MODELY, MODELOVÁNÍ, MDA A UML

Martin Molhanec
České vysoké učení technické – FEL, K-13113
Technická 2, 166 27 PRAHA 6, Dejvice, Česká republika
tel.: (++420) 2 2435 2118
mailto: molhanec@fel.cvut.cz
http://martin.feld.cvut.cz/~molhanec

ABSTRAKT
Obsahem příspěvku je několik úvah o příčinách nesprávného používání UML, zejména v oblasti analýzy. Dále se příspěvek zabývá klasifikací modelů používaných v oblasti SI (softwarové inženýrství). Věnuje se také tomu, jakým způsobem pro určitý model správně UML použít. Příspěvek navazuje na řadu přecházejících autorových příspěvků a rozvíjí dále myšlenky v nich obsažené.

KLÍČOVÁ SLOVA: softwarové inženýrství, modelování {datové, objektové, konceptuální}, UML, MDA, MDD, ontologie

1 ÚVOD

Jako každý rok, tak i letos, navazuji na své příspěvky z předešlých let uveřejněné na konferenci *Tvorba software* ([2], [4], [6], [7] a [8]) a na konferenci *Objekty* ([1], [3] a [5]). Bezprostřední příčinou pro vznik této série příspěvku, byla nespokojenost autora s tím, jaké znalosti mají studenti VŠ v oblasti modelování a v tom, jak chápou UML. Jaká je vlastně příčina této skutečnosti?

- Znalosti získané mimo školu.

  Velká část studentů se seznámí s jazykem UML, dnes velice módním, již dříve, než je jim poskytnuta základní informace o oboru softwarového inženýrství, prostřednictvím textů na Internetu, dostupných knih a ve firmách, kde již před ukončením studia pracují. Ovšem kvalita těchto zdrojů je mnohdy velice na pováženou!

  - Internet – obsahuje velké množství textů, bohužel však mnohdy velice nízké kvality. Mnohdy vytvářené jinými studenty a odrážející jejich znalosti.

  - Knihy – jsou vydávány především z komerčních důvodů, a to je zcela pochopitelné. Pro vydavatele je hlavní – rychle umístit knihu na trh a pořídit její překlad a práva k vydání co nejlevněji. Není divu, že mnoho knih trpí špatným překladem a nejsou na nejlepší odborné úrovni.

  - Firmy – jenom odrážejí výše uvedené faktory. Jejich použití UML je mnohdy poněkud pokřivené, ale pokud jim „*to*“ funguje, tak je to asi správné.

- Znalosti získané ve škole

  Bohužel ani školy neposkytují to nejlepší. Ano – nesouhlasím s tím, jak k výkladu užití UML přistupují někteří moji kolegové. Jejich výklad trpí nejvíce následujícími nedostatky:

  - Nerozlišováním toho, co z UML použít v různých úrovních modelování, tj. úrovni analytické, návrhové a implementační.
  - Setrváváním na *programátorské* a nikoliv *analytické* úrovni. Vytvářením především modelu aplikace a nikoliv reálného světa, který mají v analýze správně modelovat.
  - Nesprávným výkladem některých UML paradigmat. Například chápáním paradigmatu *případů užití* totožně s paradigmatem *kontextového diagramu*.
  - Nevhodnými příklady na použití UML. Například pro předmět softwarové inženýrství, kde jde především o analýzu a navrhování velkých informačních systémů, použít jako „*vhodný*“ příklad „řízení výtahu“ či „automat na colu“, které se dle mého názoru hodí spíše do výkladu o analýze a návrhu řídících systémů.
  - Nesprávným, nevhodným či zbytečným používáním některých pojmů, které pak přenášejí na stále další generace studentů.

Co z výše uvedených faktů vyplývá?

- Zmatek v komunikaci mezi jednotlivým odborníky. Všichni sice používají UML, ale každý ho chápe jinak!
- Setrvávání ve vlastním omezeném pojetí, a tím potenciální neschopnost dalšího růstu!

2 UML

Co je vlastně UML (Unified Modelling Language)? Obraťme se k oblíbené *Wikipedii* [13]:

> In the field of software engineering, the Unified Modeling Language (UML) is a standardized specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model.

