Česká zemědělská univerzita v Praze
Provozně ekonomická fakulta

Automatizované modelování

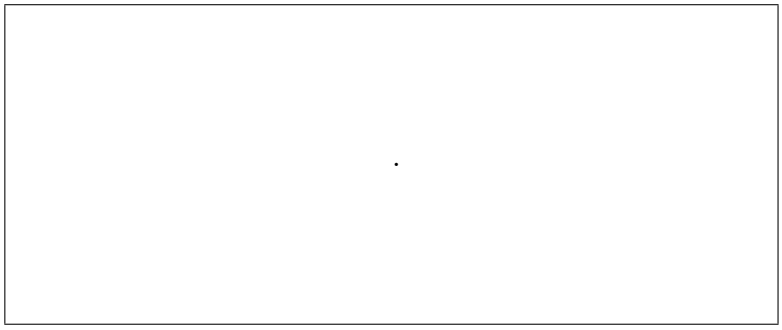
Diplomová práce

Vedoucí práce:
Doc. Ing. Vojtěch Merunka, Ph.D.

Bc. Jan Tomsa

Praha 2010

(zde bude vloženo zadání)



Děkuji vedoucímu diplomové práce Vojtěchu Merunkovi za cenné podněty a metodickou podporu.

Prohlašuji, že jsem tuto práci vypracoval samostatně s použitím literatury, kterou uvádím v seznamu.

V Praze dne 9. dubna 2010 .....

Abstract

Tomsa, J. Automated modeling.

This thesis concerns current attitudes and procedures aimed towards automated modeling. By modeling we mean portraying of the real world objects as well as modeling information systems in the process of their construction.
On the example of an information system for decision support it demonstrates the need of adoption of the Model Driven Architecture (MDA) an it presents possible solution using one particular platform.
The solution is demonstrated on the object oriented platform of Smalltalk programming language.
Key words: Automated modeling, MDA, UML, transformation, Pharo, Seaside, XMI, EMF

Abstrakt

Tomsa, J. Automatizované modelování.

Tato práce pojednává o aktuálních přístupech a postupech směřujících k automatizovanému modelování. Modelováním se zde myslí jak zobrazování skutečností reálného světa, tak i modelování informačních systémů v rámci jejich tvorby.
Na příkladu informačního systému pro podporu rozhodování je demonstrována potřeba vývoje v duchu Architektury řízené modelem (MDA) a možné řešení s použitím konkrétní platformy. Řešení je demonstrováno na objektově orientovaná platforma jazyku Smalltalk.
Klíčová slova: MDA, Automatizované modelování, UML, transformace, Pharo, Seaside, XMI, EMF

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Tato práce pojednává o roli modelování v ekonomické činnosti podniků a ve tvorbě informačních systémů, které tyto činnosti podporují. Zaměřuje se na to, jak se tyto zdánlivě vzdálené oblasti přibližují a jak se díky aktuálním trendům v tvorbě software modelování reality prolíná s modelováním systémů, které tuto realitu zpracovávají. Práce popisuje některé z těchto trendů, jako je posun k tzv. Architektuře řízené modelem (Model Driven Architecture) a kriticky hodnotí jejich připravenost a vhodnost k reálnému použití v současně vytvářených informačních systémech.

1.1 Východiska práce

Při psaní této práce jsem vycházel zejména ze zkušeností získaných na pozici vedoucího vývoje ve společnosti NESS Czech s.r.o. Dále jsem čerpal ze znalostí nabytých na školení k certifikaci *IBM Certified Solution Designer - Object Oriented Analysis and Design, vUML 2.* Uvedené zkušenosti a znalosti mi umožnily kritičtější pohled a utvoření vlastního názoru na problematiku. Zároveň ale jistě ovlivnily mé názory a preference v oblasti metodiky tvorby informačních systémů a použitých systémů.

Tématika modelování mi je blízká vzhledem k profesi vedoucího vývoje, ve které se snažím vést vývojový tým k maximální efektivitě a zároveň souladu implementace s modelem. Téma práce *automatizované modelování* proto výborně zapadá do mé oblasti zájmu a věřím, že tak tato diplomová práce bude mít i praktický přínos pro moji další práci.

V zaměstnání jsem se setkal i s generováním aplikací a měl jsem tak možnost načerpat zkušenosti, které lze stěží, pokud vůbec, získat z odborné literatury. Mnoho věcí, které se v článcích a literatuře jeví velmi slibně a téměř dokonale pak v praxi nefunguje tak dobře, nebo jsou vykoupeny řadou nepříjemností a omezení.

Vedoucí diplomové práce mne upozornil na existenci *Magritte*, nástroje pro meta-modelování vytvořeného v prostředí *Pharo*, čímž ovlivnil výběr platformy, na které jsem koncept rozvinul v kapitole 6.

Myšlenku vytváření modelu software a jeho následné transformace na funkční aplikace jsem demonstroval na příkladu vývoje aplikace pro podporu rozhodování. V rámci tohoto procesu je ukázána identifikace specifického návrhového vzoru,

namodelování transformace doménových tříd modelu do tohoto návrhového vzoru, vytvoření *doménově specifického jazyka* pro podporu této transformace (v podobě UML profilu). Následuje ukázka automatizace transformace implementovaná v objektově orientovaném prostředí jazyka Smalltalk a ukázka výsledku této transformace (generovaný zdrojový kód aplikace).

1.2 Typografické konvence

V textu práce jsou použity následující typografické konvence:

`Strojopis`	Názvy a hodnoty tříd, proměnných, konstant, názvy souborů případně celé úryvky zdrojového kódu.
Zvýrazněné písmo	Názvy programových aplikací, jejich částí, položek menu a obrazovek.

1.3 Vysvětlení zkratek

Zkratky použité v dokumentu jsou vysvětleny v tabulce 1 na straně 22.

1.4 Slovník cizích pojmů

V oblasti automatizovaného modelování a v oblasti tvorby informačních systémů vůbec jsou často používána anglická slova, pro která neexistuje nebo se zatím neustálil jednoslovný český překlad. Tato slova proto v textu používám bez překladu, neboť se domnívám, že používání víceslovných překladů nebo opisů by text učinilo hůře čitelným a pochopitelným. Tato slova jsou proto stručně vysvětlena v následujícím slovníku cizích pojmů

1.4.1 Framework

> ***Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji.***
>
> (zdroj: `http://cs.wikipedia.org/wiki/Framework`)

1.4.2 Refaktorování / Refaktoring

(počeštěné slovo *refactoring*)

> ***Refaktorování je disciplinovaný proces provádění změn v softwarovém systému takovým způsobem, že nemají vliv na vnější chování kódu, ale vylepšují jeho vnitřní strukturu s minimálním rizikem vnášení chyb. Při refaktoringu provádíme malé až primitivní změny, ale celkový efekt je velký a to v podobě čistšího, průhlednější a čitelnější kódu, kód se také lépe udržuje a rozšiřuje.***
>
> (zdroj: `http://cs.wikipedia.org/wiki/Refaktorování`)

Tabulka 1: Slovník zkratek

API	Application Program Interface (rozhraní aplikačního programu)
CASE	Computer Aided Software Engineering (Počítačem podporované softwarové inženýrství); používáno i pro označení nástroje CASE
CPM	Critical path method
DB	Databáze
DSL	Domain-specific language
EMF	Eclipse Modeling Framework
GMF	Graphical Modeling Framework
HTML	HyperText Markup Language
J2EE	Java 2 Enterprise Edition
JSF	JavaServer Faces
JSP	JavaServer Pages
MDA	Model Driven Architecture
MOF	Meta Object Facility
PERT	Program Evaluation and Review Technique
OMG	The Object Management Group
RTF	Rich text format
SQL	Structured Query Language
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

2 Cíl práce

Práce se v první části zaměřuje obecně na modelování jako disciplínu. V další části pak na analýzu dostupných modelovacích nástrojů z pohledu jejich použitelnosti v rámci různých metodik tvorby informačních systémů. Následuje část věnovaná transformačním modelovacím jazykům a opět jejich možnosti využití při tvorbě informačních systémů. Kapitola Vlastní projekt se pak zaměřuje na samotnou ukázku realizace systému za použití vhodné podmnožiny nástrojů a technik zmíněných v předchozích kapitolách.

Přínos práce by měl být zejména v ozřejmění problematiky, v kritickém zhodnocení použitelnosti současně dostupných nástrojů a technik a naznačit možný směr, jak v při vytváření informačních systémů efektivně postupovat.

2.1 Motivační příklad

Problematika je ilustrována na situaci malého zemědělského podniku, který v rámci svého strategické analýzy identifikuje potřebu nového informačního systému pro podporu rozhodování. Podnik zvažuje metodiku a způsob vytvoření nového systému a v neposlední řadě též platformu, na které by zamýšlený systém měl být vytvořen a provozován. Informační systém by měl být vytvořen takovým způsobem, aby jeho vytvoření a počáteční provoz nevyžadovaly příliš velké investice. Zároveň však musí být zajištěna budoucí rozšiřitelnost jak co do funkcionality, tak z pohledu výkonnosti.

3 Metodika

Jak už jsem zmínil v úvodu, při psaní této práce jsem vycházel z uvedené literatury a ze zkušeností v roli vedoucího vývoje ve společnosti NESS Czech s.r.o. Dále jsem vycházel ze znalostí získaných na školení k certifikaci IBM Certified Solution Designer - Object Oriented Analysis and Design, vUML 2.

Tomu odpovídá zvolená metodika vývoje (viz kapitola Vlastní projekt) i použité nástroje, jako je Enterprise Architect použitý pro objektovou analýzu a design.

Při psaní diplovové práce jsem postupoval takto:
Našel jsem si modelový příklad potřeby modelování v běžném, např. zemědělském podniku. Zde mi byl inspirací mj. studovaný předmět *Systémová analýza a modelování* a *SW aplikace operačního výzkumu.*

Poměrně dosti práce mi pak dalo vymyšlení koncepce, jak problematiku zajímavě uchopit a přitom postihnout všechny aspekty, které mne na problému zajímají.

V době, kdy jsem uvažoval o vzorové implementaci v prostředí *Squeak* (open source implementaci Smalltalku), nasměroval mne vedoucí diplomové práce mne na projekt *Pharo*, což je větev *Squeaku* určená pro profesionální vývoj - zejména - webových aplikací.

Z Internetu jsem si proto stáhl vývojářský „obraz“ (image) a *Pharo Virtual Machine*. Obraz určený pro vývojáře má některé vlastnosti, které značně usnadňují vývoj: zvýraznění syntaxe kódu, automatické doplňování, klávesové zkratky apod. V tomto prostředí jsem naprogramoval řešení úlohy standardního lineárního programování pomocí simplexové metody.

V CASE nástroji Enterprise Architect jsem pak namodeloval požadaky (případy užití) a doménový model základní verze informačního systému, který by funkčnost výše uvedeného řešení úlohy standardního lineárního programování pomocí simplexové metody zpřístupnil běžnému uživateli.

Při studiu vývoje webových aplikací za pomoci frameworku *Seaside* jsem narazil na demonstrační příklad *Los Boquitas* s názornými video ukázkami vytvořený společností *GemStone Systems, Inc.* na adrese
`http://seaside.gemstone.com/tutorial.html`,
podle kterého jsem si v „obrazu“ *Seaside One-Click Experience* příklad prošel. V rámci tohoto návodu jsem se blíže seznámil s balíčkovacím a instalačním nástrojem *Monticello*, který jsem pak použil k exportu vytvořené Seaside aplikace a do

lokálního úložiště. Do stejného úložiště jsem vyexportoval i balík tříd týkající se simplexové metody.

Následně jsem si stáhl nejaktuálnější verzi vývojářského obrazu *Pharo* a do něj nainstaloval nejnovější verzi *Seaside 3.0*. Do té jsem pak naimportoval i zmíněné dva balíčky z lokálního úložiště.

Ze vzorového příkladu *Los Boquitas* jsem extrahoval návrhový vzor evidenční webové aplikace v prostředí Seaside a aplikoval jsem jej na jednu entitu (Model) z doménového modelu zamýšleného systému.

Abych mohl automatizovat aplikaci tohoto návrhového vzoru na všechny ostatní entity, musel jsem vyřešit, jakým způsobem jejich model transformovat.

V rámci podrobnějšího zkoumání vlastností CASE nástrojů jsem vyřešil i jednoduché generování Smalltalk kódu v tzv. „chunk“ formátu z Enterprise Architect (viz kapitola Enterprise Architect). Pro výše uvedený účel jsem však potřeboval nejen generování, ale zejména poměrně netriviální transformaci jedné třídy na čtyři jiné třídy. Ačkoli Enterprise Architect podle dokumentace umožňuje i to, připadalo mi, že se kvůli tomu budu muset naučit další specifický skriptovací jazyk a pak se ještě potýkat s jeho případnými omezeními. Zvolil jsem proto možná pracnější, avšak univerzálnější řešení v podobě importu modelu do prostředí *Pharo* a naprogramování transformace ve Smalltalku.

Nejprve jsem do prostředí musel doinstalovat podporu XML, což díky propracovaným konfiguračním nástrojům byla záležitost asi pěti minut. Dále jsem vyexportoval model z CASE do formátu XMI a podrobněji nastudoval jeho strukturu.

Namodeloval jsem metamodel UML (v EA) a v prostředí *Pharo* jsem jej implementoval.

Dále jsem naprogramoval import ze struktur XMI do metamodelu UML a transformaci modelů (entita -> čtyři implementační třídy).

Následně jsem nstudoval problematiku dynamické deklarace tříd v prostředí *Pharo* (platí zřejmě obecně pro Smalltalk) a na základě této znalosti jsem pak dokončil transformaci z UML do tříd přímo v prostředí *Pharo*.

Takto vytvořené třídy jsem pak odladil (pochopitelně spolu s opravami chyb ve všech krocích transformace) a následně napojil do existující aplikace, jejíž výsledný kód uvádím v příloze.

Část I

Literární rešerše

4 Přehled vlastností modelovacích nástrojů

4.1 Úloha modelování v běžném životě

V každodenním životě se neustále setkáváme s modelováním a s rozhodováním na základě jeho výsledků. Pro drtivou většinu modelů nám však postačuje vlastní představivost a proto si tuto činnost nejspíše většina z nás příliš neuvědomuje.

Modelování má veliký význam v řešení problémů reálného světa. Většina úloh počítačového zpracování dat zahrnuje modelování, i když si to uživatelé těchto modelů často neuvědomují.

Například sestavení rodinného rozpočtu v tabulkovém procesoruu není nic jiného než vytvoření jednoduchého modelu. Jestliže si např. vytvoříme tabulku průměrných měsíčních příjmů a výdajů (viz Obr. 1), pak jsme vlastně vytvořili model. Analýzou takovéhoto modelu pak rodina zjistí, zda si letos může dovolit dovolenou u moře, nebo např. na Šumavě. Tabulkový procesor se tak navíc stává nástrojem pro podporu rozhodování.

Použití výpočetní techniky není dokonce nezbytnou podmínkou takovéhoto typu modelování. Sečtení příjmů a výdajů by jistě bylo možné provést např. na papíře nebo dokonce z hlavy. Výhoda počítačového zpracování je zejména ve snadnosti generování mnoha variant (oproti pracnému postupu v papírové podobě modelu) a v možnosti zahrnutí velkého množství vstupních parametrů (oproti velmi omezenému rozsahu lidské paměti).

	A	B	C	D	E	F	G	H
1								
2			Příjmy		Zbývá (úspora)		Výdaje	
3		**Celkem**	**31 000**		=C3-G3		**25 500**	**Celkem**
4		Mzda	30 000				6 000	Splátky úvěrů
5		Státní podpora (průměr)	1 000				4 000	Benzín
6							12 000	Domácnost (průměr)
7							1 500	Ošacení (průměr)
8							2 000	Útrata (průměr)
9								
10								
11								

Obrázek 1: Model v MS Excel

Dalšími méně zjevnými modelovacími nástrojem jsou např. nástroje na řízení projektů – Microsoft Project, Primavera, Taskjuggler apod. Jedná se vlastně o

databázi úkolů a jejich vzájemných vazeb, které modelují skutečné nebo zamýšlené činnosti, k jejichž naplánování byl tento model použit. Jedná se vlastně o databázi úkolů a zdrojů a jejich vzájemných vazeb, nad kterou operuje matematický model CPM a PERT. Tento model tedy je naplněn daty a následně jsou mu kladeny dotazy. Např.:

- Kdy projekt skončí?
- Kdo má kdy co dělat?
- Co je třeba změnit, aby projekt skončil ke zvolenému datu?
- atd.

Na základě takto zjištěných odpovědí jsou pak činěna rozhodnutí.

Příklad:

Předpokládejme, že se chystáme např. přestěhovat větší skříň z jedné místnosti do jiné přes několik různě širokých a vysokých dveří.

Můžeme postupovat tak, že jednoduše začneme se stěhováním a uvidíme, zda projdeme. Jestliže u jedněch z dveří zjistíme, že jimi skříň neprojde a že se musí rozebrat, pak nám může stát, že zrovna v daném místě pro rozebrání není místo a musíme se vracet. Naivní přístup se tedy příliš neosvědčil.

Nebo můžeme změřit rozměry skříně svinovacím metrem a s jeho pomocí pak ověřit prostupnost úzkých míst na cestě. Problém odhalíme s daleko větší pravděpodobností a budeme ho tak moci vyřešit efektivněji – tj. skříň rozebrat např. již před započetím stěhování. V tomto případě byl vlastně vytvořen model skříně reprezentovaný jejími rozměry a na tomto modelu byl simulován průchod úzkými místy. V konkrétním místě byl výsledek simulace neuspokojivý a na základě toho bylo učiněno rozhodnutí o změně postupu – rozebrání skříně. Pochopitelně by mělo následovat vytvoření nového modelu a opakování simulace, ale je možné, že nově vzniklá situace bude triviální a „model“ a „simulaci“ zvládne stěhovák provést na mentální úrovni své představivosti.

Výše uvedený model byl velmi jednoduchý, neboť takový byl i problém, pro který byl vytvořen. Pro složitější situace je třeba vytvářet složitější modely s vyšší mírou formalizace. Vzhledem k omezeným možnostem lidské představivosti a paměti se při modelování velmi složitých problémů neobejdeme bez použití výpočetní techniky.

> ***Modelování je jakousi myšlenkovou imitací, abstrakcí, reprodukcí reálně existujícího systému pomocí speciálně konstruovaných modelů – analogů. Modelování je tedy jednou z forem poznání, zvláštním prostředkem reprodukce reality.***
>
> [Polák,Merunka,Carda, 2003]

Vrátíme-li se k příkladu použití tabulkového procesoru z úvodu kapitoly, pak model domácího rozpočtu obsahoval toto pravidlo: $Uspora = \sum prijem - \sum vydaj$ Jednotlivé položky příjmů a výdajů jsou pak vstupní data tohoto modelu. Jestliže *informace je význam přiřazený datům*, pak tím, co vstupním datům (čísla 30000, 1000, 6000, 4000, 12000, 1500, 2000) přiřadilo význam, byl právě onen model podpořený automatizací v podobě tabulkového procesoru Microsoft Excel.

4.1.1 Vhodnost použití objektových nástrojů pro modelování a transformace

> ***Objektově orientovaná technologie překonává zatím nejlépe ze všech jiných technologií nebezpečí, že tvůrce vyčerpá svoji energii na jednotlivstech, vynucených obtížnou implementací detailů projektu na konkrétním operačním systému a programovacím jazyce.***
>
> [Polák,Merunka,Carda, 2003]

Domnívám se, že blízkost softwarových objektů k jejich předobrazům v reálném světě činí objektově orientované jazyky a prostředí obzvláště vhodné pro vytváření počítačových modelů. Model vytvořený objektově orientovanými technologiemi je srozumitelnější, což má za následek nižší chybovost a zároveň větší pravděpodobnost že skutečnosti s jeho pomocí zjištěné budou správně interpretovány.

V dalším textu se zaměřím na nástroje pro modelování software. K modelování v širším slova smyslu se vrací kapitola 6.

4.2 Architektura řízená modelem - Model Driven Architecture

4.2.1 The Object Management Group

The Object Management Group je mezinárodní obchodní asociace zaregistrovaná jako nezisková organizace ve Spojených státech a s pobočkami po celém světě. Tato

organizace byla vytvořena s cílem pomoci omezit složitost, snížit náklady a uspíšit vznik nových softwarových aplikací. OMG tohoto svého cíle dosahuje zavedením architektonického frameworku Model Driven Architecture (MDA) - Architektura řízená modelem, spolu s podporou vzniku jeho detailních specifikací. Tyto specifikace mají vést softwarový průmysl směrem ke spolupracujícím, znovupoužitelným a přenositelným softwarovým komponentám a datovým modelům založeným na standardních modelech. [OMG, 2003]

4.2.2 Základní cíle a přístupy MDA

Záměrem MDA je umožnit definici aplikačních a datových modelů ve formě čitelné strojem, které mají umožnit dlouhodobou flexibilitu implementace, integrace, údržby, testování a simulace. Pod pojmem model MDA chápe formální specifikaci systému ve standardu UML. Z pohledu jeho strojového zpracování se pak jedná o jeho reprezentaci v jazyku eXtensible Model Interchange (XMI), případně EMF core (Ecore) - viz Kapitola věnovaná EMF.

MDA poskytuje postupy a umožňuje vznik nástrojů pro
- specifikace systémů nezávisle na platformě, která je podporuje,
- specifikace platforem,
- výběr konkrétní platformy pro konkrétní systém a
- transformace specifikace systému do podoby uzpůsobené konkrétní platformě.

4.2.3 Platforma

Platforma je sada subsystémů a technologií, které poskytují souvislou množinu funkcionality prostřednictvím rozhraní a specifikovaných vzorů užití. Jakákoli aplikace podporovaná touto platformou může tuto funkcionalitu využívat bez nutnosti starat se, nebo dokonce vědět, jak je daná poskytovaná funkcionalita implementována. [OMG, 2003]

Příklady platforem:
- Java Enterprise Edition
- Microsoft .NET™
- Cincom VisualWorks
- Pharo (open-source implementace prostředí Smalltalk)
- ... a mnoho dalších.

4.2.4 Hierarchie modelů dle MDA

MDA definuje následující hierarchii modelů:

- Model nezávislý na počítačovém zpracování - Computation Independent Model (CIM)
- Model nezávislý na platformě - Platform Independent Model(PIM)
- Model specifický ke konkrétní platformě - Platform Specific Model (PSM)
- Zdrojový kód aplikace

4.2.5 Model nezávislý na počítačovém zpracování

Požadavky na systém jsou modelovány v podobě nezávislé na tom, jakým způsobem budoucí automatizace. Tento model se někdy nazývá též Doménový model nebo Business model. CIM zobrazuje systém v přostředí, ve kterém bude provozován a popisuje přesně *CO* má systém dělat. Je také často zdrojem společného slovníku pojmů pro použití v dalších modelech.

4.2.6 Model nezávislý na platformě

Model nezávislý na platformě (PIM) popisuje *JAK* bude počítačový systém fungovat, avšak nejde do detailu ohledně toho, jak tento systém bude využívat vlastnosti některé konkrétní platformy.

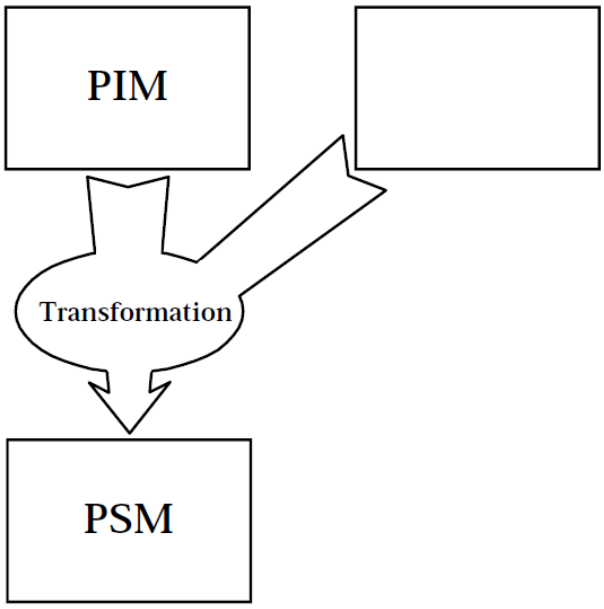
UML profily

Jeden ze způsobů vyjádření PIM je UML model využívající specifický UML profil.

UML profily jsou způsobem, jak rozšířit UML pro modelování entit specifické problémové domény. Jsou založeny na stereotypech a "oštítkovaných hodnotách" (tagged values), které jsou aplikovány na elementy, atributy, metody, vazby, balíky atd.

Tímto způsobem může být UML rozšířeno např. pro modelování grafického uživatelského rozhraní apod.

Např. CASE Enterprise Architect (viz níže) obsahuje UML profil pro modelování XSD (XML Schema Definition) souborů.



Obrázek 2: Transformace PIM do PSM

Doménově specifický jazyk - Domain-specific language

Další možností vyjádření PIM (případně i PSM) je nějaký doménově specifický jazyk.

Jedná se specifický způsob reprezentace problému v určité problémové doméně.

Příkladem může být např. jazyk BPEL (Business Process Execution Language).

Jiným příkladem (avšak již zřejmě nikoli v souvislosti s MDA) je např. programovací jazyk *Karel* používaný pro výuku programování nebo *UnrealScript* pro programování logiky hry *Unreal Tournament*.

V určitém slova smyslu by bylo možné za doménově specifický jazyk považovat i SQL, neboť jeho doménou je manipulace s daty v relačních databázích. Tato "doména" je ale tak široká, že je sporné, zda lze skutečně hovořit o nějaké specifičnosti.

4.2.7 Mapování a značkování

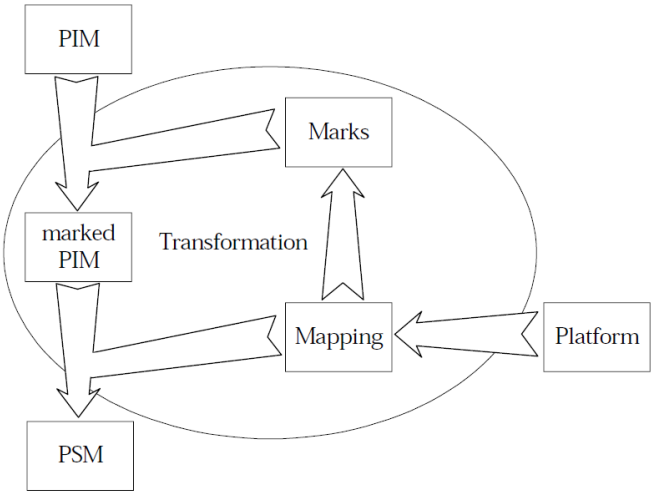
MDA specifikuje obecný rámec, jak by elementy PIM měly být mapovány na elementy PSM. Popisuje, jak by měla být definována pravidla a značky pro generování PSM.

Příklad:

Mapování PIM v UML na implementaci pomocí EJB obsahuje značky, které řídí transformaci PIM na PSM. Mohou také obsahovat šablony nebo vzory pro generování kódu a konfiguraci serveru (Java EE kontejneru). Označení třídy v UML značkou *Session* povede s ohledem na mapování k transformaci této třídy do session bean a dalších podpůrných tříd.

Mluvíme-li zde o pravidlech, značkách, šablonách či vzorech, myslí se tím v ideálním případě strojově zpracovatelné artefakty jako soubory a generátory. V méně ideálním případě se pak jedná o slovně popsaná pravidla, na základě kterých probíhá jednoznačná (i když třeba poměrně složitá) manuální transformace.

Lze předpokládat, že když uživatelé a tvůrci nástrojů získají zkušenosti a techniky modelování sémantiky se zdokonalí, zmenší se množství nutné manuální intervence do transformačního procesu. [Soley, 2000]



Obrázek 3: Transformace PIM do PSM s pomocí mapování a značek

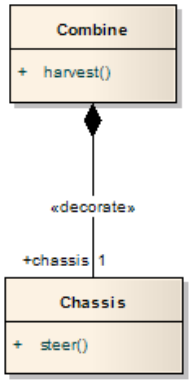
4.2.8 Model specifický ke konkrétní platformě

Platformně specifický model, který je výsledkem transformace, je modelem stejného systému specifikovaného PIM, avšak obsahuje navíc informaci, jak tento systém bude využívat zvolenou platformu.

