

**Vysoká škola ekonomická v Praze**

Fakulta informatiky a statistiky

Katedra informačních technologií

## **Nástroje meta-CASE**

**(charakteristika, vývoj, přehled trhu, trendy)**

Přednášející: doc. Ing Václav Řepa, CSc.



Seminární práce z předmětu 4IT450

-

CASE (Computer Aided System Engineering)

Jan Blaha

Jan Hrabal

Zimní semestr 2008/2009

# Obsah

1. Úvod .....	3
2. Metamodelování.....	3
3. Ukázka metamodelování.....	3
4. Vrstvení metamodelů.....	4
4.1. COMMA.....	5
4.2. GOPRR.....	6
4.3. MOF.....	6
5. Domain specific Modeling.....	7
5.1. UML Profily.....	7
6. Cílové skupiny.....	8
7. Katalog metaCASE nástrojů.....	9
7.1. Eclipse GMF .....	9
7.1.1. Základní práce s GMF.....	9
7.1.2. Závěrečné shrnutí Eclipse GMF.....	12
7.2. Microsoft DSL Tools .....	12
7.2.1. Ukázka vytváření metamodelu pomocí DSL Tools.....	14
7.2.2. Generování kódu.....	16
7.2.3. Validace.....	16
7.2.4. Závěrečné zhodnocení MS DSL Tools.....	17
7.3. MetaEdit+ .....	17
8. Závěr.....	18
9. Použité zdroje.....	19

# 1. Úvod

Cílem této práce je seznámit čtenáře s nástroji metaCASE a principy metamodelování. Práce je strukturovaná tak, že nejprve je metamodelování ukázáno na příkladu, a až dále jsou popsány základní principy metamodelování jako vrstvenost modelů, DSL a UML profily. Velká část práce je tvořena katalogem existujících metaCASE nástrojů. Rozhodli jsme se, že popíšeme především nástroje, které v minulých pracích nebyly zahrnuté, a to zejména Eclipse GMF a Microsoft DSL Tools . V závěru zhodnotíme současnou situaci v oblasti metamodelování a uvedeme vlastní názor na jeho praktickou využitelnost.

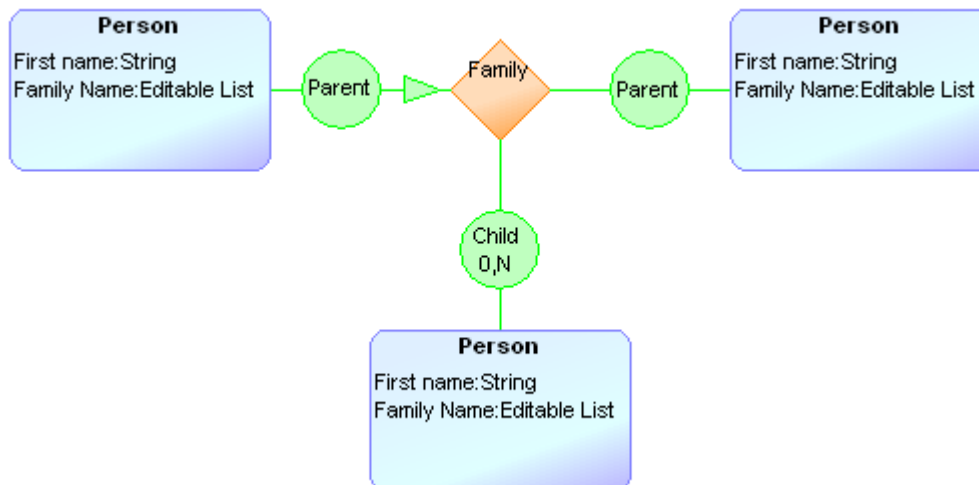
## 2. Metamodelování

K vymezení pojmu metamodelování je potřeba nejprve popsat, co je metamodel. Předpona meta vychází z řeckého jazyka a znamená přidání vrstvy abstrakce. Tedy pokud model je popis skutečnosti v reálném světě, tak metamodel je popis struktury a vlastností daného modelu (schéma modelu). Jinými slovy je metamodel modelem modelu. Nejjednodušší příklad nalezneme v analogii termínů data a metadata, kde metadata můžou představovat schéma relační databáze a data skutečné hodnoty v tabulkách.

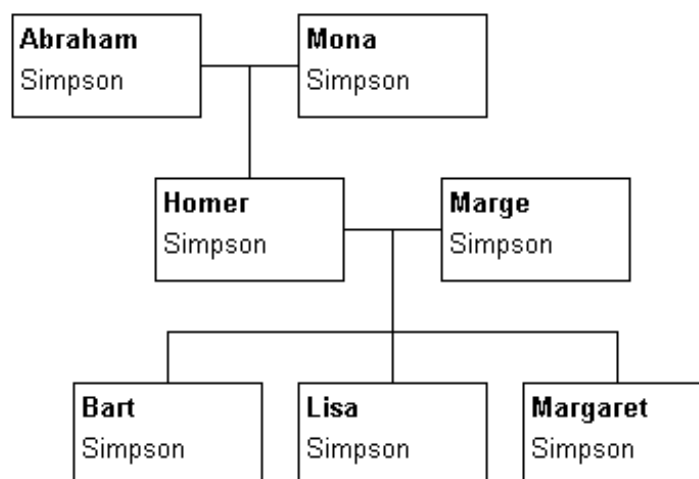
Metamodelování je modelování metamodelů. Pro usnadnění metamodelování byly vyvinuty nástroje metaCASE. V následující kapitole uvedeme jednoduchou ukázkou, jak může metamodelování vypadat, a dále se pak vrátíme k hlubšímu vysvětlení pojmů.

## 3. Ukázka metamodelování

Jako ukázkou metamodelování jsme vybrali metamodel rodinného stromu. Na následujících dvou obrázcích je nejprve zobrazeno, jak vypadá zjednodušený metamodel rodinného stromu a následně jak může vypadat konkrétní model nad ukázaným metamodelem.



