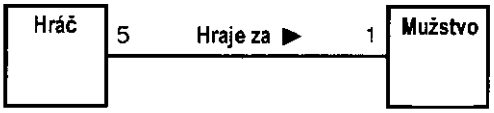


Obrázek 4.7:
Asociační třída je modelem atributů a operací asociace
S asociací je spojena tečkovanou čarou a může být asociována s jinou třídou.

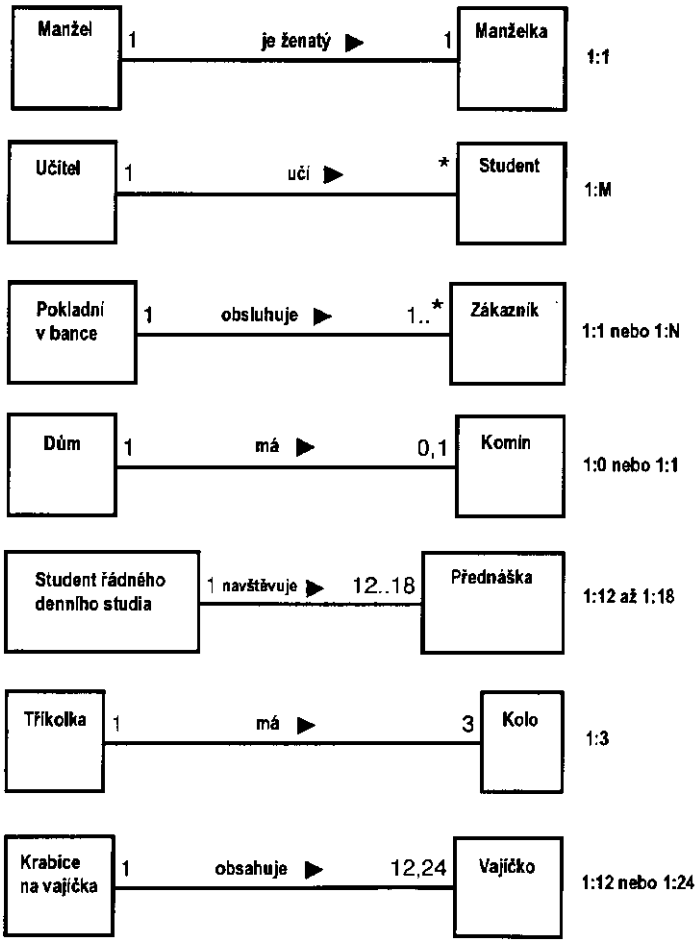
Kde je chyba? Ve výše zmíněné referenční publikaci autoři UML vysvětlují rozdíl mezi *vztahovou třídou* a *zhmotnělým vztahem* (v originále *reified association*), který se ovšem kreslí jako úplně normální třída, takto (volně přeformulováno). *Vztahová třída* je v podstatě čistá *vazební třída*, jejíž *primární klíč* je složen z klíčů tříd na obou stranách vztahu. Pokud takto vytvořený *klíč* není pro jednoznačnou identifikaci výskytů *vztahové třídy* dostatečný, jedná se o *zhmotnělý vztah*. Vzhledem k tomu, že jeden hráč může s jedním mužstvem

postupně v čase uzavřít více *smluv*, je zřejmé, že *primární klíč* nemůže být složen pouze z *primárních klíčů* hráče a mužstva, jedná se tedy o příklad na *zhmotnělý vztah*!

Také další obrázek vyhlíží docela správně, ale jenom do okamžiku než si uvědomíme, že nikde není jako ukázka všech možných různých vztahů také uveden vztah **M : N**! A opravdu! V celé kapitole o vztazích nikde není explicitně nakreslen vztah *mnoho ku mnoho*! Čím pak to asi je? A přijde na to, že vztah **M : N** také existuje, čtenář této učebnice sám od sebe?

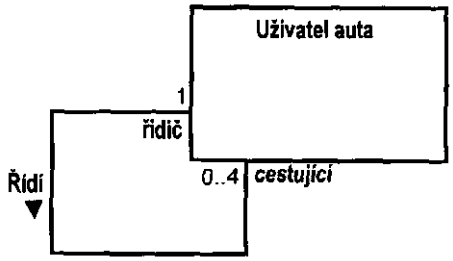


Obrázek 4.9:
Násobnost označuje počet objektů jedné třídy, které se mohou vztahovat k jednomu objektu asociované třídy.



Obrázek 4.10:
Možné násobnosti a jejich znázornění v jazyku UML.

