Pielikums nr.2

Valsts pētījumu programmas Kiberfizikālās sistēmas, ontoloģijas un biofotonika drošai&viedai pilsētai un sabiedrībai(SOPHIS)

Projekts Nr. 2 Uz ontoloģijām balstītas tīmekļa videi pielāgotas zināšanu inženierijas tehnoloģijas

1.posma

Zinātniskā atskaite

SATURS

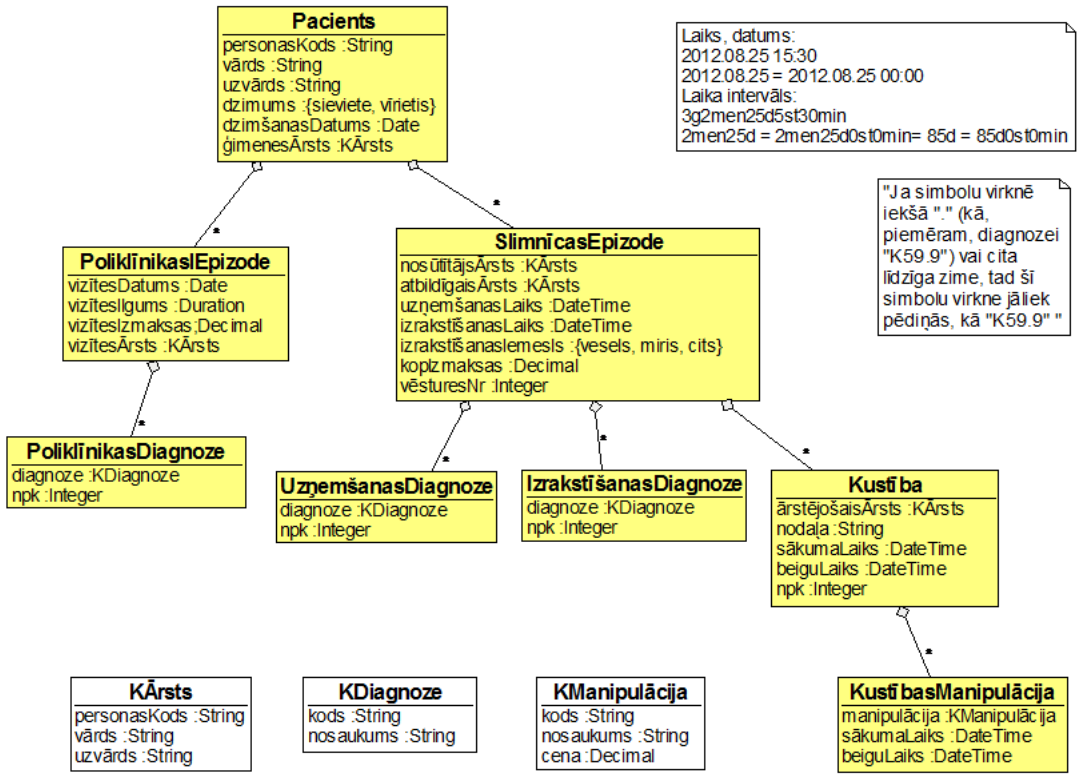
Pielikums 2.1

Ātro vaicājumu valoda

Dotais valodas apraksts ir paredzēts tās lietotāju galvenajai kategorijai – nozaru speciālistiem, kuri nav profesionāli programmētāji. Tādēļ apraksts sākas ar detalizētu datu ontoloģijas skaidrojumu

1.Datu ontoloģija

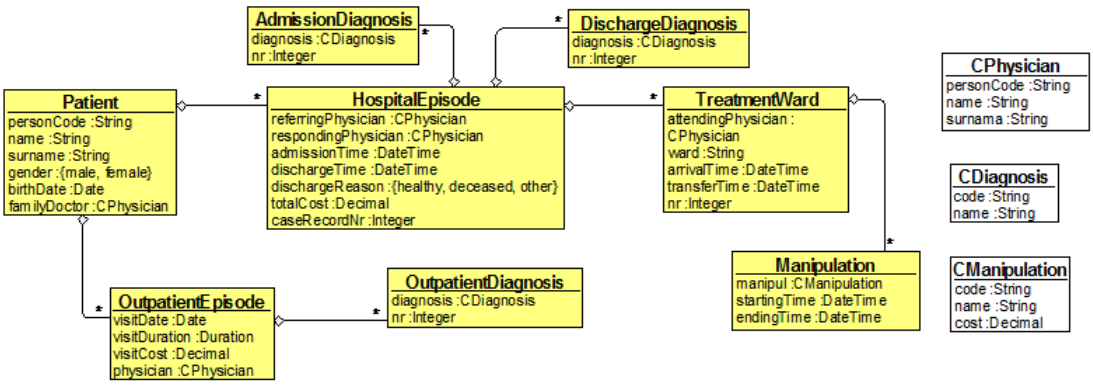
Piedāvātās vaicājumu valodas lietojums konkrētā nozarē, piemēram, medicīnā vienmēr balstās uz šīs nozares datu aprakstu ar attiecīgās datu shēmas palīdzību. Zīm.1 ir attēlota ļoti vienkāršota datu shēma priekš slimnīcas. Šāda veida datu shēmas sauc par **datu ontoloģijām**. Taču, pirms skaidrojam formālos apzīmējumus, pacentīsimies saturīgi paskaidrot, kādi dati mūsu ļoti vienkāršotajā slimnīcas datu bāzē tiek glabāti (tad arī mēs labāk sapratīsim turpmākos formālos apzīmējumus):



Zīm.1

1) Par katru pacientu tiek glabāti viņa personīgie dati – personas kods, vārds, uzvārds, dzimums, dzimšanas datums, kā arī ziņas par viņa ģimenes ārstu;
2) Papildus par katru pacientu tiek glabāts saraksts ar viņa slimnīcas epizodēm (slimnīcas epizode – tas ir viens ārstēšanās kurss slimnīcā, tādu vienam pacientam var būt vairākas);
3) Par katru slimnīcas epizodi tiek glabāti tās pamatdati – kas ir nosūtītājs ārsts, kas ir ārstējošais ārsts, uzņemšanas laiks, izrakstīšanas laiks, kopizmaksas par doto epizodi, kā arī vēstures numurs (katrai epizodei tiek piešķirts savs unikāls numurs);

4) Papildus par katru slimnīcas epizodi tiek glabāts vēl saraksts ar t.s. kustībām (ar kustību tiek saprasta pacienta ārstēšana konkrētā slimnīcas nodaļā, epizodes laikā pacients var pabūt vairākās nodaļās);
5) Par katru kustību tiek glabāti tās pamatdati – kas ir ārstējošais ārsts, kāda nodaļa atbilst šai kustībai, kustības sākuma laiks, kustības beigu laiks (t.i. kad pacients pārvests uz citu nodaļu vai izrakstīts), kustības numurs dotās epizodes ietvaros (t.i. kustības tiek numurētas – pirmajai kustībai dotajā epizodē tiek piešķirts numurs=1, otrajai numurs=2, utt.);
6) Papildus par katru kustību vēl tiek glabāts saraksts ar manipulācijām (ārstēšanas vai izmeklēšanas procedūrām), kas tiek veiktas šīs kustības laikā;
7) Tālāk, par katru manipulāciju tiek glabāti tās dati – kas tā par manipulāciju, manipulācijas sākuma laiks, manipulācijas beigu laiks;
8) Atgriezīsimies vēl pie slimnīcas epizodēm, par katru epizodi tiek glabāts arī saraksts ar uzņemšanas diagnozēm un saraksts ar izrakstīšanas diagnozēm; diagnozes epizodes ietvaros tiek numurētas, uzņemšanas diagnoze ar numuru=1 ir galvenā uzņemšanas diagnoze, līdzīgi, izrakstīšanas diagnoze ar numuru=1 ir galvenā izrakstīšanas diagnoze;
9) Mūsu slimnīcā ir arī ambulatoro pacientu poliklīnika, tādēļ datu bāzē par katru pacientu tiek glabāts ne tikai slimnīcas epizožu saraksts, bet arī poliklīnikas epizožu saraksts;
10) Savukārt, par katru poliklīnikas epizodi tiek glabāti tās pamatdati – vizītes datums, vizītes ilguma, vizītes izmaksas, vizītes ārsts, kā arī saraksts ar poliklīnikā uzstādītajām diagnozēm (tās var būt vairākas, diagnoze ar numuru=1 nozīmē, ka tā ir galvenā).



Zīm.2.

Tagad, balstoties uz zīm.1, paskaidrosim turpmāk lietotos formālos jēdzienus. Ar taisnstūra simboliem tiek attēlotas t.s. **datu klases**. Klases vārds ir dots taisnstūra augšējā daļā treknpierakstā. Zīm.1 ir šādas klases: Pacients, PolEpizode, PolDiagnoze, SlimEpizode, }UznemDiagnoze, IzrakstDiagnoze, Kustība, Manipulacija, kā arī t.s. klasifikatoru klases KArsts, KDiagnoze, KManipulacija. Šie klašu vārdi ir saīsinātie nosaukumi tiem jēdzieniem, kas jau parādījās mūsu slimnīcas datu bāzes saturīgajā skaidrojumā. Zem klases vārda ir uzskaitīti šai klasei atbilstošie atribūti (aiz divpunktes seko atribūtu tipi). Piemēram, klasei Pacients ir šādi atribūti: personasKods, vards, uzvards, dzimums, dzDatums, gimArsts. SlimEpizodei savukārt ir atribūti nosArsts, atbilArsts, uznLaiks, izrLaiks, izrIemesls, kopizmaksas, utt.

Klases vārds ir jāuztver kā sugas vārds, to rakstām vienskaitlī: Pacients, SlimEpizode utt. Taču datu bāzē, kā likums, ir informācija par daudziem konkrētiem Pacientiem, daudzām konkrētām SlimEpizodēm, utt. Šos konkrētos Pacientus mēs turpmāk sauksim par klase

Pacients **instancēm**; tāpat šīs konkrētās SlimEpizodes mēs sauksim par klases SlimEpizode **instancēm**, utt. (citiem vārdiem sakot "konkrēts Pacients" vietā lietosim terminu "Pacienta instance").

Tagad konkrētāk, kas šajā "datu pasaulē" tiek saprasts ar kādas klases instanci, piemēram, Pacienta instanci? Mūsu vaicājumu valodā var uzrakstīt šādu komandu:

```
PARĀDĪT 1 Pacientu
```

(saturīgi tas nozīmē "parādīt vienu Pacienta instanci no mūsu datu bāzes"). Ja mēs nospiedīsim pogu "Izpildīt", uz ekrāna parādīsies, piemēram, šāds teksts:

```
Pacients {
_personasKods=230614-104802
_vārds=Anna
_uzvārds=Priede
_dzimums=female
_dzDatums=2014-06-23
_gimArsts=nill
}
```

Kā redzam no šī piemēra, ar pacienta instanci tiek saprasti dati (t.i. atribūtu vērtības) par konkrētu pacientu. Starp citu, šajā piemērā parādās "gimArsts=nill". Tas nozīmē, ka datu bāzē šim konkrētajam pacientam nav norādīts (t.i. nav ievadīts), kas ir viņa "gimArsts". Ja kādai instancei kāda atribūta vērtība nav ievadīta, tad mūsu vaicājumu valoda uzskata, ka šī atribūta vērtība ir **nill.** Varam tagad precizēt mūsu vaicājumu:

```
PARĀDĪT 1 Pacientu KURAM gimArsts<>nill
```

Rezultātā uz ekrāna parādīsies, piemēram, šāds teksts:

```
Pacients {
_personasKods=280913-10406
_vārds=Andrejs
_uzvārds=Ozoliņš
_dzimums=male
_dzDatums=2013-09-28
_gimArsts=KArsts {
__ personasKods =120155-10245
__ vārds=Juris
__ uzvārds=Krams
_}
}
```

Šajā gadījumā tiek parādīta atribūta "gimArsts" konkrēta vērtība - tā ir klases "KArsts" konkrēta instance.