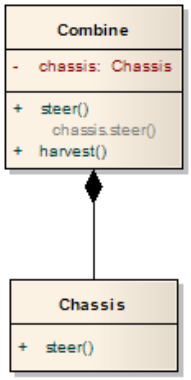
Tento model by měl být již tak specifický, že by z něj mělo být možné přímo generovat zdrojový kód.

Příklad:

Mějme v PSM třídu `Chassis` (podvozek), která je vazbou kompozice propojena s třídou `Combine` (kombajn). Jedná se o návrhový vzor *Decorator*, tj. kombajn "vylepšuje" v sobě zahrnutý podvozek tím, že mu přidává nějakou další funkčnost - sklízení (operace `harvest()`). Vazba je proto označena stereotypem ≪decorate≫. Třída `Combine` pak bude do zdrojového kódu transformována tak, že převezme protokol třídy `Chassis` a tyto převzaté metody naplní kódem pro převolání identických metod členské proměnné `chassis` typu `Chassis`. Viz Obrázky 4 a 5.



Obrázek 4: PSM



Obrázek 5: Implementace

4.2.9 Zdrojový kód aplikace

Samotný zdrojový kód konkrétního programovacího jazyka je z perspektivy MDA v podstatě již nezajímavý, neboť by v něm neměla probíhat žádná hodnototvorná činnost. Jedná se pouze o prostředek k vykonatelnosti platformně specifického modelu, obdobně jako pro programátora ve vyšším programovacím jazyku je obvykle nepodstatný byte-kód nebo strojový kód, do kterého kompilátor přeloží jím vytvořený program.

4.3 MDA a Oracle Designer

Ačkoli iniciativa MDA pod tímto názvem vznikla koncem devadesátých let 20. století, myšlenka tvorby informačních systémů metodou postupných, na sebe navazujících transformací modelů je zde již mnohem déle. Za příklad lze vzít nástroj *Oracle Designer*. Ten, spolu s metodikou *CASE*Method*, sloužil k automatizaci tvorby informačních systémů od fáze modelování obchodních procesů přes tvorbu relačního datového modelu až po generování hotových aplikací v Oracle Forms a dalších platformách. Tento nástroj, původně prodávaný pod názvem *CASE*Dictionary* a *CASE*Designer* existoval již v roce 1988. V dnešní době již nový vývoj tohoto nástroje neprobíhá. Jeho dodavatel, společnost Oracle, se v současnosti zaměřuje na nástroje pro podporu vývoje na platformě *Java EE* postavené na vývojovém prostředí *JDeveloper*.

4.4 Vlastní zkušenost

Dle mé vlastní zkušenosti je třeba přísliby populárních, ale i odborných publikací a článků o MDA brát s určitou rezervou.

V případě generování aplikací z modelu, resp. transformace modelů, je třeba vždy zvažovat jak přínosy tak nevýhody.

V bakalářské práci [Tomsa, 2007] popisuji situaci na projektu údržby a rozvoje Informačního systému katastru nemovitostí (ISKN). Hlavní část tohoto systému je tvořena databází Oracle a klientskou aplikací v *Oracle Forms* a *Oracle Reports*. Datový model, logika v databázi i klientská část jsou z velké části generovány z CASE *Oracle Designer*.

Ačkoli tento přístup v minulosti měl své výhody, domnívám se, že ve stávající fázi projektu (po více než deseti letech od začátku) již nevýhody převyšují přínosy.

Síla generování je, kromě zvýšení úrovně abstrakce, v jeho hromadnosti. Změnou šablony, resp. obecně úpravou parametrů transformace, lze hromadně změnit velké množství nebo všech artefaktů, např. obrazovek aplikace.

Nevýhody (oproti „ručnímu“ programování) jsou podle mého názoru následující:

- omezení vyplývající z použití generátoru - některé problémy je třeba obcházet, to, co by v ručně upraveném kódu bylo provedeno za hodinu, může v CASE trvat i řádově déle.
- programování v prostředí CASE

  - tj. v podstatě se nejedná o modelování, nýbrž o snahu přimět generátor, aby vytvořil něco, co by vývojář sám zvládl několikanásobně rychleji.

  - CASE není pro programování nejvhodnější - oproti standardnímu vývojovému prostředí chybí mnoho funkcí, které vývojáři usnadňují práci (obecně např. zvýraznění syntaxe, automatické doplňování, refaktoring)
- svázanost s prostředím, které se nerozvíjí
- pro nově příchozí vývojáře nutnost naučit se další nástroj - to snižuje flexibilitu co do velikosti týmu vývojářů
- v případě problémů (např. chyb) se tím zvyšuje počet jejich potenciálních zdrojů - vývojář musí řešit více otázek: „Udělal jsem chybu v kódu?“, „Došlo k chybě při generování?“, „Přizpůsobil jsem svůj kód dostatečně generátoru?“, apod.
- několikanásobně komplikovanější konfigurační řízení

Zatímco hromadné akce se typicky dějí zřídka (v pozdější fázi vývoje/údržbě s rozestupem v řádu let), běžné úpravy a opravy můžou probíhat i denně.

Jestliže jsme nuceni odmítat požadavky uživatelů protože „platforma to sice umožňuje, ale my to neumíme vygenerovat“, nebo jsme nuceni vymýšlet krkolomné konstrukce, jak v generovaném prostředí dosáhnout funkcionality, kterou bychom „ručně“ naprogramovali v řádu minut, pak podle mého názoru nazrál čas k přehodnocení přístupu a zvážení, zda by v danou chvíli již nebylo ekonomičtější postupovat klasickým stylem, tj. ručními lokálními úpravami.

4.5 Vlastnosti modelovacích nástrojů

Z nepřeberného množství existujících CASE nástrojů jsem vybral ty, které jsem měl možnost ve větší míře vyzkoušet, nebo ke kterým jsem alespoň měl dostatek

podkladových materiálů.

Snažil jsem se u nich vyhodnotit zejména následující vlastnosti:

- Podpora pro sběr požadavků
- Podpora vizuálního modelování (v UML2)
- Podpora životního cyklu vývoje - iterací
- Verzování
- Podpora MDA
    - Podpora exportu do XMI
    - Podpora transformací modelů (XSLT, MDF ...)
    - Podpora skriptování na úrovni modelu
    - Podpora generování kódu
    - Možnost vytváření vlastních UML profilů (včetně grafické reprezentace)
    - Podpora Doménově specifických jazyků

Vzhled a některé vlastnosti nástrojů jsou demonstrovány na příkladu objektového modelu Mealyho konečného stavového automatu (Finite State Machine - FSM).

Vyhodnocované nástroje:

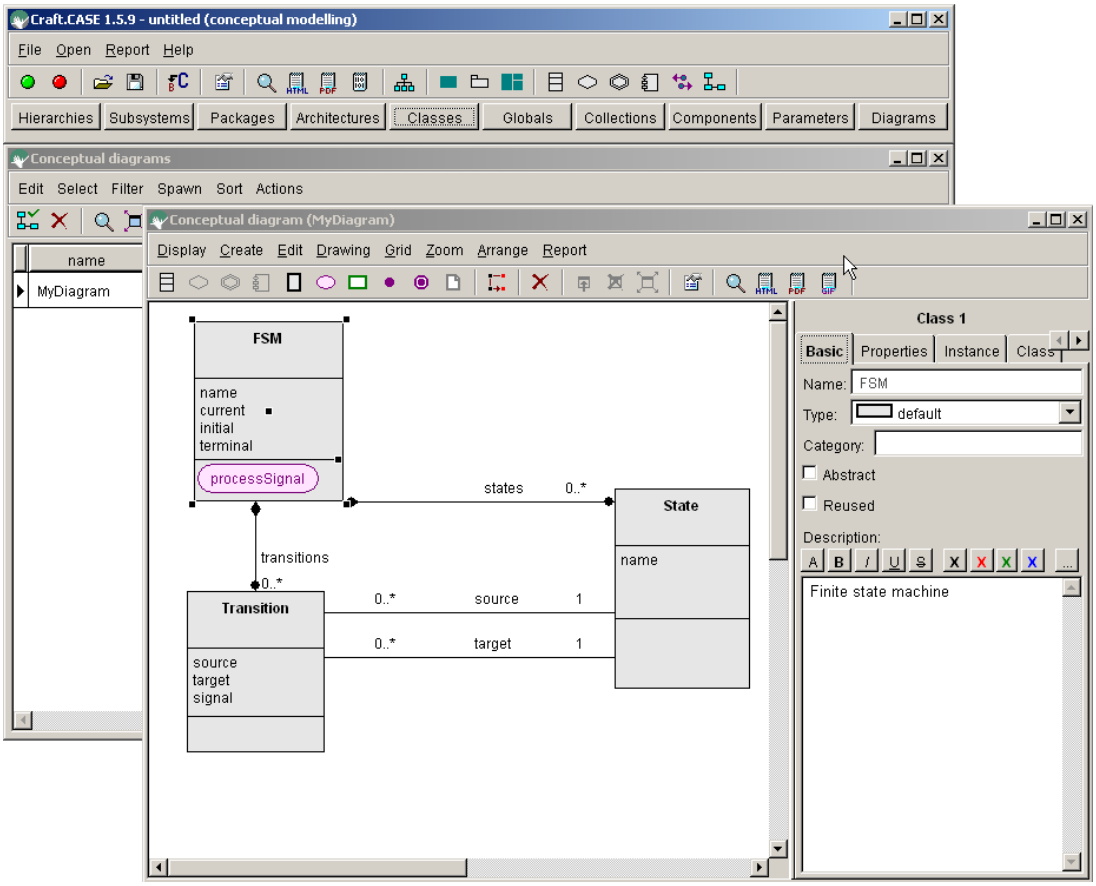
- Craft.CASE
- Eclipse Modeling Framework
- EclipseUML2
- Enterprise Architect

4.6 Craft.CASE

Výrobce: CRAFT.CASE LIMITED

Tento CASE se zcela vymyká obvyklým měřítkům, a to zejména protože je založen na podpoře jiné metodiky vývoje software, než (Rational) Unified Process, a to na metodice *Business Object Relation Modeling (BORM)*. Druhým důvodem jeho výjimečnosti je to, že neslouží (specificky) k návrhu a vývoji aplikací v jazyku Java.

Sběr požadavků a jejich další transformace je ústředním bodem *BORM* a tedy i *Craft.CASE*. Jedna ze zabudovaných kontrol modelu, které v sobě má nástroj



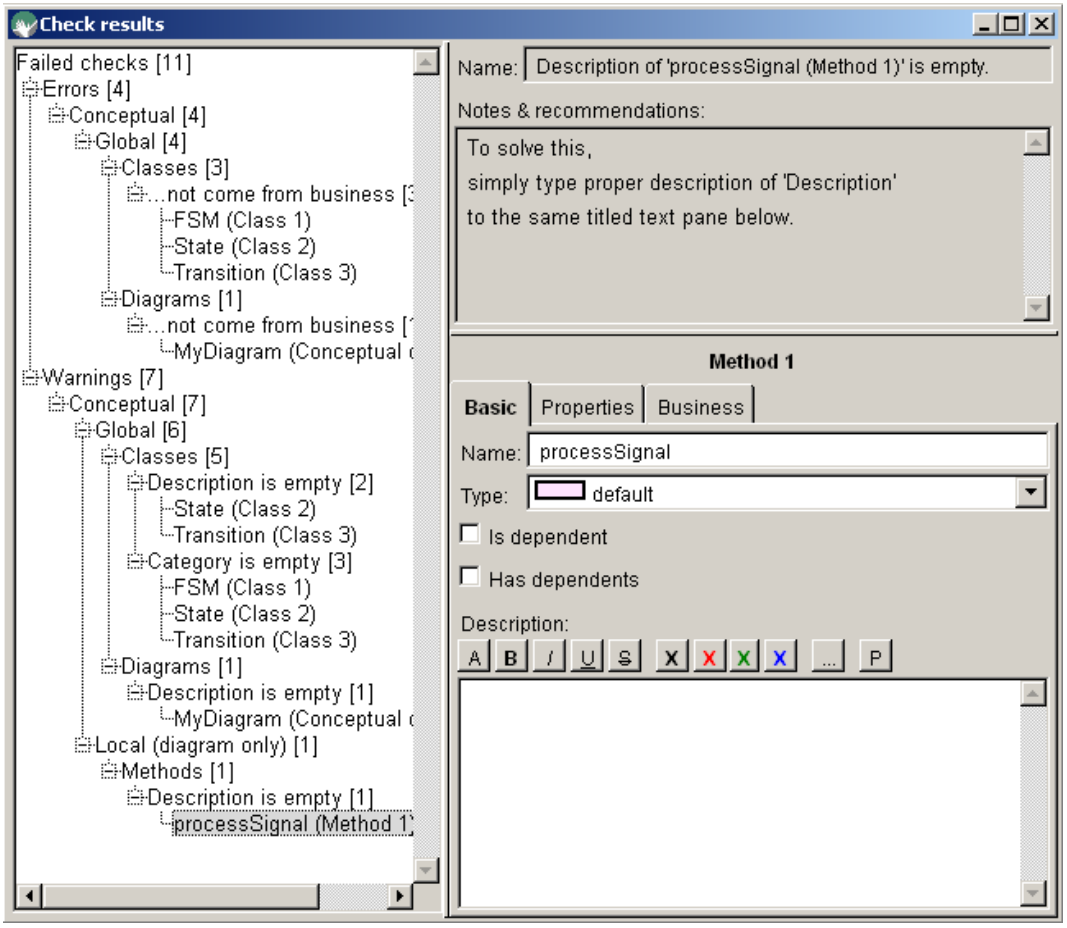
Obrázek 6: Craft.CASE - Diagram tříd

zabudovány, je právě návaznost modelovaných artefaktů na business požadavky - viz Obr. 7.

Co se týče vizuálnosti modelování, jsou mé dojmy rozporuplné. Na jednu stranu nástroj velmi dobře vizualizuje procesní stránku systému, na druhou stranu v snadnosti a intuitivnosti ovládání a v grafickém provedení tento nástroj proti některým dalším poněkud pokulhává. Grafické provedení se může jevit jako relativně nepodstatné, jestliže nástroj poskytne dostatečnou funkcionalitu. Mám zkušenost, že vzhledem k tomu, že výstupy z CASE jsou často užívány při komunikaci se zákazníkem, tak určitá úroveň výstupů může mít na jejich přijetí nezanedbatelný vliv.

Craft.CASE podporuje životní cyklus vývoje v pojetí *BORM*, nicméně ve verzi 1.5.9, kterou jsem měl možnost zkoumat jsem nenarazil na podporu verzování. Aktuální verze v době psaní této práce je 2.1.

Craft.CASE podporuje transformace konceptů prostřednictvím simulátorů business procesů, modelování na úrovni instancí a sady transformačních pravidel



Obrázek 7: Craft.CASE - Výsledky kontrol modelu. V pravé horní části je popis chyby (resp. varování) včetně návodu na její odstranění. Pravá dolní části zobrazuje přímo editační prostředí pro daný prvek. Jeho vlastnosti tedy lze na základě návodu rovnou upravit a tím odstranit chybu.

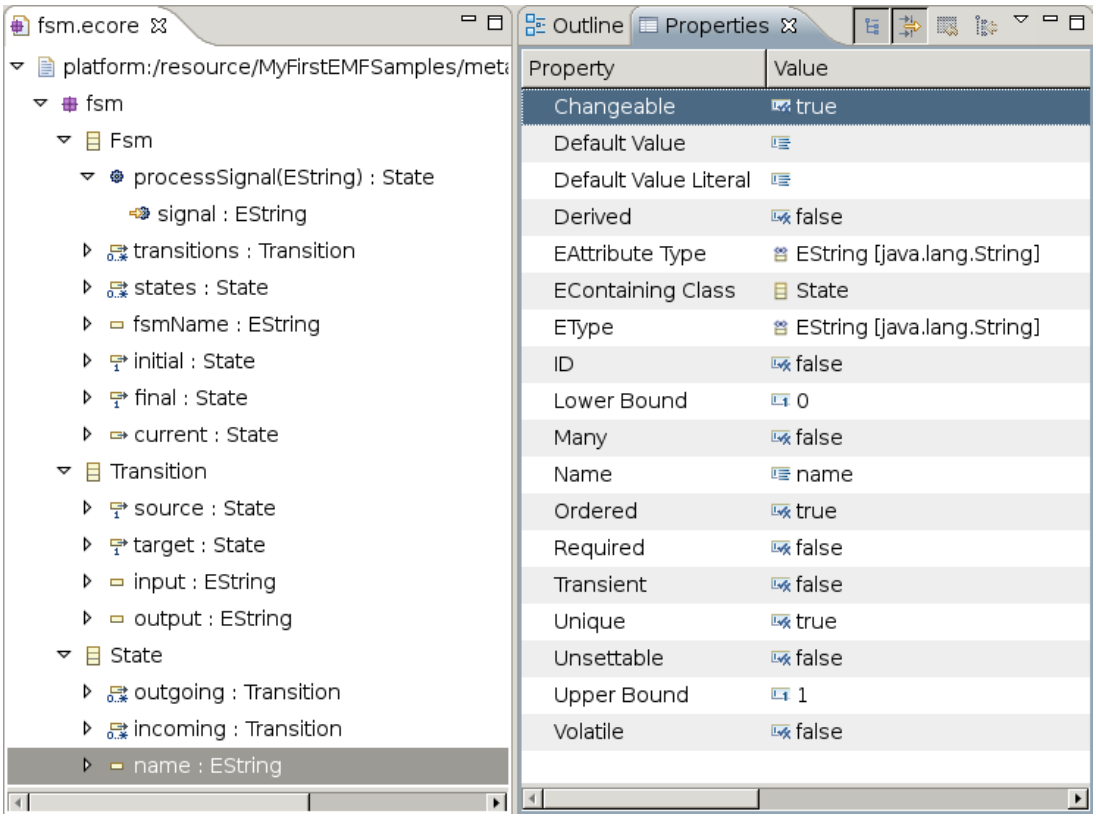
popisujících, jak z předchozích konceptů odvodit následující. [Merunka,Nouza,Brožek,2008]

Export do formátu XMI a řadu dalších rysů podporujících koncept MDA zde zajišťuje zejména skriptovací jazyk *C.C language* (viz kapitola 5.3).

4.7 Eclipse Modeling Framework

Eclipse Modeling Framework (dále jen EMF) je Java framework a příslušenství pro generování kódu pro vytváření nástrojů a jiných aplikací založených na strukturovaném modelu. Snaží se poskytnout výhody formálního modelování s velmi nízkými vstupními náklady. Jeho cílem je pomoci při transformaci mod-

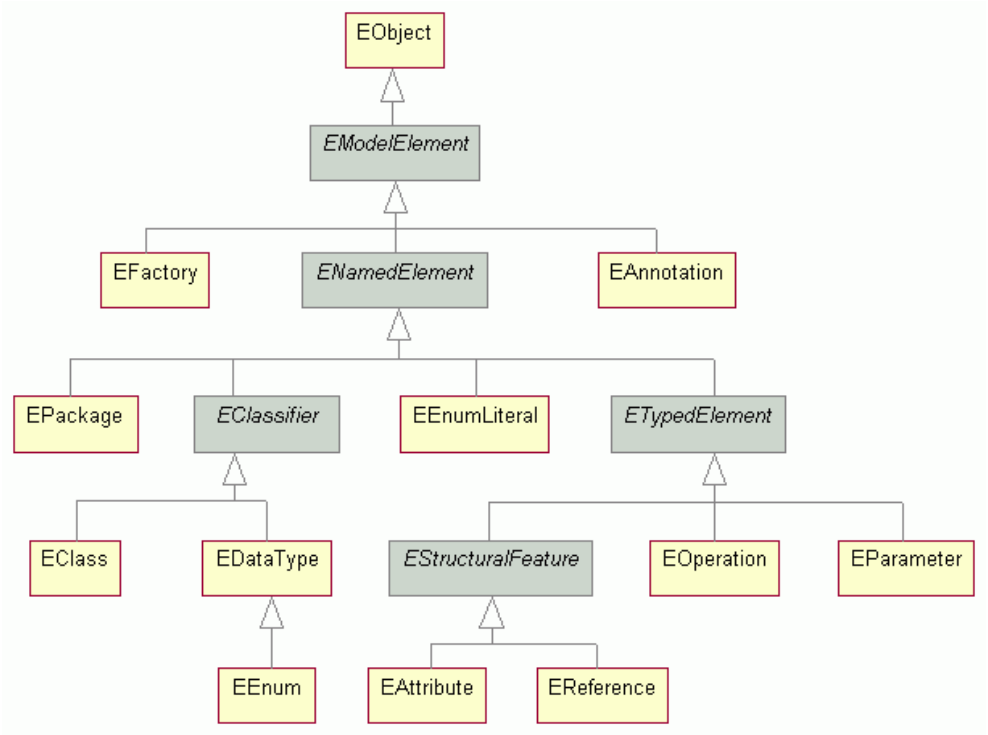
elů do efektivního, správného a snadno přizpůsobitelného kódu v jazyce Java. [Budinsky,Steinberg,Merks,Ellersick,Grose, 2003]



Obrázek 8: EMF - Finite State Machine

Když mluvíme o modelování, obecně máme na mysli Diagramy tříd, Diagramy spolupráce, Stavové diagramy apod. Pro tyto diagramy je notace definována standardem UML. Použitím kombinace UML diagramů může být specifikován kompletní model aplikace. Tento model může být použit čistě pro dokumentační účely nebo, v případě použití vhodných nástrojů, může být použit jako vstup, ze kterého se vygeneruje část nebo, v jednodušších případech, celá aplikace. Modely mohou být vytvářeny s použitím anotatovaného Java kódu, XML dokumentů, nebo modelovacími nástroji jako Rational Rose (nebo např. Enterprise Architect). Tyto modely jsou následně importovány do EMF.[Eclipse,2009]

EMF sestává ze dvou základních frameworků: *core framework* a *EMF.Edit*. *Core framework* poskytuje základní podporu pro generování implementačních tříd pro model v jazyku Java. *EMF.Edit* staví na *core frameworku* a rozšiřuje jej přidáním



Obrázek 9: EMF - metamodel

podpory pro vytváření editorů strukturovaných informací na základě formálně popsaného modelu.

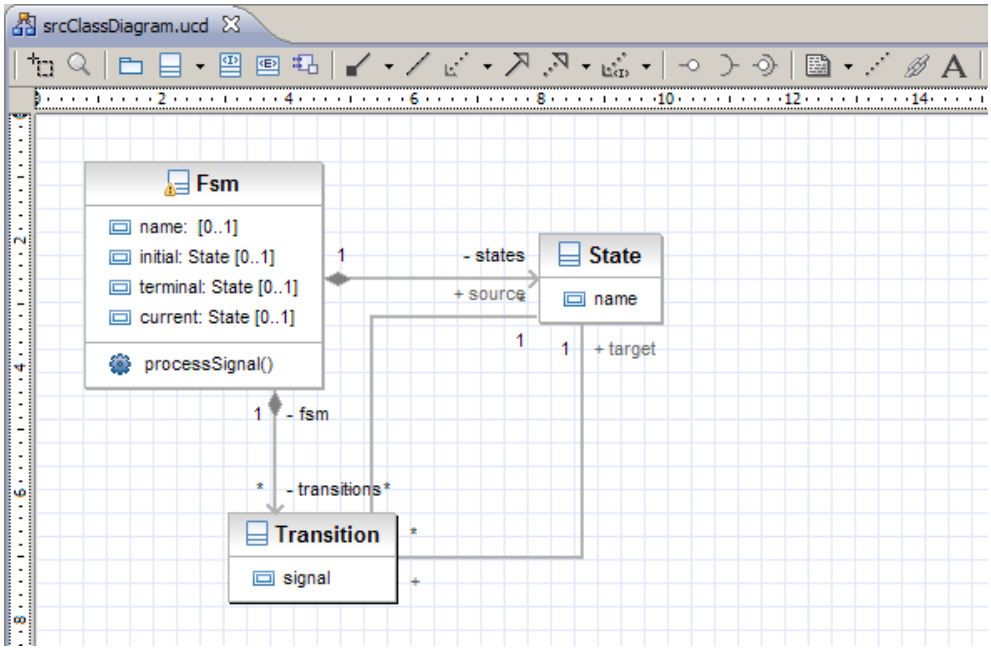
S použitím *Graphical Modeling Framework* (GMF) lze pak vytvořit grafický editor, který obsažené informace vizualizuje libovolnou formou – např. obdobně jako UML diagram tříd vizualizuje strukturu a vztahy v modelu tříd nějakého objektově orientovaného jazyka.

Tato technologie je v současné době ve velmi ranném stádiu a nelze proto předpokládat její široké použití v komerční sféře. Přesto vzhledem k velikosti komunity lze v budoucnu očekávat použitelnost.

EMF nelze považovat za CASE v pravém slova smyslu. Jedná se spíše o jakýsi základ, na kterém může být takový nástroj postaven.

4.8 Omondo EclipseUML2

Jedním z příkladů komerčního využití EMF uvedeného výše je zásuvný modul *EclipseUML2*.



Obrázek 10: EclipseUML2 - Diagram tříd

Projekt *EclipseUML2* je implementací metamodelu UML 2.1 na platformě *Eclipse* založenou na EMF. Cílem tohoto projektu je poskytnutí

- použitelné implementace metamodelu pro podporu vývojových a modelovacích nástrojů,
- společné XMI schéma pro usnadnění výměny sémantických modelů a
- testovacích případů jako prostředku k validování specifikace (EMF).

Podpora MDA je zde klíčová a souvisí mj. s tím, že nástroj vznikl jako vzorová implementace EMF. Poněkud slebší se mi jevila podpora životního cyklu a verzování, ale to může být způsobeno omezeným časem, který jsem tomuto nástroji mohl věnovat.

4.9 Enterprise Architect

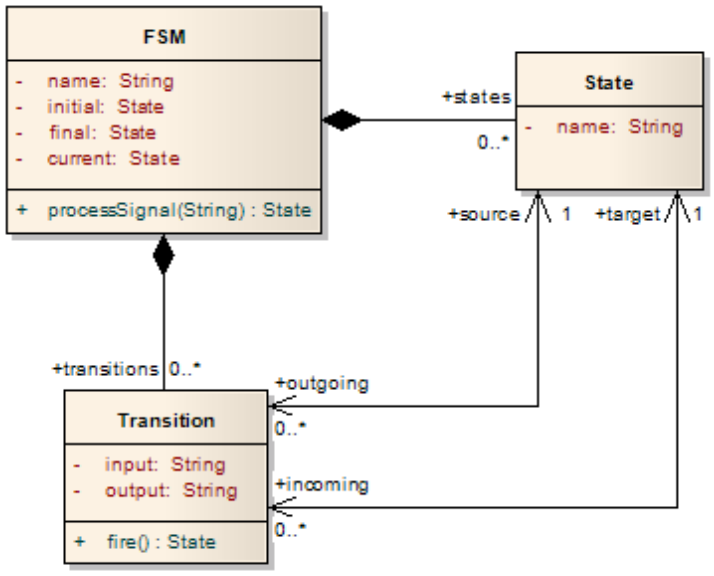
Výrobce: Sparx Systems

Platforma: Microsoft Windows

Enterprise Architect 7.5 je profesionální prostředí pro vizuální modelování se silnou podporou UML 2.1 a příbuznými standardy. Je uzpůsoben zejména pro tvorbu software v rámci *Unifikovaného procesu (Unified process)*. Podporuje životní cyklus projektu od sběru požadavků až po generování kódu aplikace.

Tento CASE má jako výchozí úložiště databázi MS Access, avšak v tzv. *Corporate Edition* podporuje i použití externí relační databáze jako Oracle nebo MS SQL Server. Pro souběžný přístup k repository pěti a více uživatelů se jedná o doporučovanou variantu z výkonnostních důvodů.

