IEGULDĪJUMS TAVĀ NĀKOTNĒ

Eiropas Reģionālās attīstības fonds

Prioritāte: 2.1. Zinātne un inovācijas

Pasākums: 2.1.1. Zinātne, pētniecība un attīstība

Aktivitāte: 2.1.1.1. Atbalsts zinātnei un pētniecībai

Projekts: ”Multi - modeļu izstrādes tehnoloģija .NET pielietojumu projektiem”

Projekta sākuma datums: 2014.gada 1.janvāris.

Projekta beigu datums: 2015.gada 30.jūnijs.

Līguma Nr. 2013/0031/2DP/2.1.1.1.0/13/APIA/VIAA/010

ESF finansējuma saņēmējs: SIA, SWH SETS

Sadarbības partneris: Elektronikas un datorzinātņu institūts (EDI)

Projekta aktivitātes Nr.1.1 ”Meta metamodeļu izpēte" progresa pārskats

Pārskats Nr.1 par periodu no 2014.gada 1.janvāra līdz 2014.gada 30.jūnijam.

SATURS

1. Kopsavilkums....3
2. Ievads....4
3. Meta-metamodeļi....5
3.1. UML un MOF....5
3.2. CDIF....6
3.3. Prasības metamodelim....6
4. Bāzes metametamodelis....8
4.1. Metamodelis....9
4.2. Metaentītija un metaentītiju hierarhija....10
4.3. Metakomplekss un metadaļa....10
4.3.1. Metaatribūts....10
4.3.2. Metasaite....11
4.4. Metametaatribūti....11
4.5. Metaobjekts....12
5. Paplašinātais metametamodelis....13
5.1. Mantošana....13
5.2. Metamodeļa precizējumi....13
5.2.1. Metametatribūtu vērtības....13
5.2.2. Loģiskie metametaatribūti....13
5.2.3. Predefinētie metametaatribūti....14
5.2.4. Predefinētās saites....14
6. Metamodeļu mantošana....15
7. Secinājumi....16
8. Literatūras saraksts....17

1. Kopsavilkums

Pārskata periodā (2014-01-01 – 2013-06-30.) projekta „Multi - modeļu izstrādes tehnoloģija .NET pielietojumu projektiem” aktivitātes Nr.1.1 "Meta metamodeļu izpēte" ietvaros veikti šādi darbi:

1. Pastāvošo metametamodeļu izpēte. Projekta gaitā tika pētīts UML un MOF, kā arī CDIF.

2. Identificētas prasības metamodelim.

3. Bāzes metametamodeļa izstrāde, kas ietver metamodeļa, metaentītijas un metaentītiju hierarhijas, metakompleksa un metadaļas, metaatribūtu un metaobjektu definēšanu.

4. Paplašinātā metametamodeļa izstrādi, kas ietver mantošanas, metamodeļu precizējumu, metamodeļu mantošanas definēšanu.

5. Aktivitātes pētnieciskā darbība apspriesta ik nedēļas projekta semināros.

2. Ievads

Informatīvo sistēmu izstrādes laikā tiek veikta sistēmas analīze, lai noskaidrotu, kādi ir sistēmai svarīgie objekti. Analīzes rezultātā veidojas modelis, kurš ar noteikti definētas notācijas palīdzību apraksta sistēmas modeli. Eksistē daudzas notācijas, kas ļauj aprakstīt to vai citu sistēmas aspektu, piemēram, ER-modelēšana, UML (*Unified modeling language*). To, ko var aprakstīt ar modeļa palīdzību nosaka metamodelis. **Metamodelis** ir modelis, ar kura palīdzību apraksta modeļus. Definējot metamodeli, mums ir iespēja veidot modeli atbilstoši metamodelim. Ja ir vajadzīga iespēja definēt dažādus metamodeļus, nepieciešams definēt modeli, ar kuru varētu aprakstīt metamodeļus. **Metametamodelis** ir modelis, ar kura palīdzību apraksta metamodeļus. Metametamodelim jābūt pietiekoši universālam. Pārskata periodā tika izstrādāts metametamodelis, uz kura balstās tālāk projekta gaitā izstrādātie metamodeļi.