Rezumējot iepriekš teikto, konkrētu instanci raksturo tas, ka tās atribūtiem ir norādītas konkrētas vērtības (arī "nill" tiek uzskatīta par atribūta vērtību).

Vēl viena tehniska piezīme – turpmākajā tekstā klašu instances mēs attēlosim nedaudz ekonomiskākā pierakstā:

```
Pacients { personasKods=280913-10406, vārds=Andis, uzvārds=Ozoliņš, dzimums=male,
dzDatums=2013-09-28, gimArsts=KArsts { personasKods =120155-10245, vārds=Juris,
uzvārds=Krams } }
```

Turpinām ontoloģijas skaidrojumu. Zīm.1 attēlotajā ontoloģijā visas klases ir sadalītas divās grupās. Pirmajā grupā ir t.s. **pamatklases**, kas ir saistītas viena ar otru, sākot ar klasi Pacients, tā veidojot kokveida struktūru. Mēs mūsu vaicājumu valodā izmantosim tikai šādas kokveida struktūras. Otrajā grupā ir t.s. **klasifikatoru klases**, šajā piemērā tās ir trīs: KArsts, KDiagnoze, KManipulacija.

Starp pamatklasēm ir attiecības, kas zīm.1 ir attēlotas līnijas formā, kurai vienā galā ir rombiņš, bet otrā galā ir simbols zvaigznīte "*". Dabīgā valodā šīs attiecības ir izsakāmas ar šādu vienkārši paplašinātu teikumu palīdzību:

Katram Pacientam atbilst vairākas SlimEpizodes (zvaigznīte nozīmē "vairākas", rombiņš nozīmē, ka katrai SlimEpizodei atbilst tieši viens Pacients).
Katrai SlimEpizodei atbilst vairākas UznemDiagnozes, vairākas IzrakstDiagnozes un vairākas Kustības.
Katrai Kustībai atbilst vairākas Manipulacijas.
Vēl katram Pacientam atbilst vairākas PolEpizodes.
Katrai PolEpizodei atbilst vairākas PolDiagnozes.

Šie dabīgās valodas teikumi viennozīmīgi paskaidro attiecību jēgu starp pamatklasēm.

Tagad aplūkosim **atribūtus** un to **tipus**. Atribūtiem var būt divu veidu tipi:

Pirmie ir t.s. **datu tipi:**
a) String (šī datu tipa vērtības ir simbolu virknes, piemēram, "Bērziņš", "K2.29"),
b) Integer (šī datu tipa vērtības ir veseli skaitļi, piemēram, 25, 120),
c) Decimal (šī datu tipa vērtības ir decimālskaitļi, piemēram, 0.25, 55.05),
d) Date (piemēram, 2015.10.25, 2014.01.01)
e) DateTime (pimēram, 2015.10.25 15:30:00, 2014.01.01 00:30:00),
f) Duration (piemēram, 3g2men25d5st30min, 85d0st0min, 85d)
g) pārskaitāmais tips - nozīmē to, ka atribūta iespējamās vērtības ir uzskaitītas tipa vārda vietā, liekot tās figūriekavās, piemēram,
dzimums:{sieviete, vīrietis}, izrlemesls:{vesels, miris, cits}.

Otrie ir t.s. **klasifikatoru tipi**, piemēram,
gimArsts :KArsts.
Tas nozīmē, ka atribūta gimArsts vērtības ir klases KArsts instances, piemēram,
gimArsts=KArsts { personasKods =120155-10245, vārds=Juris, uzvārds=Krams }

Ievada noslēgumā vēl izpildīsim mūsu vaicājumu valodas komandu:

PILNPARĀDĪT 1 Pacientu

Rezultātā mēs ieraudzīsim uz ekrāna lielu instanču koku, kurā izvēlētajam pacientam tiek parādītas visas viņa SlimEpizodes instances, katrai SlimEpizodes instancei tiek parādītas visas viņas Kustību instances, katrai Kustību instancei tiek parādītas visas tās laikā veiktās manipulāciju instances utt.

2. Piemēri

Tagad dosim tipiskus vaicājumu piemērus, balstītus uz zīm.1 redzamo datu ontoloģiju. Šie vaicājumi būs formulēti dabīgajā valodā, vienīgi daži vārdi būs rakstīti ar lielajiem burtiem – tie būs vārdi, kas tiks lietoti arī formālajā vaicājumu valodā (ko "saprot dators"), pie tam

tādā pašā nozīmē, kā šajos piemēros. Šajos dabīgās valodas vaicājumu formulējumos izmantosim arī dažus formālos apzīmējumus, lai padarītu vaicājumus kompaktākus un vieglāk uztveramus. Izmantosim arī vispārzināmos aritmētisko operāciju simbolus + (plus), - (mīnus), * (reizināt), / (dalīt), kā arī salīdzināšanas operācijas = (vienāds), <> (nav vienāds), < (mazāks), <= (mazāks vai vienāds), > (lielāks), >= (lielāks vai vienāds). Vaicājumu formulējumos izmantosim arī tādus dabīgās valodas vārdus kā UN, VAI, EKSISTĒ, NEEKSISTĒ, kā arī saikli KUR (attiecīgajos locījumos), arī šie vārdi vēlāk pārceļos uz formālo valodu, kur tie tiks lietoti līdzīgā nozīmē. Zemāk vaicājumi dabīgajā valodā būs formulēti melnas krāsas fontā, bet to pieraksts formālajā vaicājumu valodā - zilas krāsas fontā. Jūs redzēsiet, ka pat nesākuši lasīt formālās vaicājumu valodas aprakstu, jūs šos formālās valodas vaicājumus vairumā gadījumu sapratīsiet, jo mūsu piedāvātā formālā vaicājumu valoda ir tuva dabīgajai valodai (un tā ir galvenā piedāvātās formālās vaicājumu valodas pamatideja!).

SASKAITĪT Pacientus, KURIEM dzimums ir sieviete UN ģimenesĀrsta vārds ir Māris

SKAITS Pacienti, KURIEM dzimums =sieviete UN ģimenesĀrsts.vārds =Māris

(Pacienti, KURIEM dzimums = sieviete UN ģimenesĀrsts.vārds =Māris).SKAITS

SASKAITĪT Pacientus, KURIEM dzimums ir sieviete, ģimenesĀrsta vārds ir Māris, ģimenesĀrsta uzvārds ir Ozoliņš UN KURI ir pabijuši slimnīcā un šajā reizē ir izārstējušies, bet koplzmaksas šai epizodei ir bijušas lielākas par 500.00, t.i. tiem EKSISTĒ SlimnīcasEizode, KURAI izrakstīšanasIemesls ir vesels UN koplzmaksas>500.00

SKAITS Pacienti, KURIEM dzimums=sieviete UN ģimenesĀrsts.vārds=Māris UN ģimenesĀrsts.uzvārds=Ozoliņš UN EKSISTĒ SlimnīcasEizode, KURAI izrakstīšanasIemesls=vesels UN koplzmaksas>500.00

SASKAITĪT Pacientus, KURI ir pabijuši slimnīcā vismaz 2 reizes ar mazāku par 30 dienām intervālu un abas reizes izrakstīšanas iemesls ir bijis vesels, t.i. SASKAITĪT Pacientus, KURIEM EKSISTĒ SlimnīcasEpizode e1 UN EKSISTĒ SlimnīcasEpizode e2, KURĀM e1<>e2 UN e1.izrakstīšanasIemesls=vesels UN e2.izrakstīšanasIemesls=vesels UN e1.izrakstīsanasLaiks < e2.uzņemšanasLaiks UN e2.uzņemšanasLaiks – e1.izrakstīšanasLaiks < 30 dienām

SKAITS Pacienti, KURIEM EKSISTĒ SlimnīcasEpizode e1 UN EKSISTĒ SlimnīcasEpizode e2, KURĀM e1<>e2 UN e1.izrakstīšanasIemesls=vesels UN e2.izrakstīšanasIemesls=vesels UN e1.izrakstīsanasLaiks < e2.uzņemšanasLaiks UN e2.uzņemšanasLaiks – e1.izrakstīšanasLaiks < 30d

SASKAITĪT SlimnīcasEpizodes, kurās pacients ir izārstēts, t.i. KURĀM izrakstīšanasIemesls= vesels, priekš Pacientiem vīriešiem, t.i. Pacientiem, KURIEM dzimums=vīrietis

SKAITS SlimnīcasEpizodes, KURĀM izrakstīšanasIemesls=vesels UN Pacients.dzimums=vīrietis

(Pacienti, KURIEM dzimums=vīrietis).(SlimnīcasEpizodes, KURĀM izrakstīšanasIemesls=vesels).SKAITS ??????vai vajadzīgs

Atrast koplzmaksu MAX vērtību, aplūkojot SlimnīcasEpizodes, KURĀM izrakstīšanasIemesls = vesels, priekš Pacientiem, KURIEM dzimums=vīrietis

(Pacienti, KURIEM dzimums=vīrietis).(SlimnīcasEpizodes, KURĀM izrakstīšanasIemesls=vesels).koplzmaksas.MAX ??????vai šādi

Atrast kopIzmaksu SUMMU, aplūkojot SlimnīcasEpizodes, KURĀM izrakstīšanasIemesls = vesels, priekš Pacientiem, KURIEM dzimums=vīrietis

(Pacienti, KURIEM dzimums=vīrietis).(SlimnīcasEpizodes, KURĀM izrakstīšanasIemesls=vesels).kopIzmaksas.SUM ??????vai šādi

Atrast kopIzmaksu SUMMU, aplūkojot SlimnīcasEpizodes, KURĀM, pirmkārt, izrakstīšanasIemesls= vesels UN, otrkārt, pacienta vecums epizodes sākuma brīdī ir mazāks par 1 gadu, priekš Pacientiem, KURIEM dzimums=vīrietis

(Pacienti p KURIEM dzimums=vīrietis).(SlimnīcasEizodes KURĀM izrakstīšanasIemesls=vesels UN uzņemšanasLaiks - p.dzimšanasDatums<1g).kopIzmaksas.SUM ??????vai šādi

Atrast biežāko UzņemšanasDiagnozes kodu Pacientiem, KURI SlimnīcasEpizodes sākuma brīdī jaunāki par 1 gadu

(Pacienti p).(SlimnīcasEizodes KURĀM uzņemšanasLaiks-p.dzimšanasDatums<1g).UzņemšanasDiagnoze.diagnoze.kods.BIEŽ ??????vai šādi

3.Daži palīgjēdzieni celā uz formālo vaicājumu valodu

Kā jau iepriekš bija minēts, datu ontoloģija sastāv no pamatklasēm un klasifikatoru klasēm. Vaicājumi galvenokārt attieksies uz pamatklasēm. Taču, lai definētu vaicājumu valodu, ir nepieciešami vairāki palīgjēdzieni.

Pieņemsim, ka esam izvēlējušies klasi SlimnīcasEpizode. Tad Pacientu klasi mēs sauksim par šīs SlimnīcasEpizodes **vecāku** klasi. Līdzīgi, ja mēs esam izvēlējušies klasi Kustība, tad tai būs divas **vecāku** klases – SlimnīcasEpizode un Pacients, kur SlimnīcasEpizode būs pirmā līmeņa vecāks, Pacients – otrā līmeņa vecāks.