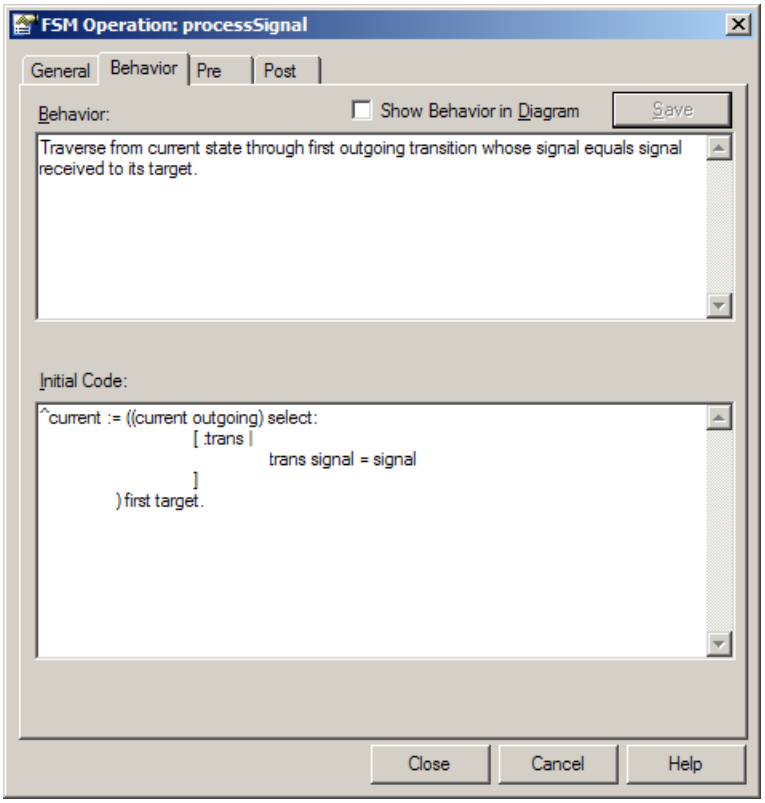
Co se týče podpory verzování, *Enterprise Architect* toto přímo nepodporuje. V každém okamžiku tedy lze pracovat a prohlížet pouze aktuální verzi modelu. Existuje však možnost vytváření tzv. *základní konfigurace (baseline)*, proti níž lze následně porovnávat aktuální stav, resp. jinou baseline. Jelikož se s tímto nástrojem setkávám denně v zaměstnání, musím konstatovat z vlastní zkušenosti, že pro řízení vývojových prací tato informace není dostatečná. Je-li model používán jako zadání pro vývojáře, pak vydefinování práce v druhé a další iteraci pouze na základě výše popsaného porovnání modelů je nepoužitelné a tvůrci modelu tak musí doplňovat dodatečné informace týkající se změn modelu.



Obrázek 11: Enterprise Architect - Diagram tříd FSM

4.9.1 Podpora MDA

Nástroj *Enterprise Architect* se hrdě hlásí k podpoře Architektury řízené modelem. Ačkoli některé funkcionality by jistě bylo možné vyřešit lépe a intuitivněji, celkově se s tímto tvrzením dá souhlasit, jak ukáži v dalším textu na konkrétních vlastnostech.



Obrázek 12: Enterprise Architect - Editace chování operace. Je zde patrné, že lze přímo do modelu zapsat zdrojový kód v libovolném programovacím jazyce, avšak bez jakékoli syntaktické kontroly, o zvýraznění syntaxe a automatickém doplňování nemluvě.

Podpora vlastních UML profilů

Enterprise Architect umožňuje vytváření vlastních UML profilů včetně možnosti úpravy grafické reprezentace elementů na základě jejich stereotypů a tagovaných hodnot.

UML profil lze vytvořit přímo v prostředí *Enterprise Architectu* jako diagram obsahující metatřídy a stereotypy. Viz např. Obrázek 21 nebo **??** (na straně 68).

Takto vytvořený diagram se vyexportuje do formátu XMI (příklad viz příloha D) a následně naimportuje jako UML profil.

Při vytvoření metadřídy CASE uživateli dá na výběr elementů, které mají být metadřídou reprezentovány. Tento výběr je bohužel omezen - chybí např: *Control* a *Table*, přičemž *Control* (reprezentující třídu se stereotypem ≪control≫) hraje v metodice *Unified Process* velmi významnou roli.

Toto chování naznačuje, že se *Enterprise Architect* ke elementům chová různě nikoli jen na základě jejich stereotypů, ale i na základě „způsobu vzniku“, což je nestandardní a někdy to může komplikovat práci.

Pro úpravu grafické reprezentace elementů v rámci profilu nabízí *Enterprise Architect* možnost použití formátu Windows metafile, případně skriptovací jazyk (tzv. *Shape script*). Ten umožňuje uzpůsobit si profil až do vlastního grafického doménově specifického jazyka. Bohužel tento skriptovací jazyk má opět některá nepříjemná omezení, která tuto (jinak vítanou) vlastnost poněkud degradují.

Generování kódu

V souladu s koncepcí MDA *Enterprise Architect* nabízí generování kódu z modelu. Slouží mu pro to tzv. *Code Template Framework* tvořený metamodelem a jednoduchým skriptovacím jazykem.

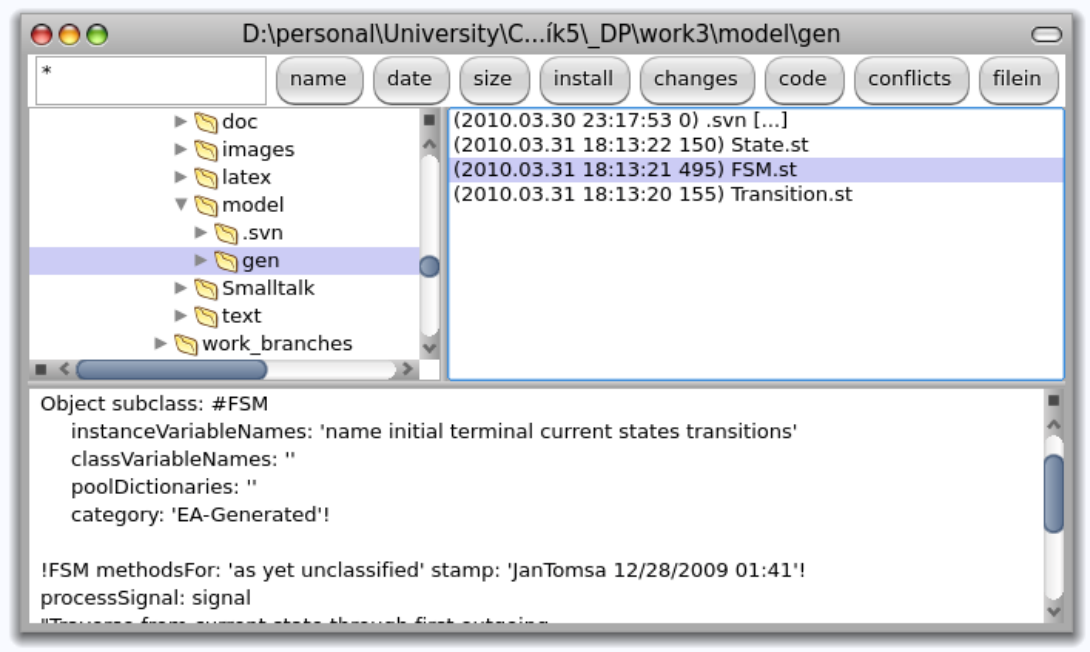
Tento CASE obsahuje výchozí sadu šablon pro jazyky C, C#, C++, Java, Delphi, PHP, Python a ActionScript. Umožňuje zároveň vytváření vlastní podpory pro generování téměř libovolného programovacího jazyka.

V příloze B je ukázán příklad nově vytvořené šablony pro generování tříd v jazyku Smalltalk. Na obrázku 14 je pak vidět třída Smalltalku naimportovaná v prostředí *Pharo*. Tomuto zobrazení ještě předchází import vygenerovaného souboru `FSM.st` do prostředí *Pharo* - viz obrázek 13. Opět demonstrováno na příkladu objektového modelu Mealyho konečného automatu. Kompletní vygenerovaný zdrojový kód je pak možno nalézt v příloze C.

Obrázku 12 ukazuje, že v Enterprise Architectu sice lze ve vlastnostech operace (záložka *Behavior* - chování) zapsat kód v libovolném programovacím jazyku. Zároveň ale ukazuje slabinu takovéhoto přístupu, neboť zde chybí jakékoli syntaktická kontrola, o refaktoringu, zvýraznění syntaxe a automatickém doplňování ani nemluvě.

Transformační šablony

Enterprise Architect (opět v duchu MDA) umožňuje rovněž transformaci mezi modely. Transformační šablony jsou silně založeny na šalonách pro generování kódu a používají tedy stejný skriptovací jazyk s poměrně malými možnostmi rozšíření.



Obrázek 13: Pharo - import vygenerovaného souboru (tlačítkem filein).

Zabudované jsou transformační šablony pro DDL, C#, Java, EJB a XSD. Vytvoření šablony vlastní je dle dokumentace možné, avšak troufám si odhadnout, že vzhledem k ne zcela intuitivnímu proprietárnímu jazyku to nebude zrovna snadné.

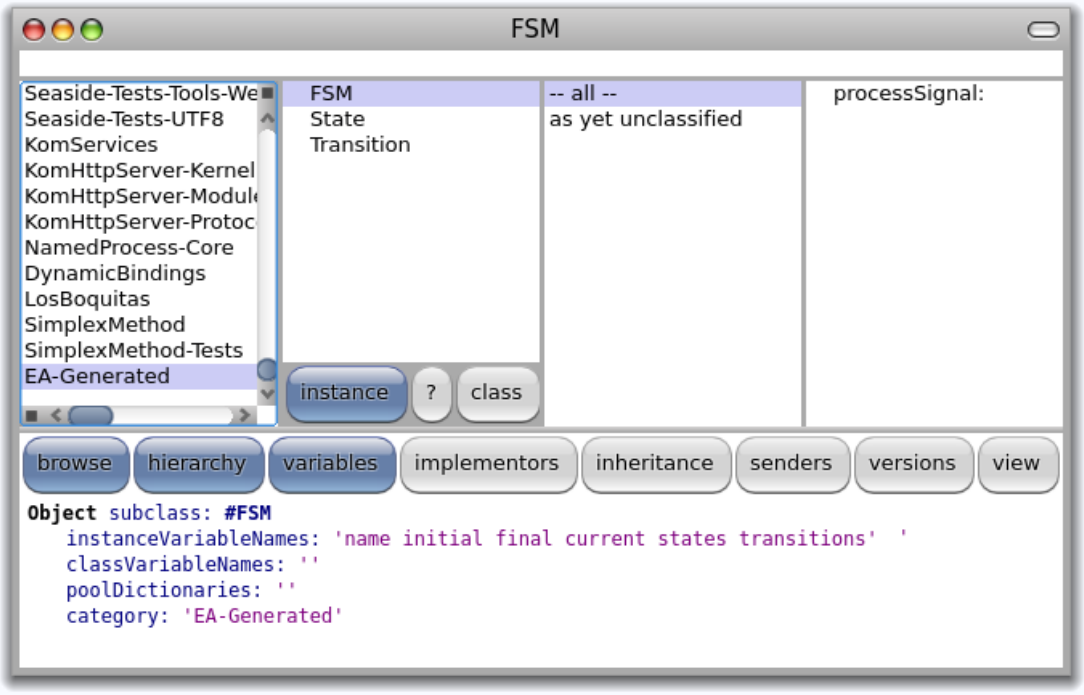
MDG technologie

Enterprise Architect obsahuje funkcionalitu pro vytvoření tzv. *MDG technologie* (MDG = Model Driven Generator), což je sada UML profilů (včetně speciálních typů diagramů), transformačních a generovacích šablon a obrázků (např. pro vlastní ikony elementů). Pomocí takovéto sady lze rozšířit možnosti tohoto CASE o zcela nové vlastnosti a přízpůsobit ho tak např. nějaké specifické doménové oblasti.

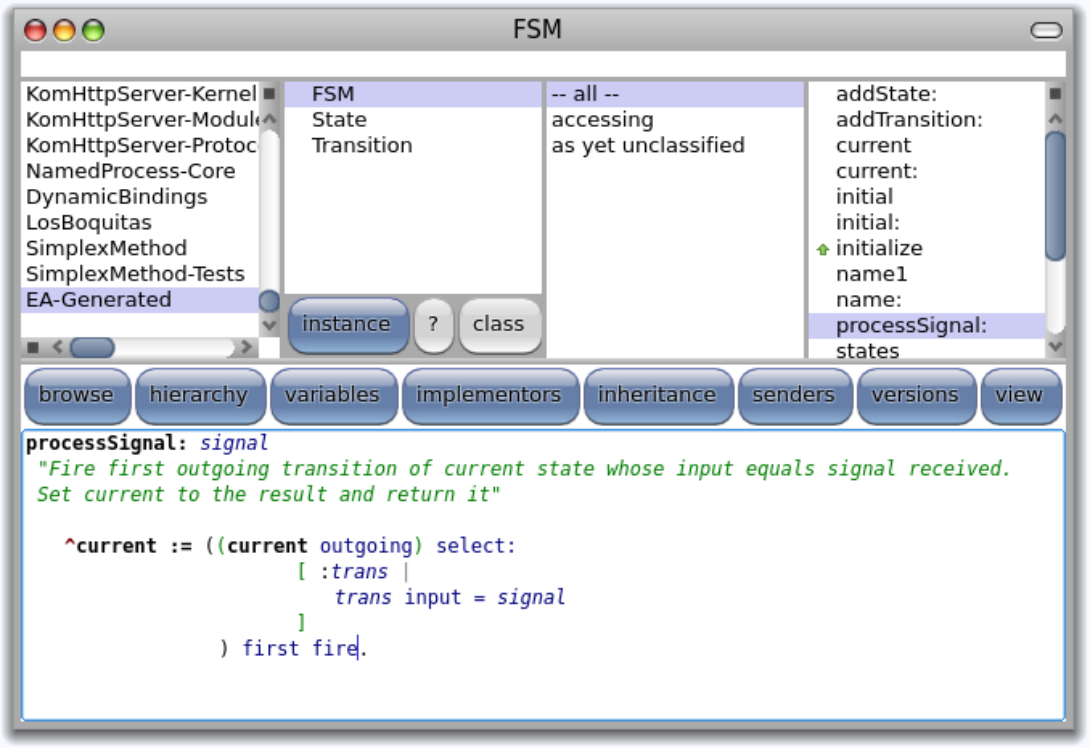
Pomocí této formátu lze např. *Enterprise Architect* rozšířit o podporu *BPMN - Business Process Modeling Notation* - grafický jazyk určený pro popis business procesů.

Shrnutí

Vzhledem k tomu, že s nástrojem Enterprise Architect mám nejbohatší zkušenosti, použil jsem jej pro realizaci projektu v kapitole 6.



Obrázek 14: Pharo - Class Browser s vygenerovanou třídou FSM.



Obrázek 15: Pharo - Class Browser s metodou processSignal vygenerované třídy FSM. Na obrázku je patrné zvýraznění syntaxe různými barvami a styly textu. Dále jsou patrné metody pro přístup k členským proměnným (*accessors*) vytvořené automaticky refaktorováním.

5 Transformační modelovací jazyky

Pro podporu metamodelování a transformací vznikly specializované transformační modelovací jazyky, z nichž některé v této kapitole popíšu.

5.1 KerMeta

KerMeta je Modelově orientovaný jazyk založený na paradigmatu metamodelování s možností vykonatelnosti modelu v objektově orientovaném prostředí. Je vytvořen jako rozšíření MOF směrem k vykonavatelnosti. [Tanguy,Vojtisek,Faucher, 2006]

Jeho vývojovým i běhovým prostředím je *Eclipse IDE* (integrované vývojové prostředí).

Model *KerMeta* lze vytvořit buď ručně nebo konverzí z *Ecore EMF* metamodelu.

Výsledek při použití Ecore modelu konečného stavového automatu (viz Obrázek 8) zásuvný modul Kermeta vygeneruje následující zdrojový soubor `fsm.kmt`:

```
@uri "platform:/resource/MyFirstEMFSamples/metamodels/fsm.ecore"
package fsm;

require "kermeta"
require "http://www.eclipse.org/emf/2002/Ecore"
class Fsm
{
     attribute transition : Transition[0..*]
     attribute state : State[0..*]
     reference initial : State[1..1]
     reference final : State[1..1]
     reference current : State[1..1]
     operation prettyprint() : ecore::EString is
           abstract
}
class Transition
{
     attribute input : ecore::EString
     attribute output : ecore::EString
```

```
    reference source : State[1..1]
    reference target : State[1..1]
}
class State
{
    attribute name : ecore::EString
    reference outgoing : Transition[0..*]
    reference incoming : Transition[0..*]
}
```

Tento formát je vhodný pro psaní kódu, zatímco formát `*.km` lze zobrazit a upravovat v *Sample Reflexive Ecore Model Editoru*, případně jiném *GMF* editoru.

Modely je možné libovolně konvertovat mezi oběma formáty pomocí funkcí zásuvného modulu Kermeta.

Vlastnosti jazyka

Jazyk *KerMeta* obsahuje moderní objektové konstrukty jako jsou generické třídy a operace a v neposlední řadě lambda výrazy. To dává jeho uživateli širokou škálu možností a jeví se jako poměrně slibná platforma.

Bohužel se zdá, že vývoj tohoto nástroje poněkud ustrnul a na webových stránkách jeho autorů jsou k dispozici zatím pouze velmi jednoduché příklady bez skutečného užitku. Přesto *KerMeta* vypadá slibně a bude určitě zajímavé sledovat, kam se vývoj tohoto jazyka bude dále ubírat.

5.2 Eclipse Modelling Framework

Jak již bylo popsáno v kapitole 4.7, EMF je framework, jehož síla je v transparentní transformaci mezi XMI, XML Schema a anotovaným zdrojovým kódem v Javě.

S pomocí jeho API lze z Javového kódu přistupovat k entitám modelu a tak vytvářet transformované modely nebo generovat libovolný kód.

Jedná se o obdobný přístup, jaký jsem aplikoval v kapitole **??** s tím rozdílem, že tento framework je omezen skutečně čistě na Javu.

5.3 C.C language

Jazyk C.C je funcionální programovací jazyk se syntaxí blízkou jazyku PASCAL s několika imperativními konstrukty a s několika vlastnostmi inspirovanými jazyky PROLOG, Erlang, Ruby, Python a Smalltalk. Jedná se o interpretovaný jazyk provozovaný v prostředí CASE nástroje *Craft.CASE* na platformě *VisualWorks*.

C.C se používá pro následující účely:

- skriptovací jazyk - Procedury v C.C jsou schopny procházet projektovou databází a vytvářet nejrůznější dokumentační sestavy
- precizní simulace procesů - Procedury v C.C mohou vypočítávat různá simulační data, řídit průběh simulace apod.
- automatické manipulace s modelem - např. aplikování návrhových vzorů, refaktoring, objektová normalizace apod.
- kontroly konzistence a integrity nad projektovou databází - tato vlastnost pokrývá stejnou funkcionalitu jako OCL.
- export dat v různých formátech (jmenovitě XMI a binární formáty některých jiných CASE nástrojů).
- import dat z různých datových zdrojů (ODBC, CSV atd.)

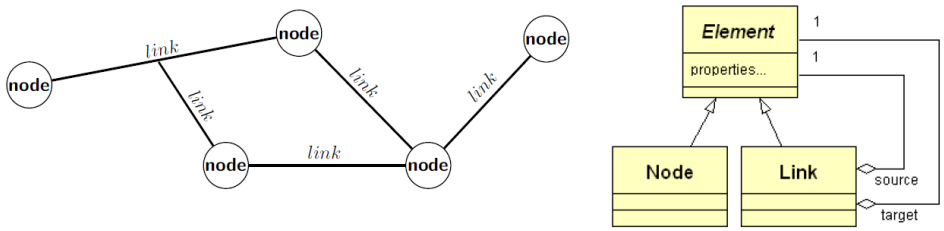
Motivation

Jazyk C.C pokrývá vlastnosti různých, již existujících nástrojů. Jeho autoři požadovali zakomponování těchto vlastností do modelovacího prostředí Craft.CASE zejména pro dosažení následující funkcionality:

- interaktivní transformace modelů
- prototypování a testování na úrovni instancí modelovaných objektů (dotazování a manipulace s daty konkrétních instancí objektů vytvořených z definice v modelu) a
- simulace.

[Merunka,Nouza,Brožek,2008]

Jazyk C.C zprostředkovává repository modelu prostřednictívm metamodelu na obrázku 16.



Obrázek 16: C.C metamodel repository Craft.CASE. [Merunka,Nouza,Brožek,2008]

5.4 XSLT

Za nejuniverzálnější transformační jazyk by se dal zřejmě označit jazyk *eXtensible Stylesheet Language Transformations (XSLT)*.

Vzhledem k tomu, že většina současných CASE nástrojů umožňuje export do formátu XMI, lze takovýto soubor transformovat (pomocí XSLT procesoru) do jednoho či více souborů buď opět ve formátu XMI (pak by se tedy jednalo o transformaci modelu do modelu) nebo ve formátu nějakého programovacího jazyka.

Jedná se o zcela obecný jazyk pro transformaci textů ve formátu XML, takže neobsahuje žádnou podporu metamodelu XMI. Veškerou logiku by si tak tvůrce transformační šablony musel naprogramovat sám. Reálně tedy o jeho použití lze zřejmě uvažovat spíše pro velmi jednoduché až triviální transformace.

Část II

Projekt

6 Vlastní projekt

6.1 Úvod

V této kapitole na realizaci informačního systému pro podporu rozhodování ukážu, jak v rámci iterativního postupu může být využit přístup Architektury řízené modelem. Bude ukázáno, jaké výhody tento přístup může přinést, ale budou rovněž naznačena omezení, se kterými je potom třeba počítat.

Zemědělský podnik X vydefinuje své potřeby. Realizační tým provede analýzu požadavků a navrhne vytvoření informačního systému *DecisionMaker*. V první iteraci implementuje komponentu pro evidenci *modelů*. Při implementaci evidence *modelů* odhalí návrhový vzor společný všem evidovaným entitám. Realizační tým tento návrhový vzor popíše a navrhne UML profil pro usnadnění zápisu všech potřebných údajů do modelu. Analytici upraví model s použitím vytvořeného profilu. Implementace dalších entit by byla z větší části mechanickou interpretací vzniknuvšího *Doménově specifikého jazyka*. Proto následně realizační tým vytvoří generátor, který tuto rutinní práci odstraní. S pomocí generátoru pak realizační tým vytvoří komponenty pro všechny doménové entity a ručně je pospojuje do funkční aplikace.

6.2 Výchozí situace

Vedení menšího zemědělského podniku v rámci implementace nové strategie zamýšlí optimalizovat výrobu. Chce za tímto účelem doplnit podnikový informační systém o aplikaci pro podporu rozhodování na základě aplikace lineárního programování.

Stávající podnikový informační systém sestává z

- PIS Helios Orange (na SQL Serveru 2008)
- zemědělský informační systém GC ÚPRAVY®(Moduly Užívací vztahy, GC Mapa, Registr zvířat)
- Zápůjčky techniky (Vlastní vyvinutá aplikace provozovaná v prostředí Pharo)
- Servisní zásahy (Vlastní vyvinutá aplikace provozovaná v prostředí Pharo)

Pharo

je open-source prostředí Smalltalk dostupné zdarma. Jedná se o pokračování (větev) projektu *Squeak*. Na rozdíl od *Squeaku*, který byl a je určen především pro děti pro výuku programování, je určen především pro profesionální vývojáře, zejména se zaměřením na vývoj webových aplikací ve frameworku *Seaside*. [Black,Ducasse,Nierstrsz,Pollet 2009]
Více informací viz: `http://www.pharo-project.org/home`

Seaside

je framework v prostředí Smalltalk určený pro vývoj webových aplikací. *Seaside* je založen na komponentovém přístupu, kdy každá vizuální komponenta webové aplikace je implementovaná vlastní třídou, potomka třídy `WAComponent`. *Seaside* je k dispozici pro řadu Smalltalk platforem, mj. *Pharo*.
Více informací viz: `http://www.seaside.st/`

Vzhledem k existujícím aplikacím v prostředí Pharo se podnik rozhodne realizovat na stejné platformě i tento nový projekt.

Protože chce postupovat v souladu s osvěčenou metodikou Unified Process, hodlá se držet i jejích hlavních zásad (tzv. „Best practices“):

- Iterativní vývoj
- Vizuální modelování
- Komponentová architektura
- Správa požadavků
- Kontinuální verifikace kvality
- Řízení změn

Pro vizuální modelování použije nástroj Enterprise Architect, neboť za přijatelnou cenu poskytuje přehledné uživatelské rozhraní, umožňuje generování dokumentace ve formátu RTF nebo HTML a v neposlední řadě velmi dobře podporuje koncepci Architektury řízené modelem (MDA).

7 Požadavky na informační systém

7.1 Funkční požadavky na informační systém

Vedení podniku od nového informačního systému očekává, že s jeho pomocí bude moci snadno a rychle vyhodnocovat situaci a činit rozhodnutí o následujících obchodních aktivitách.

Prostředí zemědělských podniků je extrémně dynamické, neboť je ovlivňováno řadou legislativních omezení, měnících se kvót, sazeb, proměnlivých cen paliva, hnojiv, osiv, výkupních cen plodin a dalších faktorů.

Za takové situace je velmi těžké dělat rozhodnutí a dá se předpokládat, že bez použití matematických modelů podpořených výpočetní technikou by úspěšnost takových rozhodnutí byla mizivá.

Je proto požadováno vytvoření informačního systému *Decision Maker*.

Tento systém má umožňovat definování linárního optimalizačního modelu, jehož proměnnými budou např. osevní plochy jednotlivých plodin, počty kusů chovaného dobytka, ale třeba též rozsah ploch ponechaných ladem. Parametry těchto modelů budou výše uvedené údaje jako ceny výrobních faktorů, realizační ceny vyráběných produktů apod. Omezujícími podmínkami pak budou např. kapacity chovných zařízení, celkové výměry osevních ploch atd.

Tyto modely pak budou automaticky řešeny za pomoci *Simplexové metody*.

Do budoucna je zamýšlen rozvoj systému do takové podoby, ve které si bude umět parametry a omezující podmínky automaticky aktualizovat pomocí webových služeb např. ze systému GC Úpravy, z on-line burzy plodin či meteoslužby předpovědi počasí. Na základě těchto nových údajů by pak měl systém automaticky přepočítávat a hlásit změny výstupních hodnot jako podklady pro operativní rozhodování managementu.

7.2 Nefunkční požadavky na systém

Systém z počátku nemusí být stavěn pro velké zatížení. Jeho návrh však musí být dostatečně flexibilní a do budoucna škálovatelný pro případ, že by se podnik rozrostl a systém by začalo používat více lidí.

Zvolená platforma *Pharo* a framework *Seaside* tomuto požadavku odpovídají, neboť aplikace napsané ve Smalltalku lze relativně jednoduše portovat např. na

platformu *VisualWorks* nebo *GemStone*, které sice již nejsou zdarma, ale zato dokáží zaručit solidní škálovatelnost.

Vedení podniku zároveň chce být připraveno na případné zahraniční akvizice nebo asociace a proto požaduje, aby veškerý zdrojový kód systému byl psán v angličtině. Uživatelské rozhraní pak musí podporovat vícejazyčnost, alespoň na úrovni konfigurace. (Přepínání aktivního jazyka konfiguračním parametrem v kódu.)

8 Analýza

8.1 Model případů užití

V rámci analýzy byly identifikovány následující případy užití a doménový objektový model.