Šis pārskats ir veltīts projekta apakšaktivitātes Nr.1.1 "Meta metamodeļu izpēte" ietvaros veiktajiem pētījumiem. Apraksts sastāv no 4 daļām. Pirmajā daļā ir apskatīti pazīstamākie meta-metamodeļi, Otrajā daļā aprakstīts bāzes metametamodelis, kurš tiks izmantots projektā. Trešajā daļā aprakstīti bāzes matametamodeļa paplašinājumi, kas saistīti ar metamodeļa realizāciju (ērtāka metametamodeļa veidošana, realizācijas vides ierobežojumi, metametamodeļa precizējumi). Ceturtajā daļā aprakstīta metmetamodeļu mantošanas shēma un tai atbilstošie modeļa paplašinājumi.

Diagrammās ir izmantoti termini, kuri veidoti uz angļu valodas bāzes, bet tekstā termini, kuri veidoti uz latviešu valodas bāzes.

3. Meta-metamodeļi

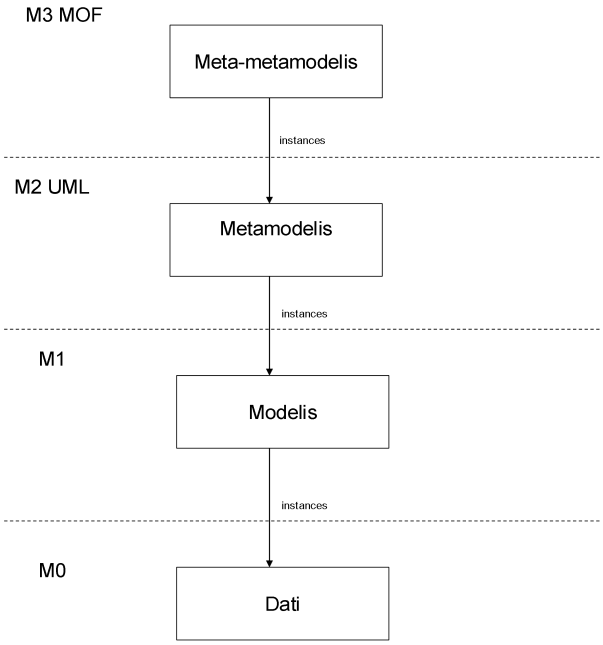
Termins metametamodelis parādījās pagājušā gadsimta deviņdesmitajos gados. Tieši deviņdesmitajos gados parādījās UML (Unified Modelling Language) [1] un EIA/CDIF (Electronic Industries Alliance/CASE Data Interchange Format)[2].

3.1. UML un MOF

UML ir izstrādājusi organizācija OMG (Object Management Group). Sākotnēji UML tika radīta kā grafiska valoda, kas varētu aprakstīt informatīvās sistēmas, vālāk tika izveidota metamodelēšanas arhitektūra MOF (Meta-Object Facility) ar kuras palīdzību var aprakstīt UML, ka arī citas līdzīgas modelēšanas valodas.

Meta-metmodeļa jēdziens ir svarīga UML sastāvdaļa. Metametamodelis ir modelis, kurš definē valodu metamodeļu aprakstīšanai. Meta-metamodeļa un metamodeļa saistība ir analoga ar saistību starp metamodeli un modeli. Šis abstrakcijas līmenis ir svarīgs tikai izstrādes rīku veidotajiem. UML ir definēts, izmantojot MOF meta-metamodeli. [3]

UML ir četru līmeņu metamodeļu hierarhija.



Zīmējums 1.