Ja esam izvēlējušies vienu Kustības instanci x, tad tai atbildīs tieši viena SlimnīcasEizodes instance, kuru apzīmēsim: x.SlimnīcasEpizode. Tālāk, šai x.SlimnīcasEpizodei atbildīs tieši viena Pacienta instance, kuru apzīmēsim x.SlimnīcasEpizode.Pacients, jeb, īsāk, x.Pacients .

Nākamais jaunais jēdziens ir izvēlētās klases **bērnu** klases. Piemēram, ja mēs esam izvēlējušies klasi SlimEpizode, tad tās bērnu klases būs:
UzņemšanasDiagnoze, IzrakstīšanasDiagnoze, Kustība, kā arī Manipulācija (klase Manipulācija ir arī klases Kustība bērnu klase). Pirmās 3 klases sauksim par pirmā līmeņa bērniem, klasi Manipulācija – par otrā līmeņa bērnu.

Savukārt, klase SlimnīcasEpizode, kā arī visas viņas bērnu klases, ir arī klases Pacients bērnu klases. Ņemot vērā ontoloģijas kokveida formu, Pacients ir visu bērnu klašu "patriarhs".

Ja esam izvēlējušies vienu SlimnīcasEpizodes instanci x, tad tai atbildīs *vairāka*s Kustību instances (uz to, ka ir *vairākas* Kustību instances, norāda mazā zvaigznīte saites galā) , šīs instances sauksim par instances x bērniem, šo instanču kopu apzīmēsim: x.Kustības .
Šāda veida apzīmējumu lietosim ne tikai priekš tiešajiem bērniem, bet arī priekš bērnu bērniem utt. Piemēram, ar x.Kustības.Manipulācijas, jeb, īsāk, x.Manipulācijas apzīmēsim visu to Manipulāciju instanču kopu, kuras var sasniegt no SlimnīcasEpizodes instances x, ejot caur attiecīgajām Kustību instancēm no kopas x.Kustibas. Viena noruna: šajos apzīmējumos

uzskatāmības labad bērnu un bērnu klašu vārdus parasti rakstīsim daudzskaitlī, lai uzsvērtu, ka attiecīgais apzīmējums nozīmē nevis vienu klases instanci (kā tas x.Pacients gadījumā), bet gan instanču kopu, kā tas ir, piemēram, x.Kustības gadījumā (no datoriskās realizācijas viedokļa galotnei nav nozīmes, datoriskā realizācija izmantos tikai vārda sakni, vārdu galotnes ir svarīgas cilvēkam – lasītājam).

Precīzākai vaicājumu valodas semantikas izpratnei vajadzīgi vēl daži apzīmējumi (pirmajā lasīšanas reizē tos var izlaist). Pieņemsim, ka ar x ir apzīmēta SlimnīcasEpizodes konkrēta instance. Iepriekš bija izskaidroti apzīmējumi tipa x.Pacients, kur Pacients ir SlimnīcasEpizodes vecāks, un x.Kustības, kur Kustība ir SlimnīcasEpizodes bērns. Taču sarežģītāku vaicājumu gadījumā svarīgi būs arī "brāļi" un "brālēni". Konkrēti, kā saprast apzīmējumu x.SlimnīcasEpizodes, kur pats x jau ir SlimnīcasEpizodes instance? Vienosimies par šādu izpratni: x.SlimnīcasEpizodes = y.SlimnīcasEpizodes, kur y=x.Pacients, t.i. tuvākais vecāks. Līdzīgi, ar x.PoliklīnikasEpizodes apzīmēsim y.PoliklīnikasEpizodes, kur y=x.Pacients. Līdzīgi, ja ar x' ir apzīmēta Kustības konkrēta instance, tad x'.Kustības = (x.SlimnīcasEpizode).Kustības, x.PoliklīnikasEpizodes = (x.SlimnīcasEizode.Pacients).PoliklīnikasEizodes. Citiem vārdiem sakot, mēs "atkāpjamies" līdz tuvākajam vecākam, no kura var nokļūt pie attiecīgā "brālēna".

Rezultātā jebkuras klases (apzīmēsim to ar Aclass) instancei x un jebkurai klasei Bclass (tai skaitā Bclass=Aclass) ir definēta apzīmējuma x.Bclass nozīme (semantika). Šeit iet runa par ontoloģijas pamatklasēm. Pie reizes norunāsim arī, ko saprast ar apzīmējumu x.Aclass, ja x ir klasifikātoru klases Kclass konkrēta instance, bet Aclass ir ontoloģijas pamatklase: tās ir visas tās Aclass instances, no kurām var nokļūt līdz Kclass instancei x, izmantojot gan vecākus, gan bērnus, gan atribūtus ar tipu Kclass. (Vēl palikusī kombinācija x.Kclass, kad x – klasifikātoru klases instance vai pamatklases instance, mums nebūs vajadzīga).

4. Skalārās izteiksmes

Zemāk definētās skalārās izteiksmes būs galvenie "darba instrumenti" mūsu vaicājumu valodā. Pārējā daļa mūsu vaicājumu valodā būs kontrolētajā dabīgajā valodā, tās semantiku būs viegli uzminēt. Zemāk definētās izteiksmes šajos kontrolētās dabīgās valodas vaicājumos zināmā nozīmē parādīsies kā "svešvārdi", kuriem ir sava speciāli definēta sintakse un semantika. Definēt šo sintaksi un semantiku – tas ir šīs sadaļas uzdevums. Formālie jēdzieni būs attēloti sarkanā krāsā, tiem sekos precīzs skaidrojums, dažos gadījumos tas var būt grūti uztverams. Tālāk sekos piemēri zilā fontā, tie ir galvenie no izpratnes viedokļa – ja tos jūs sapratīsiet, variet pārāk neiedziļināties formālajos skaidrojumos.

4.1. Atribūtizteiksmes

Pieņemsim, ka Aclass – patvaļīga mūsu ontoloģijas klase (pamatklase vai klasifikatoru klase). Tad:

1. Ja a ir Aclass atribūts, tad **a** ir atribūtizteiksme, piemēram, SlimnīcasEizodes gadījumā:
   nosūtītājsĀrsts, uzņemšanasLaiks, kopIzmaksas,
   KĀrsts gadījumā: **personasKods, vārds, uzvārds**
2. Ja a ir Aclass atribūts, kura tips ir klasifikatoru klase Kclass (tā tas var būt tikai pamatklases gadījumā), un k ir KClass atribūts, tad **a.k** ir atribūtizteiksme,

piemēram, SlimnīcasEpizodes gadījumā:
nosūtītājsĀrsts.uzvārds

3. Ja w ir atribūtizteiksme un tās tips ir String, tad atribūtizteiksme būs arī **w.SUBSTRING(i,j),** kur i, j – naturāli skaitļi, i<=j, un SUBSTRING – operācijas vārds, kas no simbolu virknes izgriež fragmentu no i-tā līdz j-tajam simbolam. Piemēram, klases Persona gadījumā:
personasKods.SUBSTRING(1,6) – paņems no personasKoda pirmos 6 simbolus, kas kodē personas dzimšanas datumu, piemēram, 230157,
klases UzņemšanasDiagnoze gadījumā:
diagnoze.kods.SUBSTRING(1,2) - paņem no diagnozes koda pirmos 2 simbolus, kas parasti kodē slimības grupu,
klases KDiagnoze gadījumā:
kods.SUBSTRING(1,2) – paņem no diagnozes koda pirmos 2 simbolus;
4. Ja w ir atribūtizteiksme un tās tips ir Date, tad atribūtizteiksmes būs arī **w.YEAR**, **w.MONTH, w.DAY**, kur YEAR, MONTH, DAY – operāciju vārdi, kas no datuma izgriež, attiecīgi, gadu, mēnesi un dienu, šo operāciju vērtības ir Integer tipa. Piemēram, klases Persona gadījumā:
dzimšanasDatums.YEAR, dzimšanasDatums.MONTH, dzimšanasDatums.DAY
- ja dzimšanasDatums=2012.06.25, tad dzimšanasDatums.YEAR=2012, dzimšanasDatums.MONTH=6, dzimšanasDatums.DAY=25;
5. Ja w ir atribūtizteiksme un tās tips ir DateTime, tad atribūtizteiksmes būs arī **w.YEAR**, **w.MONTH, w.DAY, w.HOUR, w.MINUTE, w.SECOND**, **w.DATE**, kur YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, DATE – attiecīgo operāciju vārdi. Piemēram, klases SlimnīcasEpizode gadījumā:
uzņemšanasLaiks.YEAR, uzņemšanasLaiks.MONTH, uzņemšanasLaiks.DAY, uzņemšanasLaiks.HOUR, uzņemšanasLaiks.MINUTE, uzņemšanasLaiks.SECOND, uzņemšanasLaiks.DATE
- ja uzņemšanasLaiks=2015.02.25 18:30:00, tad uzņemšanasLaiks.YEAR=2015, uzņemšanasLaiks.MONTH=2, uzņemšanasLaiks.DAY=25, uzņemšanasLaiks.HOUR=18, uzņemšanasLaiks.Minute=30, uzņemšanasLaiks.SECOND=0, bet uzņemšanasLaiks.DATE=2015.02.25;
pirmo 6 operāciju vērtības ir Integer tipa, pēdējās (t.i. DATE) vērtība ir Date tipa;
6. Ja w ir atribūtizteiksme un tās tips ir Duration, tad atribūtizteiksmes būs arī **w.YEARS**, **w.MONTHS, w.DAYS, w.HOURS, w.MINUTES, w.SECONDS**, kur YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS – attiecīgo operāciju vārdi, kas laika intervālu w pārvērš norādītajās vienībās. Piemēram, klases AmbulatorāEpizode gadījumā:
vizītesIlgums.DAYS, vizītesIlgums.HOURS, vizītesIlgums.MINUTES.
- ja vizītesIlgums=2st15min, tad vizītesIlgums.DAYS=0.09, vizītesIlgums.HOURS=2.25, vizītesIlgums.MINUTES=135, šo operāciju vērtības ir Decimal tipa;
7. Ja w1 un w2 ir atribūtizteiksmes vai konstantes ar tipiem Ineger vai Decimal, tad **(w1+w2), (w1-w2), (w1*w2), (w1/w2)** arī ir atribūtizteiksmes ar tipiem Integer vai Decimal. Piemēram, klases KustībasManipulācija gadījumā:
(manipulācija.cena*1.25)
8. Ja w1 un w2 ir atribūtizteiksmes ar tipiem Date vai DateTime, tad **(w1-w2)** arī ir atribūtizteiksme, tās tips ir Duration. Piemēram klases SlimnīcasEpizode gadījumā:
(izrakstīšanasLaiks-uzņemšanasLaiks),
(izrakstīšanasLaiks.DATE-uzņemšanasLaiks.DATE);

šīs izteiksmes var turpināt, piemēram,
(izrakstīšanasLaiks.DATE-uzņemšanasLaiks.DATE).DAYS.