8.1.1 Případy užití

Modul MainPage

UC001-Zobraz výchozí obrazovku
UC002-Vyber činnost
UC100-Spravuj Skupiny parametrů
UC101-Zobraz seznam Skupin parametrů
UC102-Zobraz Skupinu parametrů
UC103-Vytvoř Skupinu parametrů
UC104-Zruš Skupinu parametrů
UC105-Uprav atributy Skupiny parametrů
UC106-Přidej člena do Skupiny parametrů
UC107-Vyřaď člena ze Skupiny parametrů
UC108-Obnov hodnoty všech Parametrů ve Skupině

UC200-Spravuj Parametry

UC201-Zobraz seznam Parametrů
UC202-Zobraz Parametr
UC203-Vytvoř Parametr
UC204-Zruš Parametr
UC205-Uprav Parametr
UC206-Obnov hodnotu Parametru

UC300-Spravuj Datové zdroje

UC301-Zobraz seznam Datových zdroju
UC302-Vytvoř Datový zdroj
UC303-Zruš Datový zdroj
UC304-Testuj Datový zdroj
UC310-Uprav Datový zdroj

UC311-Uprav Datový zdroj typu Web service
UC312-Uprav Datový zdroj typu Lambda

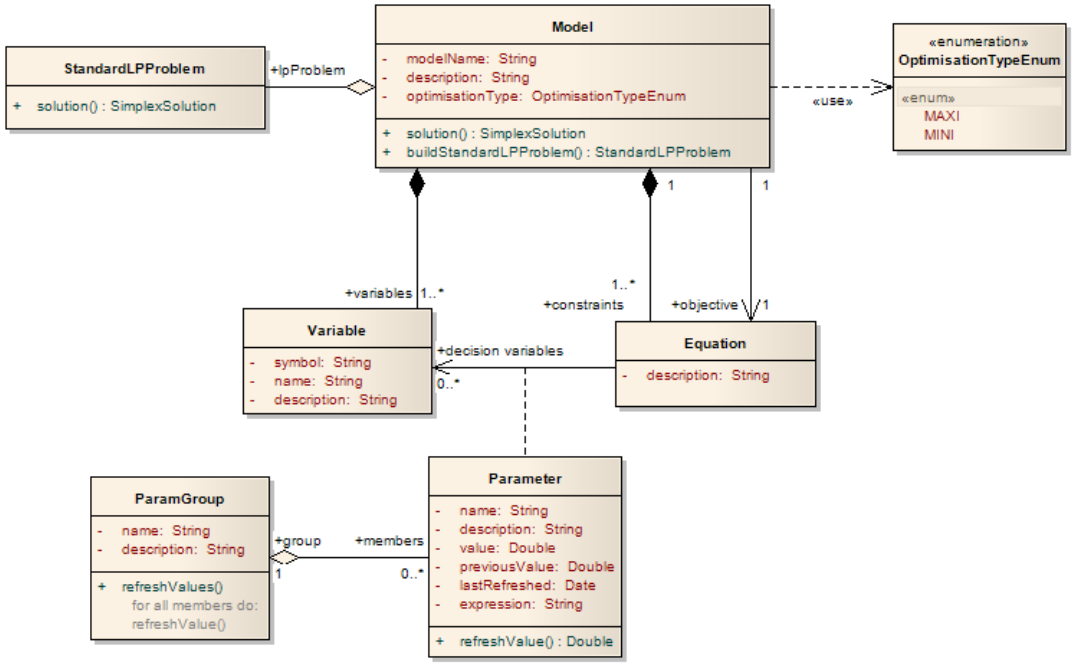
UC400-Spravuj Modely
UC401-Zobraz seznam Modelů
UC402-Zobraz Model (Zobraz seznam rovnic modelu)
UC403-Vytvoř Model
UC404-Zruš Model
UC405-Uprav Model
UC420-Zobraz seznam proměnných modelu
UC421-Zobraz proměnnou modelu
UC422-Vytvoř proměnnou modelu
UC423-Zruš proměnnou modelu

8.2 Doménový objektový model

V rámci analýzy byly identifikovány následující doménové třídy:

- Model
- Equation (rovnice)
- Parameter (parametr)
- Variable (proměnná)
- ParamGroup (skupina parametrů)

Atributy tříd viz obrázek 17.

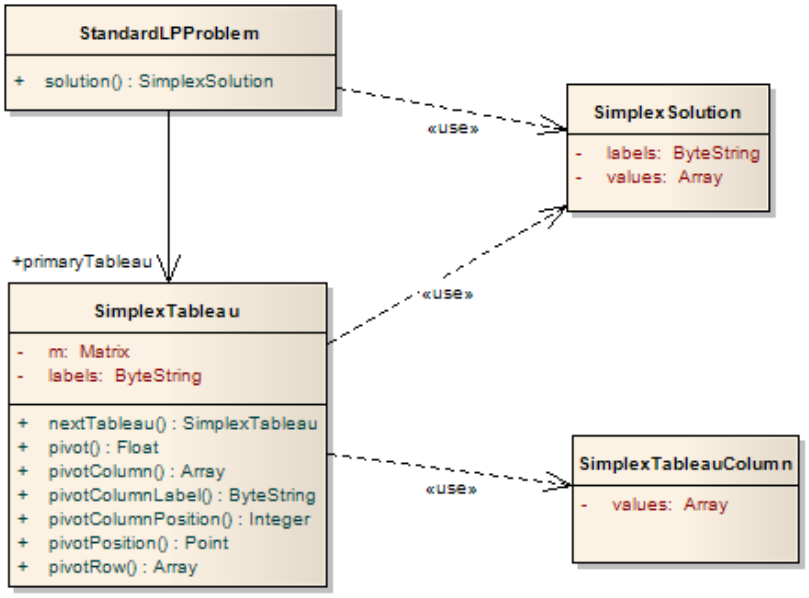


Obrázek 17: Diagram tříd doménového modelu

9 Design informačního systému

9.1 Diagram implementačních tříd systému Decision Maker

Pro implementaci jádra systému, tj. algoritmu pro řešení lineárního optimalizačního problému simplexovou metodou byl navržen objektový model na obrázku 18.

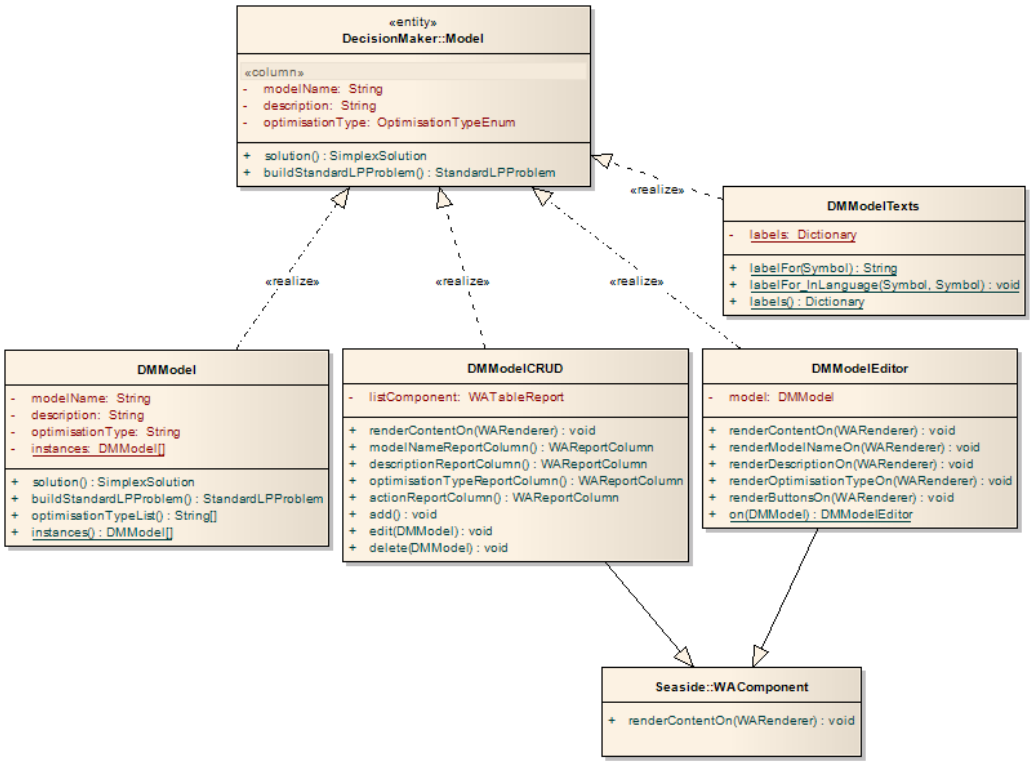


Obrázek 18: Diagram tříd pro podporu Simplexové metody

V rámci první iterace vývoje je třeba vytvořit v prostředí webového frameworku Seaside vzorovou implementaci sady obrazovek pro editaci jedné entity. Pro tuto vzorovou implementaci je zvolena entita `Model`. Pro implementaci editace jsou navrženy následující implementační třídy:

- DMModel
- DMModelCRUD (Create Read Update Delete)
- DMModelEditor
- DMModelTexts (sada textů)

  Detaily jednotlivých tříd viz obrázek 19.



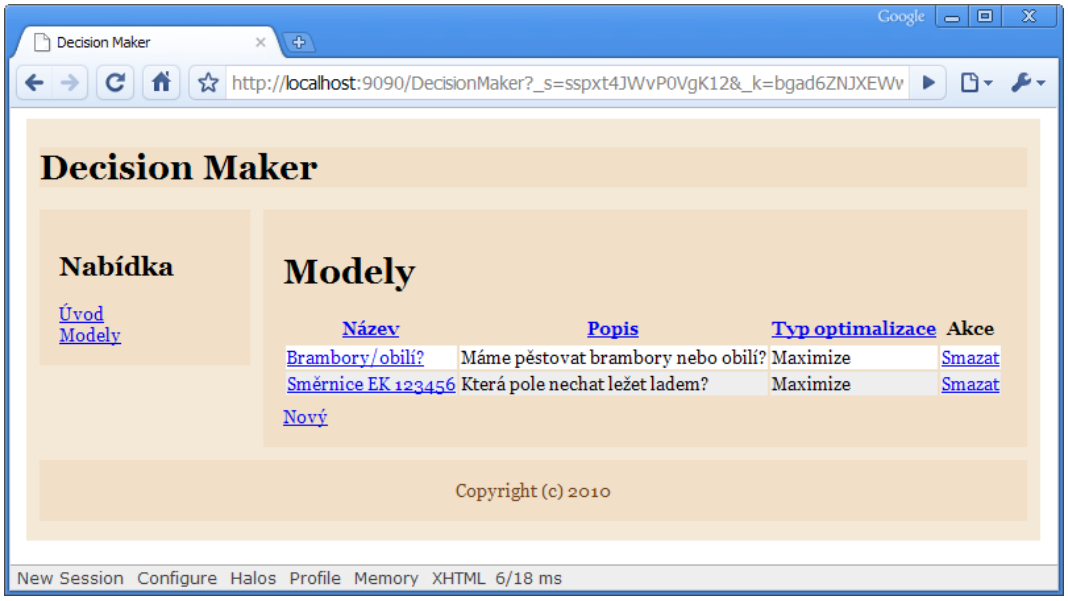
Obrázek 19: Diagram tříd pro editaci entitních dat v prostředí frameworku *Seaside*.

10 Aplikace Architektury řízené modelem (MDA)

10.1 Identifikace návrhového vzoru

Analýzou implementačních tříd entity Model (viz obrázek 19) bylo zjištěno, že se jedná o obecný návrhový vzor, který by měl být použit i pro implementaci ostatních entit.

Implementace v prostředí Seaside je patrná na obrázku 20.



Obrázek 20: DecisionMaker - obrazovka po první iteraci.

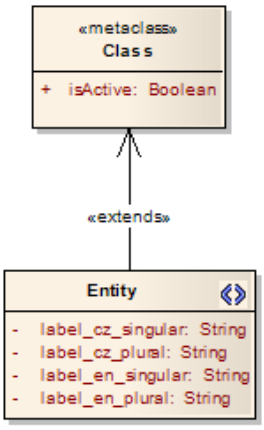
V první iteraci byla implementována správa entity *Model*. Výsledkem jsou čtyři třídy - *model*, *CRUD (Create Read Update Delete)*, *editor* a *sada textů*.

Je proto rozhodnuto, že tento vzor bude popsán a bude pro vytvořen UML profil pro doplnění všech informací potřebných pro implementaci.

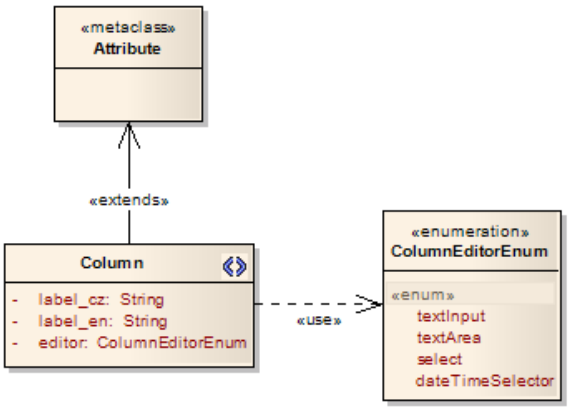
Tento UML profil je realizován, jak je patrné na obrázcích 21 a 22.

Doménový model je doplněn o stereotypy a tagované hodnoty UML profilu. Viz obrázky 23, 26 a figTaggedValues2.

Dále vývojový tým identifikuje, že takto doplněný model obsahuje všechny potřebné informace pro vygenerování kódu. Rozhodne se proto poněkud nabourat



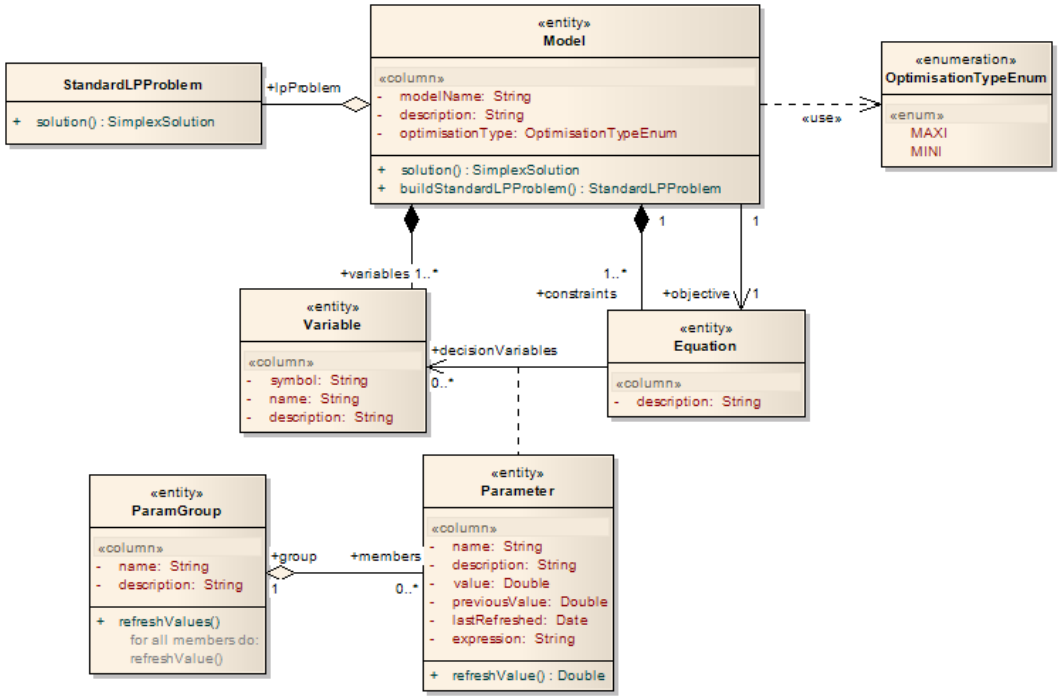
Obrázek 21: UML profil - stereotyp ≪entity≫



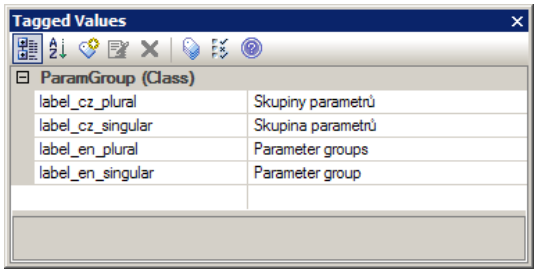
Obrázek 22: UML profil - stereotyp ≪column≫

plán vývoje a věnovat určitý čas implementaci generátoru v prostředí *Pharo* s tím, že toto zdržení se následně vyplatí vzhledem k ušetřené rutinní práci.

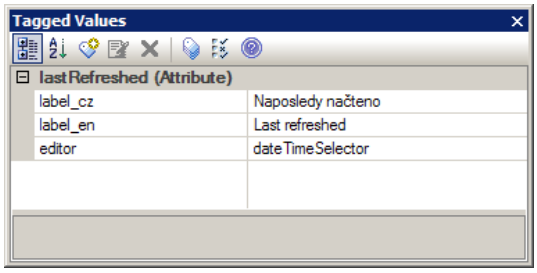
> Poznámka: Zde je ovšem třeba dát si pozor, aby se práce na generátoru zbytečně neprodlužovala, neboť existuje riziko, že pro vývojáře bude taková činnost zajímavější než vývoj samotného koncového systému a mohou tak mít tendenci generátor neustále vylepšovat ne úkor práce na hlavním systému.



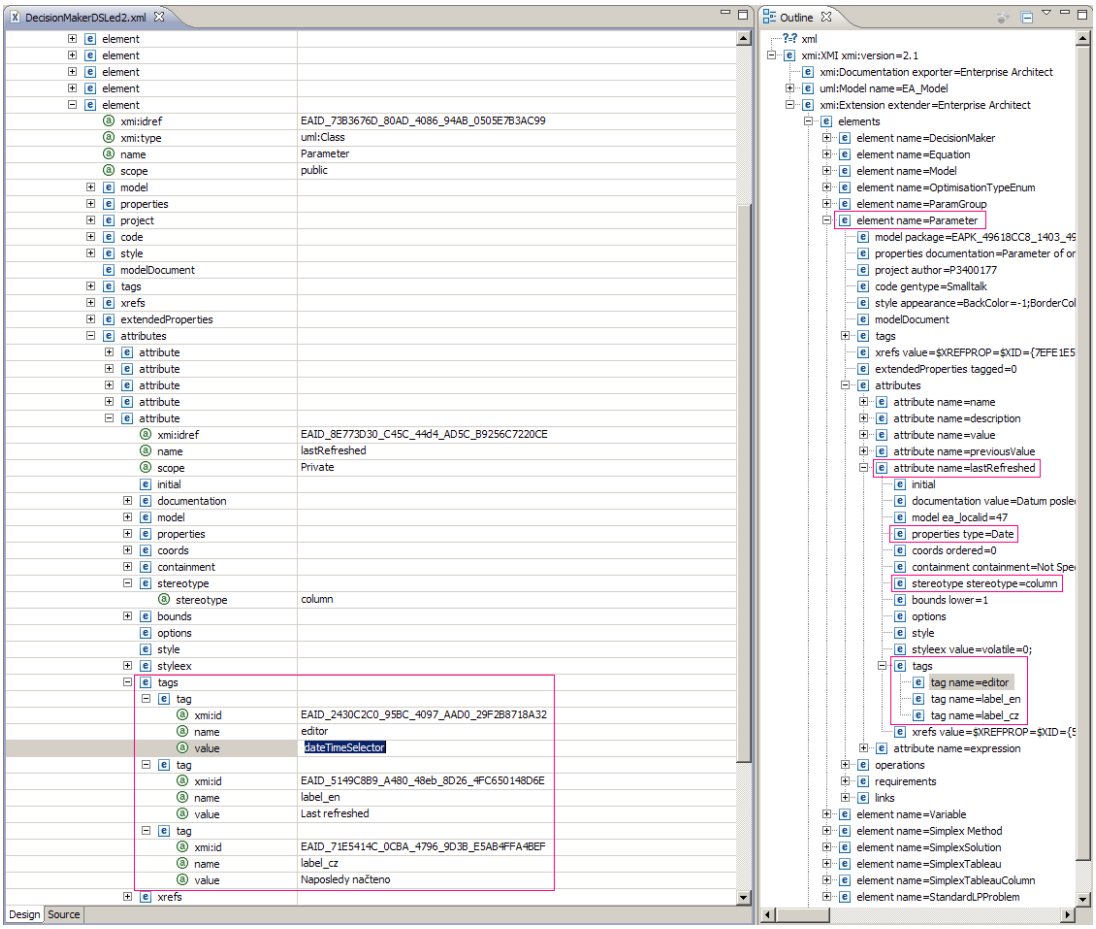
Obrázek 23: Doménový model po aplikaci UML profilu.



Obrázek 24: Doménový model po aplikaci UML profilu - tagované hodnoty entity `ParamGroup`.



Obrázek 25: Doménový model po aplikaci UML profilu - tagované hodnoty atributu `lastRefreshed`.



Obrázek 26: Doménový model po aplikaci UML profilu - vyexportovaný do formátu XMI. Na obrázku jsou zvýrazněny tagované hodnoty atributu `lastRefreshed` třídy `Parameter`. Patrný je rovněž stereotyp ≪column≫. Pro přehlednost zobrazeno v XML editoru prostředí *Eclipse*.

11 Vývoj generátoru

Pro vyvtvoření generátor musí být nejprve implementovány třídy metamodelu. Výsledek viz příloha E.

Dále musí být implementován mechanismus pro import modelu. Použije se standardní formát XMI. Výsledek viz: příloha E

Na základě naimportovaného modelu z CASE pak musí proběhnout transformace do pomocného modelu (entita -¿ model, CRUD, editor a sada textů) a z tohoto pomocného modelu se pak vygenerují výsledné implementační třídy. Realizace transformátoru viz příloha F.

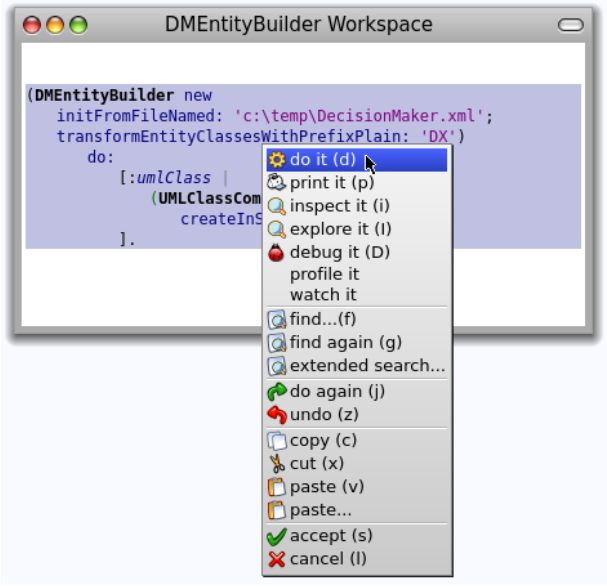
12 Generování kódu z modelu

Veškeré implementační třídy modelovaných entit se vygenerují jedním příkazem ve *Workspace*:

```
(DMEntityBuilder new
    initFromFileNamed: 'c:\temp\DecisionMaker.xml';
    transformEntityClassesWithPrefixPlain: 'DM')
        do:
            [:umlClass |
                (UMLClassCompiler new: umlClass)
                    createInSmalltalk.
            ].
```

viz též obrázek 27.

Výsledné vygenerované třídy jsou zobrazeny v *System Browseru* na obrázku 28

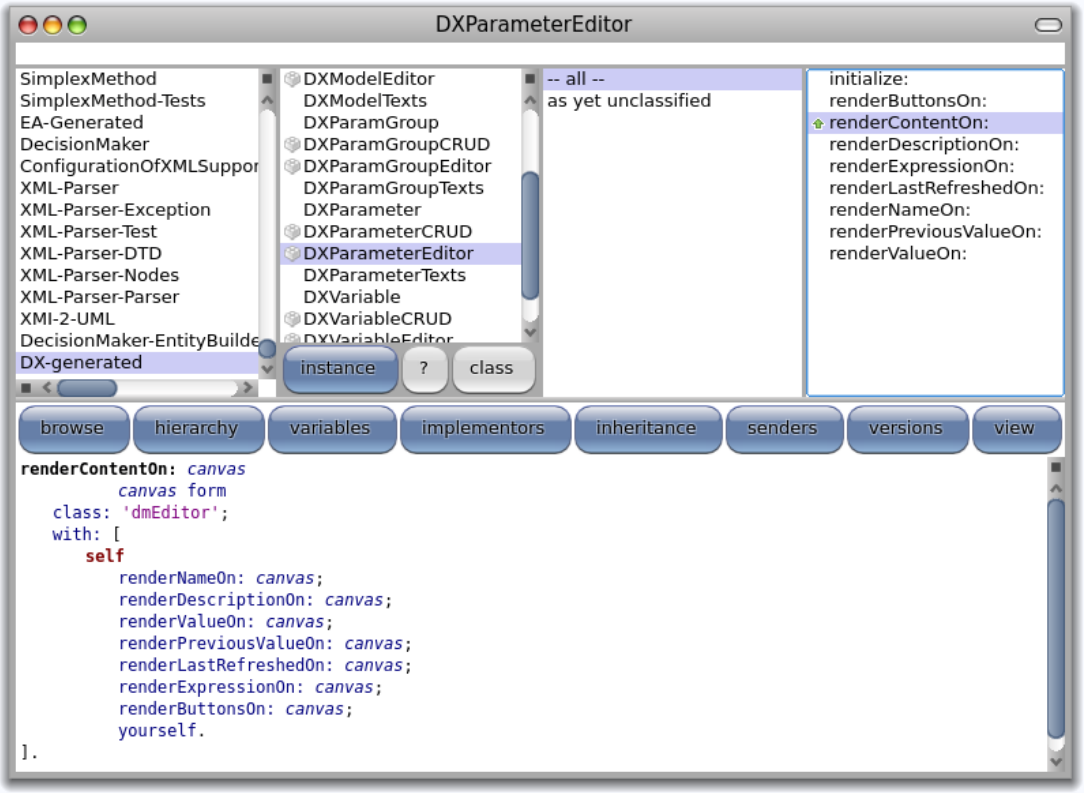


Obrázek 27: Spuštění transformace modelu na implementační třídy v prostředí *Pharo*.

Prvotní zakomponování vygenerovaného kódu do aplikace se provede doplněním zdrojového kódu medody `renderSidebarOn:` třídy `DMMain`, která slouží pro zobrazení menu aplikace.

Úprava viz níže:

```
renderSidebarOn: canvas
    canvas div
        id: 'sidebar';
        class: 'section';
        with: [
            canvas heading
                level2;
                with: DMTexts MENU.
            canvas anchor
                callback: [ mainArea := DMHome new ];
                with: DMTexts HOME.
            canvas break .
            canvas anchor
                callback: [ mainArea := DMModelCRUD new ];
                with: (DMModelTexts labelFor: #Models).
            "========= call GENERATED classes ========"
            canvas break .
            canvas anchor
                callback: [ mainArea := DXEquationCRUD new ];
                with: (DXEquationTexts labelFor: #Instances).
            canvas break .
            canvas anchor
```



Obrázek 28: Zobrazení vygenerovaných implementačních tříd v System Browseru.

```
            callback: [ mainArea := DXVariableCRUD new ];
            with: (DXVariableTexts labelFor: #Instances).
        canvas break .
        canvas anchor
            callback: [ mainArea := DXParamGroupCRUD new ];
            with: (DXParamGroupTexts labelFor: #Instances).
        canvas break .
        canvas anchor
            callback: [ mainArea := DXParameterCRUD new ];
            with: (DXParameterTexts labelFor: #Instances).
        "======== /call GENERATED classes ========"
    ];
yourself.
```

Výsledná aplikace po doplnění generovaných částí je zobrazena na obrázku 29.

Takto vytvořená aplikace bude pochopitelně dále upravována v rámci dalších vývojových iterací. Pokud to bude vyhodnoceno jako efektivní, může být opět použito generování - zejména pokud by se významně změnil výchozí doménový model a zároveň objem ručně provedených zásahů do generovaných tříd by byl minimální.

Obrázek 29: Vzhled aplikace po doplnění generovaných částí.

V každém případě je však potřeba tento postup pokaždé zvažovat s ohledem na očekávané přínosy v porovnání s vynaloženou prací.

13 Závěr

V diplomové práci jsem se zabýval problematikou modelování skutečností reálného světa i modelování informačních systémů s využitím výpočetní techniky.

Popsal jsem některé modelovací nástroje a transformační jazyky jako EMF, KerMeta, C.C language a XSLT a posoudil jejich použitelnost pro různé účely.

Představil jsem koncepci Architektury řízené modelem (Model Driven Architecture - MDA) a její aplikaci jsem ilustroval na příkladu tvorby informačního systému pro podporu rozhodování založeného na řešení úloh lineárního programování simplexovou metodou.

V rámci tohoto příkladu jsem simuloval dvě iterace vývoje. V první iteraci byl identifikován a popsán společný návrhový vzor. V mezifázi byl vytvořen doménově specifický jazyk (Domain-specific language) v podobě UML profilu. Pro podporu modelování tohoto návrhového vzoru byl vytvořen transformátor a s jeho pomocí byla realizována druhá iterace vývoje.