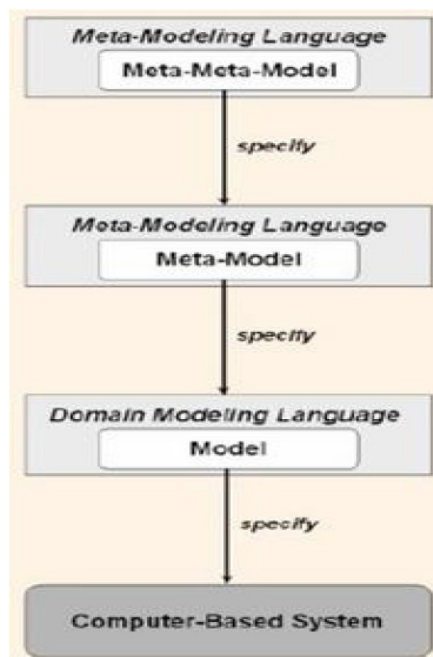
*Ilustrace 1: Metamodel rodinného stromu*



*Ilustrace 2: Konkrétní model*

## 4. Vrstvení metamodelů

Tak jako k modelu existuje jeho metamodel, je možné přidat ještě další vrstvu abstrakce, metametamodel. Metametamodel typicky obsahuje seznam a popis objektů, které se mohou v metamodelech a dále v modelech vyskytovat. Metamodel pak specifikuje, jak jsou tyto objekty využity. Např. metamodel diagram tříd v UML je postaven na metametamodelu MOF [odkaz na kapitolu]. MOF popisuje, že metamodely na něm postavené obsahují třídy, objekty, vztahy atd. Vrstvení metamodelů zobrazuje následující obrázek.



*Ilustrace 3: Hierarchie modelů*

Vrstev abstrakce je možné přidat rekurzivně libovolně mnoho. Obvykle se však tento počet pohybuje mezi dvěma a čtyřmi.

Pro tvorbu metamodelu je tedy potřeba znát metametamodel, na kterém metamodel postavíme. Použitým metametamodelem se pak také odlišují jednotlivé metaCASE nástroje. V dalších podkapitolách popíšeme metametamodely COMMA, GOPPR a MOF. Z této trojice je zdaleka nejpoužívanější MOF, nicméně ostatní přístupy uvádíme z historický důvodů.

#### **4.1. COMMA**

Projekt COMMA (Common meta model architecture) popisuje metamodel na bázi pojmů ze světa objektově orientovaných metodologií. COMMA používá tyto základní pojmy:

- Pojem (Concept) – má jméno a atributy
- Dědění (Inheritance) – vyjadřuje relaci specializace
- Asociace (Association) – vyjadřuje vztah mezi pojmy
- Agregace (Aggregation) – vyjadřuje skládání, je to speciální případ asociace
- Role (Role) – objevuje se, když objekt přijímá charakteristiky jiného objektu. Role je dočasná a objekt může mít i více rolí najednou.

Projekt COMMA už bohužel zanikl, nyní se již tedy metaCASE nástroje používající tento přístup

nerozvíjí.

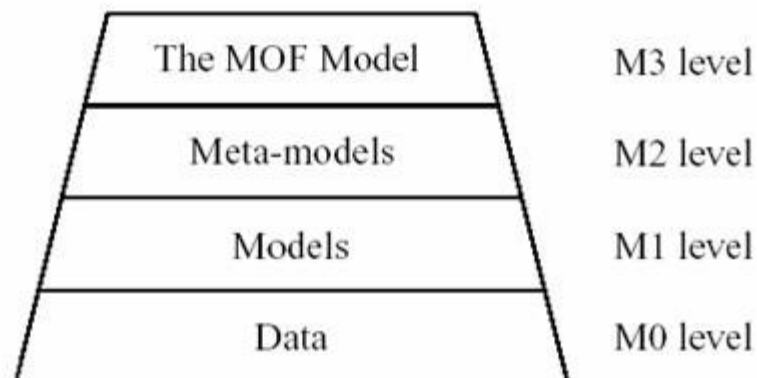
## 4.2. GOPRR

Přístup GOPRR (graph-object-property-role-relationship) nejvíce proslavil nástroj MetaEdit+, kterému se věnuje kapitola [7.3]. Základní prvky GOPRR jsou:

- Diagram (Graph) – je kolekce objektů, vztahů a rolí, která definuje co a jak lze spojovat dohromady.
- Objekt (Object) – definuje entitu, která může existovat sama o sobě.
- Vlastnost (Property) – charakterizuje graf, objekt, roli nebo vztah.
- Role (Role) – existuje mezi vztahem a objektem.
- Vztah (Relationship) – existuje mezi dvěma a více objekty.

## 4.3. MOF

MOF (Meta Object Facility) je standardem konsorcia *OMG (Object Management Group)* pro metametamodel [4]. MOF je založen na představě čtyř vrstev, z nichž vrstva vyšší je metavrstvou k vrstvě nižší. Znázornění vrstvenosti je vidět na obrázku [4]. Jako vrstvu M2 je možné si představit například jazyk UML, jako vrstvu M1 konkrétní model a jako vrstvu M0 vygenerovaný kód z modelu. MOF je dobře popsán v [2], proto zde zmíníme spíše související standardy jako XMI a JMI.



*Ilustrace 4: Vrstvenost v MOF*

Pro přenos metadat namodelovaných pomocí MOFu byl vyvinut konsorciem *OMG* standard XML metadata interchange (XMI). XMI je v podstatě jazyk na bázi XML, který byl vyvinut k popisu modelů na bázi MOFu. Pomocí XMI je tak možné přenést UML model mezi dvěma nástroji, pokud tento standard podporují.

Přestože je velice často zmiňovaný vztah MOFu a UML, tak hlavní náplní MOFu není metametamodel pro UML. Největší využití nachází MOF v oblasti datových skladů a modelování

metadat. Pokud chtějí mezi sebou komunikovat dvě aplikace využívající jiné datové úložiště, tak typicky využijí webových služeb, které je nutno explicitně vytvořit. Pokud by však obě datová úložiště využívaly pro popis svých dat MOF, byla by možná jejich komunikace čistě na bázi metadat. Takovou to komunikaci usnadňuje framework Java Metadata Interface (JMI). JMI umožňuje z popisu metadat v XMI vygenerovat konkrétní Java třídy s příslušnými metodami a s těmi dále pracovat pomocí jazyka Java.

## 5. Domain specific Modeling

Jedním z důvodů vzniku metamodelování byla potřeba vytvářet modely pro specifické oblasti, kde jeden metamodel, jako UML, nemusí být plně dostačující. Modelování pro specifické oblasti, jako např. finance, výrobní průmysl a další, se nazývá Domain specific modeling (DMS). Podstatná část DMS je tvořena generováním jazyka (Domain specific language - DSL) z výsledných modelů. Z modelu popisujícího např. výrobní proces je pak možné vygenerovat program zapsaný v jazyce, který se používá k řízení výrobní linky.