4.2. x-skalārās izteiksmes

Pieņemsim, ka Aclass – patvaļīga mūsu ontoloģijas klase (pamatklase vai klasifikatoru klase). Apzīmēsim ar x patvaļīgu šīs klases instanci. (No formālā viedokļa tas nozīmē, ka Aclass ir mainīgais, kas par vērtībām pieņem dotās ontoloģijas pamatklases, bet x ir mainīgais, kas par vērtībām pieņem Aclass instances). Lai definētu x-skalāro izteiksmi (t.i. izteiksmi, uzbūvētu uz instances x bāzes), mums būs vajadzīgs vēl viens jēdziens - "Bclass y selekcijas izteiksme atkarīga no x", kur Bclass – patvaļīga ontoloģijas pamatklase, y – patvaļīga tās instance. Precīza šī jēdziena definīcija tiks dota 6.sadaļā, šeit to paskaidrosim ar piemēru palīdzību. Pieņemsim, ka Aclass ir Pacientu klase, kur x – Pacienta instance, un Bclass ir SlimnīcasEpizodu klase, kur y – SlmnīcasEpizodes instance. Dažas tipiskas "SlimnīasEpizodes y selekcijas izteiksmes atkarīgas no x":
SlimnīcasEpizodes y KURĀM izrakstīšanasIemesls=vesels UN kopizmaksas>500.00
– šīs izteiksmes vērtība būs visu to SlimnīcasEpizodu innstanču kopa, kurām izpildās norādītais nosacījums, y apzīmē patvaļīgu šīs kopas elementu (šajā piemērā vēl neparādās atkarība no x),
SlimnīcasEpizodes y KURĀM izrakstīšanasIemesls=vesels UN kopizmaksas>500.00 UN x.dzimums=sieviete,
SlimnīcasEpizodes y KURĀM izrakstīšanasIemesls=vesels UN kopizmaksas>500.00 UN x.dzimums=sieviete UN (uzņemšanasLaiks-x.dzimšanasDatums<=1g)
- pēdējos 2 piemēros parādās atkarība no x.

Pārejam pie x-skalārās izteiksmes definīcijas:

1. Ja a ir Aclass atribūtizteiksme, tad **x.a** ir x-skalārā izteiksme, tās vērtība ir atribūtizteiksmes a vērtība uz instances x.
2. Ja Bclass ir ontoloģijas pamatklase, tad x-skalārās izteiksmes ir **x.Bclass** un **x.("Bclass y selekcijas izteiksme atkarīga no x")**, kur Bclass ir ontoloģijas pamatklase. Šo izteiksmju vērtības ir atkarīgas no Aclass un Bclass savstarpējām attiecībām:
   a) ja Bclass ir Aclass vecāks, tad x.Bclass vērtība ir Bclass instance, kuras bērns ir x (tāda ir viena vienīga); ja Bclass vietā ir "Bclass y selekcijas izteiksme atkarīga no x", tad attiecīgās izteiksmes vērtība ir vai nu iepriekš minētā viena vienīgā Bclass instance, vai arī tukša kopa, ja selekcijas nosacījums vēl tālāk sašaurina pieļaujamās Bclass instances; iepriekš minēto vienu vienīgo instanci (vienveidības dēļ ar nākamo gadījumu) mēs arī uztversim kā kopu, sastāvošu no vienas vienīgas instances;
   b) ja Bclass ir Aclass bērns vai brālis (ja Bclass ir vienāda ar Aclass, tad arī mēs sakām, ka Bclass ir Aclass brālis), tad x.Bclass vērtība ir visu to Bclass instanču kopa, kuras ir sasniedzamas no instances x; ja Bclass vietā ir "Bclass y selekcijas izteiksme atkarīga no x", tad attiecīgās izteiksmes vērtība ir tās x.Bclass instances, kuras papildus vēl apmierina doto selekcijas nosacījumu;
3. Ja b ir Bclass atribūtizteiksme, tad x-skalārās izteiksmes ir **x.Bclass.b** un **x.("Bclass y selekcijas izteiksme atkarīga no x").b**. Šo izteiksmju vērtību definēsim sekojoši: pieņemsim, ka x.Bclass (attiecīgi, x.("Bclass y selekcijas

izteiksme atkarīga no x") vērtība ir Bclass instanču kopa {B1, B2, …, Bn}. Tad x.Bclass.b (attiecīgi, x.("Bclass y selekcijas izteiksme atkarīga no x").b) vērtība ir multikopa {B1.b, B2.b, …, Bn.b}, kur Bi.b ir atribūtizteiksmes b vērtība uz instances Bi (multikopa nozīmē, ka viena un tā pati vērtība var atkārtoties vairākas reizes). Pasvītrosim, ka **x.("Bclass y selekcijas izteiksme atkarīga no x").b** ir ļoti svarīgs un plaši lietojams izteiksmes gadījums tabulu būvē, kad selekcijas nosacījums ir tāds, ka tā rezultāts ir kopa sastāvoša no viena vienīga elementa (vai nill), ko var attēlot tabulas vienā šūnā. Piemēram, ja x ir SlimnīcasEizodes instance, tad izteiksmes **x.(UzņemšanasDiagnoze KURAI npk=1).diagnoze.kods, x.(Kustība KURAI npk=3).nodaļa** raksturosies ar to, ka tām būs viena vienīga vērtība (vai nill, ja Kustiba ar npk=3 neeksistē);

4. Izteiksmes **x.Bclass.SKAITS** un **x.("Bclass y selekcijas izteiksme atkarīga no x").SKAITS** ir x-skalārās izteiksmes, to vērtība ir elementu skaits attiecīgajās kopās.
5. Ja atribūtizteiksme b ir ar tipu, Integer, Decimal, Duration, tad izteiksmes **[x.Bclass.b / x.("Bclass y selekcijas izteiksme atkarīga no x")].[SUM / MAX / MIN / VID]** ir x-skalāra izteiksmes, to semantika ir acīmredzama.
6. Ja atribūtizteiksme b ir ar tipu String, tad izteiksmes **[x.Bclass.b / x.("Bclass y selekcijas izteiksme atkarīga no x")].BIEŽ** ir x-skalāras izteiksmes, to semantika arī ir acīmredzama – attiecīgajā multikopā biežāk sastopamā vērtība (ja tādas ir vairākas, tad nejauši tiek izvēlēta viena no tām). Tipisks lietojums, kad x ir klases Pacients instance: x.(UzņemšanasDiagnoze KURAI npk=1).diagnoze.kods.BIEŽ (Šādai izteiksmei ir saturīga jēga priekš Pacientiem, kuri ir daudzas reizes pabijuši slimnīcā, t.i. tiem ir daudzas SlimnīcasEpizodes).
7. Izteiksmes, kas sastādītas no x-skalārajām izteiksmēm un konstantēm, lietojot tipiem atļautās operācijas (Integer un Decimal tās ir operācijas +, -, *, /; Date un DateTime tā ir operācija – (mīnus)), arī ir x-skalārās izteiksmes. Piemēram, kad x ir klases Pacients instance:
   x.SlimnīcasEpizode.kopizmaksas.SUM +
   x.PoliklīnikasEpizode.vizītesIzmaksas.SUM ,
   (x.SlimnīcasEpizode.kopizmaksas.SUM +
   x.PoliklīnikasEpizode.vizītesIzmaksas.SUM)*1.25 ,
   x.KustībasManipulācija.manipulācija.cena.VID
8. Ja Cclass ir ontoloģijas pamatklase, tad izteiksmes **x.[Bclass / "Bclass y selekcijas izteiksme atkarīga no x"]. [Cclass / "Cclass z selekcijas izteiksme atkarīga no x,y"]** arī ir x-skalāras izteiksmes. Ja nepieciešams, šo hierarhiju var turpināt, piemēram, var ņemt Dclass un aplūkot x.("Bclas y selekcijas izteiksme atkarīga no x").(Cclass z selekcijas izteiksme atkarīga no x,y").(Dclass v selekcijas izteiksme atkarīga no x, y, z") – arī tā būs x-skalāra izteiksme. Paskaidrosim šāda tipa izteiksmju vērtību uz piemēra **x.("Bclass y selekcijas izteiksme atkarīga no x").("Cclass z selekcijas izteiksme atkarīga no x, y")** - šīs izteiksmes vērtība ir Cclass instanču kopa {C1, C2, …, Ck}=B1.("Cclass z selekcijas izteiksme atkarīga no x, y") + B2. ("Cclass z selekcijas izteiksme atkarīga no x, y") + … + Bn.("Cclass z selekcijas izteiksme atkarīga no x, y"), kur "+" nozīmē kopu apvienošanu un { B1, B2, …,Bn} ir izteiksmes x.("Bclass y selekcijas izteiksme atkarīga no x") vērtība – attiecīga Bclass instanču kopa. Tālāk, var lietot operāciju SKAITS, piemēram, **x.("bclass y**

selekcijas izteiksme atkarīga no x").("Cclass z selekcijas izteiksme atkarīga no x").SKAITS. Tālāk, ja c ir Cclass atribūtizteiksme, tad, līdzīgi kā 6. un 7. puktos, var lietot operācijas SUM, MAX, MIN, BIEŽ, piemēram, **x.("Bclass y selekcijas izteiksme atkarīga no x").("Cclass z selekcijas izteiksme atkarīga no x, y").c.SUM .** Piemēri, kad x ir klases Pacients instance: **x.(SlimnīcasEpizodes y KURĀM y.uzņemšanasLaiks.DATE>=2015.01.01 UN y.izrakstīšanasLaiks.DATE<=2015.03.31).(Kustības z KURĀM z.nodaļa="12-05").(KustībasManipulācijas v KURĀM v.beiguLaiks-v.sākumaLaiks>30min) .manipulācija.cena.SUM**

Nobeigumā vēl daži x-skalāro izteiksmju piemēri, kad x ir klases SlimnīcasEpizode instance:
x.kopizmaksas, x.nosūtītājsĀrsts, x.nosūtītājsĀrsts.personasKods, x.nosūtītājsĀrsts.personasKods.SUBSTRING(1,6), x.nosūtītājsĀrsts.personasKods.SUBSTRING(5,6), x.uzņemšanasLaiks.DATE, x.uzņemšanasLaiks.HOUR, (x.izrakstīšanasLaiks-x.uzņemšanasLaiks).DAYS, x.Pacients.ģimenesĀrsts.uzvārds

Vēl viens jēdziens būs vajadzīgs. Pieņemsim, ka t, u, ... attiecīgi klašu Tclass, Uclass , ... instances. **"x-skalāras izteiksmes atkarīgas t, u, ..."** - tās ir x-skalārās izteiksmes, t-skalārās izteiksmes, u-skalārās izteiksmes, kā arī izteiksmes, izveidotas no iepriekš minētajām skalārajām izteiksmēm, lietojot tipiem atļautās operācijas.

4.3. Tālāk –kontrolētā dabīgā valoda.

Kā jau iepriekš bija minēts, skalārās izteiksmes – galvenais formālais jēdziens (kopā ar atbilstošo formālo notāciju), uz kura balstās mūsu vaicājumu valoda. Šis formālisms ir "jāiemācās no galvas". Tālākā vaicājumu valodas daļa jau vairāk vai mazāk atbildīs dabīgajai valodai (ieskaitot tās sintaksi un semantiku). Pilnības labad mēs tomēr dosim precīzu sintaksi un semantiku arī šai vaicājumu valodas daļai, bet to būs ievērojami vieglāk uztvert un atcerēties pateicoties līdzībai ar dabīgo valodu.

5. Loģiskās izteiksmes

Līdzīgi kā iepriekš, pieņemsim, ka Aclass – patvaļīga mūsu ontoloģijas klase. Apzīmēsim ar x patvaļīgu šīs klases instanci. Atzīmēsim, ka x vietā varēja izvēlēties jebkuru citu vārdu, piemēram, y vai y1, vienīgā prasība ir, lai šis vārds nesakristu ar ontoloģijas klašu vai atribūtu vārdiem, kā arī vaicājumu valodas priekšdefinētajiem vārdiem, kurus parasti mēs rakstīsim ar lielajiem burtiem.