Jak samotný vzorový informační systém, tak transformátor modelu byl implementován v jazyku Smalltalk v prostředí *Pharo*. Dynamičnost tohoto objektově orientovaného prostředí se dle mého názoru pro tento účel velmi osvědčila.

Domnívám se proto, že pro aplikaci principů Architektury řízené modelem je prostředí založené na jazyku Smalltalk dobrou volbou.

Literatura

[Polák,Merunka,Carda, 2003] Polák, J.; Merunka, V.; Carda, A. *Umění systémového návrhu* Grada Publishing a.s., 2003. 196 s. ISBN 80-247-0424-2

[Merunka,Nouza,Brožek,2008] Vojtěch Merunka, Oldřich Nouza, and Jiří Brožek Automated Model Transformations Using the C.C Language. In *Advances in Enterprise Engineering I. 4th International Workshop CIAO! and 4th International Workshop EOMAS, held at CAiSE 2008, Montpellier, France, June 16-17, 2008. Proceedings.* Berlin Heidelberg: Springer-Verlag, 2008 Part 4. Model Transformation and Simulation. s. 137-151.

[Black,Ducasse,Nierstrsz,Pollet 2009] Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz and Damien Pollet. *Pharo by Example* Version of 2009-10-28. Bern: Square Bracket Associates, Switzerland, 2009. ISBN 978-3-9523341-4-0.

[OMG, 2003] The Object Management Group *MDA Guide V1.0.1* Dokument ve formátu PDF dostupný z <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf>

[Eclipse,2009] The Eclipse Foundation *Eclipse documentation - Current Release[online]* c2009 Dokumentace dostupná z <http://help.eclipse.org/>

[OMG, 2006] Object Management Group, Inc. *Meta Object Facility (MOF) Specification, v.2.0* 2006. Dokument dostupný z <http://www.omg.org/spec/MOF/2.0/>

[Budinsky,Steinberg,Merks,Ellersick,Grose, 2003] Frank Budinsky; David Steinberg; Ed Merks; Raymond Ellersick; Timothy J. Grose *Eclipse Modeling Framework: A Developer's Guide* Addison Wesley Professional, 2003. ISBN-10: 0-13-142542-0.

[Soley, 2000] Richard Soley, Object Management Group, Inc. *Model Driven Architecture* Dokument dostupný z <http://www.omg.org/cgi-bin/doc?omg/00-11-05>

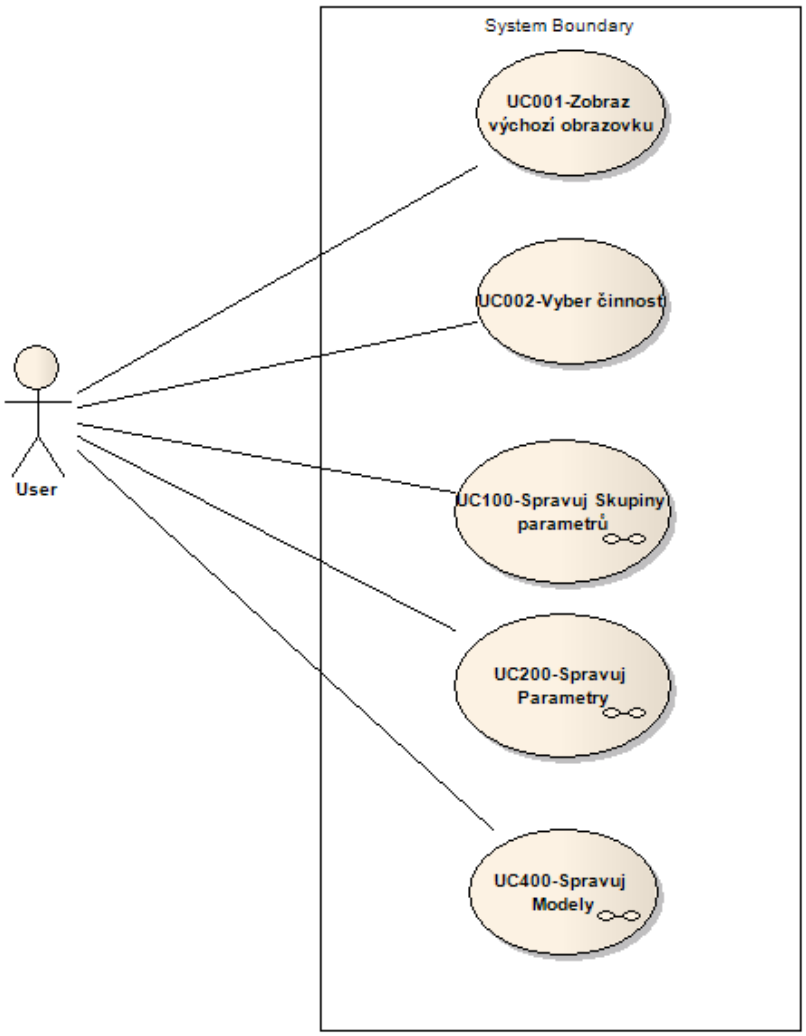
[Tanguy,Vojtisek,Faucher, 2006] François Tanguy, Didier Vojtisek, Cyril Faucher *Kermeta tutorial - How to add behavior to a metamodel* Dokument dostupný z <http://www.kermeta.org/documents/manual/>

[Tomsa, 2007] Tomsa Jan *Analýza konfiguračního řízení Informačního systému katastru nemovitostí* Bakalářská práce

Přílohy

A Případy užití

A.1 Hlavní případy užití



Obrázek 30: Hlavní případy užití

A.1.1 UC001-Zobraz výchozí obrazovku

Po spuštění aplikace se uživateli zobrazí výchozí obrazovka.

Simple Scenario - Simple

Obsah obrazovky:

Nadpis "DecisionMaker".

Obrázek se zemědělskou tematikou (volitelně vyjadřující rozhodování).

V levé části obrazovky menu.

Položky menu:

- Modely
- Skupiny parametrů
- Parametry

A.1.2 UC002-Vyber činnost

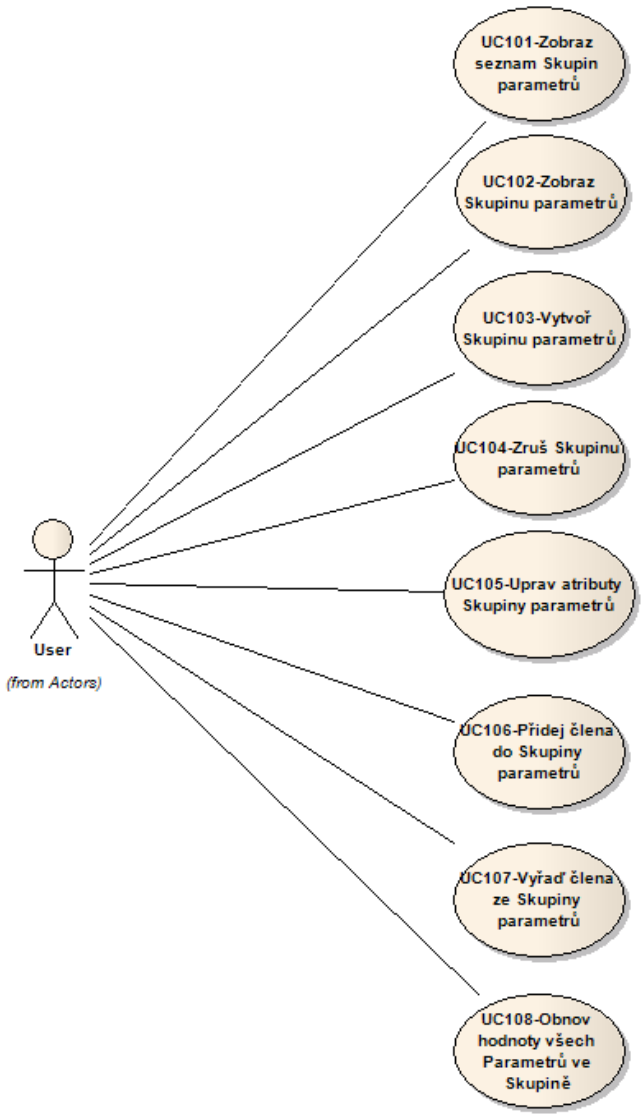
Užvatel zvolí požadovanou činnost výběrem z menu

Simple Scenario - Simple

1. Uživatel má na výběr z následujících činností:

- Zobrazení seznamu Modelů (Správa modelů) -> **UC400**
- Zobrazení seznamu Skupin parametrů (Správa skupin parametrů) -> **UC100**
- Zobrazení seznamu Parametrů (Správa parametrů) -> **UC200**

A.2 Správa skupin parametrů



Obrázek 31: UC100-Spravuj Skupiny parametrů

A.2.1 UC101-Zobraz seznam Skupin parametrů

Basic scenario - Basic path 1. Systém zobrazí seznam *Skupin parametrů* ve formě tabulky.

Zobrazované údaje:

- Název
- Popis
- Počet členů

Třídění: podle Názvu vzestupně

2. Uživatel může zvolit akci se skupinou parametrů nebo s jednotlivým záznamem

2.1 Uživateli budou dostupné následující akce:

- Vytvořit skupinu parametrů -> **UC103**

2.2. Pro každý záznam bude možné vyvolat následující akce:

- Zobrazit detail -> **UC102**
- Upravit atributy -> **UC105**
- Zrušit -> **UC104**

A.2.2 UC102-Zobraz Skupinu parametrů

Basic scenario - Basic path

1. Pro vybranou skupinu parametrů systém zobrazí následující údaje:

- Název
- Parametry ve skupině - zobrazeny ve formě tabulky - zobrazené údaje:

  Název

  Hodnota (?)

  Datum posledního obnovení hodnoty

  Datový zdroj

2. Uživatel může provést akci se skupinou nebo s parametrem ve skupině

2.1. Pro aktuální skupinu parametru bude možné vyvolat následující akce:

- Obnovení hodnot všech parametru -> **UC108**
- Přidání parametru do skupiny -> **UC106**

2.2. Pro každý parametr ve skupině bude možné vyvolat následující akce:

- Vyrazení parametru ze skupiny -> **UC107**
- Obnovení hodnoty parametru -> **UC206**

- Zobrazení detailu parametru -> **UC202**

A.2.3 UC103-Vytvoř Skupinu parametrů

Basic scenario - Basic Path

1. Uživatel zadá údaje:

- Název
- Popis

2. Uživatel zvolí jednu z voleb 2.1. Uživatel zvolí Vytvoření Skupiny parametrů
2.1.1. Vytvoří se nová Skupina parametrů s atributy naplněnými údaji zadanými uživatelem.
2.1.2. -> **UC102**

Alternate scenario - Alternate

2.2. Uživatel zvolí Zrušení
2.2.1. -> **UC102**

A.2.4 UC104-Zruš Skupinu parametrů

Pre-condition Vybrána existující skupina parametrů

Basic scenario - Basic Path

1. Systém zruší vybranou skupinu parametrů.
2. -> **UC101**

A.2.5 UC105-Uprav atributy Skupiny parametrů

Pre-condition Vybrána existující skupina parametrů

Basic scenario - Basic Path

1. Uživatel zadá nové hodnoty atributů:

- Název

2. Uživatel buď potvrdí nebo zruší úpravy
2.1. Uživatel potvrdí úpravy
2.1.1. Hodnoty atributů vybrané Skupiny parametru se nahradí hodnotami zadanými uživatelem
2.1.2. -> **UC102**

Alternate scenario - Alternate

2.2. Uživatel zruší úpravy

2.2.1 -> **UC102**

A.2.6 UC106-Přidej člena do Skupiny parametrů

Pre-condition Vybrána existující skupina parametrů

Basic scenario - Basic Path

1. Systém zobrazí seznam Parametrů, které nejsou členy vybrané skupiny parametru.

Zobrazené údaje:

- Název
- Hodnota
- Datum posledního obnovení hodnoty

2. Uživatel vybere Parametr.

3. Parametr se přidá do množiny členů vybrané Skupiny parametrů.

4. -> **UC102**

A.2.7 UC107-Vyřaď člena ze Skupiny parametrů

Pre-condition Vybraný člen vybrané Skupiny parametrů

Basic scenario - Basic Path

1. Systém vyřadí Vybraného člena z vybrané Skupiny parametrů

2. -> **UC102**

Post-condition Vybraný člen není členem vybrané Skupiny parametrů.

A.2.8 UC108-Obnov hodnoty všech Parametrů ve Skupině

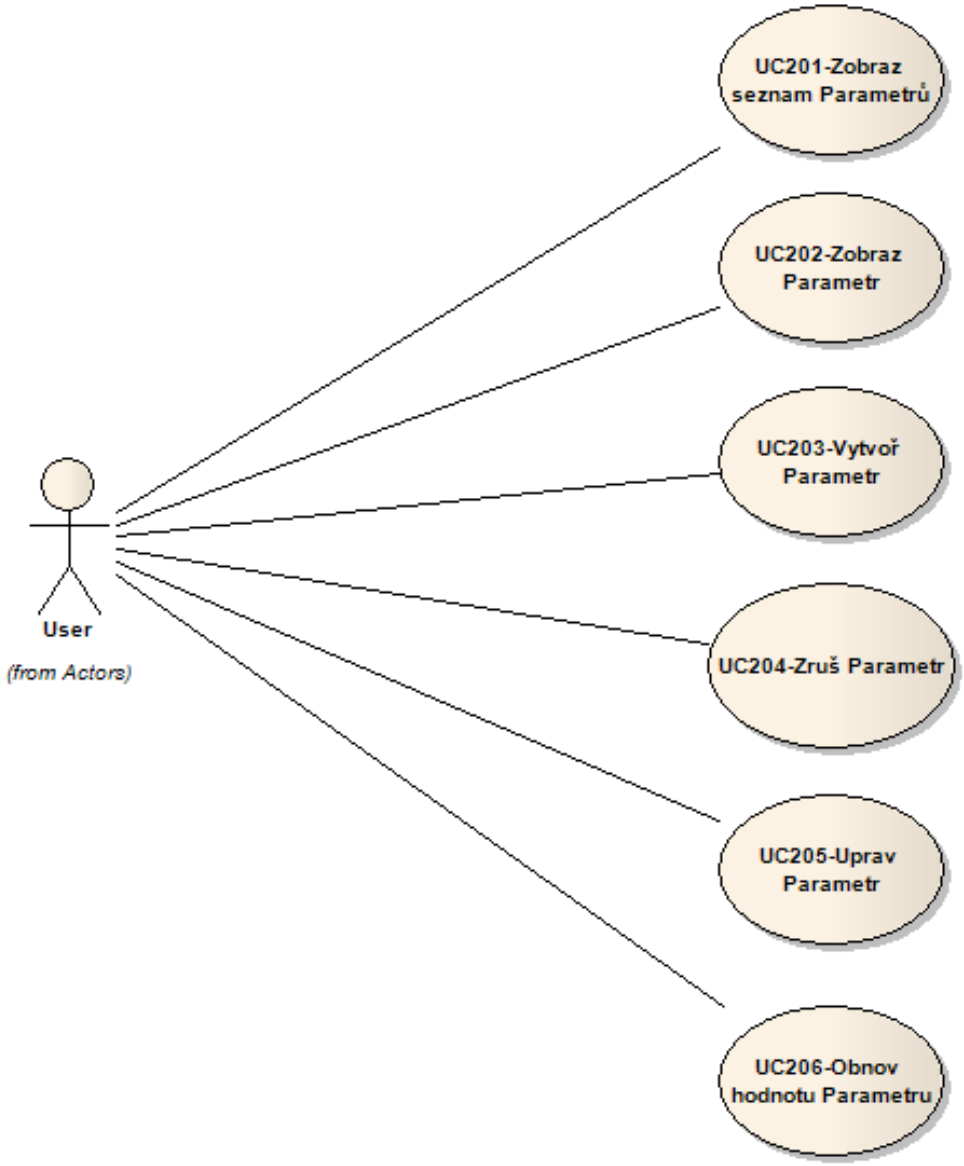
Pre-condition Vybrána Skupina parametrů.

Basic scenario - Basic Path

1. Pro všechny parametry ve vybrané skupině systém provede: **UC206**

2. -> **UC102**

A.3 Správa parametrů



Obrázek 32: UC200-Spravuj Parametry

A.3.1 UC201-Zobraz seznam Parametrů

Basic scenario - Basic Path

1. Uživateli se zobrazí seznam parametrů.

2. Pro každý Parametr zobrazeny atributy:

- Název
- Příslušnost ve skupinách
- Hodnota
- Datum posledního obnovení hodnoty
- Datový zdroj

Třídění: dle názvu parametru (bez možnosti změny)

3. Pro každý Parametr bude možné vyvolat následující akce:

- Obnovit hodnotu Parametru -> **UC206**
- Zobrazit detail Parametru -> **UC202**
- Zrušit Parametr -> **UC204**

A.3.2 UC202-Zobraz Parametr

Pre-condition Vybraný Parametr

Basic scenario - Basic Path

1. Zobrazí se atributy Vybraného Parametru:

- Název
- Hodnota
- Datum posledního obnovení hodnoty
- Datový zdroj
- Skupina parametrů

2. Uživatel má dostupné následující akce:

- Upravit hodnoty Parametru -> **UC205**
- Obnovit hodnotu Parametru z datového zdroje -> **UC206**
- Vyřadit ze Skupiny parametrů (funkce je dostupná, pokud Vybraný Parametr je členem nějaké skupiny)
- Zobrazit seznam parametrů (návrat na seznam) -> **UC201**
- Zobrazit Skupinu parametrů (funkce je dostupná, pokud Vybraný Parametr je členem nejaké skupiny)

- Zrušit Parametr -> **UC204**

A.3.3 UC203-Vytvoř Parametr

Basic scenario - Basic Path

1. Uživatel zadá údaje:
- Název
- Popis
- Datový zdroj
- Skupinu parametru

2. Uživatel zvolí akci:

2.1. Potrvzení vytvoření

2.1.1. Vytvoří se Parametr se zadanými hodnotami

2.1.2. -> **UC201**

Alternate Scenario - Alternate

2.2. Zrušení vytvoření

2.2.1. -> **UC201**

A.3.4 UC204-Zruš Parametr

Pre-condition Vybraný Parametr.

Basic scenario - Basic Path

1. Systém odstraní Vybraný Parametr ze všech Skupin parametrů ve kterých je přítomen.

2. Systém zruší Vybraný Parametr.

3. -> **UC201**

A.3.5 UC205-Uprav Parametr

Pre-condition Vybraný Parametr.

Basic scenario - Basic Path

1. Uživatel zadá nové hodnoty atributu:
- Název
- Datový zdroj (výběr ze seznamu hodnot)

- Skupina parametrů (výěer ze seznamu hodnot)

  2. Zobrazeny následující hodnoty

- Hodnota
- Datum posledního obnovení hodnoty

  3. Uživatel zvolí jednu z akcí

  3.1. uložení hodnot

3.1.1. Atributy Vybraného Parametru se změní na nové hodnoty zadané uživatelem

3.1.2. -> **UC202**

Alternate Scenario - Alternate

3.2. zrušení úprav

3.2.1. -> **UC202**

A.3.6 UC206-Obnov hodnotu Parametru

Pre-condition Vybraný Parametr.

Basic scenario - Basic Path

Pro Vybraný Parametr se provede následující:

1. Z Datového zdroje se načte hodnota.
2. Pokud je načtení hodnoty úspěšné:

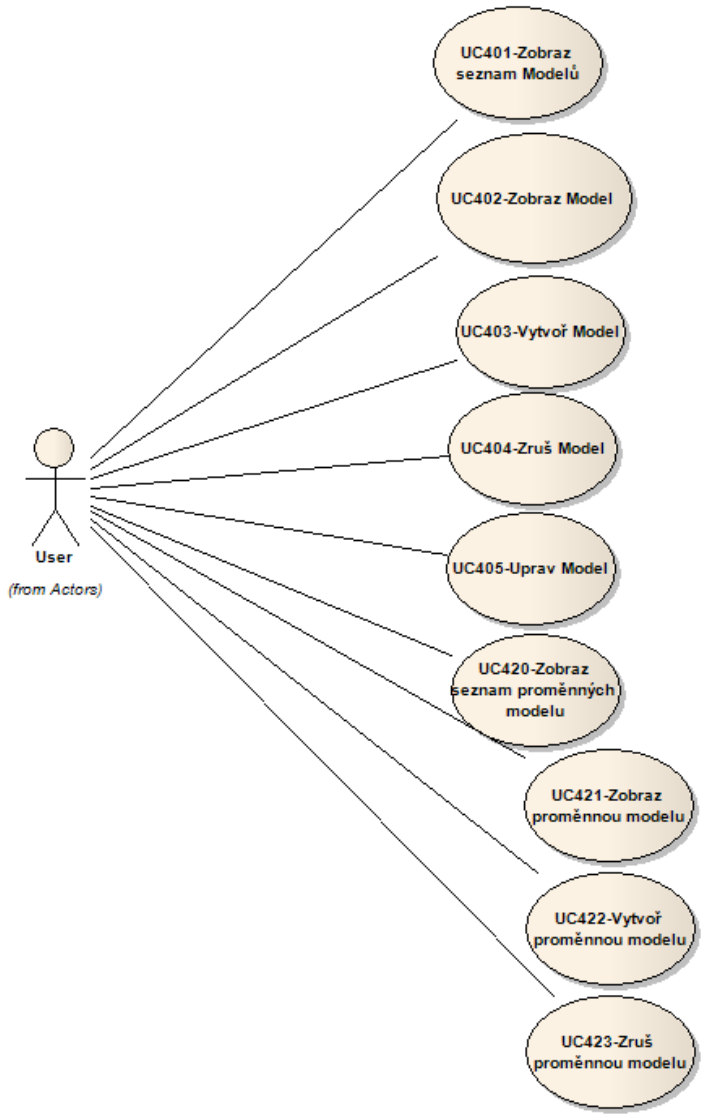
   •Hodnota atributu Hodnota se přiřadí do atributu Předchozí hodnota.

   •Načtená hodnota se přiřadí do atributu Hodnota.

   •Atribut Datum posledního obnovení hodnoty se nastaví na systémové datum a čas.
3. Pokud při načtení hodnoty dojde k chybě:

   •Do Logu se zapíše zpráva "Parametr ¡Vybraný Parametr¿: chyba při načtení hodnoty z datového zdroje Datový zdroj"

A.4 Správa modelů



Obrázek 33: UC400-Spravuj Modely

A.4.1 UC401-Zobraz seznam Modelů

Basic scenario - Basic Path

1. Uživateli se zobrazí seznam Modelů.
   Pro každý Model se zobrazí atributy:
   - •Název
   - •Popis
   - •Typ optimalizace Třídění: dle názvu Modelu (bez možnosti změny)
2. Pro každý Model bude možné vyvolat následující akce:
   - •Zobrazit detail modelu -> **UC402**
   - •Zobrazit seznam proměnných modelu -> **UC420**

A.4.2 UC402-Zobraz Model (Zobraz seznam rovnic modelu)

Pre-condition Vybraný Model.

Basic scenario - Basic Path

1. Systém zobrazí seznam rovnic vybraného Modelu. Pro každou rovici zobrazí tyto údaje:
   - •Popis
   - •Rovnici ve formátu $ax + by + cz + p = A1$
2. Uživatel může provést jednu z následujících akcí:
   - •Zobrazit proměnnou modelu -> **UC421**
   - •Vytvořit proměnnou modelu -> **UC422**
   - •Zrušit proměnnou modelu -> **UC423**

A.4.3 UC403-Vytvoř Model

Basic scenario - Basic Path

1. Systém zobrazí formulář pro zadání vlastností Modelu:
   - •Popis
2. Pokud uživatel zvolí akci Vytvořit
   - •Systém vytvoří model na základě zadaných údajů
   - •-> **UC405**

A.4.4 UC404-Zruš Model

Pre-condition Vybraný Model.

Basic scenario - Basic Path

1. Systém zruší všechny proměnné a všechny rovnice ve vybraném Modelu.
2. Systém zruší vybraný Model.
3. -> **UC401**

Pre-condition Vybraný Model je zrušený.

A.4.5 UC420-Zobraz seznam proměnných modelu

Pre-condition Vybraný Model.

Basic scenario - Basic Path

1. Systém zobrazí všechny proměnné vybraného Modelu.

A.4.6 UC421-Zobraz proměnnou modelu

Pre-condition Vybraná proměnná modelu.

Basic scenario - Basic Path

1. Systém zobrazí detaily vybrané proměnné vybraného Modelu.

A.4.7 UC422-Vytvoř proměnnou modelu

Pre-condition Vybraný Model.

Basic scenario - Basic Path

1. Systém zobrazí formulář pro zadání vlastností Proměnné:
   - •Symbol
   - •Název
   - •Popis
2. Pokud uživatel zvolí akci Vytvořit
   - •Systém vytvoří Proměnnou na základě zadaných údajů
   - •-> **UC405**

A.4.8 UC423-Zruš proměnnou modelu

Pre-condition Vybraná proměnná modelu.

Basic scenario - Basic Path

1. Systém zruší vybranou Proměnnou modelu.
2. -> **UC405**

Post-condition Vybraná proměnná modelu je zrušena.