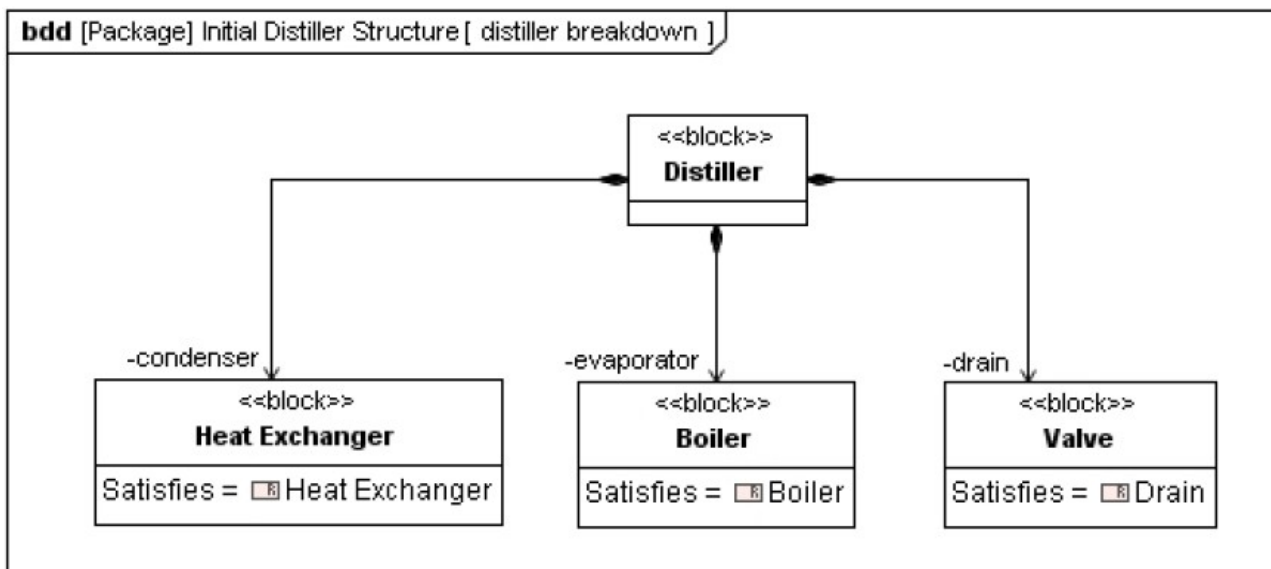
Představme si výrobní halu ovládanou pomocí specifického DSL jazyka. Pokud skupina technických expertů vytvoří metamodel a popis transformování modelů do daného jazyka, který algoritmus výroby řídí, bude možné všechny výrobní procesy jednoduše modelovat a automaticky spouštět bez jakéhokoliv ručního zásahu programátorů do kódu ovládacích programů. Tím se maximálně usnadní přechod od modelů přímo k výrobku. Zde je možnost, aby se prosadily metaCASE nástroje, jako například MetaEdit+, který transformace do DSL podporuje.

Nicméně tvůrci dnes nejmasověji rozšířeného modelovacího jazyka UML se rozhodli, že umožní použití UML i v oblasti DSM, a zavedli standard UML Profily, kterým se věnuje následující podkapitola.

### 5.1. UML Profily

Profily se definují pomocí značek (stereotype, tag), a omezujících pravidel (constraints), které se aplikují na konkrétní elementy modelu, jako jsou třída, atribut, aktivita atd. Pomocí profilů je možné přizpůsobit modely pro konkrétní oblast (finanční, zdravotní, výrobní) nebo pro konkrétní platformu (J2EE, .NET).

Nejznámějším UML Profilem je Systems Modeling Language SysML (<http://www.omg.sysml.org/>). Ukázka diagramu v SysML je na následujícím obrázku.



*Ilustrace 5: Aplikace UML Profilu SysML*

Pomocí profilů je možné např. barevně označit elementy podle typu jejich stereotypů nebo přidat standardním elementům nové vlastnosti. UML profily podporuje většina kvalitnějších CASE nástrojů. V CASE nástrojích si pak můžete vytvořit vlastní profily nebo importovat existující.

Nicméně UML Profily neposkytují tolik možností rozšíření jazyka jako vytvořený metamodel v metaCASE nástroji. Profily nám neumožní například komplexní generování kódu z modelů nebo implementaci validací. Na druhou stranu, protože je jazyk UML velice známý, nekladou vysokou náročnost na schopnosti a vědomosti uživatele, jako je tomu při tvorbě metamodelů.

## 6. Cílové skupiny

Metamodelování je poměrně náročná činnost, protože když tvoříme metamodely, tak spolu s nimi vytváříme i metodiku. Tuto činnost mohou vykonávat jen specialisté. Z toho důvodu jsou cílové skupiny uživatelů používajících nástroje metaCASE a metamodelování poměrně malé.

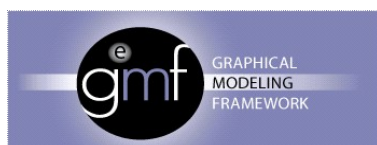
Metamodelování je určeno zejména dvěma cílovým skupinám.

- První skupina jsou uživatelé, kteří si sami vytváří vlastní metamodely a tím tedy i vlastní metodiky. Tato skupina je poměrně malá a největší zastoupení uživatelů má v akademickém světě.
- Druhá varianta je mnohem častější. Výrobci CASE nástrojů často vyvíjí metaCASE, ale před vydáním z něj vytvoří klasický CASE nástroj. Tím zvyšují prodejnost nových verzí. Inovace je pak pro ně mnohem jednodušší, např. při přechodu na novou verzi UML jim stačí jen pozměnit metamodel. Tento postup mimo jiné také sníží počet chyb v nových verzích

## 7. Katalog metaCASE nástrojů

Nástroje popsané v minulých pracích, snad až na MetaEdit+, již přestaly být rozšiřované nebo nejsou nástroji metaCASE tak, jak my je chápeme. Rozhodli jsme, že popíšeme především nástroje, které v minulých pracích nebyly zahrnuté, jen závěrem uvedeme MetaEdit+, který je v současnosti považován za komerčně nejspěšnější.

### 7.1. Eclipse GMF



Eclipse GMF (Graphical modeling framework) je sada zásuvných modulů do nástroje Eclipse, umožňujících vytváření metamodelů a modelovacích editorů. Při práci s GMF je využívána řada dalších modulů. Mezi ty nejdůležitější patří EMF (Eclipse modeling framework) a GEF (Graphical editor framework). Všechny modelovací moduly je naštěstí možné zdarma stáhnout a nainstalovat najednou ze stránek [6] pod záložkou stahování a odkazem Eclipse Modeling Tools.

GMF poskytuje celou paletu možností a funkcí, které rozšiřují možnosti metamodelování. Jsou jimi např. validování modelů, tvorba kontextových menu nebo generování kódu.