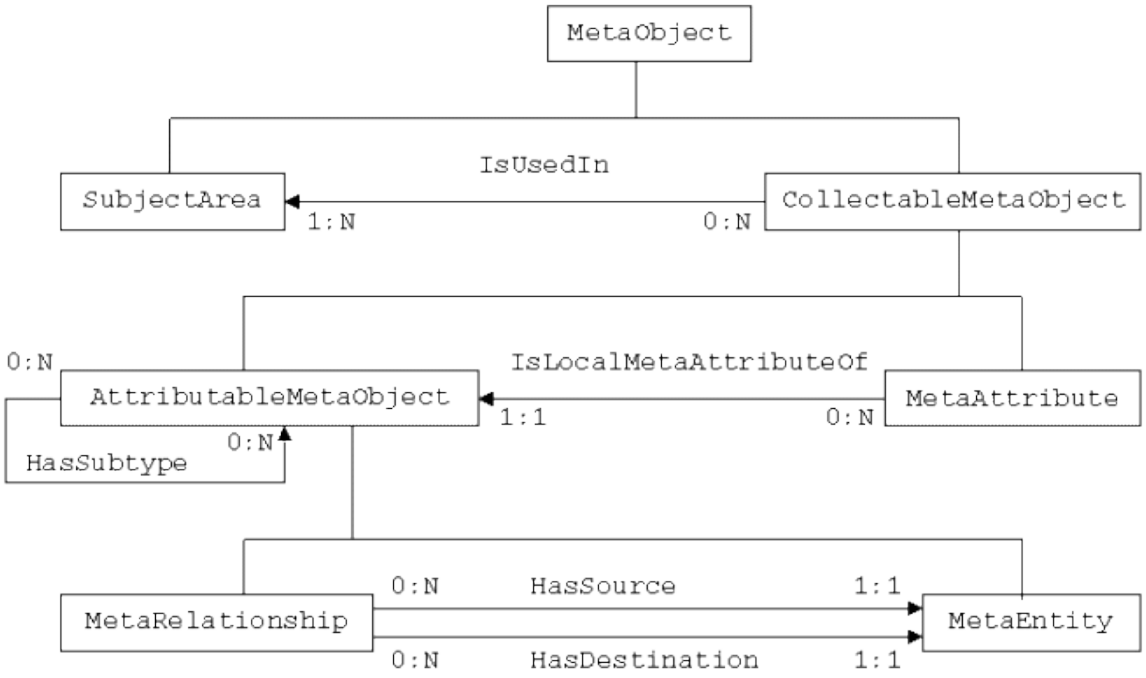
UML izstrādātāji šos līmeņus apzīmē ar M0, M1, M2, M3. M0 līmenis ir izstrādātā sistēma, M1 – sistēmas modelis, t.i. modelis, ar kuru strādā sistēmas izstrādātājs. M2 ir valoda, kuru izmanto, lai aprakstītu modeli, piemēram UML, M3 ir meta-metamodelis MOF, ar kura palīdzību tiek aprakstīts UML un citas līdzīgas modelēšanas valodas.

UML ir veidots tā, kā MOF faktiski ir apakškopa no UML klašu diagrammas. Var teikt, ka UML ir sevi aprakstošs.

3.2. CDIF

CDIF metamodeļa ideja parādījās, mēģinot standartizēt modeļa datu apmaiņu starp dažādiem CASE (computer-aided software engineering) rīkiem. Kā rezultāts radās EIA/CDIF (Electronic Industries Alliance/CASE Data Interchange Format), kura pamatā ir meta-metamodelis, ar kura palīdzību var definēt EIA/CDIF metamodeļus. [4] Līdzīgi kā UML/MOF, CDIF izmanto četru līmeņu hierarhiju: M0, M1, M2, M3, kur M0 ir lietotāja dati, bet M3 ir meta-metamodelis.

CDIF meta-metamodelis redzams attēlā (attēls no [2]):

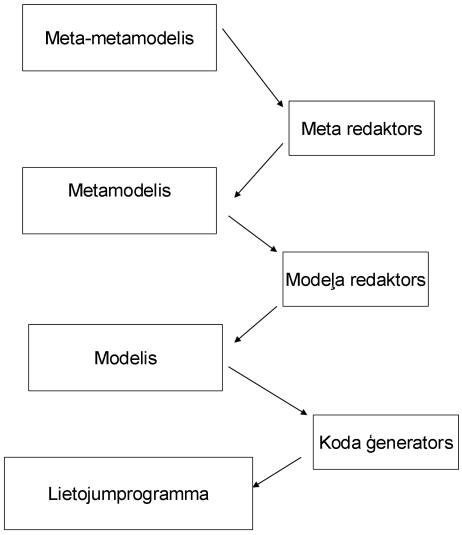


Zīmējums 2.

3.3. Prasības metamodelim

Pielietojumprojektu izstrādē būtu saprātīgi izmantot četru līmeņu metamodeļu hierarhiju. M0 līmenī būtu meta-metamodelis, uz kura balstoties meta redaktorā ar diagrammu

palīdzību varētu definēt metamodeļus. Balstoties uz metamodeļu definīcijām, ar speciāla modeļa redaktora palīdzību varētu definēt sistēmas modeļus. Savukārt, no atbilstoši definētam sistēmas modelim, koda ģenerators varētu ģenerēt lietojumprogrammas sagatavi.