Proti této definici nelze zřejmě nic namítat. Nicméně z ní plyne několik závažných faktů:

- UML není metodika. Na rozdíl od svých předchůdců a zdrojů, jimiž jsou OMT (*Object Modeling Technique*), *Booch Method* a OOSE (*Object-Oriented Software Engineering*), UML nepopisuje striktně, v jakém pořadí se jednotlivé metody (diagramy) používají a jak spolu souvisejí. Toto ponechává tvůrcům metodik, které

využívají UML jako modelovací jazyk. Můžeme si na tomto místě uvést například jednu z nejrozšířenějších metodik postavených na UML a to metodiku RUP (*Rational Unified Process*) či její předlohu metodiku UP (*Unified Process*).

  - Následkem toho, se velice často při výkladu UML autor soustředí pouze na samotné UML. Nebo čtenáři představí pouze jedinou metodiku založenou na UML a to nejčastěji nejrozšířeněji používaný RUP, a utají mu tak možnost užití dalších metodik.

- UML je určen pro *objektové modelování*. Otázka je však, co se pod pojmem *objektové modelování* vlastně skrývá! Zdá se, že nejjednodušší odpověď je ta, že se jedná vhodný způsob modelování systémů, které jsou složené z objektů. Je zřejmé, že v úrovni implementace je určeno především pro modelování systémů vytvořených pomocí OOP (*Object-Oriented Programming*) jazyků.

  - Malá, ale velice důležitá poznámka. ERM (*Entity-Relationship Model*) je typ *konceptuálního* datového modelu. Entity jsou objekty/třídy a v názvu tohoto modelu se mluví o *vztahu* a nikoliv o *relaci*, což nepoučeného odvádí k představě jakési souvislosti s *relační databází*! Avšak – ERM je *objektově orientované paradigma* a nikoliv *relační!*

- Je určeno pro všechny úrovně modelovaní (analýza, návrh, implementace). Z toho však nevyplývá nutnost používat ve všech úrovních všech jeho konstruktů!

  - Typickým jevem této skutečnosti je to, že student v analytickém diagramu neuvede u vztahu žádnou kardinalitu, ani parcialitu, ale nakreslí tam šipku, která však znamená navigabilitu či závislost, což jsou však konstrukty odpovídající nižší, implementační úrovni! Vzhledem k tomu, že většina učitelů se shodně se svými studenty pohybuje též pouze v oblasti objektově-logického datového modelu a myslí si, že toto je vrchol abstrakce, k žádnému vytýkání chyby obvykle nedojde!

Pokud odhlédneme od UML, připomeňme si další typické chyby týkající se modelování obecně, respektive zejména objektově-orientovaného modelování!

- Chyba spočívající v nejasném rozlišení mezi vztahem s kardinalitou 1 : M a kompozicí/agregací. Vždy mne udivilo, že někteří studenti kreslí pouze agregace/kompozice, ale proč nekreslí také vztahy 1 : M? Důvod spočívá zřejmě v následujících dvou faktech:

  - Rozdíl mezi vztahem 1 : M a kompozicí/agregací je správně vysvětlitelný pouze prostřednictvím ontologií, bohužel však takové vysvětlení se studentům v rámci běžné výuky nedostane, naopak je někdy přímo odmítáno.

  - OOP paradigma v sobě bohužel ze tří základních vztahů OOM (dědičnost, skládání a vztah) obsahuje pouze první dva (dědičnost a skládání). Protože však student (a mnohdy i jeho učitel) myslí pouze v této na platformě závislé úrovni je přirozené, že model této úrovni přizpůsobují. Podobný omyl tomuto je například ten, kdy se při datovém ER modelování nekreslí vztahy M : N, které přeci na úrovni fyzického modelu RDB neexistují, a v relační databázi se nakonec vždycky implementují vazební entitou!

- Další typickou chybou je představa, že objektový model není normalizovaný, podrobněji například [9]. Jak jsem se snažil ve svém příspěvku [1] dokázat, je každý konceptuální model ze svého principu normalizovaný! Normalizace se totiž týká nadbytečnosti v modelu a nemá nic společného s relačními databázemi! Je však pochopitelné, že model nižší úrovně, například implementační může být naším rozhodnutím nenormalizovaný!

Na tomto místě je správné připomenout další fakt týkající se definice toho – co je to UML. Použijeme opět Wikipedii [13].

> UML is not restricted to modeling software. UML is also used for business process modeling, systems engineering modeling and representing organizational structures. The Systems Modeling Language (SysML) is a Domain-Specific Modeling language for systems engineering that is defined as a UML 2.0 profile.