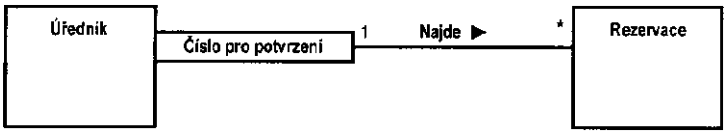
Ani příklad na rekurzivní (v knize *reflexivní*) vztah není nejlepší. Řidič řídí cestující? No vidíte a já si myslel, že řidič řídí auto! A nebylo by lepší použít pro tento příklad např. klasický *kusovník*?



Obrázek 4.12:
V reflexivní asociaci nakreslíte čáru od třídy směrem ke stejné třídě. Můžete také označit role, jméno asociace, směr asociace a její násobnost

2.1.3 Některé vymoženosti, ale pro koho?

UML poskytuje pro kreslení schémat mnoho různých konstruktů. Mnohé z nich jsou však pro kreslení analytických schémat nežádoucí. Jsou to konstrukty vhodné pro oblast návrhu nebo implementace, ale nikoliv pro oblast analýzy. Uveďme si některé konstrukty, které mám na mysli.

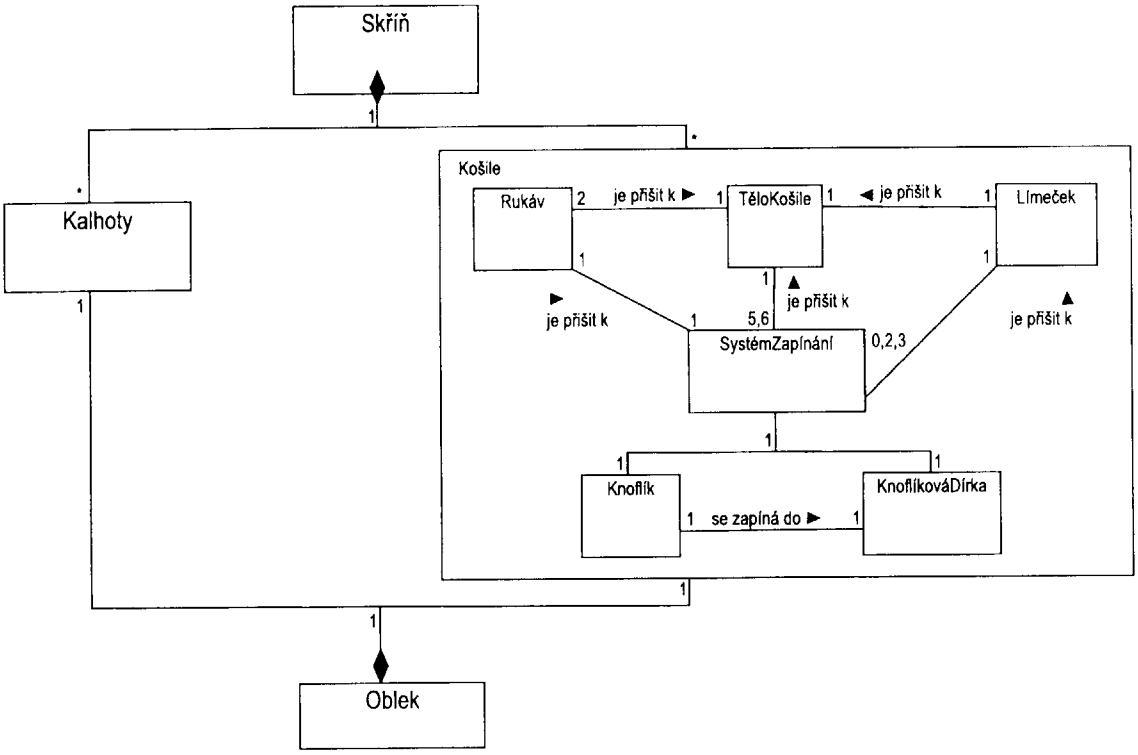


Obrázek 4.11:
Kvalifikátor v asociaci řeší problém s vyhledáváním

Konstrukt *kvalifikátoru* je zřejmě užitečný. Nicméně nejsem si jist, zdali je ve fázi analýzy potřebný. Jak je zřejmé z textu, je tento konstrukt dáván do souvislosti s *vyhledáváním*. Ale ve fázi analýzy řeším model okolního světa a programátorské záležitosti související s vyhledáváním mne přeci vůbec nezajímají! Navíc, rozumíte výše uvedenému příkladu? V učebnici, k výše uvedenému obrázku, žádné zvláštní vysvětlení není.

Obrázek 4.15:
Tečkovaná čára s šipkou vyjadřuje závislost.