Zīmējums 3.

Gan UML/MOF, gan EIA/CDIF definē savus metamodeļus, kurus var izmantot informatīvo sistēmu modelēšanā un izstrādē.

Tomēr ir vairāki faktori, kas rada problemātisku UML pielietošanu:

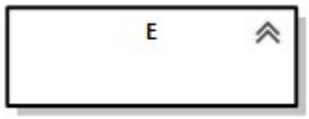
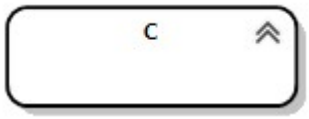
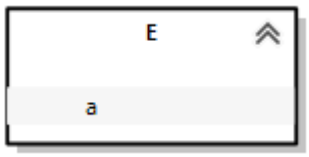
UML neļauj glabāt informāciju, kas neiekļaujas metamodelī.
UML nekādā veidā nenodrošina lietotāja interfeisa modelēšanu.
Klašu diagramma nav ideāla datubāzes modelēšanai.
UML specifikācija ir pusformāla. Precīzas semantikas trūkums var novest pie neviennozīmīgas interpretācijas. [5]

Arī CDIF metamodeļos ir līdzīgas problēmas. Ja metamodeļi jau ir definēti, nav iespējas modelēt aspektus, kas nav definēti metamodelī.

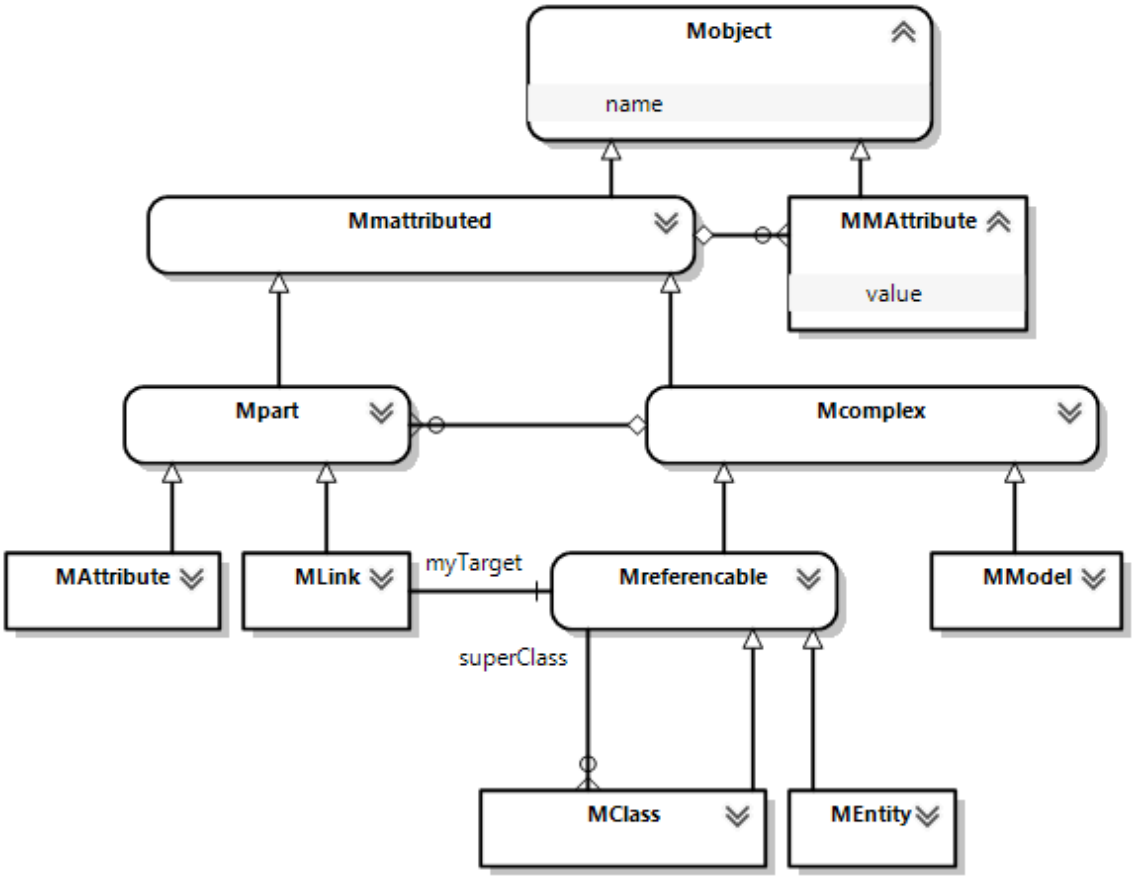
Meta-metamodelim vajadzētu būt pietiekoši vienkāršam un tajā pašā laikā pietiekoši plašam, lai tas pārklātu visu, kas nepieciešams, lai definētu metamodeļus.