5.1. x-loģiskie elementi

Tās ir izteiksmes, kas iegūtas no x-skalārajām izteiksmēm un konstantēm, savienojot tās ar salīdzināšanas operācijas zīmēm:

= ("vienāds"), <> ("ņevienāds"), < ("mazāks"), <= ("mazāks vai vienāds"), > ("ļielāks"), >= ("lielāks vai vienāds")

Pirmās 2 salīdzināšanas operācijas ir lietojamas jebkurām x-skalārajām izteiksmēm vai konstantēm, bet pārējās 4 salīdzināšanas operācijas ir lietojamas tikai Integer, Decimal, Date, DateTime, Duration, kā arī String izteiksmēm (String gadījumā šīs salīdzināšanas operācijas tiek domātas attiecībā pret alfabētisko sakārtojumu).

String gadījumā ir paredzēta vēl tā saucamā ietveres salīdzināšanas operācija ~, paskaidrosim to ar piemēra palīdzību: Pieņemsim, ka t1="abcdef" un t2="cd", tad apgalvojums t1~t2 pie šīm vērtībām būs patiess, jo t1 vērtība ietver sevī kā apakšvirkni t2 vērtību. Ja par t2 vērtību mēs būtu paņēmuši, piemēram, "dcc" , tad dotais apgalvojums būtu aplams. Īpaša loma šajā notācijā ir % zīmei: ja t2 vērtība sākas ar % zīmi, piemēram, t2=%"def", tad t1~t2 ir patiess tikai tad, ja t1 vērtība beidzas ar virknīti "def" (kā tas ir dotajā gadījumā). Līdzīgi, ja t2="ab"%, tad t1~t2 ir patiess tikai tad, ja t1 vērtība sākas ar virknīti "ab".

x-loģisko elementu piemēri, kad x ir klases SlimnīcasEpizode instance:
x.kopIzmaksas>=100,
x.nosūtītājsĀrsts=x.Pacients.ģimenesĀrsts,
x.kopIzmaksas>x.VeiktāsManipulacijas.manipulācija.cena.SUM ,
x.kopIzmaksas=x.VeiktāsManipulacijas.manipulacija.cena.SUM + (x.izrakstīsanasLaiks-x.uzņemšanasLaiks).DAYS *12
"y-loģisko elementu piemēri, kad y ir klases Pacients instance:
y.uzvārds=Priede, y.dzimums=sieviete, y.ģimenesĀrsts.uzvārds=Ozoliņš, y.dzimšanasDatums.MONTH=12, y.SlimnīcasEpizodes.SKAITS=5, y.AmbulatorāsEpizodes.SKAITS=0

Vēl viens jēdziens būs vajadzīgs. Pieņemsim, ka t, u, ... attiecīgi klašu Tclass, Uclass , ... instances. **"x-loģiskie elementi atkarīgi t, u, ..."** - tās ir izteiksmes, iegūtas no x-skalārajiem elementiem atkarīgiem no t, u, .. , lietojot iepriekš minētās salīdzināšanas operācijas.

5.2. x-loģiskās izteiksmes

Tie ir x-loģiskie elementi vai izteiksmes, kas izveidotas no x-loģiskajiem elementiem, lietojot loģisko attiecību (operāciju) vārdus UN, VAI (nepieciešamības gadījumā lietojot arī iekavas).

x-loģisko izteiksmju piemēri, kad x ir klases SlimnīcasEpizode instance:

x.kopIzmaksas>=100 UN x.Kustības.SKAITS> 5 UN (x.atbildīgaisĀrsts.uzvārds=Osis VAI x.atbildīgaisĀrsts.uzvārds=Liepa) UN x.Pacients.dzimums=sieviete

Viena piezīme par pieraksta racionalizāciju, kad attiecības kreisās puses izteiksmes ir vienādas:

x.atbildīgaisĀrsts.uvārds =Osis VAI x.atbildīgaisĀrsts=Liepa
vietā varēja rakstīt īsāk:
x.atbildīgaisĀrsts.uzvārds=Osis VAI =Liepa

Līdzīgi, izteiksmes
x.Pacients.dzimšanasDatums>2012.01.01 UN x.Pacients.dzimšanasDatums<=2013.12.31
vietā var rakstīt
x.Pacients.dzimšanasDatums>2012.01.01 UN <=2013.12.31

Vēl piemērs:

x.kopIzmaksas>=100 UN x.kopIzmaksas<200 UN (x.(UzņemšanasDiagnoze d KURAI d.npk=1).diagnoze.kods="K63.8" VAI (x.(UzņemšanasDiagnoze d KURAI d.npk=1).diagnoze.kods="K59")
vai īsāk:

x.kopIzmaksas>=100 UN <200 UN (x.(UzņemšanasDiagnoze d KURAI d.npk=1).diagnoze.kods="K63.8" VAI ="K59")

x-loģisko iztiksmju piemēri, kad x ir klases Pacients instance:

x.SlimnīcasEpizodes.SKAITS=5 UN x.PoliklīnikasEpizodes.SKAITS=0,
x.dzimšanasDatums.YEAR>2010 UN x.(SlimnīcasEpizodes y KURĀM y.izrakstīšanasIemesls=vesels).kopizmaksas.SUM +
x.PoliklīnikasEpizodes.vizītesIzmaksas.SUM >1000

Kas ir x-loģiskās izteikmes **vērtība** pie fiksētas x vērtības **xi** ? Tā ir tā saucamā loģiskā vērtība "true" vai "false" , jeb, vienkāršākiem vārdiem runājot, pie fiksētas x vērtības dotā loģiskā izteiksme vai nu "izpildās", vai "neizpildās ".

Vēl viens jēdziens būs vajadzīgs. Pieņemsim, ka t, u, … attiecīgi klašu Tclass, Uclass , … instances. **"x-loģiskās izteiksmes atkarīgas t, u, …"** - tās ir izteiksmes, iegūtas no "x-loģiskajiem elementiem atkarīgiem no t, u, … , lietojot iepriekš minētās loģiskās operācijas UN, VAI.

6. Selekcijas izteiksmes

Vispirms viena vispārīga piezīme: selekcijas izteiksmes definīcijā mēs pastarpināti izmantosim skalāras izteiksmes jēdzienu, kas savukārt izmanto selekcijas izteiksmes jēdzienu. Tā nav nekāda pretruna, šāda tipa rekursīvas definīcijas, kā likums, nākas izmantot formālo valodu sintakses aprakstos, piemēram, <vārds>::=<burts> I <burts><vārds>.

Tagad ieviesīsim pašu svarīgāko izteiksmes veidu

"Aclass selekcijas izteiksme"

Lai precīzi definētu šo jēdzienu, ir jāievieš Aclass patvaļīgas instances apzīmējums, piemēram, x (no šī apzīmējuma nebūs atkarīga izteiksmes vērtība, tam ir tikai tīri tehniska nozīme). Tātad pieņemsim, ka Aclass patvaļīgo instanci esam apzīmējuši ar x.

6.1. Vienkāršākais gadījums

Sāksim ar vienkāršāko gadījumu – definēsim

"Aclass x vienkāršu selekcijas izteiksmi"

ar šādas sintakses palīdzību:

Aclass x KURĀM IZPILDĀS "x-loģiskā izteiksme"

jeb, pierakstot nedaudz īsāk:

Aclass x KURĀM "x-loģiskā izteiksme"

Atgādināsim, ka priekšdefinēto vārdu pasvītrotā daļa, ir tā, kas jāraksta obligāti, līdz ar to vārdi KURĀM, KURIEM ir līdzvērtīgi.

Ko nozīmē šādi pierakstīta selekcijas izteiksme? Tās jēgu (semantiku) jau ir viegli uzminēt, skatoties uz tās sākotnējo (nesaīsināto) pierakstu kā uz dabīgās valodas teikumu. Tā atlasa

visas tās Aclass instances x, kurām izpildās dotā "x-loģiskā izteiksme". Šo atlasīto instanču sarakstu (sakārtotu kopu)

Instance x1, Instance x2, … , Instance xn

mēs turpmāk sauksim par dotās selekcijas izteiksmes vērtību.

Selekcijas izteiksmju piemēri:

SlimībasEpizodes x KURĀM x.kopIzmaksas>=100 UN x.Kustibas.SKAITS> 5 UN (x.atbildīgaisĀrsts.uzvārds=Osis VAI =Liepa) UN x.Pacients.dzimums=sieviete

Pacienti y KURIEM y.SlimībasEpizodes.SKAITS=5 UN y.PoliklīnikasEpizodes.SKAITS=0

Pacienti y KURIEM y.dzimšanasDatums=1998.12.31

Lai ieraudzītu uz datora "selekcijas izteiksmes" vērtību, mums ir jāizpilda komanda

PARĀDĪT VISAS "selekcijas izteiksme"

Piemēram, izpildot komandu

PARĀDĪT VISUS Pacientus x KURIEM x.dzimšanasDatums=1998.12.31

mēs varētu iegūt, piemēram, šādu Pacientu instanču sarakstu (šajā gadījumā sastāvošu no 2 Pacienta instancēm):
Pacients{personasKods=311298-10415, vārds=Jānis, uzvārds=Bērziņš, dzimums=vīrietis, dzimšanasDatums=1998.12.31, ģimensĀrsts=KĀrsts{personasKods=250155-12418, vārds=Ruta, uzvārds=Ose}},
Pacients{personasKods=311298-10255, vārds=Anna, uzvārds=Priede, dzimums=sieviete, dzimšanasDatums=1998.12.31, ģimensĀrsts=nill}

Tagad viena svarīga piezīme un īsrakstīšanas noruna: Ja selekcijas izteiksmē patvaļīgā pacienta apzīmējuma "x" vietā mēs būtu paņēmuši kādu citu burtu (vai pat veselu vārdu, tikai tas nedrīkst sakrist ar klašu un atribūtu vārdiem), piemēram, "p" vai "p1", tad no tā nemainītos selekcijas izteiksmes vērtība, t.i. selektēto instanču saraksts. Varam iet vēl tālāk – varam atļaut selekcijas izteiksmēs vispār nelietot patvaļīgo klašu instanču apzīmējumus x, y un tml. Tādā gadījumā tos nelietosim arī selekcijas nosacījumos; zinot šo norunu, pārpratumus tas neradīs, tas tikai vairāk atbildīs dabīgās valodas izteiksmes formai.

Iepriekšējie selekcijas izteiksmju piemēri, pārrakstīti saīsinātā formā:

SlimībasEpizodes KURĀM kopIzmaksas>=100 UN Kustibas.SKAITS> 5 UN (atbildīgaisĀrsts.uzvārds=Osis VAI =Liepa) UN Pacients.dzimums=sieviete

Pacienti KURIEM SlimībasEpizodes.SKAITS=5 UN PoliklīnikasEpizodes.SKAITS=0

Pacienti KURIEM dzimšanasDatums=1998.12.31

Tomēr būs viens gadījums, kad selekcijas izteiksmē instances apzīmējums tipa "x", "y" un tml. būs obligāti jālieto, un proti, kad mēs lietosim komandu ATLASĪT…IZVEIDOT TABULU , bet par to nedaudz vēlāk.

6.2. Vispārīgais gadījums

Iepriekšējā sadaļā ieviestā vienkāršā selekcijas izteiksme nepārklāj daudzus praksē svarīgus un dabīgajā valodā viegli pasakāmus gadījumus. Lai risinātu šo problēmu, ieviesīsim

"Aclas x paplašināto selekcijas izteiksmi"

Selekcijas izteiksmes paplašinājumu mēs definēsim ar sintaksi, kas no formāļās valodas viedokļa ir sarežģīta, bet tā atbilst tam, kā mēs formulētu šāda tipa apgalvojumus dabīgā valodā.