Další programátorská vymoženost jsou *závislosti*. Jedna třída je závislá na druhé. Ano, v referenční knize UML je pojem *závislost* definován, je zde definováno i jeho grafické vyjádření. Ale jaký je jeho význam v *datovém schématu*? Navíc se zde konstrukt *závislost* podle UML plete s pojmy *závislá* a *nezávislá entita*. Pojem *závislost* je navíc většinou vysvětlován na příkladu, kdy metoda jedné třídy používá jako parametr jinou třídu. Ale to už je přeci fáze implementace.



Obrázek 5.5:
Diagram kontextu systému znázorňuje komponenty třídy a vztah třídy k jiným třídám v systému.

Tak tohle je také pěkná hrůza. Ano, rozumím co chtěl autor říct, ale co vlastně mám dělat až budu převádět toto schéma do schématu relační databáze? Jaké atributy má *košile*? Jaký je vztah mezi *košilí* a *knoflíkem*? Jinak ovšem souhlasím s tím, že možnost rozkládat schémata tříd (nebo ERD) na menší a snáze myšlenkově uchopitelné celky je vlastnost žádoucí.

2.2 Základy objektově orientovaného návrhu v UML

Druhou knihu, kterou chci zkritizovat je kniha *Základy objektově orientovaného návrhu v UML* od autora *Meilir Page-Jones*, kterou vydala *GRADA Publishing* v roce 2001 v edici *moderní programování*[2].

Podobně jako kniha předchozí i tato kniha zavádí některé pojmy nebo vysvětlení, která mi nejsou pochuti. Domnívám se, že je psaná více pro programátory, nežli pro analytiky. Posuďte ostatně sami.

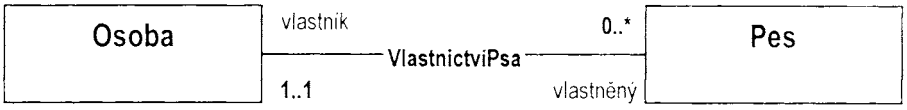
> Operace je *abstraktní*, když nemá žádnou implementaci. *Abstraktní třída* nevytváří instance objektů obvykle proto, že má definovanou alespoň jednu abstraktní operaci.

Ano, je to pěkná definice, čistě objektová, ale není příliš náročná a tím matoucí? Nebylo by lepší vysvětlit pojem *abstraktní třídy* podobně jako to například učinili pánové *Coad* a *Yourdon* ve své knize *Object-Oriented Analysis* [5]. Prostě a jednoduše, že *abstraktní třída* nemá žádné instance objektů? A žádné metody do toho nezatahovat?

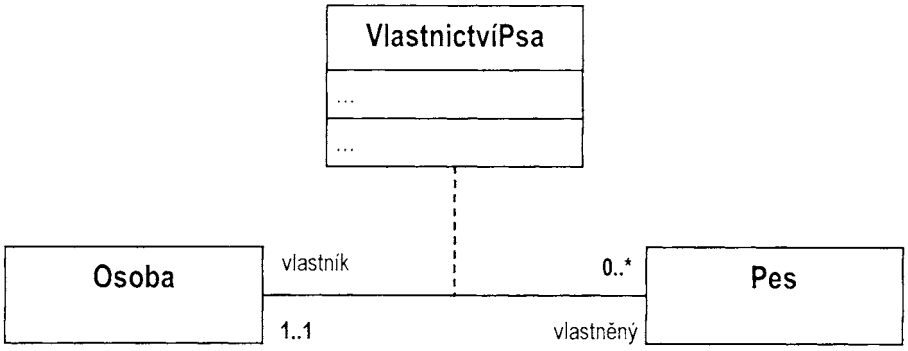
Další věc, se kterou vysloveně nesouhlasím je uvedena na stránce 108 uvedené knihy. Musíme si z ní opět kousek ocitovat.

> Asociace neboli *vztah* se v tradičním informačním modelování obvykle označuje slovesem. Modeláři v objektově orientovaném světě však dávají přednost nazvání asociace podstatným jménem v jednotném čísle. Důvod: asociace je v zásadě třída, jejíž pojmenování podstatným jménem je nejpřirozenější.

Ne, nechci být objektový modelář! Jsem informatik! Co více povědět! Rozdíl mezi třídou a vztahem je použitím podstatného jména pro třídu a slovesa pro vztah jenom zvýrazněn. Skutečnost, jak se co implementuje není rozhodující. Naopak! Pokud v implementaci bude databázová tabulka nazvaná slovesem, vím, že vznikla jako implementace vztahu a jde o vazební tabulku, pokud bude název podstatné jméno, jedná se o regulérní entitu nebo třídu. Jako příklad uvedu následující obrázek.