4. Bāzes metametamodelis

Metamodeļa attēlošanai izmantosim diagrammu Diagrammu zīmēšana ir izmantots projektā izstrādātais Metamodeļu redaktors. Meta-meta modelis ir sevi aprakstošs. Nākošajā tabulā ir aprakstīta meta redaktora diagrammās izmantotā notācija. Notācija ir visai līdzīga UML notācijai.

Simbols	Apraksts
	Meta entītija (ar vārdu E)
	Meta klase (ar vārdu C)
	Metaentītijai (ar vārdu E) piederošs metaatribūts (ar vārdu a)
	Virsklase. Bultiņa norāda uz virsklasi.
	Metasaite (ar vārdu s1) – neobligāta, var būt tikai 1 instance.
	Metasaite (ar vārdu s2) – obligāta, var būt tikai 1 instance.
	Metasaite (ar vārdu s3) – neobligāta, var būt vairākas instances.
	Metasaite (ar vārdu s4) – obligāta, var būt vairākas instances.
	Predefinētā metasaite „saturSastāvdaļu”

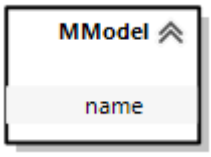
Projektā izstrādātais bāzes metametamodelis ir redzams nākošajā diagrammā.



Zīmējums 4.

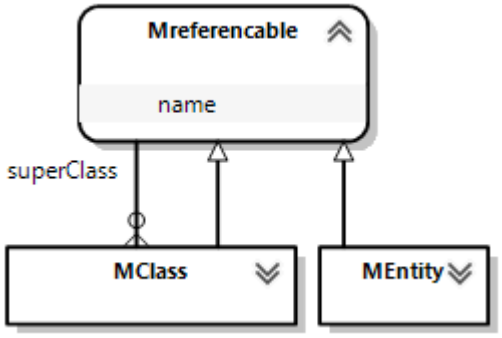
Apskatīsim sīkāk tipus, kuri tiek izmantoti metamodeļu aprakstīšanai.

4.1. Metamodelis



Metamodelis (MModel) ir objekts, kas reprezentē aprakstāmo metamodeli. Katrā metamodelī ir tieši viena metamodeļa instance. Katram metamodelim ir vārds.

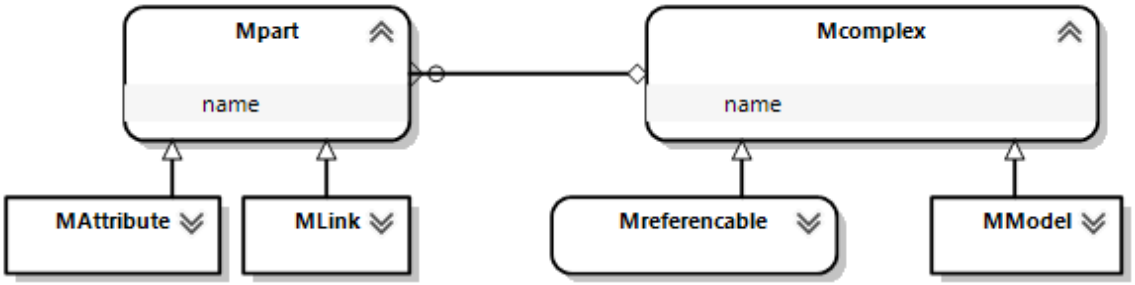
4.2. Metaentītija un metaentītiju hierarhija



Zīmējums 5.