Šī sintakse sastāv no šādām daļām:

1) Vispirms daļa, kas atbilst jau agrāk ieviestajai vienkāršajai selekcijas izteiksmei
   Aclass x KURĀM "x-loģiskā izteiksme"
2) Tad seko loģiskais saiklis
   UN
3) Tālāk seko garš paplašinājums ar šādu sintaksi
   [EKSISTĒ /NEEKSISTĒ] Cclass y KURAI "y-loģiskā izteiksme atkarīga no x"
   UN [EKSISTĒ /NEEKSISTĒ] Dclass z KURAI "z-loģiskā izteiksme atkarīga no y, x"
   … … …

Daudzpunkti šeit nozīmē to, ka klases var turpināties pēc vajadzības, piemēram,
UN [EKSISTĒ /NEEKSISTĒ] Eclass v KURAI "v-loģiskā izteiksme atkarīga no z, y, x"
utt.

Bez komentāriem **"Aclass x paplašinātās selekcijas izteiksme** sintakse ir šāda:

Aclass KURĀM "x-loģiskā izteiksme"
UN [EKSISTĒ /NEEKSISTĒ] Cclass y KURAI "y-loģiskā izteiksme atkarīga no x"
UN [EKSISTĒ /NEEKSISTĒ] Dclass z KURAI "z-loģiskā izteiksme atkarīga no y, x"
… … …

Turpmāk šīs izteiksmes fragmentu

"x-loģiskā izteiksme"
UN [EKSISTĒ /NEEKSISTĒ] Cclass y KURAI "y-loģiskā izteiksme atkarīga no x"
UN [EKSISTĒ /NEEKSISTĒ] Dclass z KURAI "z-loģiskā izteiksme atkarīga no y, x"
… … …

sauksim par **"paplašināto x-loģisko izteiksmi"**

Rezumēsim iepriekš teikto. **Paplašinātā selekcijas izteiksme** (turpmāk saukta vienkārši par **selekcijas izteiksmi**) ir izteiksme, kas tiek definēta ar šādu sintaksi:

Aclass x KURAI IZPILDĀS "paplašinātā x-loģiskā izteiksme"

Tās vērtība ir visu to Aclass instanču saraksts
Instance x1, Instance x2, … , Instance xn,
kurām izpildās (t.i. ir patiesa) dotā "paplašinātā x-loģiskā izteiksme".

Tagad aplūkosim piemērus, kas palīdzēs saprast šī garā formālā skaidrojuma sintaksi un semantiku pateicoties tā līdzībai ar dabīgo valodu.

Piemēri:

1. Pacienti x KURIEM dzimums=vīrietis

2. Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris
3. Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris UN EKSISTĒ UzņemšanasDiagnoze KURAI diagnoze.nosaukums=Vīrusmeningīts

1.piemēra semantika ir acīmredzama – tiek atlasīti visi Pacienti, kuri ir vīrieši.

2.piemēra semantika, ja to uztver kā dabīgās valodas teikumu, arī ir viennozīmīgi saprotama – tiek atlasīti visi tie Pacienti, kuri ir vīrieši un kuri ir miruši, t.i. kuriem eksistē SlimnīcasEpizode ar izrakstīšanasIemeslu=miris.

Nedaudz smalkāka situācija ir ar 3.piemēru. Atbilstoši dotajai ontoloģijai (zīm.1), vienam Pacientam var būt vairākas SlimnīcasEpizodes, kur katrai SlimnīcasEpizodei, savukārt, var būt vairākas UzņemšanasDiagnozes. Līdz ar to 3.piemēru no dabīgās valodas viedokļa var interpretēt 2 dažādos veidos:

1.inerpretācija:

Pacienti x kuriem:
1) dzimums=vīrietis UN
2) eksistē SlimnīcasEpizode, kurai:
- 2a) izrakstīšanasIemesls=miris UN
- 2b) eksistē UzņemšanasDiagnoze, KURAI diagnoze.nosaukums=Vīrusmeningīts

2.interpretācija:

Pacienti x kuriem:
1) dzimums=vīrietis UN
2) eksistē SlimnīcasEpizode, KURAI izrakstīšanasIemesls=miris UN
3) eksistē UzņemšanasDiagnoze, KURAI diagnoze.nosaukums=Vīrusmeningīts

Šīs interpretācijas atšķiras ar to, ka 1.gadījumā mēs uzskatām, ka "Vīrusmeningīts" ir bijis tieši miršanas SlimnīcasEpizodē, bet 2.gadījumā - vienalga kurā šī Pacienta SlimnīcasEpizodē (jo vienam Pacientam var būt vairākas SlimnīcasEpizods). Acīmredzot daudz dabīgāka ir pirmā interpretācija, autoru eksperimenti ar potenciālajiem sistēmas lietotājiem parādīja, ka 2.interpretācija viņiem pat prātā nebija ienākusi. Tādēļ šāda veida izteiksmēs "KUR EKSISTĒ … KUR EKSISTĒ … KUR EKSISTĒ …" par **priekšdefinēto interpretāciju** mēs uzskatīsim 1.interpretāciju.

Taču patiesībā šī interpretāciju divdomība ir sekas mūsu nedaudz "vieglprātīgajai" EKSISTĒ/NEEKSISTĒ lietošanai, kur mēs pārāk paļāvāmies uz "veselo saprātu". Atbilstoši mūsu "veselajam saprātam", kad mēs rakstām

Pacienti x KURIEM EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris ,

mēs patiesībā domājam atlasi tikai pa tām SlimnīcasEpizodēm, kuras atbilst Pacienta instancei x (nevis atlasi pa visām SlimnīcasEpizodēm, kas ir mūsu datu bāzē). Tāpēc korektāk būtu bijis, ja iepriekšējo izteiksmi mēs būtu rakstījuši šādi:

Pacienti x KURIEM EKSISTĒ **x**.SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris ,

kur viennozīmīgi pateikts, pie "EKSISTĒ" rēķināšanas atlase tiek veikta tikai pa tām SlimnīcasEpizodes instancēm, kas pieder kopai x.SlimnīcasEpizodes.

Tagad arī kļūst skaidrs, ar ko atšķiras 3.piemēra 1.interpretācija no 2.interpretācijas. 1.interpretācijas gadījumā mēs patiesībā domājam:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ **x**.SlimnīcasEpizode y KURAI izrakstīšanasIemesls=miris UN EKSISTĒ **y**.UzņemšanasDiagnoze KURAI diagnoze.nosaukums=Vīrusmeningīts

2.interpretācijas gadījumā mēs domājam šādu situāciju:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ **x**.SlimnīcasEpizode y KURAI izrakstīšanasIemesls=miris UN EKSISTĒ **x**.UzņemšanasDiagnoze KURAI diagnoze.nosaukums=Vīrusmeningīts

Par noklusēto variantu mēs uzskatīsim 1.interpretāciju, t.i. principu, kad:

Aclass x KURAI … EKSISTĒ x.Bclass y KURAI … EKSISTĒ y.Cclass z KURAI … EKSISTĒ z.Dclass…

un šajā gadījumā attiecīgos x, y, z,… prefiksus atļausim nerakstīt. Praktiski visos saturīgos piemēros pietiek ar šo noklusēto interpretāciju.

Arī atribūtiem, kur tas nerada divdomību, instances apzīmējumu prefiksus tipa x, y,… nerakstīsim. Ja tos rakstītu, tad iepriekšējais piemērs izskatītos šādi:

Pacienti x KURIEM **x**.dzimums=vīrietis UN EKSISTĒ **x**.SlimnīcasEpizode y KURAI **y**.izrakstīšanasIemesls=miris UN EKSISTĒ **x**.UzņemšanasDiagnoze z KURAI **z**.diagnoze.nosaukums=Vīrusmeningīts

Iepriekš mēs runājām par "EKSISTĒ" gadījumu, taču viss tas pats attiecas arī uz "NEEKSISTĒ" lietojumu.

Aplūkosim vēl piemērus:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris UN EKSISTĒ Kustība KURAI nodaļa=10-05

Šajā gadījumā tiek atlasīti visi Pacienti, kuri ir miruši, t.i. tiem eksistē "miršanas" SlimnīcasEpizode, un šajā SlimnīcasEpizodē pacients ir pabijis nodaļā=10-05, t.i. šai SlimnīcasEpizodei eksistē Kustība, kurai atribūts nodaļa =10-05.

Bet pieņemsim tagad, ka mēs gribam atlasīt tos Pacientus, kas miruši tieši nodaļā=10-05 , t.i. pacientus, kuru "miršanas" SlimnīcasEpizodes pēdējās Kustibas nodaļa ir tieši 10-05. Kā to izdarīt? Jāatceras no ontoloģijas skaidrojuma, ko nozīmē atribūts "npk". Šis atribūts ir arī klasei Kustība. Ar šī atribūta palīdzību ir sanumurētas klases Kustiba instances, kas atbilst vienai tiešā vecāka (šajā gadījumā SlimnīcasEpizodes) instancei, t.i. pirmajai instancei npk=1, otrajai instancei npk=2 utt. Kā jau bija minēts pie ontoloģijas skaidrojuma, šiem numuriem ir īpaša semantika, Kustības gadījumā tie atbilst reālajai kustību secībai laikā dotās SlimEpizodes instances ietvaros. Priekš dotās SlimnīcasEpizodes instances lielāko (t.i. **pēdējo**) Kustības instances npk apzīmēsim ar "*". Šis apzīmējums ir mūsu vaicājumu valodas oficiāla sastāvdaļa. Tas dod mums iespēju ļoti vienkārši pierakstīt augstāk minēto dabīgās valodas vaicājumu:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris UN EKSISTĒ Kustība KURAI nodaļa=10-05 UN npk=*

Varēja šo pašu vaicājumu pierakstīt arī bez "*" jēdziena, tikai tas būtu sarežģītāk:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris UN EKSISTĒ Kustība k1 KURAI k1. nodala=10-05 UN NEEKSISTĒ Kustība k2 KURAI k2.npk>k1.npk

Starp citu, šis vaicājums pilnā pierakstā (bez noklusējumu izmantošanas) izskatītos šādi:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ **x**.SlimnīcasEpizode y KURAI izrakstīšanasIemesls=miris UN EKSISTĒ **y**.Kustība k1 KURAI k1. nodala=10-05 UN NEEKSISTĒ **k1**.Kustība k2 KURAI k2.npk>k1.npk

Vēl viens piemērs: kā atlasīt tos Pacientus, kas "miršanas" SlimnīcasEpizodē pabijuši nodaļā=10-05 vismaz 2 reizes:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris UN EKSISTĒ Kustība k1 KURAI EKSISTĀ Kustība k2 KURAI k1<>k2 UN k1.nodaļa=10-05 UN k2.nodaļa=10-05

Šajā gadījumā, kad divas vai vairāk reizes parādās "EKSISTĒ Kustība", priekš katras no šīm kustībām ir jāievieš savs patvaļīgās instances apzīmējums, šajā piemērā tie ir k1 un k2. Pilnā pierakstā šis piemērs izskatītos šādi:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ x.SlimnīcasEpizode y KURAI izrakstīšanasIemesls=miris UN EKSISTĒ y.Kustība k1 KURAI EKSISTĀ k1.Kustība k2 KURAI k1<>k2 UN k1.nodaļa=10-05 UN k2.nodaļa=10-05

Šo pašu atlasi varēja nodefinēt arī dabīgajai valodai nedaudz tuvākā formā:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemesls=miris UN EKSISTĒ Kustība k1 UN EKSISTĒ Kustība k2 KURĀM k1<>k2 UN k1.nodala=10-05 UN k2.nodala=10-05

Pilnajā pierakstā tas izskatītos šādi:

Pacienti x KURIEM dzimums=vīrietis UN EKSISTĒ x.SlimnīcasEpizode y KURAI izrakstīšanasIemesls=miris UN EKSISTĒ y.Kustība k1 UN EKSISTĒ y.Kustība k2 KURĀM k1<>k2 UN k1.nodala=10-05 UN k2.nodala=10-05

Šeit tiek izmantots vēl viens noklusējums:

Aclass x KURAI ... EKSISTĒ Bclass UN EKSISTĒ CClass ...

atbilst šādam pilnajam pierakstam:

Aclass x KURAI ... EKSISTĒ x.Bclass UN EKSISTĒ **x**.CClass ...