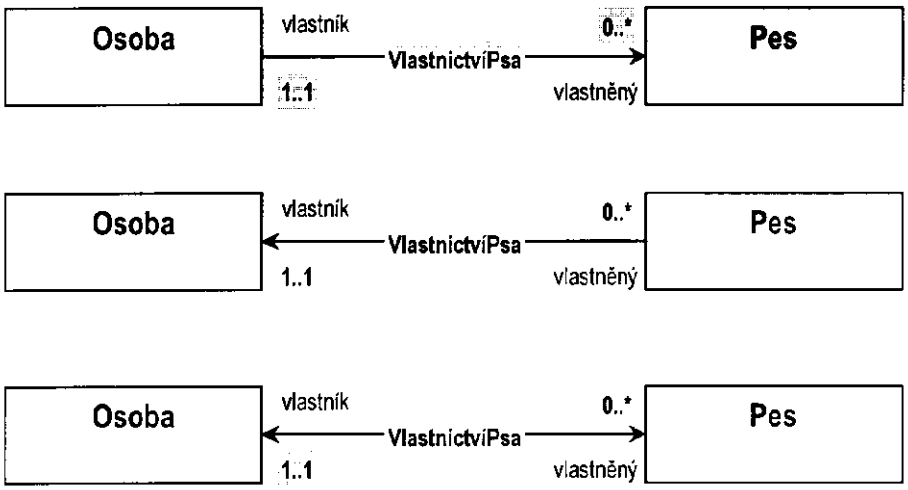
Obrázek 4.12 Základní asociace mezi dvěma třídami



*Obrázek 4.13 Název **VlasnictvíPsa**, povýšený na třídu*

Je patrné, že kdyby v prvém obrázku byl vztah nazván *vlastní*, pak by výše uvedené dva obrázky krásně ilustrovaly skutečnost, že normální vztah je nazván slovesem a vztahová třída podstatným jménem! Bohužel, není tomu tak. Navíc je otázka, nejedná-li se vlastně o zhmotnělou třídu. Proč, to už nechám na úvaze čtenáře samotného.

Další matoucí a pro fázi analýzy zbytečný pojem je *průchodnost* (*navigability*) označovaný jako šipka mezi třídami. Naštěstí v knize samotné je tento pojem spojen až s fází implementace, takže snad pečlivého čtenáře nebude příliš mást. Navíc, z hlediska implementace pomocí relačních databází nemá opravdu žádný význam. *Průchodnost* je ilustrována v námi kritizované knize následujícím obrázkem.



Obrázek 4.15 *Tři možné průchodnosti asociace mezi dvěma třídami*

Abych nezapomněl, význam *průchodnosti* je ten, že ve směru šipky je umožňován rychlý přechod. Čili, pro první příklad na výše uvedeném obrázku, pokud například znám *osobu* pak rychle zjistím, jakého *psa* vlastní. Opravdu by mne zajímalo, jak se změní můj SQL příkaz, když daný vztah bude nebo nebude mít šipku?

2.3 Shrnutí

Nechci obě knihy pochopitelně zcela odsoudit. Obě dvě jsou v mnoha směrech zajímavé a přínosné. Je v nich obsaženo množství příkladů. Bohužel, jak jsem ukázal, ne vždy zcela výstižných. Problém je v zejména v následujících skutečnostech.

- Jsou psány především pro programátory a to zejména v jazyku C++. Proto se na svět okolo nás dívají pohledem implementace a nikoliv pohledem analytika!
- Nezdůrazňují, které konstrukty se mají užívat ve fázi analýzy a které až ve fázi implementace!
- Některé příklady nejsou vhodně zvoleny nebo jsou nedostatečně popsány, takže nejsou buď pochopitelné nebo jsou zavádějící.
- Vždycky je nutné se dívat do referenční knihy autorů UML. Bez jejich vysvětlení, jak jsou jednotlivé konstrukty myšlené, je bohužel četba všech ostatních knih o UML nedostatečná.