Metaentītija (MEntity) tiek lietota, lai attēlotu metamodelī veidojamos objektus (instances). **Metaklase** (MClass) tiek izmantota hierarhijas veidošanai, lai, izmantojot saiti **virsklase** (superClass), varētu mantot meta objektu īpašības, līdzīgi, kā tas ir objektorientētajā programmēšanā. Metaklase ir abstrakts objekts – tā nav instancējama. Metaentītijai un metaklasei ir kopējs nosaukums – **m-referencējams** (MReferencable). Mreferencējamam ir netukšs unikāls vārds, t.i. metaentītijām un metaklasēm ir kopēja vārdu telpa, kurā jānodrošina unikalitāte.

4.3. Metakomplekss un metadaļa



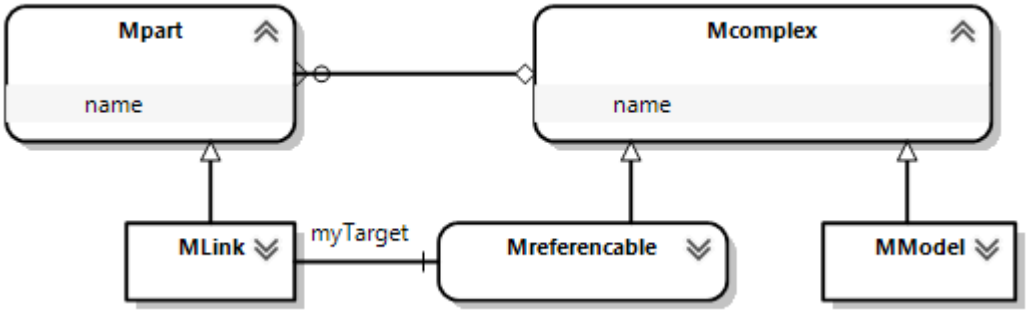
Zīmējums 6.

Gan metaentītījām, gan metamodeļiem var būt **metadaļas** (Mpart), kuras var aprakstīt kaut kādas šo objektu īpašības. Metamodeļus un metentīšu kopējais nosaukums ir **metakomplekss** (Mcomplex). Katra metadaļa pieder kādam metakompleksam un katrai metadaļai ir vārds, kurš ir unikāls metakompleksa ietvaros. Ir iespējamas divu veidu metadaļas – **metaatribūti** (MAttribute) un **metasaites** (MLink).

4.3.1. Metaatribūts

Metaatribūtam modeļa ietvaros ir vērtība un tas tiek atpazīts pēc netukša viena metakompleksa ietvaros unikāla vārda.

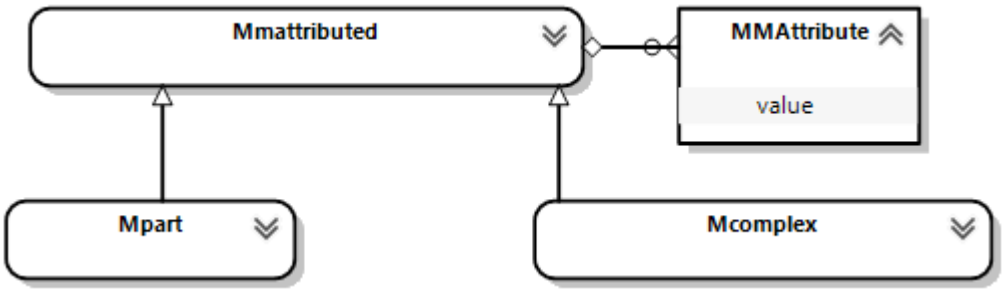
4.3.2. Metasaite



Zīmējums 7.

Metasaite veido savienojumu no objekta, kuram tā pieder, uz objektu, uz kuru norāda saite **mansMērķis** (myTarget). Metasaite var piederēt metaklasei, metaentītijai vai metamodelim un norādīt uz metklasi vai metaentītiju. Metasaite nevar norādīt uz metamodeli. Metasaites vārds, tāpat kā metaatribūta vārds ir unikāls metakompleksa ietvaros.