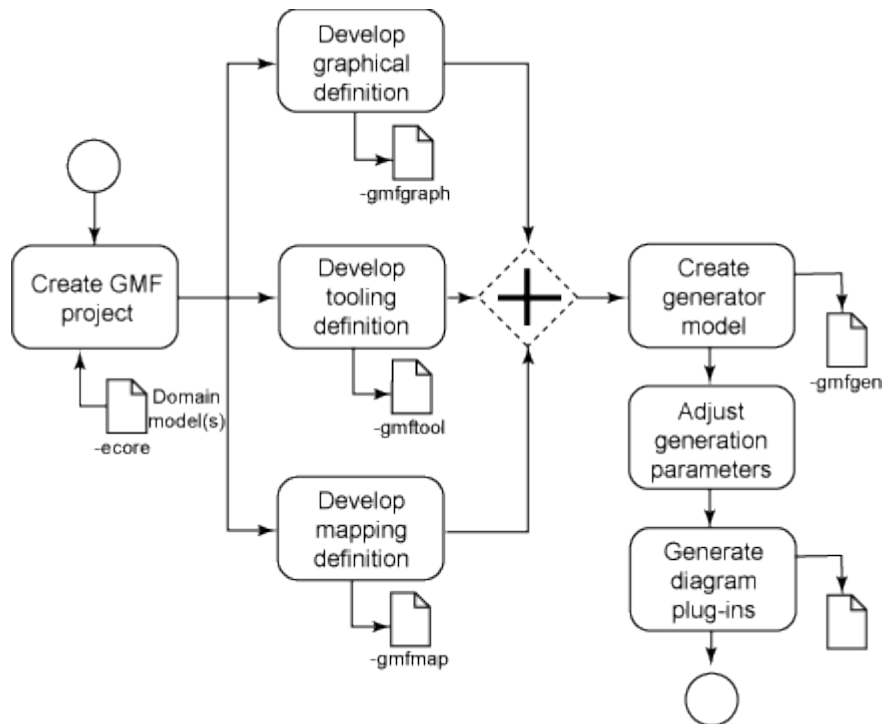
#### 7.1.1. Základní práce s GMF

Základní princip metamodelování v GMF je následující. Nejdříve se v Eclipse vytvoří speciální GMF projekt, ve kterém se založí soubory definující metamodel a chování editoru. Z takového projektu je pak možné vygenerovat další Eclipse projekt, obsahující grafický editor, který odpovídá vytvořenému metamodelu a obsahuje požadované nadefinované vlastnosti. Tento postup je poměrně složitý, ale zaručuje nám vysokou obecnost a širokou paletu možností, jak ovlivnit chování výsledně vygenerovaného editoru.

Před vygenerováním editoru, ve kterém bude možné modelovat nad vytvořeným metamodelem, je potřeba udělat několik kroků. Prvním z nich je samozřejmě tvorba metamodelu. Ten je možné vytvořit několika způsoby, jako například vymodelování pomocí návrháře EMF nebo importovat z UML diagramu či z anotovaných java rozhraní.

Další kroky vedou k přizpůsobení modelovacího editoru potřebám daného metamodelu. Vytváří se grafická reprezentace prvků metamodelu, upravuje se kontextové menu a podobně. Fakticky se

jedná o vytvoření několika XML souborů, které vlastnosti editoru nadefinují. Jejich souvislosti jsou zobrazené na následujícím obrázku.

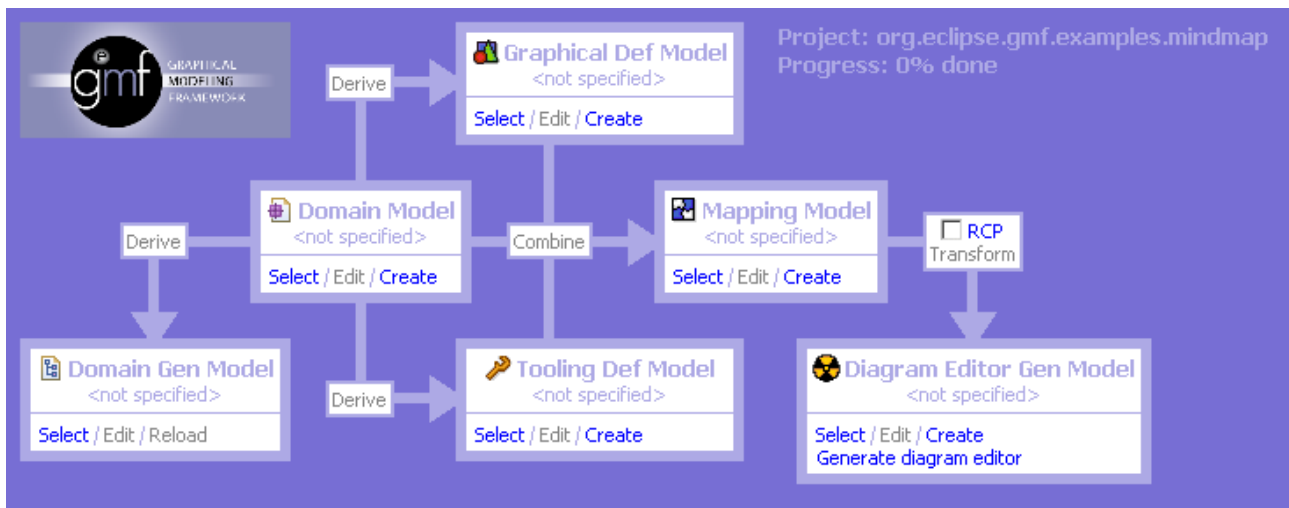


*Ilustrace 6: Soubory definující chování editoru*

Mezi základní nezbytné soubory patří soubory s příponami:

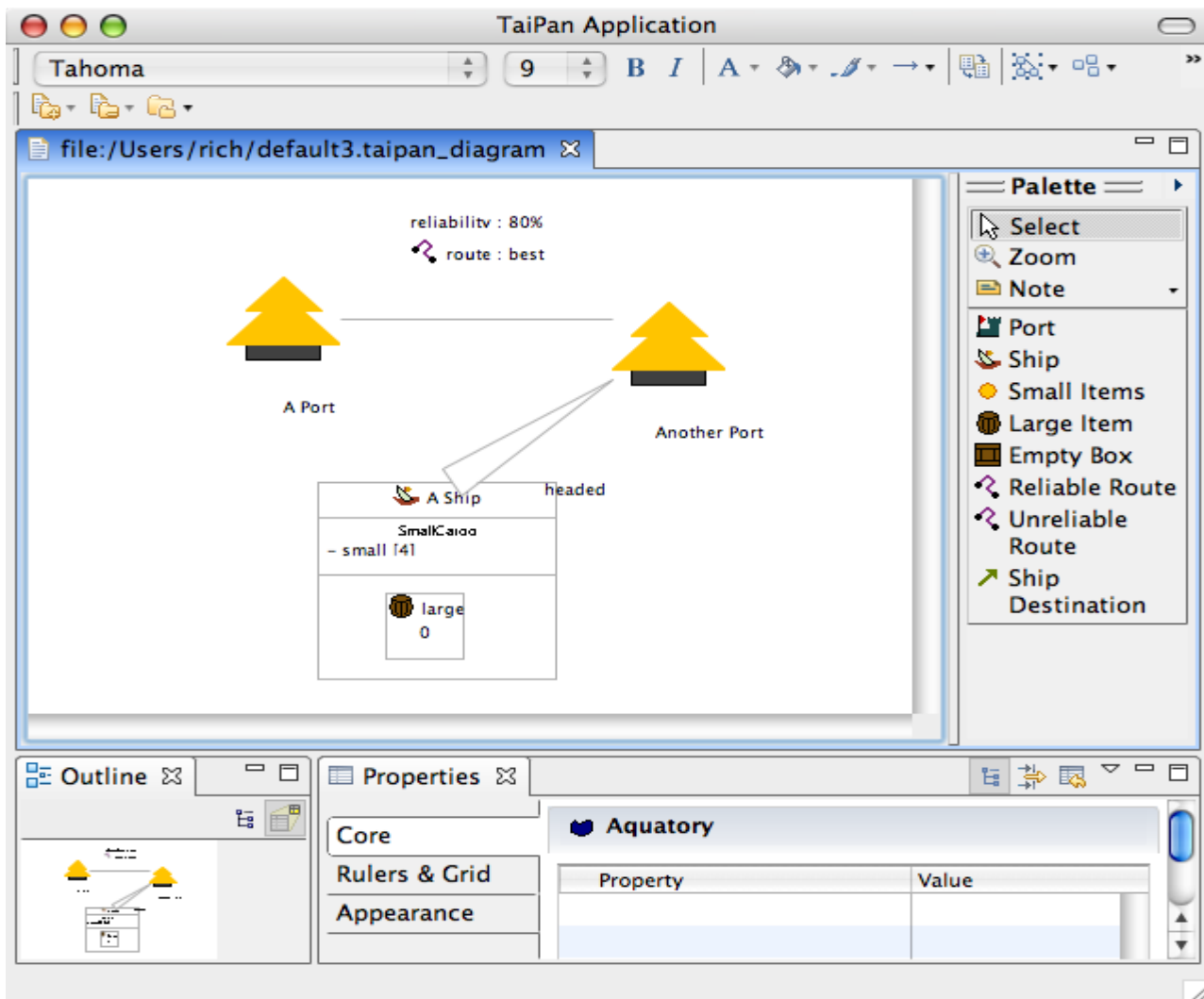
- ecore – metamodel
- gmfgraph – grafická reprezentace prvků metamodelu
- gmftool – nastavení chování generovaného editoru jako kontextové menu a obsah palet
- gmfmap – reprezentuje propojení předchozích souborů, jako například který obrázek náleží jakému prvku modelu

Pro zjednodušení procesu vytváření editoru na modelování obsahuje GMF výborný nástroj. Tento nástroj se jmenuje GMF dashboard (v Eclipse : Window > Show View > Other > GMF Dashboard). S jeho pomocí je možné jednoduše spravovat proces vývoje grafického editoru.



*Ilustrace 7: GMF dashboard*

Pomocí GMF dashboard je možné postupně a přehledně vytvořit a upravovat všechny potřebné soubory. Posledním krokem celého procesu je vygenerování grafického editoru. K tomu je potřeba nejprve vytvořit tzv. GMFGen model. Ten se vytvoří ze souboru s definovaným mapováním (gmfmap) pomocí funkce z kontextového menu „Create generator model“. Tato funkce současně vytvoří celý nový projekt, který bude obsahovat všechny potřebné zdrojové kódy, které bude nově vytvořený grafický editor potřebovat. Pak už jen stačí jednoduše spustit projekt jako „Eclipse application“ a máte grafický editor pracující s vámi vytvořeným metamodellem. Spuštěný editor může vypadat například tak, jako na následujícím obrázku.



Ilustrace 8: Ukázka spuštěného editoru

### 7.1.2. Závěrečné shrnutí Eclipse GMF

Eclipse GMF je bezesporu nejlepší zdarma stažitelný metamodelovací nástroj. Obsahuje jak možnosti vytváření metamodelů, tak následné modelování nad vytvořenými metamodely. Mezi popsányými nástroji vyniká svou rozšiřitelností. Editor, ve kterém se po vytvoření metamodelu modeluje, je možné přizpůsobit všem potřebám. Na druhou stranu je i vytvoření jednoduchého metamodelu pomocí GMF poměrně složité.

## 7.2. Microsoft DSL Tools

Microsoft Tools for Domain-Specific Languages je sada nástrojů pro vytváření, editaci, vizualizaci a používání doménově specifických dat pro automatizaci procesu vývoje enterprise systému. Největší výhodou těchto nástrojů je, že jsou všechny integrovány do dobře známého vývojového prostředí Microsoft Visual Studio (dále jen VS). Uživatel tak většinou nemusí trávit