6.3.Parametrizētas selekcijas izteiksmes

Nobeigumā atzīmēsim, ka mums vāl būs vajadzīgas parametrizētas selekcijas izteiksmes, t.i.

"Aclass x selekcijas izteiksme atkarīga no t, u, ... " ,

kur t apzīmē Tclass patvaļīgu instanci, u apzīmē Uclass patvaļīgu instanci, ...
Par pamatu ņemsim iepriekš ieviesto neparametrizētas selekcijas izteiksmes sintaksi:

Aclass KURĀM “x-loģiskā izteiksme”
UN [EKSISTĒ /NEEKSISTĒ] Cclass y KURAI “y-loģiskā izteiksme atkarīga no x”
UN [EKSISTĒ /NEEKSISTĒ] Dclass z KURAI “z-loģiskā izteiksme atkarīga no y, x”
… … …

un papildināsim to ar parametriem:

Aclass KURĀM “x-loģiskā izteiksme atkarīga no **t, u,** …”
UN [EKSISTĒ /NEEKSISTĒ] Cclass y KURAI “y-loģiskā izteiksme atkarīga no x, **t, u,** …”
UN [EKSISTĒ /NEEKSISTĒ] Dclass z KURAI “z-loģiskā izteiksme atkarīga no y, x, **t, u,** …”
… … …

Tā tad arī būs “Aclass x selekcijas izteiksme atkarīga no t, u, …”, tās semantika, ņemot vērā iepriekšējos skaidrojumus, ir acīmredzama.

Parametrizētās selekcijas izteiksmes tipisks lietojums (ar x šeit apzīmēta klases Pacients instance, tā tad arī būs dotās selekcijas izteiksmes parametrs):

```
x.(SlimnīcasEpizodes KURĀM
(uzņemšanasLaiks - x.dzimšanasDatums)<=1g).SKAITS
```

7. Vaicājumu valodas komandas

Tagad esam sagatavojuši nepieciešamos jēdzienus, lai pārietu pie mūsu vaicājumu valodas galvenās daļas – vaicājumu komandām. Sāksim ar jau apraksta sākumā pieminēto komandu:

PARĀDĪT [**k** / VISAS] “Aclass selekcijas izteiksme”

Piemēram:

```
PARĀDĪT 2 Pacientus, KURIEM dzimšanasDatums=1998.12.31
```

```
PARĀDĪT VISAS SlimnīcasEpizodes KURĀM kopIzmaksas>=100 UN Kustības.SKAITS> 5 UN
(atbildīgaisĀrsts.uzvārds=Osis VAI =Liepa) UN Pacients.dzimums=sieviete
```

Tas pats, tikai

PILNPARĀDĪT [**k** / VISAS] “Aclass selekcijas izteiksme”

```
PILNPARĀDĪT 2 Pacientus, KURIEM dzimšanasDatums=1998.12.31
```

```
PILNPARĀDĪT VISAS SlimnīcasEpizodes KURĀM kopIzmaksas>=100 UN Kustības.SKAITS> 5
UN (atbildīgaisĀrsts.uzvārds=Osis VAI =Liepa) UN Pacients.dzimums=sieviete
```

Nākamā svarīgā plaši lietojamā komanda:

SKAITS “Aclass selekcijas izteiksme”

Piemēram:

```
SKAITS Pacientiem, KURIEM dzimšanasDatums=1998.12.31
```

```
SKAITS SlimEpizodēm. KURĀM kopIzmaksas>=100 UN Kustības.SKAITS> 5 UN
(atbildīgaisĀrsts.uzvārds=Osis VAI =Liepa) UN Pacients.dzimums=sieviete
```

Šo komandu semantika ir acīmredzama.

Komandu SKAITS “Aclass selekcijas izteiksme” var pierakstīt arī citā (universālākā) formātā:

(“Aclass selekcijas izteiksme”).SKAITS

Šīs sintakses ideja, aizņemoties jēdzienus no sadaļas 3, dod iespēju definēt nākamās komandas:

(“Aclass selekcijas izteiksme”).a.SUM

(“Aclass selekcijas izteiksme”).a.MAX

(“Aclass selekcijas izteiksme”).a.MIN

(“Aclass selekcijas izteiksme”).a.VID,

kur **a** ir Aclass atribūtizteiksme ar tipu Integer, Decimal vai Duration.

Līdzīgi

(“Aclass selekcijas izteiksme”).a.BIEŽ,

kur **a** ir Aclass atribūtizteiksme ar tipu String.

Šo komandu semantika, ņemot vērā skaidrojumus sadaļā 4, ir acīmredzama.

Šīm komandām likumīgs ir arī dabīgajai valodai tuvāks pieraksts:

SUM /a/ “Aclass selekcijas izteiksme”

MAX /a/ “Aclass selekcijas izteiksme”

MIN /a/ “Aclass selekcijas izteiksme”

VID /a/ “Aclass selekcijas izteiksme”

BIEŽ /a/ “Aclass selekcijas izteiksme”

Tagad pārejam pie vienas no vissvarīgākajām komandām – tabulu būves komandas.

Šķirosim 3 gadījumus:

1. Kad tabulas bāzes klase ir kāda ontoloģijas klase,
2. Kad tabulas bāzi veido kādas klases atribūta vērtības,
3. Kad tabulas bāzi veido kāda veselo skaitļu intervāla elementi.

7.1. Pirmais gadījums: kad tabulas bāzes klase ir kāda ontoloģijas klase

Aplūkosim vistipiskāko gadījumu, kad par tabulas bāzes klasi ir izvēlēta kāda ontoloģijas klase, kuru apzīmēsim ar Aclass. Šajā gadījumā tabulas būve sākas ar komandu:

ATLASĪT [**k** / VISAS] “Aclass **x** selekcijas izteiksme”

Šajā komandā Aclass patvaļīgās instances apzīmējums “x” ir jālieto obligāti (“x” vietā var ņemt jebkuru citu apzīmējumu, piemēram, “y”).

Šīs komandas rezultāts no komandas

PARĀDĪT [k / VISAS] Aclass x KURĀM "paplašinātā x-loģiskā izteiksme"

atšķirsies ar to, ka rezultāts būs nevis vienkārši Aclass instanču saraksts:

Instance x1, Instance x2, … , Instance xn,

bet gan teksts, kurš sākas ar "x=", un tālāk seko instanču uzskaitījums:

x = Instancce x1, Instance x2, … , Instance xn,

kas nozīmē mainīgā x definīciju ar vērtībām attiecīgi **Instance x1, Instance x2, …, Instance xn.**

Komandai "atlasīt" patstāvīgas nozīmes mūsu vaicājumu valodā nav (kaut arī viņu var lietot), tā iet pārī ar komandu "izveidot tabulu":

ATLASĪT [k / VISAS] "Aclass x selekcijas izteiksme";
IZVEIDOT TABULU "Tabulas veidošanas apraksts"

Šo pāri mēs turpmāk sauksim par "**tabulas definēšanas komandu**", un tā būs viena no svarīgākajām mūsu vaicājumu valodas komandām. Ar tās palīdzību no aplūkojamās datu bāzes mēs "izvilksim" tabulas, kuras, ja nepieciešams, var ar "copy/paste" palīdzību ieeksportēt iekš Excel.

Tagad pārejam pie "Tabulas veidošanas apraksta" skaidrojuma pie nosacījuma, ka pirms tam ir izpildīta komanda "atlasīt Aclass x KURĀM …" un esam ieguvuši priekš mainīgā x visu iespējamo vērtību sarakstu x1, x2, …, xn. Tagad paskaidrosim, kā šo rezultātu izmantosim reālas tabulas konstruēšanā.

Vispirms nolemsim, ka katrai x vērtībai xi atbildīs sava rinda tabulā. Lai definētu konkrētu tabulu pie šīs norunas, mums jāveic sekojošas darbības:

1. Mums ir jāizvēlas kolonnu nosaukumi, piemēram, pasakot, ka mums būs kolonnas A, B, C;
2. Katrai kolonnai ir jānorāda, kāda būs šīs kolonnas šūnas vērtība rindā, kas atbilst attiecīgajai x vērtībai. Tā kā mums ir ieviests jēdziens "Aclass x skalāra izteiksme", tad šīs šūnas vērtību definēsim kā attiecīgās Aclass x skalārās izteiksmes vērtību.

Piemēram, ja mums Aclass vietā ir Pacienti un tabulas definēšanas pirmā komanda ir

ATLASĪT VISUS Pacientus x KURIEM dzimšanasDatums=1998.12.31,

tad mēs varam definēt tabulu, piemēram, ar kolonnām A, B, C, kur :
a) kolonnas A vērtība rindā, kas atbilst instancei xi, ir izteiksmes x.uzvārds vērtība, kad x=xi ,
b) kolonnas B vērtība rindā, kas atbilst instancei xi, ir izteiksmes x.ģimenesĀrsts.uzvārds vērtība, kad x=xi,
c) kolonnas C vērtība rindā, kas atbilst instancei xi, ir izteiksmes x.SlimnīcasEpizodes.SKAITS, kad x=xi.

Esam aprakstījuši tabulas veidošanu dabīgajā valodā. Tagad to pašu, sauktu par "Tabulas veidošanas aprakstu", uzrakstīsim formālajā vaicājumu valodā (kuru 'saprot dators'):

ATLASĪt **VISUS** Pacientus x **KUR**IEM dzimšanasDatums=1998.12.31;
IZVEIDOT TABULU x.uzvārds (**KOL**ONNA A), x.ģimeneĀrsts.uzvārds (**KOL**ONNA B),
x.SlimnīcasEpizodes.SKAITS (**KOL**ONNA C)

Drīkst nerakstīt x-prefiksus, tos 'dators sapratīs' jau no konteksta:

ATLAS VISUS Pacientus x KUR dzimšanasDatums=1998.12.31
IZVEID uzvārds (KOL A), ģimenesĀrsts (KOL B), SlimnīcasEpizodes.SKAITS (KOL C)

Lai nostiprinātu šo formālismu, aplūkosim vairākus "tabuļas definēšanas komandu" piemērus.;

Piemērs 1.