Tato definice je velice důležitá. Ukazuje totiž to, že UML není způsob, jak si kreslit strukturu *programu*, ale univerzální a standardní nástroj na modelování prakticky čehokoliv. Proto výklad UML, který tuto skutečnost zatajuje a jednotlivým UML konstruktům dává význam pouze *programovací* je zcela chybný a zavádějící. UML je možné a vhodné použít pro modelování struktury programu, ale toto užití musí být jasně deklarováno. Je zcela nevhodné v základních výukových kurzech, kde se studenti poprvé seznámí s UML, prezentovat UML pouze jako nástroj na modelování struktury programu.

Správné použití UML je neoddělitelné od správného pochopení toho, co který model vlastně znamená!

3 Typy modelů

Jaké modely máme vlastně k dispozici? Jaká je jejich klasifikace? Bohužel oblast softwarového inženýrství je zcela přeplněná různými modely. Jejich definice se z části překrývají, zčásti jsou nejasné a jindy zase totožné. Existuje mnoho důvodů pro tuto modelovou „*Babylonskou věž*“. Uveďme si některé z nich:

- Postupné zužování původního pojmu v čase.
- Postupné rozšiřování původního pojmu v čase.
- Přenesení významu původního pojmu v čase. (Čili, dnes se tím myslí něco jiného než před lety.)
- Jeden název označuje více pojmů.
- Jeden pojem má více názvů.
- Pojem se šíří díky firemní či jiné agresivitě. Je sice určitým způsobem nesprávný, ale přesto se nakonec vžije jeho nekorektní užívání.
- Pojem není správný, ale z určitých důvodů (tradice, zvyklost, hrdost) se nekorektně používá stále.
- Jiné chápání daného pojmu ve světě IT a v jiných světech mimo IT.

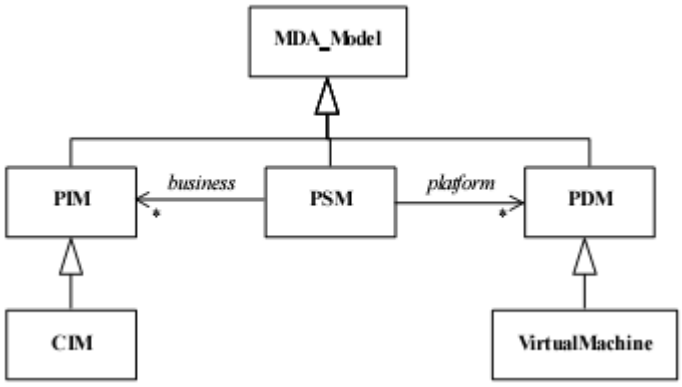
Je zřejmé, že velká část problémů v oblasti modelování vyplývá z neschopnosti se vzájemně domluvit! Kolik vlastně existuje v oblasti IT modelů? Pokusme se jenom o letmý výčet:

- Datový
- Logický
- Konceptuální
- Objektový
- Entitně-vztahový
- Relační
- Ontologický
- Na platformě nezávislý
- Na platformě závislý
- Business
- Fyzický
- Případů užití
- Dynamický
- Kontextový
- Funkční
- Procesní
- Datových toků
- Objektově-relační
- Uživatelský
- ...

Když vezmeme v úvahu to, že za každým výše uvedeným modelem se může skrývat jeho nesprávné chápání, jeho nepřesná či víceznačná definice, může být označením celé skupiny modelů nebo synonymem pro jiné označení téhož modelu, je zřejmé, že je i v oblasti IT modelování nutné učinit pořádek. Odborných prací na toto téma je však poskrovnu.

4 MDE (*Model-Driven Engineering*), MDA (*Model Driven-Architecture*), MDD (*Model-Driven Development*)

Snaha o určité uspořádání vztahů mezi různými modely je například obsažena v koncepci MDE, MDA, respektive MDD. V dalším textu budu používat termín MDA, přestože se nejedná o zcela zaměnitelné pojmy. Vzhledem k tomu, že pojem MDA znamená doslova *architektura řízená modelem*, je zřejmé, že v rámci MDA je obsažena definice různých typů modelů, které je možné v rámci životního cyklu vývoje softwarového produktu využívat. Souvislosti mezi různými typy modelů v rámci MDA dle [11] jsou na obr. 1.



obr. 1: Metamodel MDA.