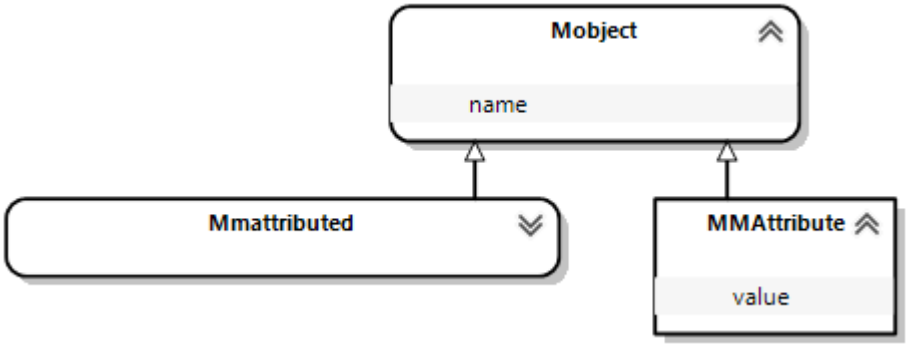
4.4. Metametaatribūti



Zīmējums 8.

Metametaatribūti ir atribūti, kuru vērtības tik uzdotas jau metamodeļa izstrādes laikā (piemēram, metaatribūtu obligātums). Metametaatribūti var tikt pierakstīti jebkuram no iepriekš apskatītajiem objektiem – gan metadaļām, gan metakompleksiem jeb **metametaatributējamiem** (Mmattributed). Metametaatribūtam ir vārds un vērtība. Vārds ir unikāls apskatāmā objekta ietvaros. Metametaatribūti modelī nepiedalās, bet parasti nosaka, kas modelī drīkst vai nedrīkst parādīties, t.i. kalpo metamodeļa papildus precizēšanai.

4.5. Metaobjekts

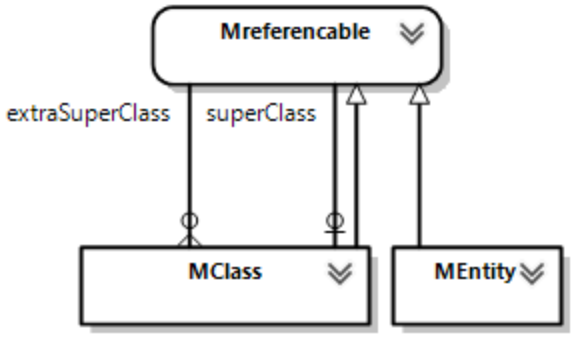


Zīmējums 9.

Metaobjekts (Mobject) sevī ietver mmatributējamus un metametaatribūtus. Katram metaobjektam ir savs vārds.

5. Paplašinātais metametamodelis

5.1. Mantošana



Zīmējums 10.

Ņemot vērā, ka .NET tiek atļauta tikai vienkāršā mantošana (var būt tikai viena virsklase) metametamodelī tiek ieviestas 2 veidu saites mantošanai – **virsklase** (superClass) un **extravirsklase** (extraSuperClass). Katrai metaentītijai vai metaklasei var būt tikai viena saite virsklase. Ja ir nepieciešama daudzkāršā mantošana, tā tiek definēta, izmantojot saiti extravirsklase. Virsklase tiek realizēta ar mantošanas palīdzību, bet extravirsklasēs mantošana tiek realizēta ar programmas koda palīdzību.

5.2. Metamodeļa precizējumi

Metamodeļa precizējumi nekādā veidā nemaina bāzes metametamodeļa loģiku, tie tikai ievieš papildus nosacījumus, kas ir derīgi modelēšanā.

5.2.1. Metametatribūtu vērtības

Metametaatribūta vērtība (value) ir simbolu rinda. Virtuāli var iedomāties, ka jebkuram objektam, kuram var būt metametaatribūti, ir pierakstīti visi iespējamie metametaatribūti. To metametaatribūtu, kas reāli nav pierakstīti, vērtība ir null.

5.2.2. Loģiskie metametaatribūti

Metametaatribūtiem nav paredzēts datu tips, tie tiek interpretēti ka simbolu rindas. Bet jebkura metametaatribūta vērtību var interpretēt arī kā loģisko vērtību pēc sekojoša likuma – vērtība ir „patiess” (true) tad un tikai tad, ja tās simboliskā vērtība ir „true”, neņemot vērā lielo un mazo burtu rakstību. Tas nozīmē, ja metametaatribūta vērtība nav norādīta, tā loģiskā vērtība ir „aplams” (false). Tālāk, runājot par loģiskajiem metametaatribūtiem, ir jālieto ši interpretācija.