B Sada šablon EA pro generování kódu v jazyku Smalltalk

B.1 File

```
$COMMENT="WARNING: DO NOT MODIFY THIS TEMPLATE BELOW THIS POINT"
%ImportSection%\n
%list="Class" @separator="\n\n"%
```

B.2 Class

```
%ClassDeclaration%
%ClassBody%
```

B.3 Class Base

```
%classBaseName%
```

B.4 Class Body

```
%list="Operation" @separator="\n\n"%
```

B.5 Class Declaration

```
$bases=%list="ClassBase"  @separator=", "%
%if $bases == ""%
$bases="Object"
%endIf%
%PI=" "%
$bases
subclass:
#%className%\n

    instanceVariableNames: '%list="Attribute" @separator=" "%'\n
    classVariableNames: ''\n
    poolDictionaries: ''\n
    category: 'EA-Generated'!\n
```

B.6 Attribute

```
%AttributeDeclaration%
```

B.7 Attribute Declaration

```
%PI=""%
%attName%
```

B.8 Linked Attribute

```
%LinkedAttributeDeclaration%
```

B.9 Linked Attribute Declaration

```
%if linkAttRole != ""%
%linkAttRole%
%else%
%REPLACE(genOptDefaultAssocAttName,"$LinkClass",linkAttName)%
%endIf%
```

B.10 Operation

```
%OperationDeclaration%
%OperationBody%
```

B.11 Operation Declaration

```
%PI=""%
!%className% methodsFor: 'as yet unclassified'
 stamp: '%classAuthor% %eaDateTime%'!\n
%opName%: %list="Parameter" @separator=" "%
```

B.12 Operation Body

```
$wrap = %genOptWrapComment=="-1" ? "-1" : "60"%
$behavior = %WRAP_LINES(opBehavior, $wrap, "", "")%
%if $behavior != ""%
"$behavior"\n
%endIf%
%if opCode != ""%
%WRAP_LINES(opCode, "-1", "\t", "")%
! !
%endTemplate%
```

B.13 Parameter

```
%paramName%
```

C Vygenerované zdrojové kódy FSM v jazyku Smalltalk

C.1 FSM.st

```
Object subclass: #FSM
    instanceVariableNames: 'name initial final current states transitions'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'EA-Generated'!

!FSM methodsFor: 'as yet unclassified' stamp: 'JanTomsa 01-IV-2010 1:38:31'!
processSignal: signal
"Fire first outgoing transition of current state whose input equals signal received.
 Set current to the result and return it."

    ^current := ((current outgoing) select:
        [ :trans |
            trans input = signal
        ]
     ) first fire.

! !
```

C.2 State.st

```
Object subclass: #State
    instanceVariableNames: 'name outgoing incoming'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'EA-Generated'!
```

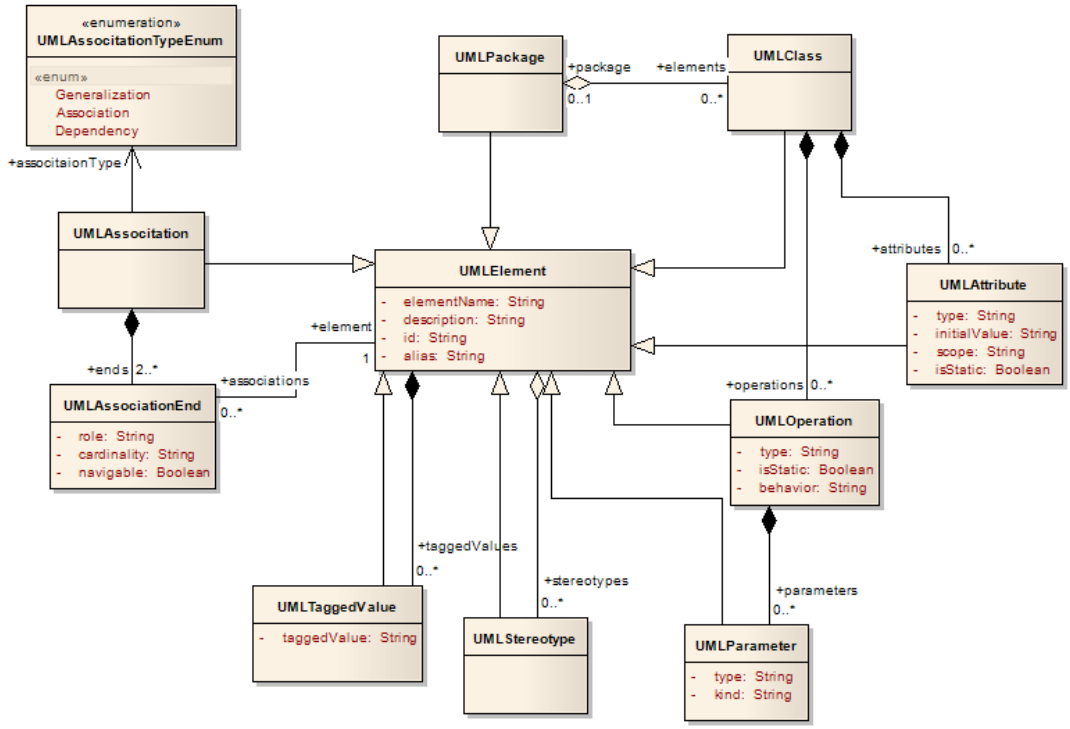
C.3 Transition.st

```
Object subclass: #Transition
    instanceVariableNames: 'input output source target'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'EA-Generated'!

!Transition methodsFor: 'as yet unclassified' stamp: 'JanTomsa 01-IV-2010 1:38:31'!

    Transcript show: output.
    ^target
! !
```

D UML profil systému DecisionMaker

```
<?xml version="1.0" encoding="UTF-8"?>
<UMLProfile>
    <Documentation id="AF7E5119-0" name="UML_Profile_Entities" version="1.0" notes="UML_Profile_Entities"/>
    <Content>    <!-- The profile content -->
        <Stereotypes>    <!-- List of stereotypes used in this profile-->
            <Stereotype name="Class" notes="" cx="" cy="" metafile="">
                <AppliesTo/>
                <TaggedValues>
                    <Tag name="isActive" description="" values="" default=""/>
                </TaggedValues>
                <Constraints/>
            </Stereotype>
            <Stereotype name="Entity" notes="" cx="" cy="" metafile="">
                <AppliesTo/>
                <TaggedValues>
                    <Tag name="label_cz_singular" description="" values="" default=""/>
                    <Tag name="label_cz_plural" description="" values="" default=""/>
                    <Tag name="label_en_singular" description="" values="" default=""/>
                    <Tag name="label_en_plural" description="" values="" default=""/>
                </TaggedValues>
                <Constraints/>
            </Stereotype>
            <Stereotype name="Attribute" notes="" cx="" cy="" metafile="">
                <AppliesTo/>
                <TaggedValues/>
                <Constraints/>
            </Stereotype>
            <Stereotype name="Column" notes="" cx="" cy="" metafile="">
                <AppliesTo/>
                <TaggedValues>
                    <Tag name="label_cz" description="" values="" default=""/>
                    <Tag name="label_en" description="" values="" default=""/>
                    <Tag name="editor" description="" values="" default=""/>
                </TaggedValues>
                <Constraints/>
            </Stereotype>
            <Stereotype name="ColumnEditorEnum" notes="" cx="" cy="" metafile="">
                <AppliesTo/>
                <TaggedValues>
                    <Tag name="textInput" description="" values="" default=""/>
                    <Tag name="textArea" description="" values="" default=""/>
                    <Tag name="select" description="" values="" default=""/>
                    <Tag name="dateTimeSelector" description="" values="" default=""/>
                </TaggedValues>
                <Constraints/>
            </Stereotype>
        </Stereotypes>
        <TaggedValueTypes/>
    </Content>
</UMLProfile>
```

E Podpůrné třídy metamodelu UML



Obrázek 34: Metamodel UML

E.1 Zdrojový kód podpůrných tříd metamodelu UML

```
Object subclass: #UMLAssociationEnd
instanceVariableNames: 'element role cardinality association'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:49'!
asString
^'<<<', (self element elementName ) asString, '>>>'! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 14:21'!
association
^ association! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 14:21'!
association: anObject
association := anObject! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
cardinality
^ cardinality! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
cardinality: anObject
cardinality := anObject! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
element
^ element! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
element: anObject
element := anObject! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:49'!
printOn: aStream
aStream nextPutAll: self asString.! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
role
^ role! !

!UMLAssociationEnd methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
role: anObject
role := anObject! !


Object subclass: #UMLAssociationTypeEnum
instanceVariableNames: 'type'
classVariableNames: ''
```

```
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 17:43'!
= anUMLAssocitaionTypeEnum
^(anUMLAssocitaionTypeEnum respondsTo: #type)
ifTrue: [(self type) = (anUMLAssocitaionTypeEnum type)]
ifFalse: [false].! !

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 13:54'!
isAssociation
^ (type = #Association)! !

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 13:55'!
isDependency
^ (type = #Dependency)! !

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 13:55'!
isGeneralization
^ (type = #Generalization)! !

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 13:56'!
printOn: aStream
aStream nextPutAll: type asString .! !

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 13:50'!
type
^ type! !

!UMLAssociationTypeEnum methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 13:50'!
type: aSymbol
type := aSymbol! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

UMLAssociationTypeEnum class
instanceVariableNames: ''!

!UMLAssociationTypeEnum class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 16:42'!
Aggregation
^self new: #Aggregation! !

!UMLAssociationTypeEnum class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 13:53'!
Association
^self new: #Association! !

!UMLAssociationTypeEnum class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 16:43'!
Composition
^self new: #Composition! !

!UMLAssociationTypeEnum class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 13:53'!
Dependency
```

```
^self new: #Dependency! !

!UMLAssociationTypeEnum class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 13:52'!
Generalization
^self new: #Generalization! !

!UMLAssociationTypeEnum class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 17:26'!
new: aSymbol
^self new
type: aSymbol;
yourself.! !


Object subclass: #UMLClassCompiler
instanceVariableNames: 'umlClass'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 22:44'!
createClassInSmalltalk
| general umlClassName instVars |
general := umlClass superType.
general ifNil: [general := 'Object'].
umlClassName := umlClass elementName.
self assert: [umlClassName notNil].
instVars := (umlClass attributes select:
[ :a | (a isStatic not) & (a elementName notNil) ])
inject: '' into: [ :a :b | a, ' ', b elementName].
Compiler evaluate:
general,' subclass: #',umlClassName,'
instanceVariableNames: ''',instVars,'''
classVariableNames: ''''
poolDictionaries: ''''
category: ''',(umlClassName copyFrom: 1 to: 2),'-generated'''.
! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 10:58'!
createClassMethodsInSmalltalk
| classMethods umlClassName |
umlClassName := umlClass elementName.
self assert: [umlClassName notNil].
classMethods := umlClass operations
select: [ :o | (o isStatic) & (o elementName notNil) ].
classMethods do: [ :o |
self createMethodInSmalltalk: o ofUMLClasNamed: umlClassName,' class'
]
! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 05:24'!
createInSmalltalk
self createClassInSmalltalk.
```

```
self declareClassVarsInSmalltalk.
self createClassMethodsInSmalltalk.
self createInstMethodsInSmalltalk.
"self class compileAll."! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 10:58'!
createInstMethodsInSmalltalk
| instMethods umlClassName |
umlClassName := umlClass elementName.
self assert: [umlClassName notNil].
instMethods := umlClass operations
select: [ :o | (o isStatic not) & (o elementName notNil) ].
instMethods do: [ :o |
self createMethodInSmalltalk: o ofUMLClasNamed: umlClassName
]
! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 21:49'!
createMethodInSmalltalk: anUMLOperation ofUMLClasNamed: anUMLClassName
| signature body params selectorParts operationName bodyWithDoubledApostrophes |
operationName := anUMLOperation elementName.
params := anUMLOperation parameters.
(params size = 0)
ifTrue: [
signature := operationName.
]
ifFalse: [
(params size = 1)
ifTrue: [
signature := operationName,
((operationName last isLetter) ifTrue: [': ']
ifFalse: [' ']),
(params first elementName)
]
ifFalse: [ "methods with more parameters must have the parts of selectors delimited by _"
signature := ''.
selectorParts := anUMLOperation elementName findTokens: '_'.
selectorParts doWithIndex: [ :sel :idx |
signature := signature, sel,': ',((params at: idx) elementName),' '
]
]
].
body := anUMLOperation behavior.
body ifNil: [ body := 'Transcript cr.' ].
bodyWithDoubledApostrophes :=
 (body inject: ''
into: [ :a :b |
a,
((b = $')
ifTrue: [ '''''' ]
ifFalse: [ b asString ])
]).
```

```
"here comes the magic"
Compiler evaluate:
anUMLClassName,' compile: ''',signature,'
',bodyWithDoubledApostrophes,'''
.'.
! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 05:47'!
declareClassVarsInSmalltalk
| classVars umlClassName |
classVars := umlClass attributes select: [ :a | (a isStatic) & (a elementName notNil) ].
umlClassName := umlClass elementName.
self assert: [umlClassName notNil].
classVars do: [ :cv |
Compiler evaluate: (umlClassName,' class addInstVarName: ''',cv elementName, '''' ).
]! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 04:58'!
umlClass
^ umlClass! !

!UMLClassCompiler methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 04:58'!
umlClass: anObject
umlClass := anObject! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

UMLClassCompiler class
instanceVariableNames: ''!

!UMLClassCompiler class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 04:58'!
new: anUMLClass
| me |
me := self new.
me umlClass: anUMLClass.
^me.! !


Object subclass: #UMLElement
instanceVariableNames: 'elementName description taggedValues stereotypes associations id alias'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 14:14'!
addAssociationEnd: anObject
associations add: anObject! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 04:42'!
addStereotype: anObject
stereotypes add: anObject! !
```

```
!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 04:43'!
addTaggedValue: anObject
taggedValues add: anObject! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 22:12'!
addTaggedValues: anArray
anArray do: [ :t | self addTaggedValue: t ]! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 22:22'!
asString
^ self elementName ifNil: [ '(unnamed)' ]! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 18:01'!
associatedElements
^self associatedElementsWithBlock: [ :ass | true ]
and: [ :end | true ].
! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 17:46'!
associatedElementsWithBlock: aBlock
^associations
ifNotNil: [
((((associations select: aBlock)
inject: ( Set new )
into: [ :a :b | a, (b ends) ]
  ) collect: [ :asEnd | asEnd element ]
 ) asSet
) select: [ :el | el ~= self ]
]
ifNil: [ nil ].
! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 17:51'!
associatedElementsWithBlock: aBlock and: anotherBlock
^associations
ifNotNil: [
(((((associations select: aBlock)
inject: ( Set new )
into: [ :a :b | a, (b ends) ]
   ) select: anotherBlock
  ) collect: [ :asEnd | asEnd element ]
 ) asSet
) select: [ :el | el ~= self ]
]
ifNil: [ nil ].
! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 17:49'!
associatedElementsWithType: anAssocitationType
^self associatedElementsWithBlock: [ :a | a associationType = anAssocitationType].! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 16:39'!
```

```
initialize
elementName := 'unnamed element'.
stereotypes := OrderedCollection new.
taggedValues := OrderedCollection new.
associations := OrderedCollection new.! !

!UMLElement methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 15:03'!
printOn: aStream
aStream nextPutAll: self asString.! !


!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 21:58'!
alias
^ alias! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 21:58'!
alias: aString
alias := aString ! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:15'!
associations
^ associations ifNil: [ associations := OrderedCollection new. ]! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 14:12'!
associations: anObject
associations := anObject! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:13'!
description
^ description! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:13'!
description: anObject
description := anObject! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:36'!
elementName
^elementName! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:36'!
elementName: aName
elementName := aName! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 02:50'!
hasStereotypeNamed: aName
^(self stereotypes select: [ :s | s elementName = aName ]) size > 0.
! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:22'!
id
^ id! !
```

```
!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:22'!
id: aString
id := aString ! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 02:47'!
stereotypeNames
^self stereotypes collect: [ :s | s elementName ].
! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:38'!
stereotypes
^ stereotypes! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:38'!
stereotypes: anObject
stereotypes := anObject! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 02:45'!
taggedValueAt: aName
^ self taggedValuesDict at: aName ifAbsent: nil.! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 02:52'!
taggedValueAt: aName ifAbsent: anAlternative
^ self taggedValuesDict at: aName ifAbsent: anAlternative.! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:38'!
taggedValues
^ taggedValues! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:38'!
taggedValues: anObject
taggedValues := anObject! !

!UMLElement methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 02:44'!
taggedValuesDict
| dict |
dict := Dictionary new.
self taggedValues do: [ :t | dict add: (t elementName)->(t taggedValue) ].
^dict.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

UMLElement class
instanceVariableNames: ''!

!UMLElement class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 22:40'!
new: aName
| newInstance |
newInstance := self new.
newInstance elementName: aName.
^newInstance.
! !
```

```
UMLElement subclass: #UMLAssociation
instanceVariableNames: 'associationType ends'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLAssociation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:21'!
addEnd: anAssociationEnd
ends add: anAssociationEnd.
anAssociationEnd association: self.
(anAssociationEnd element associations) add: self.! !

!UMLAssociation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 17:19'!
associationType
^ associationType! !

!UMLAssociation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 17:19'!
associationType: anObject
associationType := anObject! !

!UMLAssociation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
ends
^ ends! !

!UMLAssociation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:08'!
ends: anObject
ends := anObject! !

!UMLAssociation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:35'!
initialize
super initialize.
ends := OrderedCollection new.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

UMLAssociation class
instanceVariableNames: ''!

!UMLAssociation class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 17:20'!
new: anAssociationType
^self new
associationType: anAssociationType;
yourself.! !


UMLElement subclass: #UMLAttribute
instanceVariableNames: 'type initialValue scope isStatic'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!
```

```
!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:50'!
initialValue
^ initialValue! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:50'!
initialValue: anObject
initialValue := anObject! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 02:15'!
initialize
super initialize.
isStatic := false.! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/4/2010 23:23'!
isStatic
^ isStatic ! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/4/2010 23:23'!
isStatic: aBoolean
isStatic := aBoolean ! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:57'!
scope
^ scope! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:57'!
scope: anObject
scope := anObject! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:01'!
type
^ type! !

!UMLAttribute methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:01'!
type: anObject
type := anObject! !


!UMLAttribute methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 03:46'!
asString
^
( isStatic ifNotNil: [ isStatic ifTrue: [ 'static ' ] ifFalse: [''] ]
 ifNil: [''] ),
( scope ifNotNil: [ scope asString, ' ' ] ifNil: [''] ),
super asString,
( type ifNotNil: [ ' : ', type asString ] ifNil: [''] ),
(self stereotypes inject: '' into: [ :s1 :s2 | s1 asString, ' ', s2 asString ])
! !


UMLElement subclass: #UMLClass
```

```
instanceVariableNames: 'attributes operations package forcedGeneral'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:45'!
addAttribute: anObject
attributes add: anObject! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 20:40'!
addAttributes: anArray
anArray do: [ :a | self addAttribute: a ].! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:45'!
addOperation: anObject
operations add: anObject! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 20:41'!
addOperations: anArray
anArray do: [ :o | self addOperation: o ]! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/4/2010 22:47'!
asString
^'Class: ', super asString,
(self stereotypes inject: '' into: [ :s1 :s2 | s1 asString, ' ', s2 asString ]),
 (Character cr asString),
 (attributes
ifNotEmpty: [
'+-attributes: ', (Character cr asString),
(attributes
inject: ''
into: [ :a :b | a asString, '  ' ,b asString, (Character cr asString) ])
]
ifEmpty: ['']
 ),
 (operations
ifNotEmpty: [
  '+-operations: ', (Character cr asString),
(operations
inject: ''
into: [ :a :b | a asString,'  ' ,b asString, (Character cr asString) ])
]
ifEmpty: ['']
 ), (Character cr asString)! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:44'!
attributes
^ attributes! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:44'!
attributes: anObject
attributes := anObject! !
```

```
!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 03:23'!
forcedGeneral
^ forcedGeneral! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 03:23'!
forcedGeneral: anObject
forcedGeneral := anObject! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:36'!
initialize
super initialize.
attributes := OrderedCollection new.
operations := OrderedCollection new.! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 22:47'!
linkedAttributes
^associations
ifNotNil: [
((associations select:
[ :as |
(as associationType = UMLAssociationTypeEnum Association)
or: [as associationType = UMLAssociationTypeEnum Aggregation]
]
)
inject: ( Set new )
into: [ :a :b | a, (b ends select: [ :end | end element ~= self ]) ]
) collect: [ :asEnd |
((UMLAttribute new: (asEnd role))
type:
(asEnd element elementName),
((asEnd cardinality = '0..*')
ifTrue: ['[]']
ifFalse: ['']
)
;
yourself
)
]
]
ifNil: [ nil ].
! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:44'!
operations
^ operations! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:44'!
operations: anObject
operations := anObject! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:46'!
```

```
package
^ package! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 04:46'!
package: anObject
package := anObject! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 15:13'!
removeOperation: anObject
operations remove: anObject! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 17:55'!
subTypes
^ self associatedElementsWithBlock:
[ :as | as associationType = UMLAssociationTypeEnum Generalization]
and:
[ :end | end role = 'subType']! !

!UMLClass methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 03:25'!
superType
| generals |
forcedGeneral
ifNotNil: [ ^forcedGeneral ]
ifNil: [
generals := (self associatedElementsWithBlock:
[ :as | as associationType = UMLAssociationTypeEnum Generalization]
and:
[ :end | end role = 'superType']).
^generals
ifNotEmpty: [ generals first]
  ifEmpty: [ nil ].
].! !


UMLElement subclass: #UMLOperation
instanceVariableNames: 'parameters type isStatic behavior'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:22'!
behavior
^ behavior! !

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:22'!
behavior: anObject
behavior := anObject! !

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:22'!
isStatic
^ isStatic! !
```

```
!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:22'!
isStatic: anObject
isStatic := anObject! !

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:33'!
parameters
^ parameters! !

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:33'!
parameters: anObject
parameters := anObject! !

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:22'!
type
^ type! !

!UMLOperation methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:22'!
type: anObject
type := anObject! !


!UMLOperation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 05:33'!
addParameter: anObject
parameters add: anObject! !

!UMLOperation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 00:41'!
addParameters: anArray
anArray do: [ :p | parameters add: p ]! !

!UMLOperation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 03:44'!
asString
| params |
params := parameters
ifNotEmpty:
[ '(',
(parameters inject: ''
into: [ :a :b |
a asString, (a ifNotEmpty: [','] ifEmpty: ['']), b asString
]) asString,
')'
]
ifEmpty: [''].
^
( isStatic ifNotNil: [ isStatic ifTrue: [ 'static ' ] ifFalse: [''] ]
 ifNil: [''] ),
super asString,
params,
( (type notNil & (type = 'void') not) ifTrue: [ ' : ', type asString ] ifFalse: [''] )! !

!UMLOperation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 02:15'!
initialize
super initialize.
```

```
parameters := OrderedCollection new.
isStatic := false.! !


UMLElement subclass: #UMLPackage
instanceVariableNames: 'elements'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLPackage methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 20:38'!
addClass: aUMLClass
elements add: aUMLClass.
aUMLClass package: self.! !

!UMLPackage methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 20:39'!
addClasses: anArray
anArray do: [ :c | self addClass: c ].! !

!UMLPackage methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 16:36'!
initialize
super initialize.
elements := OrderedCollection new.! !

!UMLPackage methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 04:49'!
removeClass: aUMLClass
classes remove: aUMLClass.
aUMLClass package: nil.! !


UMLElement subclass: #UMLParameter
instanceVariableNames: 'type kind'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLParameter methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 03:41'!
asString
^
super asString,
( kind ifNotNil: [ kind = 'out' ifTrue: ['*'] ifFalse: [''] ] ifNil: [''] ),
( type ifNotNil: [ ' : ', type asString ] ifNil: [''] )
! !

!UMLParameter methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:31'!
kind
^ kind! !

!UMLParameter methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:31'!
kind: anObject
kind := anObject! !
```

```
!UMLParameter methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:31'!
type
^ type! !

!UMLParameter methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 05:31'!
type: anObject
type := anObject! !


UMLElement subclass: #UMLStereotype
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLStereotype methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 22:46'!
asString
^ '<<', elementName, '>>'! !


UMLElement subclass: #UMLTaggedValue
instanceVariableNames: 'taggedValue'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!UMLTaggedValue methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:37'!
taggedValue
^ taggedValue! !

!UMLTaggedValue methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 16:37'!
taggedValue: anObject
taggedValue := anObject! !


Object subclass: #UMLWorld
instanceVariableNames: 'elements'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

UMLWorld class
instanceVariableNames: ''!

!UMLWorld class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 18:25'!
findAllByName: aName
"finds all elements of given name"
^UMLElement allSubInstances select: [ :i | (i elementName) = aName ]! !

!UMLWorld class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 04:33'!
```

```
findByName: aName
"finds first element of given name"
^(self findAllByName: aName) first! !


Object subclass: #XMIUMLBuilder
instanceVariableNames: 'xmi umlModel xmiExtension packages classes stereotypes'
classVariableNames: ''
poolDictionaries: ''
category: 'XMI-2-UML'!

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/4/2010 22:20'!
buildAssociation: anAssociationDefinition
| umlAssociation sourceEnd targetEnd extAttElement |
Transcript
show: 'Association: ';
show: (anAssociationDefinition attributeAt: 'name');
cr.
umlAssociation := UMLAssociation new:
(anAssociationDefinition attributeAt: 'name' ifAbsent: '(unnamed association)').
umlAssociation id: (anAssociationDefinition attributes at: 'xmi:id').
"============= build association ends =========="
sourceEnd := UMLAssociationEnd new.
targetEnd := UMLAssociationEnd new.
extAttElement := xmiExtension firstTagNamed: 'connector'
with: [ :tag | (tag attributeAt: #'xmi:idref') = umlAssociation id ].
"TODO"! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/4/2010 22:43'!
buildAttribute: anAttribute
| umlAttribute extAttElement defValTag |
Transcript
show: 'Attribute: ';
show: (anAttribute attributeAt: 'name');
cr.
umlAttribute := UMLAttribute new: (anAttribute attributeAt: 'name').
umlAttribute id: (anAttribute attributes at: 'xmi:id').
defValTag := (anAttribute firstTagNamed: 'defaultValue').
defValTag ifNotNil: [
umlAttribute initialValue: (defValTag attributeAt: #value).
].
"--- hunt for type and other props ---"
extAttElement := xmiExtension firstTagNamed: 'attribute'
with: [ :tag | (tag attributeAt: #'xmi:idref') = umlAttribute id ].
extAttElement ifNotNil: [
umlAttribute scope: (extAttElement attributeAt: #scope).
extAttElement elements do: [ :el |
(el name = #documentation) ifTrue: [ umlAttribute description: (el attributeAt: #value) ].
(el name = #properties) ifTrue: [ umlAttribute type: (el attributeAt: #type)].
(el name = #stereotype) ifTrue: [
umlAttribute addStereotype: (self buildStereotype: (el attributeAt: #stereotype))
].
```

```
(el name = #style) ifTrue: [ umlAttribute alias: (el attributeAt: #value) ].
(el name = #tags) ifTrue: [
umlAttribute addTaggedValues:
(el elements collect: [ :elel | self buildTaggedValue: elel ])
].
].
].
^umlAttribute.! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/4/2010 22:43'!
buildClass: aClassDefinition
| umlClass extClaElement |
Transcript
show: 'Class: ';
show: (aClassDefinition  attributes at: 'name');
cr.
umlClass := UMLClass new: (aClassDefinition  attributes at: 'name').
umlClass id: (aClassDefinition attributes at: 'xmi:id').
classes add: umlClass.
"=============== build attributes ============"
umlClass addAttributes:
((aClassDefinition
elements select: [ :el | el name = #ownedAttribute ])
collect: [ :el | self buildAttribute: el ]).
"============ build operations ==========="
umlClass addOperations:
((aClassDefinition
elements select: [ :el | el name = #ownedOperation ])
collect: [ :el | self buildOperation: el ]).
"--- hunt for type and other props ---"
extClaElement := xmiExtension firstTagNamed: 'element'
with: [ :tag | (tag attributeAt: #'xmi:idref') = umlClass id ].
extClaElement ifNotNil: [
extClaElement elements do: [ :el |
(el name = #properties) ifTrue: [
umlClass addStereotype: (self buildStereotype: (el attributeAt: #stereotype))
].
(el name = #style) ifTrue: [ umlClass alias: (el attributeAt: #value) ].
(el name = #tags) ifTrue: [
umlClass addTaggedValues:
(el elements collect: [ :elel | self buildTaggedValue: elel ])
].
].
].
^umlClass.! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/3/2010 20:58'!
buildOperation: anOperation
| umlOperation |
Transcript
show: 'Operation: ';
show: (anOperation attributes at: 'name');
```

```
cr.
umlOperation := UMLOperation new: (anOperation attributes at: 'name').
umlOperation id: (anOperation attributes at: 'xmi:id').
^umlOperation ! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/3/2010 20:46'!
buildPackage: aPackageDefinition
| umlPackage |
Transcript
show: 'Package: ';
show: (aPackageDefinition  attributes at: 'name');
cr.
umlPackage := UMLPackage new: (aPackageDefinition  attributes at: 'name').
umlPackage id: (aPackageDefinition  attributes at: 'xmi:id').
packages add: umlPackage.
"=============== build classes ============"
umlPackage addClasses:
((aPackageDefinition
elements select: [ :el | (el attributes at: 'xmi:type') = 'uml:Class' ])
collect: [ :el | self buildClass: el ]).
"============ build associations =========="
(aPackageDefinition
elements select: [ :el | (el attributes at: 'xmi:type') = 'uml:Association' ])
do: [ :el | self buildAssociation: el ].! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/3/2010 22:07'!
buildRootPackages
| rootUMLPackages |
packages := OrderedCollection new.
classes := OrderedCollection new.
stereotypes := OrderedCollection new.
rootUMLPackages := (umlModel elements select: [ :el | el name = #packagedElement ]).
rootUMLPackages do: [ :pkg | self buildPackage: pkg ].! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/3/2010 22:07'!
buildStereotype: aName
| existingStereotypes newStereotype |
existingStereotypes := (stereotypes select: [ :s | s elementName = aName ]).
existingStereotypes size = 0
ifTrue: [
newStereotype := UMLStereotype new: aName.
stereotypes add: newStereotype.
^newStereotype.
]
ifFalse: [^existingStereotypes first].! !

!XMIUMLBuilder methodsFor: 'building' stamp: 'JanTomsa 4/3/2010 22:50'!
buildTaggedValue: aTVDefinition
| umlTaggedValue |
Transcript
show: 'Tagged value: ';
show: (aTVDefinition attributeAt: #name);
```

```
cr.
umlTaggedValue := UMLTaggedValue new:
(aTVDefinition attributeAt: #name).
umlTaggedValue id: (aTVDefinition attributeAt: #'xmi:id');
taggedValue: (aTVDefinition attributeAt: #value).
^umlTaggedValue.! !


!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
classes
^ classes! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
classes: anObject
classes := anObject! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
packages
^ packages! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
packages: anObject
packages := anObject! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 22:02'!
stereotypes
^stereotypes! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 22:01'!
stereotypes: anObject
stereotypes := anObject! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
umlModel
^ umlModel! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
umlModel: anObject
umlModel := anObject! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
xmi
^ xmi! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
xmi: anObject
xmi := anObject! !

!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
xmiExtension
^ xmiExtension! !
```

```
!XMIUMLBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/3/2010 18:32'!
xmiExtension: anObject
xmiExtension := anObject! !


!XMIUMLBuilder methodsFor: 'initialisation' stamp: 'JanTomsa 4/3/2010 18:32'!
initFromFileNamed: fileName
xmi := XMLDOMParser parseDocumentFrom: (FileStream oldFileNamed: fileName ).

umlModel := (((xmi elements first) elements)
select: [ :el | el name = #'uml:Model' ] ) first.

xmiExtension := (((xmi elements first) elements)
select: [ :el | el name = #'xmi:Extension' ] ) first.
! !
```