V rámci MDA se uplatňuje několik následujících modelů:

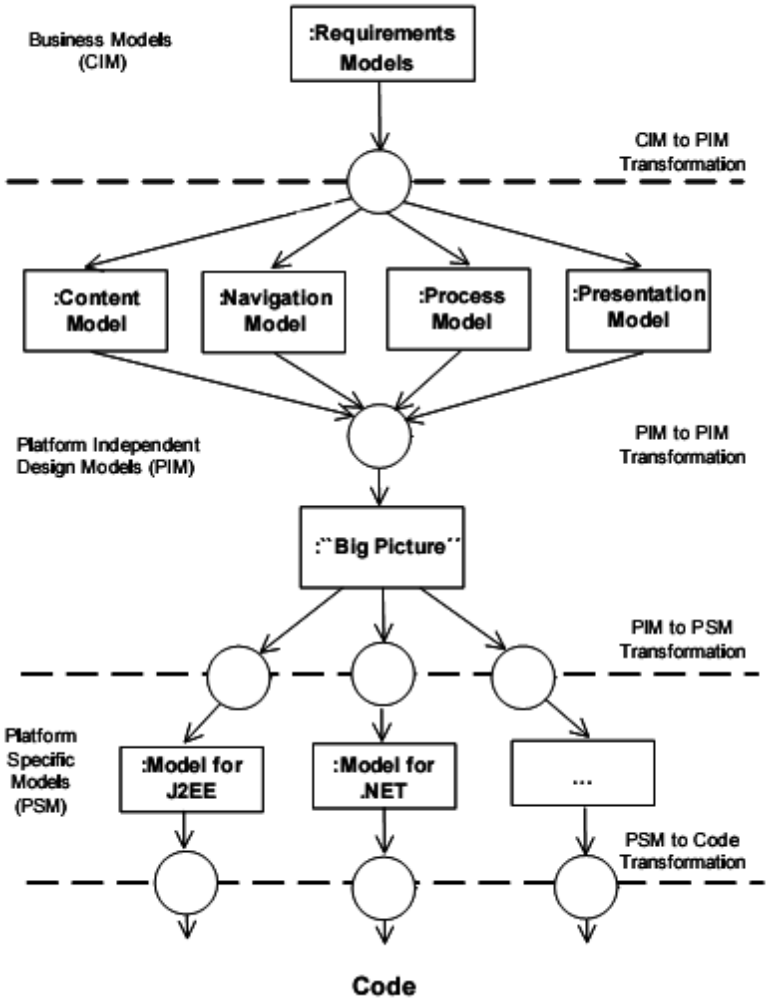
- **Business Model (CIM = *Computation Independent Model*)**
  Tento model tvoří nejvyšší *konceptuální* úroveň všech modelů. Je postaven na *ontologii* pro danou doménu. Je určen pro definování všech požadavků uživatele na činnost daného systému.

- **Platform Independent Model (PIM)**
  Jedná se o celou řadu modelů, vycházejících z *ontologií* pro různé úhly pohledu na modelovanou skutečnost. Podstatné je však to, aby jimi modelovaný systém splňoval to, co popisuje CIM model.

- **Platform Specific Model (PSM)**
  Je logický model pro určitou platformu (objektově orientovaná, relační, strukturovaná, objektově-relační, atp.)

- **Platform Description Model (PDM)**
  Je, to co se někdy nazývá, fyzický model v úrovni abstrakce odpovídající konkrétnímu programovacímu jazyku (C++, Java, Smalltalk) nebo konkrétní databázi (Oracle, Gemstone, XML).

  Je nutné však konstatovat, že samotný popis MDA není zcela přesný. Je objektem výkladů a poslední, leč nikoliv nedůležitá poznámka. V současné době je MDA ™ organizace OMG (*Object Management Group*), a tudíž je kontrolován určitou zájmovou skupinou. Další příhodný diagram ukazující vztahy mezi různými modely v rámci MDA je na obr. 2 převzatý z [11].