3. UML a persistence

Moje kritika UML není pochopitelně jediná. Po té, co jsem zjistil nedostatky ve výše zmíněných knihách, jsem hledal na internetu, zdali neexistují podobné kritiky i ve světě. Zjistil jsem, že existují. Zdá se, že skutečnost, že se UML zmocnili programátoři, kteří si s jeho pomocí dokumentují své programy v C++, vyvolala obecnou reakci kritizující možnosti využití UML v oblasti analýzy a relačních databází. Velice zajímavé články na toto téma napsal *Scott W. Ambler*. První z nich [6] s názvem *Be Realistic About the UML* se zabývá kritikou UML z hlediska nedostatečnosti jeho prostředků pro vývoj *business software*. Z článku je patrné, že autor především kritizuje neschopnost UML poskytovat prostředky pro

návrh *user interface* a *datových modelů*. Kritizuje skutečnost, že například *Activity diagram* v UML neumožňuje zachytit místo pro uložení informací, podobně jako to umí diagram DFD (*data flow diagram*).

Nedostatečností UML v oblasti datového modelování se *Scott W. Ambler* zabývá ve také svém článku [7] s názvem *Persistence Modeling in the UML*. Vyjadřuje zde myšlenku, že UML neposkytuje vhodné prostředky pro datové modelování, a proto zde navrhuje rozšíření notace UML pomocí vhodných *stereotypů*, což jsou prostředky jazyka UML pro jeho vlastní rozšiřování. Soubor určitých *stereotypů*, které rozšiřují UML pro určitou oblast užití, se nazývá *UML extensions*. Podobné názory vyjadřuje i ve svém článku [8] s názvem *Toward Executable UML*.

Přestože je možné s názory *Scott W. Amblera* souhlasit i nesouhlasit, každopádně jsou výsledkem skutečnosti, že mnozí analytici pociťují nedostatečnost UML v oblasti datového modelování.

4. Shrnutí

Záměrem tohoto článku není nikterak naznačit, že UML je špatný nástroj. Naopak, UML je velice kvalitní a bohatý nástroj. Skutečnost, že není ve všech oblastech dokonalý, není nic zavrženíhodného. Ani fakt, že jeho výuka v různých učebnicích není dokonalá, nehovoří o tom, jaké UML ve skutečnosti je. Bohužel, jako učitel zde vidím nebezpečí, že studenti se již dříve z výše kritizovaných učebnic naučí špatně UML používat, nežli jim budu moci v rámci předmětu softwarové inženýrství vysvětlit použití správné!

Nejdůležitější teze, které jsem chtěl v tomto článku vyjádřit jsou následující.

- Učebnice UML jsou v mnoha případech mírně řečeno zavádějící.
- Málokdy se v nich rozlišuje mezi užitím UML ve fázi analýzy a ve fázi implementace.
- UML není v současné době postačujícím nástrojem v oblasti datového modelování a návrhu uživatelského rozhraní.
- Je nutné pečlivě číst referenční manuál od vlastních autorů UML.

Výsledkem výše uvedených tezí jsou pak následující tvrzení.

- UML není konečným řešením v oblasti analýzy a návrhu, jak se nám snaží namluvit někteří prodejci nástrojů, které UML podporují.
- Pokud někdo používá pro vývoj svých programů UML, neznamená to automaticky nejvyšší kvalitu.
- Použití jiných nástrojů než UML, neznamená nijakou degradaci, jak se nám opět snaží namluvit někteří prodejci nástrojů podporujících UML.
- To, že UML podporují velké firmy a stojí za ním velká jména, neznamená automaticky, že nelze nic dalšího než UML použít nebo vytvořit.

Doufám, že tento článek vyvolá širší diskusi o UML – jeho použití a jeho chápání. Možná, že i mé názory nejsou vždy zcela správné. Každopádně si UML zaslouží, aby bylo správně chápáno a vykládáno.

Literatura:

1. Joseph Schmuller: Myslíme v jazyku UML. GRADA Publishing, 2001. ISBN 80-247-0029-8.
2. Meilir Page-Jones: Základy objektově orientovaného návrhu v UML. GRADA Publishing, 2001. ISBN 80-247-0210-X.
3. Booch, G., Jacobson, I., Rumbaugh, J.: The Unified Modeling Language Reference Manual. ADDISON-WESLEY, 1999. ISBN 0-201-30998-X
4. Booch, G., Jacobson, I., Rumbaugh, J.: The Unified Modeling Language User Guide. ADDISON-WESLEY, 1999. ISBN 0-201-57168-4
5. Coad, P., Yourdon, E.: Object-Oriented Analysis. YOURDON PRESS, 1991. ISBN 0-13-629981-4
6. Scott W. Ambler: Be Realistic About the UML. Published at 2001 on http://www.agilemodeling.com/essays/references.htm
7. Scott W. Ambler: Persistence Modeling in the UML. Published at 1999 on http://www.sdmagazine.com
8. Scott W. Ambler: Toward Executable UML. Published at 2002 on http://www.sdmagazine.com