F Generátor entit aplikace DecisionMaker

```
Object subclass: #DMEntityBuilder
    instanceVariableNames: 'umlBuilder'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'DecisionMaker-EntityBuilder'!

!DMEntityBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/5/2010 17:41'!
entityClasses
    ^ umlBuilder classes
            select: [ :c | c hasStereotypeNamed: 'entity' ].! !

!DMEntityBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/4/2010 22:05'!
umlBuilder
    umlBuilder ifNil: [ umlBuilder := XMIUMLBuilder new ].
    ^ umlBuilder! !

!DMEntityBuilder methodsFor: 'accessing' stamp: 'JanTomsa 4/4/2010 22:03'!
umlBuilder: anObject
    umlBuilder := anObject! !


!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/4/2010 23:43'!
CRUDClassName: anUMLClass withPrefix: aPrefix
    ^(aPrefix, anUMLClass elementName), 'CRUD'.! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/6/2010 00:12'!
buildCRUDClass: anUMLClass withPrefix: aPrefix
    | CRUDClass CR modelClassName editorClassName textsClassName |
    CR := Character cr asString.
    modelClassName := (self modelClassName: anUMLClass withPrefix: aPrefix ).
    editorClassName := (self editorClassName: anUMLClass withPrefix: aPrefix ).
    textsClassName  := (self textsClassName: anUMLClass withPrefix: aPrefix ).
    CRUDClass := UMLClass new: (self CRUDClassName: anUMLClass withPrefix: aPrefix).
    CRUDClass forcedGeneral: 'WAComponent'.
    "==================== Attributes ================="
    CRUDClass addAttribute: ( (UMLAttribute new: 'listComponent')
                                        type: 'WATableReport';
                                        yourself
    ).
    "==================== Operations ================="
    CRUDClass addOperation: ( (UMLOperation new: 'children')
                                type: 'Array';
                                behavior: '^Array with: listComponent';
                                yourself
    ).
    "--------------------- CRUD ---------------------"
    CRUDClass addOperation: ( (UMLOperation new: 'add')
            type: 'void';
            behavior: '| editor instance answer |', CR,
                        'instance := ', modelClassName, ' new.',CR,
```

```
                    'editor := ', editorClassName, ' on: instance.',CR,
                    'answer := self call: editor.',CR,
                    'answer ifTrue: [ ', modelClassName, ' instances add: instance ]'
        yourself
).
CRUDClass addOperation: ( (UMLOperation new: 'edit')
        type: 'void';
        addParameter: ( (UMLParameter new: 'anInstance') type: modelClassName );
        behavior: '| editor |', CR,
                    'editor := ', editorClassName, ' on: anInstance.',CR,
                    'self call: editor.';
        yourself
).
CRUDClass addOperation: ( (UMLOperation new: 'delete')
        type: 'void';
        addParameter: ( (UMLParameter new: 'anInstance') type: modelClassName );
        behavior: modelClassName, ' instances remove: anInstance.';
        yourself
).
"--------------------- columns ---------------------"
((anUMLClass attributes )
    select: [ :a | (a stereotypes select: [ :as | as elementName = 'column' ] ) size = 1 ])
    do: [ :a |
        CRUDClass addOperation: ( (UMLOperation new: (a elementName, 'ReportColumn'))
                                        type: 'void';
                                        behavior: '^WAReportColumn new',CR,
            '        title: (',textsClassName,' labelFor: #',a elementName,');',CR,
            '        selector: #',a elementName,';',CR,
            '        clickBlock: [ :each | self edit: each ];',CR,
            '        yourself .';
            yourself )
    ].
"------------------ action column ------------------"
CRUDClass addOperation: ( (UMLOperation new: ('actionReportColumn'))
    type: 'void';
    behavior: '^WAReportColumn new',CR,
        '    title: DMTexts ACTIONS;',CR,
        '    valueBlock: [ :anInstance :html |',CR,
        '        html anchor ',CR,
        '        onClick: (''return confirm('''''', DMTexts CONFIRMQUESTION, '''''')'');',CR,
        '        callback: [ self delete: anInstance  ];',CR,
        '        with: DMTexts DELETE',CR,
        '    ];',CR,
        'yourself .';
    yourself ).
"----------------- initialization ------------------"
CRUDClass addOperation: ( (UMLOperation new: 'initialize')
        type: 'void';
        behavior: '| columns |',CR,
                    'super initialize.',CR,
                    'columns := OrderedCollection new.',CR,
                    (((CRUDClass operations collect: [ :o | o elementName ])
```

```
                        select: [ :o | o endsWith: 'ReportColumn' ])
                        inject: '' into: [ :x :y | x,' columns add: self ',y,'.',CR]
                    ),
                    'listComponent := WATableReport new',CR,
                    '    columns: columns;',CR,
                    '    rowPeriod: 1;',CR,
                    '    yourself.';
            yourself
    ).
    "-------------------- rendering --------------------"
    CRUDClass addOperation: ( (UMLOperation new: 'renderContentOn')
            type: 'void';
            addParameter: ( (UMLParameter new: 'canvas') type: 'WARenderer' );
            behavior: 'canvas heading with: (', textsClassName, ' labelFor: #Instances); level: 3.',CR,
                        'listComponent rows: ', modelClassName, ' instances asSortedCollection.',CR,
                        'canvas render: listComponent.',CR,
                        'canvas anchor callback: [ self add ]; with: DMTexts NEW.';
            yourself
    ).
    ^CRUDClass! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/5/2010 18:22'!
buildEditorClass: anUMLClass withPrefix: aPrefix
    | editorClass CR modelClassName editorClassName textsClassName editorType |
    CR := Character cr asString.
    modelClassName := (self modelClassName: anUMLClass withPrefix: aPrefix ).
    editorClassName := (self editorClassName: anUMLClass withPrefix: aPrefix ).
    textsClassName  := (self textsClassName: anUMLClass withPrefix: aPrefix ).
    editorClass := UMLClass new: editorClassName.
    editorClass forcedGeneral: 'WAComponent'.
    "==================== Attributes ================="
    editorClass addAttribute: ( (UMLAttribute new: 'model')
                                    type: modelClassName;
                                    yourself
    ).
    "==================== Operations ================="
    "--------------------- columns ---------------------"
    ((anUMLClass attributes )
        select: [ :a | (a stereotypes select: [ :as | as elementName = 'column' ] ) size = 1 ])
        do: [ :a |
            editorType := a taggedValueAt: 'editor' ifAbsent: 'textInput'.
            editorClass addOperation: ( (UMLOperation new: 'render',(a elementName) capitalized,'On')
                                                    type: 'void';
                addParameter: ( (UMLParameter new: 'canvas') type: 'WARenderer'     );
                behavior: '| tagID |',CR,
                            'canvas div: [',CR,
                            '    canvas label',CR,
                            '        for: (tagID := canvas nextId);',CR,
                            '        with: (',textsClassName,' labelFor: #',a elementName,').',CR,
                            (self editorSourceCode: (a elementName) editorType: editorType),
                            '].';
                yourself )
```

```
    ].
"----------------- initialization ------------------"
editorClass addOperation: ( (UMLOperation new: 'initialize')
        type: 'void';
        addParameter: ( (UMLParameter new: 'anInstance') type: modelClassName );
        behavior: 'self initialize.',CR,
                    'model := anInstance.';
        yourself
).
"-------------------- rendering --------------------"
editorClass addOperation: ( (UMLOperation new: 'renderButtonsOn')
        type: 'void';
        addParameter: ( (UMLParameter new: 'canvas') type: 'WARenderer' );
        behavior: 'canvas div: [',CR,
                    '    canvas span: [',CR,
                    '        canvas cancelButton callback: [self answer: false];',CR,
                           ' with: DMTexts CANCEL.',CR,
                    '    ].',CR,
                    '    canvas span: [',CR,
                    '        canvas submitButton callback: [self answer: true];',CR,
                           ' with: DMTexts SAVE.',CR,
                    '    ].',CR,
                    '].';
        yourself
).
editorClass addOperation: ( (UMLOperation new: 'renderContentOn')
        type: 'void';
        addParameter: ( (UMLParameter new: 'canvas') type: 'WARenderer' );
        behavior: 'canvas form',CR,
                    '    class: ''dmEditor'';',CR,
                    '    with: [',CR,
                    '        self',CR,
                    (((editorClass operations collect: [ :o | o elementName ])
                        select: [ :o | o beginsWith: 'render' ])
                        inject: '' into: [ :x :y | x,'            ',y,': canvas;',CR]
                    ),
                    '            yourself.',CR,
                    '].';
        yourself
).
"-------------------- constructor --------------------"
editorClass addOperation: ( (UMLOperation new: 'on')
        type: modelClassName;
        isStatic: true;
        addParameter: ( (UMLParameter new: 'anInstance') type: modelClassName );
        behavior: '^self basicNew',CR,
                    '    initialize: anInstance;',CR,
                    '    yourself.';
        yourself
).
^editorClass! !
```

```
!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/5/2010 22:02'!
buildModelClass: anUMLClass withPrefix: aPrefix
    | modelClass |
    modelClass := UMLClass new: (self modelClassName: anUMLClass withPrefix: aPrefix).
    "==================== Attributes ================="
    anUMLClass attributes do: [ :att |
        modelClass addAttribute: ( (UMLAttribute new: (att elementName))
                                        description: att description;
                                        id: att id;
                                        alias: att alias;
                                        initialValue: att initialValue;
                                        scope: att scope;
                                        type: att type;
                                        yourself
                                    ).
    ].
    "-------------------- accessors -------------------"
    anUMLClass attributes do: [ :att |
        att elementName ifNotNil: [
        "---- getter ----"
            modelClass addOperation: ( (UMLOperation new: (att elementName))
                                        type: att type;
                                        behavior: '^',att elementName ;
                                        yourself ).
            "---- setter ----"
            modelClass addOperation: ( (UMLOperation new: (att elementName))
                                        type: 'void';
                                        addParameter: ((UMLParameter new: 'anObject')
                                                                type: att type);
                                        behavior: att elementName,' := anObject';
                                        yourself )
        ]
    ].
    "==================== Operations ================="
    anUMLClass operations do: [ :op |
        modelClass addOperation: ( (UMLOperation new: (op elementName))
                                        description: op description;
                                        id: op id;
                                        alias: op alias;
                                        type: op type;
                                        isStatic: op isStatic;
                                        behavior: op behavior;
                                        addParameters: (
                                                op parameters collect: [ :p |
                                                    (UMLParameter new: p elementName)
                                                        type: p type;
                                                        kind: p kind;
                                                        yourself
                                                ]
                                        );
                                        yourself )
    ].
```

```
    modelClass addAttribute: ( (UMLAttribute new: 'instances')
                                        isStatic: true;
                                        type: anUMLClass elementName,'[]';
                                        scope: 'Private';
                                        yourself ).
    modelClass addOperation: ( (UMLOperation new: 'instances')
                                        isStatic: true;
                                        type: anUMLClass elementName,'[]';
                                        behavior: 'instances ifNil: [ instances := IdentitySet new ].
                                        ^instances');
                                        yourself.
    "-------------------- comparator -------------------"
    modelClass addOperation: ( (UMLOperation new: '<=')
                                        type: 'Boolean';
                                        addParameter: ((UMLParameter new: 'anObject')
                                                                type: modelClass elementName);
                                        behavior: '^self ',(anUMLClass attributes first elementName),
                                                        '<= anObject ',
                                                        (anUMLClass attributes first elementName));
                                        yourself.
    ^modelClass! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/5/2010 18:53'!
buildTextsClass: anUMLClass withPrefix: aPrefix
    | textsClass CR modelClassName editorClassName textsClassName
        instanceCZ instanceEN instancesCZ instancesEN |
    CR := Character cr asString.
    modelClassName := (self modelClassName: anUMLClass withPrefix: aPrefix ).
    editorClassName := (self editorClassName: anUMLClass withPrefix: aPrefix ).
    textsClassName  := (self textsClassName: anUMLClass withPrefix: aPrefix ).
    textsClass := UMLClass new: textsClassName.
    "==================== Attributes ================="
    textsClass addAttribute: ( (UMLAttribute new: 'labels')
                                        type: 'Dictionary';
                                        isStatic: true;
                                        yourself
    ).
    "==================== Operations ================="
    textsClass addOperation: ( (UMLOperation new: 'labelFor')
            type: 'String';
            isStatic: true;
            addParameter: ( (UMLParameter new: 'aSymbol') type: 'Symbol' );
            behavior: '^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).';
            yourself
    ).

    textsClass addOperation: ( (UMLOperation new: 'labelFor_inLanguage')
            type: 'String';
            isStatic: true;
            addParameter: ( (UMLParameter new: 'aSymbol') type: 'Symbol' );
            addParameter: ( (UMLParameter new: 'language') type: 'Symbol' );
            behavior: '^(self labels at: language) at: aSymbol.';
```

```
            yourself
    ).

    instanceCZ := anUMLClass taggedValueAt: 'label_cz_singular'
                    ifAbsent: (anUMLClass class name asString).
    instancesCZ := anUMLClass taggedValueAt: 'label_cz_plural'
                    ifAbsent: (anUMLClass class name asString),'s'.
    instanceEN := anUMLClass taggedValueAt: 'label_en_singular'
                    ifAbsent: (anUMLClass class name asString).
    instancesEN := anUMLClass taggedValueAt: 'label_en_plural'
                    ifAbsent: (anUMLClass class name asString),'s'.
    textsClass addOperation: ( (UMLOperation new: 'labels')
            type: 'Dictionary';
            isStatic: true;
            behavior: '| labelsCzech labelsEnglish |',CR,
                        'labels ifNil: [',CR,
                        '    labelsCzech := Dictionary new.',CR,
                        '    labelsEnglish := Dictionary new.',CR,
                        '    labelsCzech add: #Instance->''',instanceCZ,'''.',CR,
                        '    labelsCzech add: #Instances->''',instancesCZ,'''.',CR,
                        '    labelsEnglish add: #Instance->''',instanceEN,'''.',CR,
                        '    labelsEnglish add: #Instances->''',instancesEN,'''.',CR,
                        (((anUMLClass attributes )
                            select: [ :a | a hasStereotypeNamed: 'column' ])
                                inject: '' into: [ :a :b |
                                    a,
                                    'labelsCzech add: #',b elementName,'->''',
                                        (b taggedValueAt: 'label_cz' ifAbsent: b elementName),'''.',CR,
                                    'labelsEnglish add: #',b elementName,'->''',
                                        (b taggedValueAt: 'label_en' ifAbsent: b elementName),'''.',CR
                                ]
                        ),
                        '    labels := Dictionary new.',CR,
                        '    labels add: #czech->labelsCzech.',CR,
                        '    labels add: #english->labelsEnglish.',CR,
                        '].',CR,
                        '^labels';
            yourself
    ).
    ^textsClass! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/4/2010 23:42'!
editorClassName: anUMLClass withPrefix: aPrefix
    ^(aPrefix, anUMLClass elementName), 'Editor'.
    ! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/5/2010 21:57'!
editorSourceCode: elementName editorType: editorType
    | CR sourceCode |
    CR := Character cr asString.
    (editorType ='textArea')
    ifTrue: [
```

```
        sourceCode := '    canvas textArea',CR,
            '        id: tagID;',CR,
            '        value: model ',elementName,';',CR,
            '        callback: [ :value | model ',elementName,': value ]',CR.
    ]
    ifFalse: [
        (editorType ='select')
        ifTrue: [
            sourceCode := '    canvas select',CR,
                '        id: tagID;',CR,
                '        selected: model ',elementName,';',CR,
                '        list: model ',elementName,'List;',CR,
                '        callback: [ :value | model ',elementName,': value ].',CR.
        ]
        ifFalse: [
            (editorType ='dateTimeSelector')
            ifTrue: [
                sourceCode := '    canvas span',CR,
                    '        id: tagID;',CR,
                    '        with: [ canvas render: dateTimeSelector ].',CR.
            ]
            ifFalse: [
                sourceCode := '    canvas textInput',CR,
                    '        value: model ',elementName,';',CR,
                    '        callback: [:value | model ',elementName,': value]',CR.
            ]
        ]
    ].
    ^sourceCode.
! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/4/2010 23:40'!
modelClassName: anUMLClass withPrefix: aPrefix
    ^(aPrefix, anUMLClass elementName).! !

!DMEntityBuilder methodsFor: 'builders' stamp: 'JanTomsa 4/4/2010 23:43'!
textsClassName: anUMLClass withPrefix: aPrefix
    ^(aPrefix, anUMLClass elementName), 'Texts'.
    ! !


!DMEntityBuilder methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 22:08'!
initFromFileNamed: aFileName
    self umlBuilder initFromFileNamed: aFileName.
    self umlBuilder buildRootPackages.! !

!DMEntityBuilder methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 23:01'!
transformClass: anUMLClass withPrefix: aPrefix
    | xformedClasses |
    xformedClasses := Dictionary new.
    xformedClasses add: #model->(self buildModelClass: anUMLClass withPrefix: aPrefix).
    xformedClasses add: #CRUD->(self buildCRUDClass: anUMLClass withPrefix: aPrefix).
```

```
    xformedClasses add: #editor->(self buildEditorClass: anUMLClass withPrefix: aPrefix).
    xformedClasses add: #texts->(self buildTextsClass: anUMLClass withPrefix: aPrefix).
    ^xformedClasses.! !

!DMEntityBuilder methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 17:47'!
transformEntityClassesWithPrefix: aPrefix
    | xformedEntities |
    xformedEntities := Dictionary new.
    self entityClasses do:
        [ :ec |
            xformedEntities add: (ec elementName)->(self transformClass: ec withPrefix: aPrefix).
        ].
    ^xformedEntities.! !

!DMEntityBuilder methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 17:53'!
transformEntityClassesWithPrefixPlain: aPrefix
    | xformedEntities plainList |
    xformedEntities := self transformEntityClassesWithPrefix: aPrefix.
    plainList := OrderedCollection new.
    xformedEntities do: [ :xe |
        xe do: [ :xed |
            plainList add: xed
        ]
    ].
    ^plainList.! !
```

G Zdrojový kód aplikace DecisionMaker

G.1 Iterace 1 - psáno ručně

```
WAFileLibrary subclass: #DMFileLibrary
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

!DMFileLibrary methodsFor: 'uploaded' stamp: 'JanTomsa 4/3/2010 03:32'!
dmCss
^'
.dmEditor { display: table; }
.dmEditor > div { display: table-row; }
.dmEditor > div > * { display: table-cell; }
.dmEditor textArea { height: 4em; width: 30em; }
.dmEditor div.hidden { display: none; }

body {
font-family: Georgia, "Times New Roman", Times, serif;
font-size: small;
}

#allcontent {
width: 770px;
padding-top: 5px;
padding-bottom: 5px;
background-color: #f4e8d7;
margin-left: auto;
margin-right: auto;
}

*.section {
background-color: #f1dfc7;
}

#header {
margin: 10px;
}

#main {
padding: 15px;
margin: 0px 10px 10px 10px;
width: 550px;
float: right;
}

#sidebar {
padding: 15px;
margin: 0px 600px 10px 10px;
```

```
}

#footer {
color: #704620;
text-align: center;
padding: 15px;
margin: 10px;
clear: right;
}'.! !

!DMFileLibrary methodsFor: 'uploaded'!
dmJpg
^ #(255 216 255 224 0 16 74 70 73 70 0 1 1 1 0 --- zkráceno ---) asByteArray! !


WAComponent subclass: #DMHome
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

!DMHome methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 03:35'!
renderContentOn: canvas

canvas image
altText: 'Kombajn GLEANER L2';
url: DMFileLibrary / 'dm.jpg';
yourself. ! !


WAComponent subclass: #DMMain
instanceVariableNames: 'mainArea'
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 00:02'!
children

^Array with: mainArea.! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/28/2010 00:59'!
initialize

super initialize.
mainArea := DMHome new.! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 00:01'!
renderContentOn: canvas
canvas div
id: 'allcontent';
with: [
```

```
self renderHeaderOn: canvas;
renderMainOn: canvas;
renderSidebarOn: canvas;
renderFooterOn: canvas;
yourself.
].
! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/27/2010 23:58'!
renderFooterOn: canvas
canvas div
id: 'footer';
class: 'section';
with: [
canvas text: 'Copyright (c) ' , Date today year printString.
];
yourself.
! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/27/2010 23:55'!
renderHeaderOn: canvas

canvas div
id: 'header';
class: 'section';
with: [
canvas heading
level1;
with: 'Decision Maker'; "Title on the front page"
yourself.
];
yourself.! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/28/2010 01:01'!
renderMainOn: canvas

canvas div
id: 'main';
class: 'section';
with: [canvas render: mainArea ].! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 01:00'!
renderSidebarOn: canvas
canvas div
id: 'sidebar';
class: 'section';
with: [
canvas heading
level2;
with: DMTexts MENU.
canvas anchor
callback: [ mainArea := DMHome new ];
```

```
with: DMTexts HOME.
canvas break .
canvas anchor
callback: [ mainArea := DMModelCRUD new ];
with: (DMModelTexts labelFor: #Models).
"========= call GENERATED classes ========"
canvas break .
canvas anchor
callback: [ mainArea := DXEquationCRUD new ];
with: (DXEquationTexts labelFor: #Instances).
canvas break .
canvas anchor
callback: [ mainArea := DXVariableCRUD new ];
with: (DXVariableTexts labelFor: #Instances).
canvas break .
canvas anchor
callback: [ mainArea := DXParamGroupCRUD new ];
with: (DXParamGroupTexts labelFor: #Instances).
canvas break .
canvas anchor
callback: [ mainArea := DXParameterCRUD new ];
with: (DXParameterTexts labelFor: #Instances).
"======== /call GENERATED classes ========"
];
yourself.! !

!DMMain methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 00:02'!
updateRoot: anHtmlRoot
super updateRoot: anHtmlRoot.
anHtmlRoot title: 'Decision Maker'. "Window title"
anHtmlRoot link
type: 'text/css';
beStylesheet ;
addAll;
url: DMFileLibrary / 'dm.css';
yourself .! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DMMain class
instanceVariableNames: ''!

!DMMain class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/3/2010 03:34'!
description
^'Podpora rozhodovani na zaklade matematickych modelu.'.! !

!DMMain class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/27/2010 23:33'!
initialize
"
DMMain initialize.
"
super initialize.
```

```
(self registerAsApplication: 'DecisionMaker')
preferenceAt: #sessionClass
put: WASession;
yourself.! !


Object subclass: #DMModel
instanceVariableNames: 'modelName description optimisationType objective'
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
<= anObject
^self modelName<= anObject modelName! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
buildStandardLPProblem
Transcript cr.! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
description
^description! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
description: anObject
description := anObject! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:39'!
initialize

super initialize.
modelName := 'Model #', (self class models size + 1) asString.
description := 'Popis modelu.'. "TODO: Maybe add short help here?"
optimisationType := self optimisationTypeList first.! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
modelName
^modelName! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
modelName: anObject
modelName := anObject! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
objective
^objective! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
objective: anObject
objective := anObject! !
```

```
!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
optimisationType
^optimisationType! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
optimisationType: anObject
optimisationType := anObject! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/28/2010 03:47'!
optimisationTypeList

^#( 'Maximize' 'Minimize' ).! !

!DMModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
solution
Transcript cr.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DMModel class
instanceVariableNames: 'models instances'!

!DMModel class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 00:19'!
createModels
"
DMModel createModels
"

models := nil.
self models
add: (self new
modelName: 'Co máme pestovat?';
description: 'Máme pestovat brambory nebo obilí?';
optimisationType: 'Maximize';
yourself);
add: (self new
modelName: 'Model #2';
description: 'Description of model #2';
optimisationType: 'Maximize' first;
yourself);
add: (self new
modelName: 'Model #3';
Description: 'Description of model #3';
optimisationType: 'Maximize' first;
yourself);
yourself.! !

!DMModel class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
instances
instances ifNil: [ instances := IdentitySet new ].
^instances! !
```

```
!DMModel class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/28/2010 00:09'!
models

models ifNil: [ models := IdentitySet new ].
^models .! !


WAComponent subclass: #DMModelCRUD
instanceVariableNames: 'listComponent'
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

!DMModelCRUD methodsFor: 'columns' stamp: 'JanTomsa 4/2/2010 22:43'!
actionReportColumn

^WAReportColumn new
title: DMTexts ACTIONS;
valueBlock: [ :aModel :html |
html anchor
onClick: ('return confirm(''', DMTexts CONFIRMQUESTION, ''')');
callback: [ self delete: aModel  ];
with: DMTexts DELETE
];
yourself .! !

!DMModelCRUD methodsFor: 'columns' stamp: 'JanTomsa 4/2/2010 22:37'!
descriptionReportColumn

^WAReportColumn new
title: (DMModelTexts labelFor: #description);
selector: #description;
clickBlock: nil;
yourself .! !

!DMModelCRUD methodsFor: 'columns' stamp: 'JanTomsa 4/2/2010 22:36'!
modelNameReportColumn

^WAReportColumn new
title: (DMModelTexts labelFor: #modelName);
selector: #modelName;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DMModelCRUD methodsFor: 'columns' stamp: 'JanTomsa 4/2/2010 22:36'!
optimisationTypeReportColumn

^WAReportColumn new
title: (DMModelTexts labelFor: #optimisationType);
selector: #optimisationType;
clickBlock: nil;
yourself .! !
```

```
!DMModelCRUD methodsFor: 'data management' stamp: 'JanTomsa 4/5/2010 23:57'!
add

| editor instance answer |
instance := DMModel new.
editor := DMModelEditor on: instance.
answer := self call: editor.
answer
ifTrue: [
DMModel models add: instance.
].! !

!DMModelCRUD methodsFor: 'data management' stamp: 'JanTomsa 3/28/2010 00:48'!
delete: aModel

DMModel models remove: aModel .! !

!DMModelCRUD methodsFor: 'data management' stamp: 'JanTomsa 3/28/2010 03:31'!
edit: aModel

| editor |
editor := DMModelEditor on: aModel .
self call: editor.! !


!DMModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/28/2010 00:45'!
children

^Array with: listComponent .! !

!DMModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 3/28/2010 00:44'!
initialize

| columns |
super initialize .
columns := OrderedCollection new
add: self modelNameReportColumn;
add: self descriptionReportColumn;
add: self optimisationTypeReportColumn;
add: self actionReportColumn;
yourself .
listComponent := WATableReport new
columns: columns ;
rowPeriod: 1;
yourself .! !

!DMModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 01:10'!
renderContentOn: canvas

canvas heading with: (DMModelTexts labelFor: #Models);
```

```
level: 3.
listComponent rows: DMModel models asSortedCollection .
canvas render: listComponent .
canvas anchor
callback: [ self add ];
with: DMTexts NEW.! !


WAComponent subclass: #DMModelEditor
instanceVariableNames: 'model'
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

!DMModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
initialize: anInstance
self initialize.
model := anInstance.! !


!DMModelEditor methodsFor: 'rendering' stamp: 'JanTomsa 4/5/2010 22:38'!
renderButtonsOn: canvas
canvas div: [
canvas span: [
canvas cancelButton callback: [self answer: false];
 with: DMTexts CANCEL.
].
canvas span: [
canvas submitButton callback: [self answer: true];
 with: DMTexts SAVE.
].
].! !

!DMModelEditor methodsFor: 'rendering' stamp: 'JanTomsa 4/5/2010 22:38'!
renderContentOn: canvas
canvas form
class: 'dmEditor';
with: [
self
renderModelNameOn: canvas;
renderDescriptionOn: canvas;
renderOptimisationTypeOn: canvas;
renderButtonsOn: canvas;
yourself.
].! !

!DMModelEditor methodsFor: 'rendering' stamp: 'JanTomsa 4/5/2010 22:38'!
renderDescriptionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
```

```
with: (DMModelTexts labelFor: #description).
canvas textArea
id: tagID;
value: model description;
callback: [ :value | model description: value ]
].! !

!DMModelEditor methodsFor: 'rendering' stamp: 'JanTomsa 4/5/2010 22:38'!
renderModelNameOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DMModelTexts labelFor: #modelName).
canvas textInput
value: model modelName;
callback: [:value | model modelName: value]
].! !

!DMModelEditor methodsFor: 'rendering' stamp: 'JanTomsa 4/6/2010 01:28'!
renderOptimisationTypeOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DMModelTexts labelFor: #optimisationType).
canvas select
id: tagID;
name: tagID;
selected: model optimisationType;
list: model optimisationTypeList;
callback: [ :value | model optimisationType: value ].
].! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DMModelEditor class
instanceVariableNames: ''!

!DMModelEditor class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:38'!
on: anInstance
^self basicNew
initialize: anInstance;
yourself.! !


Object subclass: #DMModelTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!
```

```
"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DMModelTexts class
instanceVariableNames: 'labels'!

!DMModelTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:45'!
labelFor: aSymbol
^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).! !

!DMModelTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 09:29'!
labelFor: aSymbol inLanguage: language
^(self labels at: language) at: aSymbol.! !

!DMModelTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/4/2010 09:24'!
labels
| labelsCzech labelsEnglish |
labels ifNil: [
labelsCzech := Dictionary new.
labelsEnglish := Dictionary new.
labelsCzech add: #modelName->'Název'.
labelsEnglish add: #modelName->'Name'.
labelsCzech add: #description->'Popis'.
labelsEnglish add: #description->'Description'.
labelsCzech add: #optimisationType->'Typ optimalizace'.
labelsEnglish add: #optimisationType->'Optimisation type'.
labelsCzech add: #Models->'Modely'.
labelsEnglish add: #Models->'Models'.
labels := Dictionary new.
labels add: #czech->labelsCzech.
labels add: #english->labelsEnglish.
].
^labels.! !


Object subclass: #DMTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DecisionMaker'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DMTexts class
instanceVariableNames: ''!

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:40'!
ACTIONS
^self ACTIONS: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:40'!
ACTIONS: language
^(language = #czech)
```

```
ifTrue: ['Akce']
ifFalse: ['Actions'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:15'!
CANCEL
^self CANCEL: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:10'!
CANCEL: language
^(language = #czech)
ifTrue: ['Zru??it']
ifFalse: ['Cancel'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:42'!
CONFIRMQUESTION
^self CONFIRMQUESTION: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:42'!
CONFIRMQUESTION: language
^(language = #czech)
ifTrue: ['Opravdu?']
ifFalse: ['Are you sure?'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:43'!
DELETE
^self DELETE: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:43'!
DELETE: language
^(language = #czech)
ifTrue: ['Smazat']
ifFalse: ['Delete'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:19'!
HOME
^self HOME: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:19'!
HOME: language
^(language = #czech)
ifTrue: ['??vod']
ifFalse: ['Home'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:20'!
MENU
^self MENU: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:20'!
MENU: language
^(language = #czech)
ifTrue: ['Nab?dka']
ifFalse: ['Menu'].! !
```

```
!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:15'!
NEW
^self NEW: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:13'!
NEW: language
^(language = #czech)
ifTrue: ['Nov??']
ifFalse: ['New'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:14'!
SAVE
^self SAVE: (self defaultLanguage)! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:11'!
SAVE: language
^(language = #czech)
ifTrue: ['Ulo??it']
ifFalse: ['Save'].! !

!DMTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/2/2010 22:45'!
defaultLanguage
" ^#english"
^#czech! !

DMMain initialize!
```