5.2.3. Predefinētie metametaatribūti

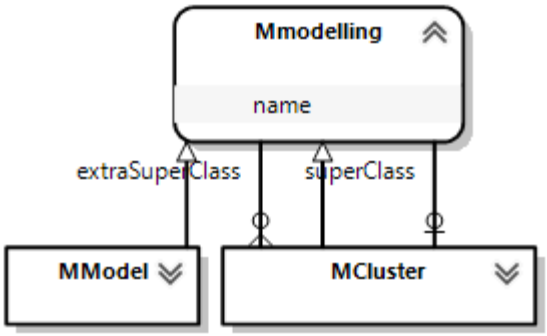
Lai atvieglotu metamodeļu veidošanu paplašinātajā metametamodelī ir ieviesti sekojoši predefinēti metametaatribūti:

Objekttips	Metametaatribūts	Vērtības tips	Apraksts
Mpart	isMandatory	loģiskais	Ja vērtība „true", metadaļa ir obligāta
Mpart	isMultiple	loģiskais	Ja vērtība „true", metadaļai var būt vairākas instances
Mreferencable,MPart	isTechnical	loģiskais	Ja vērtība „true", dotais metaobjekts tiks izmantots tehniskos nolūkos (piemēram ģenerācijā).
Mattribute	constraint	simbolu rinda	Noteiktā sintaksē aprakstītas iespējamās metaatribūta vērtības (piemēram, pārskaitījums)
Mattribute	dataType	simbolu rinda	Metaatribūta datu tips, noklusētā vērtība ir „string"
Mattribute	isName	loģiskais	Ja vērtība „true", metaatribūta vērtība kalpo kā vārds metaentītijai, kurai pieder metaatribūts.
Mattribute	isNational	loģiskais	Ja vērtība „true", metaatribūts atbalsta daudzvalodību.

5.2.4. Predefinētās saites

Saite **saturSastāvdaļu** (contains) ir metasaites paveids, kas kalpo lai norādītu, kurus objektus kā sastāvdaļas satur metaobjekts, kuram pieder dotā saite.

6. Metamodeļu mantošana.



Zīmējums 11.

Lai varētu strukturēt metamodeļus, ir ieviesta metamodeļu mantošana. Metamodeļi, no kuriem tiek mantoti citi metamodeļi tiek saukti par **metakalasteriem** (MCluster). Līdzīgi, kā ar metaentītijām, mantošana notiek, izmantojot saites **virsklase** (superType) un **extravirsklase** (extraSuperClass).

Metamoddeļi var atsaukties uz objektiem metamodeļos, kuri ir virsklase, bet tie nevar piedefinēt metaobjektiem no virsmodeļiem jaunas metadaļas.

7. Secinājumi

Pielietojumu projektu izstrādei ir radīts meta-meta modelis, ar kura palīdzību var tikt veidoti metamodeļi. Izmantojot meta-metaatribūtus, meta-meta modelis dinamiski var tikt papildināts metamodelēšanas laikā, kas rada iespēju pievienot jaunas īpašības metamodeļiem, nemainot meta-metamodeli. Tas ir svarīgi, jo uz meta-metamodeli balstās gan iekšējās datu struktūras, kuras tiks izmantotas gan metamodelēšanā, gan modelēšanā , gan arī metamodeļu un modeļu glabāšanas formāts.

Metamodeļu mantošanas mehānisms ļauj definēt metamodeļus, kuri ir izmantojami daudzos projektos, katra projekta specifiskos aspektus definējot atdalītos metamodeļos, kuri manto (jeb izmanto) no vieniem un tiem pašiem metamodeļiem.

8. Literatūras saraksts

[1] „Unified Modeling Language™ (UML®)" http://www.omg.org/spec/UML/

[2] Univ.Ass. Dr. Rony G. Flatscher "An Overview of the Architecture of EIA's CASE Data Interchange Format (CDIF)" http://wi.wu-wien.ac.at/rgf/9606mobi.html

[3] Rumbaugh, J., Jacobson, I., and Booch, G. (1999) The Unified Modeling Language Reference Manual, Reading MA: Addison-Wesley.

[4] Flatscher, Rony G., (2002), "Metamodeling in EIA/CDIF---meta-metamodel and metamodels" in ACM Transactions on Modeling and Computer Simulation Volume 12 Issue 4, October 2002 pp 322-342.

[5] Guillermo Camilo Torres Díaz National University of San Agustín of Arequipa—Perú "UML: Panacea or Malady?" http://msdn.microsoft.com/en-us/library/cc948344.aspx