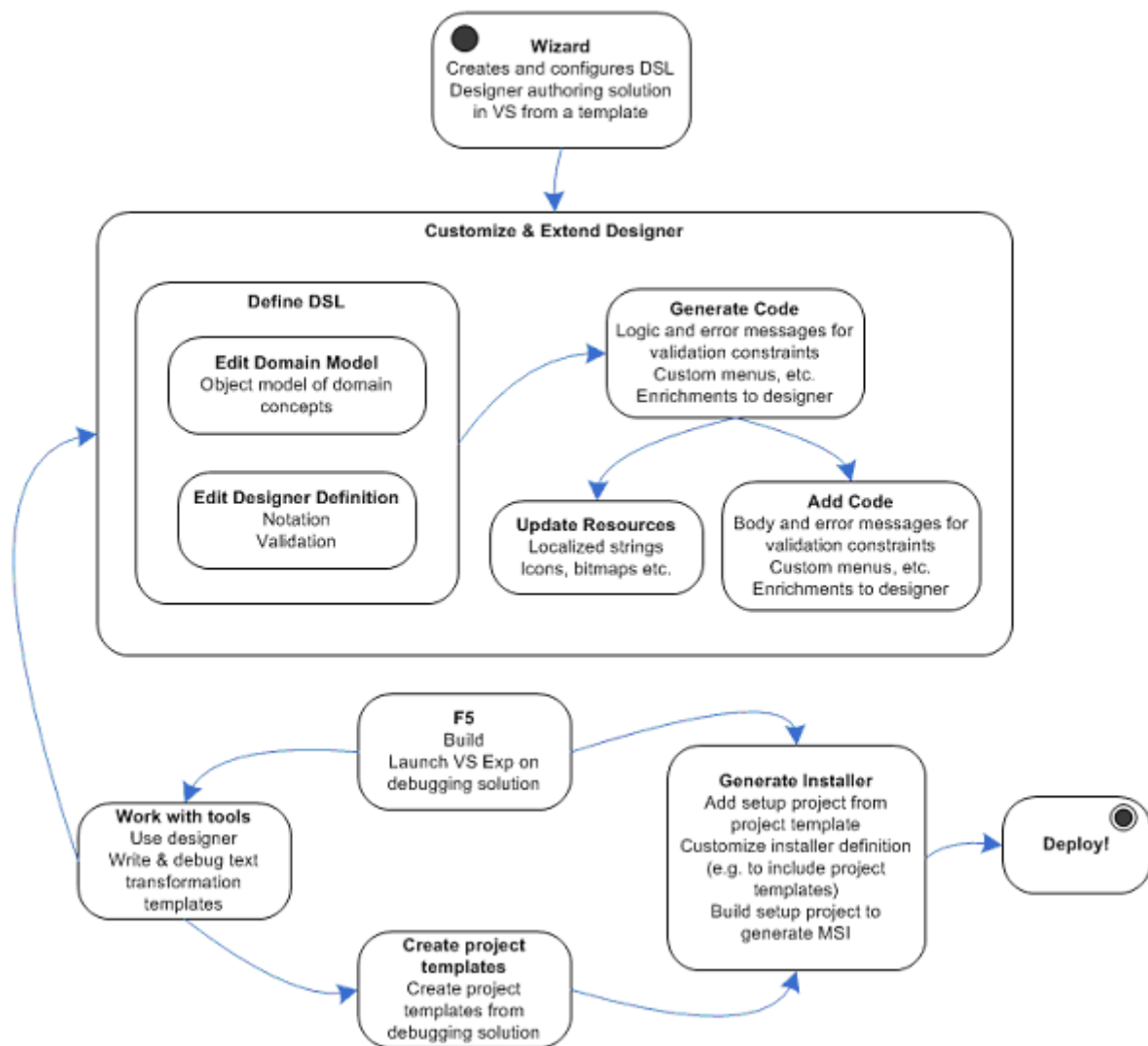
dlouhou dobu studováním dokumentace prostředí, tak jak je tomu například u velice neintuitivního nástroje MetaEdit+. Další výhodou MS DSL Tools je možnost použití jazyků implementujících .NET CLI, tedy například jazyk C# či VB.NET. Pomocí těchto jazyků pak tvůrce metamodelu může přidat kontroly modelů, generování kódu z modelu nebo ovlivnit chování VS při vytváření modelů.

Pro využití Microsoft DSL Tools musíte mít nainstalované následující komponenty:

- Visual Studio 2005.
- Visual Studio 2005 SDK.
- Domain-Specific Language Tools for Visual Studio 2005 Version November 05

Výsledná instalace pak může zabírat přes 2GB místa na disku. Jediný software z uvedené trojice, který je potřeba zakoupit, je Visual Studio 2005. Microsoft sice distribuuje i verzi Visual Studia zdarma v Express edici, ale pro DSL modelování je potřeba zakoupit minimálně verzi Profesional.

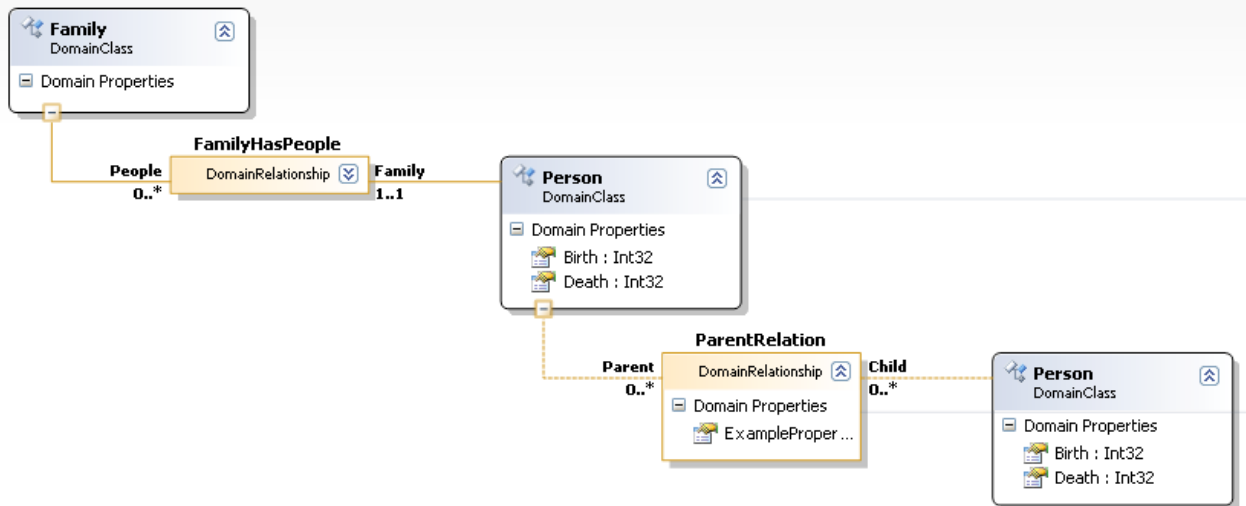
Visual studio po nainstalování uvedených komponent obsahuje jak designér pro vývoj metamodelu, tak designér pro tvorbu modelu. Nejprve se vytvoří metamodel v jedné instanci VS a poté se metamodel přeloží a spustí. Tím se otevře nová instance VS, kde se pak vytváří konkrétní modely. Tento způsob také umožňuje ladění metamodelů za pomoci funkčnosti VS Debugger. Podrobnější popis tohoto procesu je vidět na následujícím obrázku.