G.2 Iterace 2 - generováno

```
Object subclass: #DXEquation
instanceVariableNames: 'description decisionVariables'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXEquation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
<= anObject
^self description<= anObject description! !

!DXEquation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
decisionVariables
^decisionVariables! !

!DXEquation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
decisionVariables: anObject
decisionVariables := anObject! !

!DXEquation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description
^description! !
```

```
!DXEquation methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description: anObject
description := anObject! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXEquation class
instanceVariableNames: 'instances'!

!DXEquation class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
instances
instances ifNil: [ instances := IdentitySet new ].
^instances! !


WAComponent subclass: #DXEquationCRUD
instanceVariableNames: 'listComponent'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
actionReportColumn
^WAReportColumn new
title: DMTexts ACTIONS;
valueBlock: [ :anInstance :html |
html anchor
onClick: ('return confirm(''', DMTexts CONFIRMQUESTION, ''')');
callback: [ self delete: anInstance  ];
with: DMTexts DELETE
];
yourself .! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
add
| editor instance answer |
instance := DXEquation new.
editor := DXEquationEditor on: instance.
answer := self call: editor.
answer ifTrue: [ DXEquation instances add: instance ]! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:54'!
children
^Array with: listComponent! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
delete: anInstance
DXEquation instances remove: anInstance.! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
descriptionReportColumn
```

```
^WAReportColumn new
title: (DXEquationTexts labelFor: #description);
selector: #description;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
edit: anInstance
| editor |
editor := DXEquationEditor on: anInstance.
self call: editor.! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:54'!
initialize
| columns |
super initialize.
columns := OrderedCollection new.
 columns add: self descriptionReportColumn.
 columns add: self actionReportColumn.
listComponent := WATableReport new
columns: columns;
rowPeriod: 1;
yourself.! !

!DXEquationCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:54'!
renderContentOn: canvas
canvas heading: (DXEquationTexts labelFor: #Instances) level: 3.
listComponent rows: DXEquation instances asSortedCollection.
canvas render: listComponent.
canvas anchor callback: [ self add ]; with: DMTexts NEW.! !


WAComponent subclass: #DXEquationEditor
instanceVariableNames: 'model'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXEquationEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize: anInstance
self initialize.
model := anInstance.! !

!DXEquationEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderButtonsOn: canvas
canvas div: [
canvas span: [
canvas cancelButton callback: [self answer: false];
 with: DMTexts CANCEL.
].
canvas span: [
canvas submitButton callback: [self answer: true];
```

```
 with: DMTexts SAVE.
].
].! !

!DXEquationEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas form
class: 'dmEditor';
with: [
self
renderDescriptionOn: canvas;
renderButtonsOn: canvas;
yourself.
].! !

!DXEquationEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderDescriptionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXEquationTexts labelFor: #description).
canvas textArea
id: tagID;
value: model description;
callback: [ :value | model description: value ]
].! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXEquationEditor class
instanceVariableNames: ''!

!DXEquationEditor class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
on: anInstance
^self basicNew
initialize: anInstance;
yourself.! !


Object subclass: #DXEquationTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXEquationTexts class
instanceVariableNames: 'labels'!

!DXEquationTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
```

```
labelFor: aSymbol
^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).! !

!DXEquationTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol inLanguage: language
^(self labels at: language) at: aSymbol.! !

!DXEquationTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labels
| labelsCzech labelsEnglish |
labels ifNil: [
labelsCzech := Dictionary new.
labelsEnglish := Dictionary new.
labelsCzech add: #Instance->'Rovnice'.
labelsCzech add: #Instances->'Rovnice'.
labelsEnglish add: #Instance->'Equation'.
labelsEnglish add: #Instances->'Equations'.
labelsCzech add: #description->'Popis'.
labelsEnglish add: #description->'Description'.
labels := Dictionary new.
labels add: #czech->labelsCzech.
labels add: #english->labelsEnglish.
].
^labels! !


Object subclass: #DXModel
instanceVariableNames: 'modelName description optimisationType objective'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
<= anObject
^self modelName<= anObject modelName! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
buildStandardLPProblem
Transcript cr.! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description
^description! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description: anObject
description := anObject! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
modelName
^modelName! !
```

```
!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
modelName: anObject
modelName := anObject! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
objective
^objective! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
objective: anObject
objective := anObject! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
optimisationType
^optimisationType! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
optimisationType: anObject
optimisationType := anObject! !

!DXModel methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
solution
Transcript cr.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXModel class
instanceVariableNames: 'instances'!

!DXModel class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
instances
instances ifNil: [ instances := IdentitySet new ].
^instances! !


WAComponent subclass: #DXModelCRUD
instanceVariableNames: 'listComponents'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
actionReportColumn
^WAReportColumn new
title: DMTexts ACTIONS;
valueBlock: [ :anInstance :html |
html anchor
onClick: ('return confirm(''', DMTexts CONFIRMQUESTION, ''')');
callback: [ self delete: anInstance  ];
with: DMTexts DELETE
];
yourself .! !
```

```
!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
add
| editor instance answer |
instance := DXModel new.
editor := DXModelEditor on: instance.
answer := self call: editor.
answer ifTrue: [ DXModel instances add: instance ]! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:21'!
children
^Array with: listComponents! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
delete: anInstance
DXModel instances remove: anInstance.! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
descriptionReportColumn
^WAReportColumn new
title: (DXModelTexts labelFor: #description);
selector: #description;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
edit: anInstance
| editor |
editor := DXModelEditor on: anInstance.
self call: editor.! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize
| columns |
super initialize.
columns := OrderedCollection new.
 columns add: self modelNameReportColumn.
 columns add: self descriptionReportColumn.
 columns add: self optimisationTypeReportColumn.
 columns add: self actionReportColumn.
listComponents := WATableReport new
columns: columns;
rowPeriod: 1;
yourself.! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
modelNameReportColumn
^WAReportColumn new
title: (DXModelTexts labelFor: #modelName);
selector: #modelName;
clickBlock: [ :each | self edit: each ];
yourself .! !
```

```
!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
optimisationTypeReportColumn
^WAReportColumn new
title: (DXModelTexts labelFor: #optimisationType);
selector: #optimisationType;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXModelCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas heading: (DXModelTexts labelFor: #Instances) level: 3.
listComponents rows: DXModel instances asSortedCollection.
canvas render: listComponents.
canvas anchor callback: [ self add ]; with: DMTexts NEW.! !


WAComponent subclass: #DXModelEditor
instanceVariableNames: 'model'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize: anInstance
self initialize.
model := anInstance.! !

!DXModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderButtonsOn: canvas
canvas div: [
canvas span: [
canvas cancelButton callback: [self answer: false];
 with: DMTexts CANCEL.
].
canvas span: [
canvas submitButton callback: [self answer: true];
 with: DMTexts SAVE.
].
].! !

!DXModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas form
class: 'dmEditor';
with: [
self
renderModelNameOn: canvas;
renderDescriptionOn: canvas;
renderOptimisationTypeOn: canvas;
renderButtonsOn: canvas;
yourself.
```

```
].! !

!DXModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderDescriptionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXModelTexts labelFor: #description).
canvas textArea
id: tagID;
value: model description;
callback: [ :value | model description: value ]
].! !

!DXModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderModelNameOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXModelTexts labelFor: #modelName).
canvas textInput
value: model modelName;
callback: [:value | model modelName: value]
].! !

!DXModelEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderOptimisationTypeOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXModelTexts labelFor: #optimisationType).
canvas select
id: tagID;
selected: model optimisationType;
list: model optimisationTypeList;
callback: [ :value | model optimisationType: value ].
].! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXModelEditor class
instanceVariableNames: ''!

!DXModelEditor class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
on: anInstance
^self basicNew
initialize: anInstance;
yourself.! !
```

```
Object subclass: #DXModelTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXModelTexts class
instanceVariableNames: 'labels'!

!DXModelTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol
^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).! !

!DXModelTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol inLanguage: language
^(self labels at: language) at: aSymbol.! !

!DXModelTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labels
| labelsCzech labelsEnglish |
labels ifNil: [
labelsCzech := Dictionary new.
labelsEnglish := Dictionary new.
labelsCzech add: #Instance->'Model'.
labelsCzech add: #Instances->'Modely'.
labelsEnglish add: #Instance->'Model'.
labelsEnglish add: #Instances->'Models'.
labelsCzech add: #modelName->'Název'.
labelsEnglish add: #modelName->'Name'.
labelsCzech add: #description->'Popis'.
labelsEnglish add: #description->'Description'.
labelsCzech add: #optimisationType->'Typ optimalizace'.
labelsEnglish add: #optimisationType->'Optimisation type'.
labels := Dictionary new.
labels add: #czech->labelsCzech.
labels add: #english->labelsEnglish.
].
^labels! !


Object subclass: #DXParamGroup
instanceVariableNames: 'name description'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXParamGroup methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
<= anObject
^self name<= anObject name! !
```

```
!DXParamGroup methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description
^description! !

!DXParamGroup methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description: anObject
description := anObject! !

!DXParamGroup methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
name
^name! !

!DXParamGroup methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
name: anObject
name := anObject! !

!DXParamGroup methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
refreshValues
Transcript cr.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXParamGroup class
instanceVariableNames: 'instances'!

!DXParamGroup class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
instances
instances ifNil: [ instances := IdentitySet new ].
^instances! !


WAComponent subclass: #DXParamGroupCRUD
instanceVariableNames: 'listComponents'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
actionReportColumn
^WAReportColumn new
title: DMTexts ACTIONS;
valueBlock: [ :anInstance :html |
html anchor
onClick: ('return confirm(''', DMTexts CONFIRMQUESTION, ''')');
callback: [ self delete: anInstance  ];
with: DMTexts DELETE
];
yourself .! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
add
```

```
| editor instance answer |
instance := DXParamGroup new.
editor := DXParamGroupEditor on: instance.
answer := self call: editor.
answer ifTrue: [ DXParamGroup instances add: instance ]! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:20'!
children
^Array with: listComponents ! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
delete: anInstance
DXParamGroup instances remove: anInstance.! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
descriptionReportColumn
^WAReportColumn new
title: (DXParamGroupTexts labelFor: #description);
selector: #description;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
edit: anInstance
| editor |
editor := DXParamGroupEditor on: anInstance.
self call: editor.! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize
| columns |
super initialize.
columns := OrderedCollection new.
 columns add: self nameReportColumn.
 columns add: self descriptionReportColumn.
 columns add: self actionReportColumn.
listComponents := WATableReport new
columns: columns;
rowPeriod: 1;
yourself.! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
nameReportColumn
^WAReportColumn new
title: (DXParamGroupTexts labelFor: #name);
selector: #name;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParamGroupCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas heading: (DXParamGroupTexts labelFor: #Instances) level: 3.
```

```
listComponents rows: DXParamGroup instances asSortedCollection.
canvas render: listComponents.
canvas anchor callback: [ self add ]; with: DMTexts NEW.! !


WAComponent subclass: #DXParamGroupEditor
instanceVariableNames: 'model'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXParamGroupEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize: anInstance
self initialize.
model := anInstance.! !

!DXParamGroupEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderButtonsOn: canvas
canvas div: [
canvas span: [
canvas cancelButton callback: [self answer: false];
 with: DMTexts CANCEL.
].
canvas span: [
canvas submitButton callback: [self answer: true];
 with: DMTexts SAVE.
].
].! !

!DXParamGroupEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas form
class: 'dmEditor';
with: [
self
renderNameOn: canvas;
renderDescriptionOn: canvas;
renderButtonsOn: canvas;
yourself.
].! !

!DXParamGroupEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderDescriptionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParamGroupTexts labelFor: #description).
canvas textArea
id: tagID;
value: model description;
callback: [ :value | model description: value ]
```

```
].! !

!DXParamGroupEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderNameOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParamGroupTexts labelFor: #name).
canvas textInput
value: model name;
callback: [:value | model name: value]
].! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXParamGroupEditor class
instanceVariableNames: ''!

!DXParamGroupEditor class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
on: anInstance
^self basicNew
initialize: anInstance;
yourself.! !


Object subclass: #DXParamGroupTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXParamGroupTexts class
instanceVariableNames: 'labels'!

!DXParamGroupTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol
^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).! !

!DXParamGroupTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol inLanguage: language
^(self labels at: language) at: aSymbol.! !

!DXParamGroupTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 01:13'!
labels
| labelsCzech labelsEnglish |
labels ifNil: [
labelsCzech := Dictionary new.
labelsEnglish := Dictionary new.
labelsCzech add: #Instance->'Skupina parametrů'.
```

```
labelsCzech add: #Instances->'Skupiny parametrů'.
labelsEnglish add: #Instance->'Parameter group'.
labelsEnglish add: #Instances->'Parameter groups'.
labelsCzech add: #name->'Název'.
labelsEnglish add: #name->'Name'.
labelsCzech add: #description->'Popis'.
labelsEnglish add: #description->'Description'.
labels := Dictionary new.
labels add: #czech->labelsCzech.
labels add: #english->labelsEnglish.
].
^labels! !


Object subclass: #DXParameter
instanceVariableNames: 'name description value previousValue lastRefreshed expression'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
<= anObject
^self name<= anObject name! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description
^description! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description: anObject
description := anObject! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
expression
^expression! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
expression: anObject
expression := anObject! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
lastRefreshed
^lastRefreshed! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
lastRefreshed: anObject
lastRefreshed := anObject! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
name
^name! !
```

```
!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
name: anObject
name := anObject! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
previousValue
^previousValue! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
previousValue: anObject
previousValue := anObject! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
refreshValue
Transcript cr.! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
value
^value! !

!DXParameter methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
value: anObject
value := anObject! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXParameter class
instanceVariableNames: 'instances'!

!DXParameter class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
instances
instances ifNil: [ instances := IdentitySet new ].
^instances! !


WAComponent subclass: #DXParameterCRUD
instanceVariableNames: 'listComponent'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
actionReportColumn
^WAReportColumn new
title: DMTexts ACTIONS;
valueBlock: [ :anInstance :html |
html anchor
onClick: ('return confirm(''', DMTexts CONFIRMQUESTION, ''')');
callback: [ self delete: anInstance  ];
with: DMTexts DELETE
];
yourself .! !
```

```
!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
add
| editor instance answer |
instance := DXParameter new.
editor := DXParameterEditor on: instance.
answer := self call: editor.
answer ifTrue: [ DXParameter instances add: instance ]! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:53'!
children
^Array with: listComponent! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
delete: anInstance
DXParameter instances remove: anInstance.! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
descriptionReportColumn
^WAReportColumn new
title: (DXParameterTexts labelFor: #description);
selector: #description;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
edit: anInstance
| editor |
editor := DXParameterEditor on: anInstance.
self call: editor.! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
expressionReportColumn
^WAReportColumn new
title: (DXParameterTexts labelFor: #expression);
selector: #expression;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:53'!
initialize
| columns |
super initialize.
columns := OrderedCollection new.
 columns add: self nameReportColumn.
 columns add: self descriptionReportColumn.
 columns add: self valueReportColumn.
 columns add: self previousValueReportColumn.
 columns add: self lastRefreshedReportColumn.
 columns add: self expressionReportColumn.
 columns add: self actionReportColumn.
listComponent := WATableReport new
```

```
columns: columns;
rowPeriod: 1;
yourself.! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
lastRefreshedReportColumn
^WAReportColumn new
title: (DXParameterTexts labelFor: #lastRefreshed);
selector: #lastRefreshed;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
nameReportColumn
^WAReportColumn new
title: (DXParameterTexts labelFor: #name);
selector: #name;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
previousValueReportColumn
^WAReportColumn new
title: (DXParameterTexts labelFor: #previousValue);
selector: #previousValue;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 23:53'!
renderContentOn: canvas
canvas heading: (DXParameterTexts labelFor: #Instances) level: 3.
listComponent rows: DXParameter instances asSortedCollection.
canvas render: listComponent.
canvas anchor callback: [ self add ]; with: DMTexts NEW.! !

!DXParameterCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
valueReportColumn
^WAReportColumn new
title: (DXParameterTexts labelFor: #value);
selector: #value;
clickBlock: [ :each | self edit: each ];
yourself .! !


WAComponent subclass: #DXParameterEditor
instanceVariableNames: 'model'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize: anInstance
```

```
self initialize.
model := anInstance.! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderButtonsOn: canvas
canvas div: [
canvas span: [
canvas cancelButton callback: [self answer: false];
 with: DMTexts CANCEL.
].
canvas span: [
canvas submitButton callback: [self answer: true];
 with: DMTexts SAVE.
].
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas form
class: 'dmEditor';
with: [
self
renderNameOn: canvas;
renderDescriptionOn: canvas;
renderValueOn: canvas;
renderPreviousValueOn: canvas;
renderLastRefreshedOn: canvas;
renderExpressionOn: canvas;
renderButtonsOn: canvas;
yourself.
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderDescriptionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParameterTexts labelFor: #description).
canvas textArea
id: tagID;
value: model description;
callback: [ :value | model description: value ]
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderExpressionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParameterTexts labelFor: #expression).
```

```
canvas textInput
value: model expression;
callback: [:value | model expression: value]
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderLastRefreshedOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParameterTexts labelFor: #lastRefreshed).
canvas span
id: tagID;
with: [ canvas render: dateTimeSelector ].
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderNameOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParameterTexts labelFor: #name).
canvas textInput
value: model name;
callback: [:value | model name: value]
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderPreviousValueOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParameterTexts labelFor: #previousValue).
canvas textInput
value: model previousValue;
callback: [:value | model previousValue: value]
].! !

!DXParameterEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderValueOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXParameterTexts labelFor: #value).
canvas textInput
value: model value;
callback: [:value | model value: value]
].! !
```

```
"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXParameterEditor class
instanceVariableNames: ''!

!DXParameterEditor class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
on: anInstance
^self basicNew
initialize: anInstance;
yourself.! !


Object subclass: #DXParameterTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXParameterTexts class
instanceVariableNames: 'labels'!

!DXParameterTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol
^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).! !

!DXParameterTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol inLanguage: language
^(self labels at: language) at: aSymbol.! !

!DXParameterTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 01:09'!
labels
| labelsCzech labelsEnglish |
labels ifNil: [
labelsCzech := Dictionary new.
labelsEnglish := Dictionary new.
labelsCzech add: #Instance->'Parametry'.
labelsCzech add: #Instances->'Parametry'.
labelsEnglish add: #Instance->'Parameter'.
labelsEnglish add: #Instances->'Parameters'.
labelsCzech add: #name->'Název'.
labelsEnglish add: #name->'Name'.
labelsCzech add: #description->'Popis'.
labelsEnglish add: #description->'Description'.
labelsCzech add: #value->'Hodnota'.
labelsEnglish add: #value->'Value'.
labelsCzech add: #previousValue->'Předchozí hodnota'.
labelsEnglish add: #previousValue->'Previous value'.
labelsCzech add: #lastRefreshed->'Naposledy načteno'.
labelsEnglish add: #lastRefreshed->'Last refreshed'.
```

```
labelsCzech add: #expression->'Výraz'.
labelsEnglish add: #expression->'Expression'.
labels := Dictionary new.
labels add: #czech->labelsCzech.
labels add: #english->labelsEnglish.
].
^labels! !


Object subclass: #DXVariable
instanceVariableNames: 'symbol name description'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 00:07'!
<= anObject
^self name<= anObject name! !

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description
^description! !

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
description: anObject
description := anObject! !

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
name
^name! !

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
name: anObject
name := anObject! !

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
symbol
^symbol! !

!DXVariable methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
symbol: anObject
symbol := anObject! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXVariable class
instanceVariableNames: 'instances'!

!DXVariable class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
instances
instances ifNil: [ instances := IdentitySet new ].
^instances! !
```

```
WAComponent subclass: #DXVariableCRUD
instanceVariableNames: 'listComponent'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
actionReportColumn
^WAReportColumn new
title: DMTexts ACTIONS;
valueBlock: [ :anInstance :html |
html anchor
onClick: ('return confirm(''', DMTexts CONFIRMQUESTION, ''')');
callback: [ self delete: anInstance  ];
with: DMTexts DELETE
];
yourself .! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
add
| editor instance answer |
instance := DXVariable new.
editor := DXVariableEditor on: instance.
answer := self call: editor.
answer ifTrue: [ DXVariable instances add: instance ]! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 00:06'!
children
^Array with: listComponent! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
delete: anInstance
DXVariable instances remove: anInstance.! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
descriptionReportColumn
^WAReportColumn new
title: (DXVariableTexts labelFor: #description);
selector: #description;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
edit: anInstance
| editor |
editor := DXVariableEditor on: anInstance.
self call: editor.! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 01:12'!
initialize
```

```
| columns |
super initialize.

columns := OrderedCollection new.
 columns add: self symbolReportColumn.
 columns add: self nameReportColumn.
 columns add: self descriptionReportColumn.
 columns add: self actionReportColumn.
listComponent := WATableReport new
columns: columns;
rowPeriod: 1;
yourself.! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
nameReportColumn
^WAReportColumn new
title: (DXVariableTexts labelFor: #name);
selector: #name;
clickBlock: [ :each | self edit: each ];
yourself .! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 01:18'!
renderContentOn: canvas
canvas heading level: 3; with: (DXVariableTexts labelFor: #Instances).
listComponent rows: DXVariable instances asSortedCollection.
canvas render: listComponent.
canvas anchor callback: [ self add ]; with: DMTexts NEW.! !

!DXVariableCRUD methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
symbolReportColumn
^WAReportColumn new
title: (DXVariableTexts labelFor: #symbol);
selector: #symbol;
clickBlock: [ :each | self edit: each ];
yourself .! !


WAComponent subclass: #DXVariableEditor
instanceVariableNames: 'model'
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

!DXVariableEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
initialize: anInstance
self initialize.
model := anInstance.! !

!DXVariableEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderButtonsOn: canvas
canvas div: [
canvas span: [
```

```
canvas cancelButton callback: [self answer: false];
 with: DMTexts CANCEL.
].
canvas span: [
canvas submitButton callback: [self answer: true];
 with: DMTexts SAVE.
].
].! !

!DXVariableEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderContentOn: canvas
canvas form
class: 'dmEditor';
with: [
self
renderSymbolOn: canvas;
renderNameOn: canvas;
renderDescriptionOn: canvas;
renderButtonsOn: canvas;
yourself.
].! !

!DXVariableEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderDescriptionOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXVariableTexts labelFor: #description).
canvas textArea
id: tagID;
value: model description;
callback: [ :value | model description: value ]
].! !

!DXVariableEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderNameOn: canvas
| tagID |
canvas div: [
canvas label
for: (tagID := canvas nextId);
with: (DXVariableTexts labelFor: #name).
canvas textInput
value: model name;
callback: [:value | model name: value]
].! !

!DXVariableEditor methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
renderSymbolOn: canvas
| tagID |
canvas div: [
canvas label
```

```
for: (tagID := canvas nextId);
with: (DXVariableTexts labelFor: #symbol).
canvas textInput
value: model symbol;
callback: [:value | model symbol: value]
].! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXVariableEditor class
instanceVariableNames: ''!

!DXVariableEditor class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
on: anInstance
^self basicNew
initialize: anInstance;
yourself.! !


Object subclass: #DXVariableTexts
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'DX-generated'!

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

DXVariableTexts class
instanceVariableNames: 'labels'!

!DXVariableTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol
^self labelFor: aSymbol inLanguage: (DMTexts defaultLanguage).! !

!DXVariableTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/5/2010 22:45'!
labelFor: aSymbol inLanguage: language
^(self labels at: language) at: aSymbol.! !

!DXVariableTexts class methodsFor: 'as yet unclassified' stamp: 'JanTomsa 4/6/2010 01:12'!
labels
| labelsCzech labelsEnglish |
labels ifNil: [
labelsCzech := Dictionary new.
labelsEnglish := Dictionary new.
labelsCzech add: #Instance->'Proměnná'.
labelsCzech add: #Instances->'Proměnné'.
labelsEnglish add: #Instance->'Variable'.
labelsEnglish add: #Instances->'Variables'.
labelsCzech add: #symbol->'Symbol'.
labelsEnglish add: #symbol->'Symbol'.
labelsCzech add: #name->'Název'.
labelsEnglish add: #name->'Name'.
```

```
labelsCzech add: #description->'Popis'.
labelsEnglish add: #description->'Description'.
labels := Dictionary new.
labels add: #czech->labelsCzech.
labels add: #english->labelsEnglish.
].
^labels! !
```