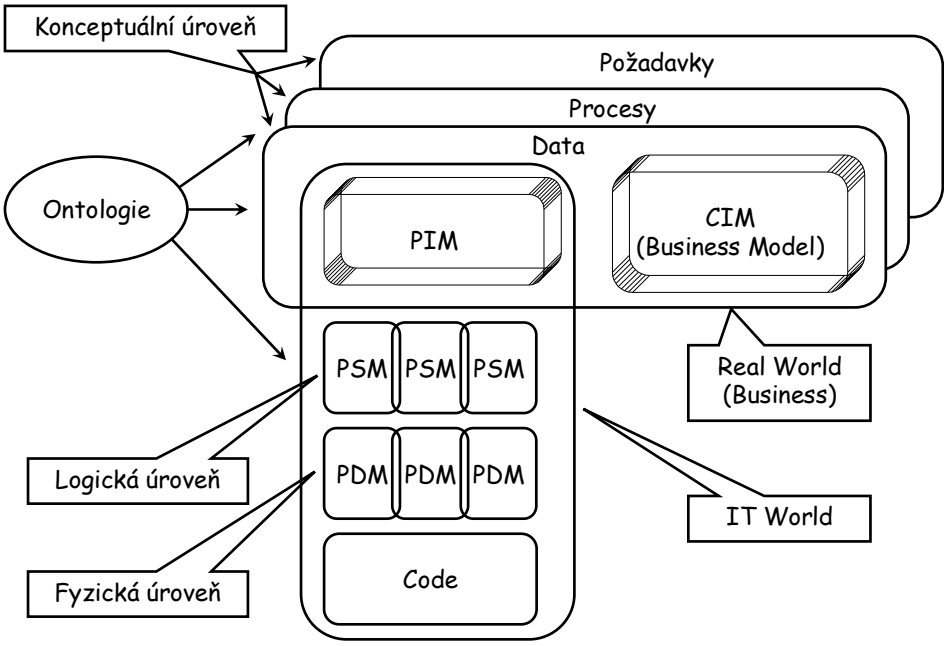


obr. 2: Vztahy mezi modely v rámci MDA.

5 Model modelů

Pokusme se nyní o vlastní pojetí modelu modelů (čili o metamodel modelů). Naše koncepce je na obr. 3. Pokusme se stručně vyjádřit její základní vlastnosti.

- Koncepce modelů je založena na ontologii. Jedině ontologie dovoluje dostatečně přesně stanovit vlastnosti a vztahy mezi různými typy modelů.
- Ontologie je metamodel pro modelování celého universa. To je však v praxi neuchopitelné.
- Z ontologie je možné odvodit různé konceptuální modely, které představují různý úhel pohledu na svět. Například pohled na statickou stránku universa (datový model), pohled na chování objektů v universu (procesní model), pohled na funkcionalitu systému v universu (model požadavků), aj.
- Různé konceptuální modely lze chápat jako podtypy univerzálního ontologického modelu (ontologie).
- Extenze konceptuálních modelů může být rozdílná. Zajímá nás buď extenze vztažená na samotný IT systém (*IT model*), a nebo na všechny objekty určitého systému tento IT systém obsahující (*business model*).
- S ohledem na určitou platformu (paradigma: relační, objektové, strukturované, aj.) se definují modely nižší úrovně (PSM) na základě restrikce intenze modelu PIM.
- Modely PDM představují metamodel pro určitou konkrétní softwarovou platformu. Například: C++, Pascal, RDBMS, OODBMS, XML, atp.
- Existují transformace mezi modely PIM ↔ PSM ↔ PDM.



obr. 3: Metamodel struktury modelů.

Výše uvedená koncepce se snaží o uspořádání modelů na základě ontologie. Je však pouze počáteční ideou další vědecké práce. Jako příklad prací zabývajících se touto problematikou, čili vytvořením ucelené teorie modelů, lze uvést například práce [10] a [12].

6 UML a systém modelů

Jeden z problémů správného užití UML spočívá v nesprávném pochopení sémantiky jednotlivých konstruktů pro daný konkrétní typ modelu, který pomocí UML právě modelujeme. Uveďme si jednoduchý příklad. Konceptuální datový model obvykle rozpoznává tři typy vztahů (což právě vyplývá z jeho ontologie), a to: typovost (***I**nheritance*), skládání (***C**omposition*) a vlastnost (***R**elationship*). Jaké možné modely mohou vzniknout různou restrikcí množiny těchto vztahů? Podívejme se tedy proto na následující tabulku (Tabulka 1).

Tabulka 1: Matice různých typů PSM pro statický datový koncept.