Ilustrace 9: Proces vývoje DSL pomocí MS DSL Tools

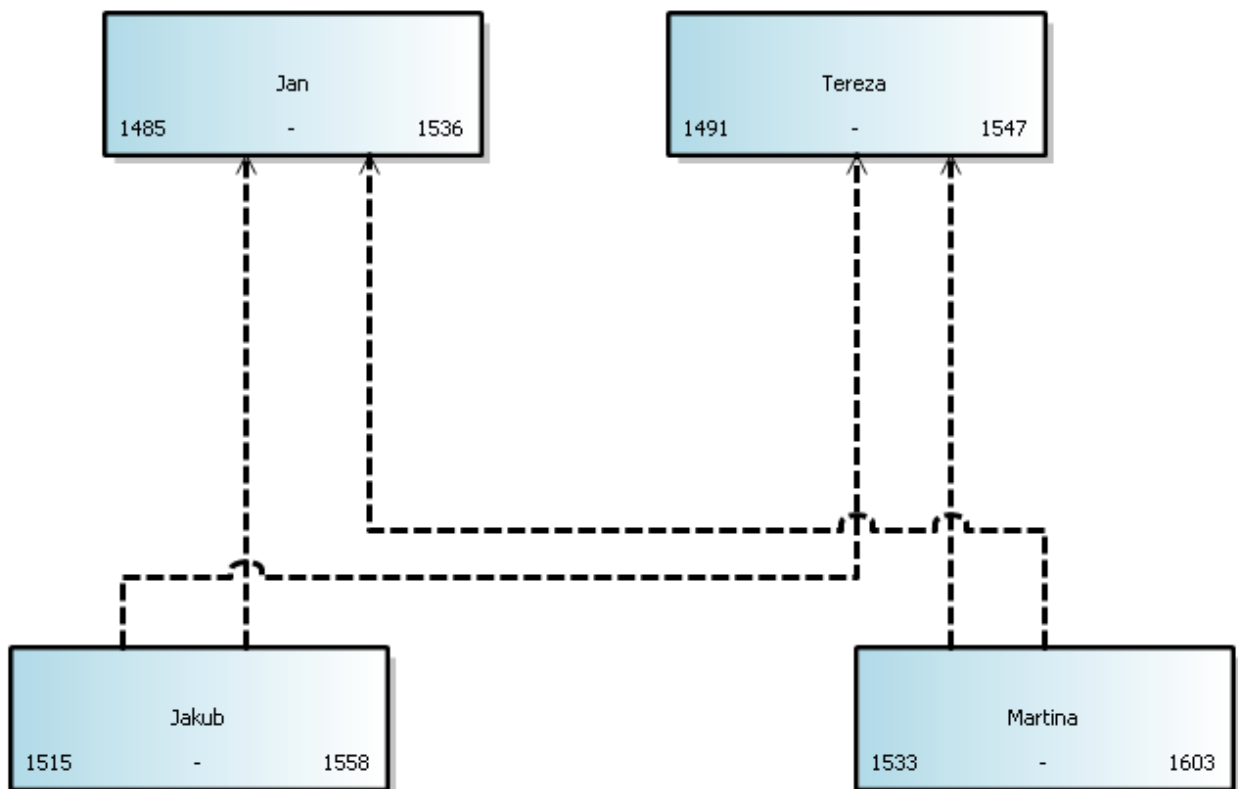
### 7.2.1. Ukázka vytváření metamodelu pomocí DSL Tools

Vytvořit metamodel pomocí DSL Tools je velice snadné. Ve VS si založíme standardním způsobem nový projekt typu Domain Specific Language Designer. Průvodce předpřipraví sám všechny potřebné soubory a strukturu projektu. V připraveném projektu pak už jen stačí otevřít soubor DSLDefinition.dsl a objeví se designér. V designéru se už může začít modelovat, potřebné objekty jsou v panelu nástrojů. Za pár minut je možné vytvořit metamodel, jako například rodinný strom, který je zobrazen na následujícím obrázku.



*Ilustrace 10: Metamodel rodinný strom v MS DSL Tools*

Pokud již máme metamodel hotový, stiskneme ctrl+F5. Tím se projekt přeloží a spustí se nová instance VS, kde se už dají vytvářet konkrétní modely nad metamodelem rodinný strom, který jsme před chvílí vytvořili. Modeluje se opět běžným způsobem pomocí myši a panelu nástrojů. Výsledný model může pak vypadat jako na následujícím obrázku.



*Ilustrace 11: Model rodinného stromu v MS DSL Tools*

## 7.2.2. Generování kódu

DSL Tools umožňují generovat z namodelovaných modelů libovolný kód. Generování kódu patří mezi přední vlastnosti metamodelování, a proto nachází v DSL Tools značnou podporu. Algoritmus generování kódu implementuje návrhář metamodelu v jazyce C# či VB.Net. K tisku kódu do souboru se používá obdobná technika jako u technologie ASP.Net. Velkým nedostatkem je v současné verzi chybějící podpora automatického doplňování kódu (intelysense). Na následujícím obrázku je ukázka tisku modelu do HTML.

```
<#@ template inherits="Microsoft.VisualStudio.TextTemplating.VSHost.ModelingTextTransformation"##>
<#@ output extension=".htm" #>
<#@ examplemodel processor="Language7DirectiveProcessor" requires="fileName='Sample.xyz'"
provides="ExampleModel=ExampleModel" #>
<html>
<body>
<h1>A report on the contents of Sample.xyz</h1>
<#
<p>Files like this can generate code or other text.</p>
<h2>Example Model</h2>
<p>Model: <#=this.ExampleModel.Name#></p>
<#
    foreach ( ExampleClass box in this.ExampleModel.Elements )
    {
<#>
    <p>Box: <#=box.Name#></p>
    <ul>
<# |
        foreach (ExampleClass linkedTo in box.Target)
        {
<#>
            <li>Linked to: <#= linkedTo.Name#>
<#
        }
<#>
    </ul>
    <ul>
<#
        foreach (ExampleClass linkedFrom in box.Source)
        {
<#>
            <li>Linked from: <#= linkedFrom.Name#>
<#
        }
<#>
    </ul>
<#
    }
<#>
</body>
</html>
```

*Ilustrace 12: Generování kódu*

## 7.2.3. Validace

Velice užitečnou součástí DSL Tools je možnost implementování validací pro vytvářené modely. Validace se implementují v jazyce C# či VB.Net. Pro každý objekt je možné vytvořit třídu, která bude obsahovat jeho validaci. U každé metody takovéto třídy určí návrhář pomocí anotací, kdy se

má validace spustit a jaká chybová hláška bude vytištěna do panelu s chybami ve VS. Na následující ukázce je validován objekt Person. Kontroluje, se zda datum smrti následuje až po datu narození a zda potomek dané osoby nemá nastavené datum narození na pozdější, než je datum narození rodiče. Pomocí anotací je u obou validací nastaveno, kdy se mají spustit.

```
[ValidationState(ValidationState.Enabled)]
public partial class Person
{
    [ValidationMethod(ValidationCategories.Open |
        ValidationCategories.Save |
        ValidationCategories.Menu)]
    public void ValidateDates(ValidationContext context)
    {
        if (this.Birth > this.Death)
        {
            context.LogError("Death must be after Birth", "FamilyPersonDateError", this);
        }
    }

    [ValidationMethod(ValidationCategories.Open |
        ValidationCategories.Save |
        ValidationCategories.Menu)]
    public void ValidateParentBirth(ValidationContext context)
    {
        foreach (Person parent in this.Parent)
        {
            if (this.Birth <= parent.Birth)
            {
                context.LogError("Birth must be after Parent's birth", "FamilyParentBirthError", this, parent);
            }
        }
    }
}
```

*Ilustrace 13: Validace v DSL Tools*

## 7.2.4. Závěrečné zhodnocení MS DSL Tools

Microsoft DSL Tools přináší oproti nástrojům popsaným v předchozích pracích řadu výhod, díky kterým je převyšuje. Je to zejména integrovanost do, většině lidem známého, prostředí MS Visual Studia a možnost využití C# a VB.Net s veškerým komfortem, který poskytují. Je vidět, že Microsoft investoval i do relativně okrajového prostředí, jako je DSL modelování, nemalé prostředky. Jedna z mála nevýhod je snad jen nemalá pořizovací cena Visual Studia.

## 7.3. MetaEdit+

MetaEdit+ je metamodelovací a modelovací nástroj. To znamená, že obsahuje jak rozhraní metamodelování, tak následné modelování nad vytvořeným metamodelem. MetaEdit+ je postaven na metametamodelu OPRR [4.2]. Nástroj byl vyvinut na finské univerzitě Jyväskylā. V současné době je distribuován firmou Metacase a jeho zkušební verzi je možné si stáhnout na adrese [www.metacase.com](http://www.metacase.com). Tento nástroj je v předchozích pracích [1] popsán jako nejkvalitnější nástroj na

metamodelování. My s tímto názorem nesouhlasíme a uvádíme seznam nedostatků, které nám přišly jako zásadní:

- Nepřívětivé grafické rozhraní,
- chybějící podpora MOF a XMI,
- generování kódu využívá vlastní jazyk,
- vysoká pořizovací cena.

Bližší popis tohoto nástroje je možné nalézt např. zde [3].

## 8. Závěr

V této práci jsme popsali základy metamodelování, jeho uplatnění a představili tři nástroje usnadňující metamodelování. Také jsme ukázali UML Profily jako alternativu k metamodelování a nastínili jejich přednosti a nedostatky. Závěrem zhodnotíme popsané nástroje a pokusíme se odhadnout budoucnost metamodelování.

MetaCASE nástroje prozatím zdaleka nedosahují takových kvalit jako CASE nástroje. MetaCASE nástrojů je poměrně málo, což spolu s nízkým počtem uživatelů snižuje motivaci dodavatelů metaCASE nástrojů k vylepšování jejich produktů. Nástroje uvedené v kapitole [7] prozatím nenacházejí velké uplatnění v komerční oblasti, pravděpodobně díky masově rozšířenému jazyku UML a kvalitních CASE nástrojích jej podporujících.

Cílová skupina, která by nástroje metaCASE používala, je velmi malá. Naprosté většině potencionálních uživatelů těchto nástrojů postačují již existující metamodely implementované v odpovídajících CASE nástrojích. Současná situace, kdy je používáno jen omezené množství metamodelů, je podle nás správná, neboť snižuje nároky na znalosti uživatelů.

I přes tuto nepřízeň však můžeme ocenit především nástroje Eclipse GMF a Microsoft DSL Tools. Tyto nástroje díky možnosti využití jazyků jako Java nebo C# k implementaci validací modelů nebo generování kódu přináší těmto dvěma nástrojům velkou výhodu. Naopak jsme zklamaní nástrojem MetaEdit+, který je inzerován jako nástroj s nejvíce funkcemi a nejvíce užívaný. Tento nástroj má natolik neintuitivní ovládání, že musí odradit většinu potencionálních uživatelů. MetaEdit+ byl zpočátku vyvíjen v akademickém prostředí, což znamenalo jeho uživatelskou přívětivost. Do budoucna očekáváme další rozvíjení nástrojů Eclipse GMF a Microsoft DSL Tools a naopak úpadek všech ostatních metaCASE nástrojů.

Úplným závěrem bychom doporučili dalším skupinám vyhledávat informace o nových nástrojích spíše pod termínem domain specific modeling tool než nástroje metaCASE. MetaCase je firma v současnosti distribuující nástroj MetaEdit+. Myslíme si, že z toho důvodu vznikla iluze v předchozích pracích, která nástroj MetaEdit+ označují za nejkvalitnější.

## 9. Použité zdroje

[1] APFELTHALER, Jan, BURIANOVÁ, Monika, CERMAN, Michal, CIBULKA, Ondřej, FOŘT, David, LORENC, Ondřej. *Nástroje meta-CASE (charakteristika, vývoj, přehled trhu, trendy)*. Praha: Vysoká škola ekonomická, Katedra informačních technologií. Seminární práce. 2006, 33 s. Dostupný z: [http://www.panrepa.org/CASE/zima2006/meta\\_case\\_zima06.pdf](http://www.panrepa.org/CASE/zima2006/meta_case_zima06.pdf) [dokument ve formátu [PDF]

[2] Burian Petr, Gottwald Tomáš, Příkryl Jan. *Nástroje meta-CASE (charakteristika, vývoj, přehled trhu, trendy)*. Praha: Vysoká škola ekonomická, Katedra informačních technologií. Seminární práce. 2007, 34 s. Dostupný z: [http://www.panrepa.org/CASE/zima2006/meta\\_case\\_jaro2007.pdf](http://www.panrepa.org/CASE/zima2006/meta_case_jaro2007.pdf) [dokument ve formátu PDF]

[3] [www.metacase.com](http://www.metacase.com) 08.12.2008

[4] [http://wiki.eclipse.org/index.php/GMF\\_Tutorial](http://wiki.eclipse.org/index.php/GMF_Tutorial) 29.12.2008

[5] <http://www-128.ibm.com/developerworks/opensource/library/os-ecl-gmf/> 29.12.2008

[6] <http://www.eclipse.org/> 29.12.2008

[7] <http://www.omg.org/mof/> 29.12.2008