Vlastnosti	Model	poznámka
ICR	Konceptuální	Toto je zřejmé z vlastní definice konceptuálního modelu. Také bychom ho mohli nazvat *„plně objektový“*!
R	ER	Klasický ERM, dle Chena.
IR	XER	Tzv. rozšířený ERM, například IDEF1X
C	hierarchický	Hierarchické DBS?, XML?
IC	OO	Také bychom ho mohli nazvat *„objektově programátorský“*. Je to *logický* OOM odpovídající principům OOP!
RC	???	Model, který má skládání a vztahy, ale nemá dědičnost?
---	???	Takový model má sice objekty, ale žádné třídy, skládání a vztahy. Ze statického hlediska nemá příliš o čem vypovídat. Může však být použit v jiném úhlu pohledu, kdy můžeme využít zde nediskutovaného paradigmatu *zasílání zpráv* mezi objekty! (*Model spolupráce.*)

7 ZÁVĚR

Problematika správného užití modelů při modelování je nesmírně důležitá. Zvláště s ohledem na skutečnost, že UML má jako univerzální modelovací jazyk podporu modelování širokého spektra různých modelů. Přes skutečnost, že samotné UML má pro různé diagramy v něm obsažené specifikovánu jejich základní sémantiku rozšiřitelnou pomocí profilů, je nutné pro správné užití UML s určitým konkrétním modelem, porozumět jeho *„zjemnělé“* sémantice!

Předkládaný článek je první pohled autora na tuto důležitou a doposud nikoliv zcela dobře teoreticky popsanou problematiku, která s nástupem MDD, čili *vývoje řízeného modelem*, nabývá na stále větší důležitosti, podobně jako třeba ontologie, které jsou základním stavebním kamenem porozumění, nejen modelů, ale celého světa – universa ve kterém žijeme.

8 LITERATURA

[1] Molhanec, M., Ontologické základy konceptuální normalizace, In: Objekty 2006. Ostrava: Technická universita Ostrava - Vysoká škola báňská, 2006, s. 81-90.

[2] Molhanec, M., Konceptuální modelování, formální základy a ontologie, In: Tvorba softwaru 2006. Ostrava - Poruba: VŠB - Technická univerzita Ostrava, 2006, s. 121-127. ISBN 80-248-1082-4.

[3] Molhanec, Martin. „Konceptuální modelování“, In Objekty 2005, Ostrava: VŠB, Technická Universita, 2005. ISBN 80-248-0595-2.

[4] Molhanec, Martin. „Zásady konceptuálního totálně objektově orientovaného modelování“ In: Tvorba softwaru 2005. Ostrava: VŠB, 2005, s. 153-158. ISBN 80-86840-14-X.

[5] Molhanec, Martin. „Několik poznámek k porozumění objektového paradigmatu“. Sborník konference Objekty 2004, Praha. ČZU PEF 2004, s. 189-197. ISBN 80-248-0672-X.

[6] Molhanec, Martin. „Kritika některých výkladů objektově orientovaného paradigmatu“. Sborník konference Tvorba software 2004. Ostrava. Tanger, 2004, s. 163-172. ISBN 80-85988-96-8.

[7] Molhanec, Martin. „Objektové metodologie – jejich užití a výklad“. Sborník konference Tvorba software 2003. Ostrava. Tanger, 2003, s. 111-115. ISBN 80-85988-83-6.
On line: http://martin.feld.cvut.cz/~molhanec/VaV/files/publik/2003/OOkrit-co.pdf

[8] Molhanec, Martin. „UML – několik kritických poznámek“. Sborník konference Tvorba software 2002. Ostrava. Tanger, 2002, s. 150-159. ISBN 80-85988-74-7.
On line: http://martin.feld.cvut.cz/~molhanec/VaV/files/publik/2002/UML.pdf

[9] Merunka, Vojtěch, „Datové modelování“, ALFA Publishing, Praha 2006. ISBN 80-86851-54-0.

[10] MDA components: Challenges and Opportunities, Jean Bézivin, S. Gérard, Pierre-Alain Muller, L. Rioux, Workshop on Metamodelling for {MDA}, page 23-41 – 2003

[11] Nora Koch, Gefei Zhang and María José Escalona. Model Transformations from Requirements to Web System Design. In Proc. of 6th International Conference on Web Engineering (ICWE 2006), ACM, 281–288, Palo Alto, USA, July 2006.

[12] J. Bezivin. On the Unification Power of Models. Software and System Modeling (SoSym) 4(2):171--188.

[13] UML, Wikipedia, On line: http://en.wikipedia.org/wiki/Unified_Modeling_Language

Ing. Martin Molhanec, CSc.
V Praze 26. března 2007