ATLASĪT VISUS Pacientus x KURIEM x. SlimnīcasEpizodes.SKAITS>5 UN
x.PoliklīnikasEpizodes.SKAITS=0; IZVEIDOT TABULU x.uzvārds (KOLONNA PacUzvārds),
x.SlimnīcasEpizodes.SKAITS (KOLONNA EpizodSkaits), x.SlimnīcasEpizodes.kopIzmaksas.SUM
(KOLONNA Izmaksas), x.SlimnīcasEpizodes.(izrakstīšanasLaiks–uzņemšanasLaiks).DAYS.SUM
(KOLONNA IlgumsDinasSlimnica)

Kā jau bija minēts, "x" prefiksu var nerakstīt:

ATLASĪT VISUS Pacientus x KURIEM SlimnīcasEpizodes.SKAITS>5 UN
PoliklīnikasEpizodes.SKAITS=0; IZVEIDOT TABULU uzvārds (KOLONNA PacUzvārds),
SlimnīcasEpizodes.SKAITS (KOLONNA EpizodSkaits), SlimnīcasEpizodes.kopIzmaksas.SUM
(KOLONNA Izmaksas), SlimnīcasEpizodes.(izrakstīšanasLaiks–uzņemšanasLaiks).DAYS.SUM
(KOLONNA IlgumsDinasSlimnica)

Bet pirms turpinām piemērus, vēl atzīmēsim, ka tabulas definēšanas komandu vēl var papildināt ar komandām

ATLASĪt RINDAS KURĀM KolonnasVārds = konstante;
("=" vietā var būt arī citas salīdzināšanas operācijas, ja to atļauj kolonnas elementu tipi)

SAKĀRTOT [**AUG**OŠI / **DILST**OŠI] **PĒC** KOLONNAS KolonnasVārds;

ATSTĀT [**PIRM**ĀS / **PĒD**ĒJĀS] 5 RINDAS; ('5' vietā var būt jebkurš cits naturāls skaitlis)

Piemērs 2:

ATLASĪT VISUS Pacientus x KURIEM SlimnīcasEpizodes.SKAITS>5 UN
PoliklīnikasEpizodes.SKAITS=0;
IZVEIDOT TABULU uzvārds (KOL PacUzvārds), SlimnīcasEpizodes.SKAITS (KOL EpizodSkaits),
SlimnīcasEpizodes.kopIzmaksas.SUM (KOL Izmaksas), SlimnīcasEpizodes.(izrakstīšanasLaiks–
uzņemšanasLaiks).DAYS.SUM (KOL IlgumsDienasSlimnica);
ATLASĪT RINDAS KURĀM IlgumsDienasSlimnica>100d;
SAKĀRTOT DILSTOŠI PĒC KOLONNAS Izmaksas;
ATSTĀT PIRMĀS 10 RINDAS

7.2. Otrais gadījums: kad tabulas bāzi veido kādas klases atribūta vērtības

ATLASĪt NO "Aclass selekcijas izteiksme" **ATRIB**ŪTA "atribūta vārds" [k / **VISAS**] ATŠĶIRĪGĀS **VĒRT**ĪBAS x ; **IZVEID**OT TABULU …

Alternatīvais pieraksts:

ATRAST "Aclass selekcijas izteiksme" **ATRIBŪTA** "atribūta vārds" [k / **VISAS**] ATŠĶIRĪGĀS **VĒRTĪBAS x ; IZVEIDOT TABULU …**

Paskaidrosim šo gadījumu tikai ar piemēriem:

ATLASĪT NO Kustībām KURĀM npk=* UN SlimnīcasEpizode.izrakstīšanasIemesls=miris
ATRIBŪTA nodaļa VISAS ATŠĶIRĪGĀS VĒRTĪBAS x ;
IZVEIDOT TABULU x (Kol Nodaļa), (SlimnīcasEpizodes KURĀM izrakstīšanasIemesls=miris UN EKSISTĒ Kustība KURAI npk=* UN nodaļa=x).SKAITS (kol MirušoSkaits)

ATLASĪT NO Kustībām atribūta nodaļa VISAS ATŠĶIRĪGĀS VĒRTĪBAS x;
IZVEIDOT TABULU x (KOL Nodaļa), (SKAITS SlimnīcasEpizodes KURĀM izrakstīšanasIemesls=miris UN (Kustība KURAI npk=*).nodaļa=x) (kol MirušoSkaits)

7.3. Trešais gadījums: kad tabulas bāzi veido veselo skaitļu intervāla elementi

ATLASĪT NO **INTERV**ĀLA **(i, j)** [k / **VISAS**] **VĒRT**ĪBAS x ; **IZVEID**OT TABULU…

Alternatīvais pieraksts:

ATRAST INTERVĀLA **(i, j)** [k / **VISAS**] **VĒRT**ĪBAS x ; **IZVEID**OT TABULU…

Arī šo gadījumu paskaidrosim tikai ar piemēriem:

ATLASĪT INTERVĀLA (1,12) VISAS VĒRTĪBAS x ;
IZVEIDOT TABULU x (kol Mēnesis), (SlimnīcasEpizodes KURĀM uzņemšanasLaiks.MONTH=x).SKAITS (kol UzņemtoSkaits)

ATLASĪT INTERVĀLA (1915, 2015) VISAS VĒRTĪBAS X;
IZVEIDOT TABULU x (KOL DzimšanasGds), (SKAITS Pacienti KURIEM dzimšanasDatums.YEAR=x UN EKSISTĒ SlimnīcasEpizode KURAI izrakstīšanasIemess=miris) (kol MirSkaits)

9. Kopsavilkums: vaicājumu šabloni

Piezīme 1: Komandas vārda "**atlasīt**" vietā ar tādu pašu nozīmi var lietot arī vārdu "**atrast**".

Piezīme 2. Lielie un mazie burti, kā arī mīkstināšanas un garumzīmes iepriekšējā tekstā ir lietoti tikai uzskatāmības dēļ, no komandu semantikas viedokļa tiem nav nozīmes (tas **neattiecas** uz String tipa konstantēm!). Tādēļ turpmākajā tekstā priekšdefinētos vārdus mēs parasti rakstīsim ar mazajiem burtiem. Zemāk redzamajos komandu šablonos priekšdefinēto vārdu obligātā daļa ir attēlota treknpierakstā un pasvītrota, zilā fontā ir pierakstīta "piemēram" daļa.

[parādīt / pilnparādīt] [10 / **visas**] Epizodes [e] **kurām** <no e -atkar log izt>
skaits Epizodēm [e] **kurām** <no e atkar log izt>

[max/min/bież/vid/sum] **/**<e-atr vai e-skalara izt>**/** Epizodes [e] **kurām** <no e -atkar log izt>
atrast [10 / **visas**] Epizodes [x] **kurām** <no x -atkar log izt> ,
izveidot tabulu <x-atkar izt1> (**kol**onna A), <x-atkar izt2> (**kol**onna B), <x-atkar izt3> (**kol**onna C),
atlasīt rindiņas kurām C<5,
sakārtot **[aug**oši/**dilst**oši**] pēc** kolonnas B,
atstāt **[pirm**ās/**pēd**ējās**]** 5 rindiņas
atrast Epizodēm [e] , **kur**ām <no e atkar log izt>, **atrib**ūta <e-atr vai izt> [10 / **visas**] atšķirīgās **vērt**ības x **,**
izveidot tabulu <x-atkar izt1> (**kol**onna A), <x-atkar izt2> (**kol**onna B), <x-atkar izt3> (**kol**onna C),
.....
atrast **interv**āla (8–23) [10 / **visas**] **vērt**ības x **,**
izveidot tabulu <x-atkar izt1> (**kol**onna A), <x-atkar izt2> (**kol**onna B), <x-atkar izt3> (**kol**onna C),
.....

10. Reāli vaicājumu piemēri uz BKUS datu ontoloģijas

"paradit 2 Epizodes",

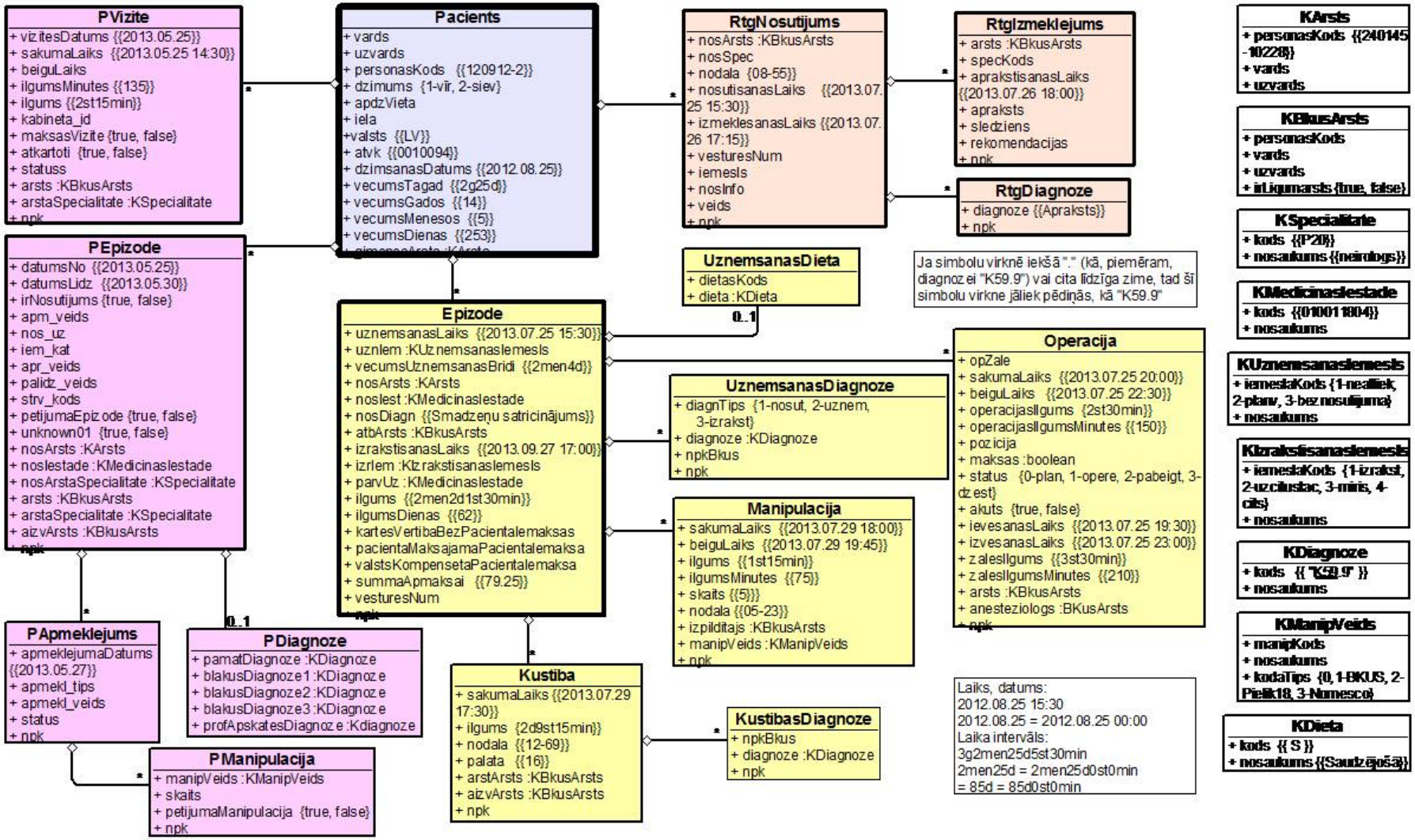
"pilnparādīt 2 Epizodes",

"parādīt 2 epizodes, kurām uzņemšanaslaiks.date>=2014.06.25 un uzņemšanaslaiks.date<2014.12.30 un nosārsts.uzvārds=Bērziņa",

"atrast visas epizodes, kurām uzņemšanaslaiks.date>=2014.06.25 un uzņemšanaslaiks.date<2014.07.30 un nosārsts.uzvārds=Bērziņa, izveidot tabulu pacients.uzvārds (kolonna Uzvārds), uznemsanaslaiks (kolonna Uznem laiks), izrakstisanaslaiks (kolonna Izrakst laiks), atbarsts.vārds (kolonna AtbildVārds), atbarsts.uzvards (kolonna AtbildUzvārds) ",

"atrast visas epizodes x, kurām uzņemšanaslaiks.date>=2014.06.25 un uzņemšanaslaiks.date<2014.07.30 un nosārsts.uzvārds=Bērziņa, izveidot tabulu x.pacients.uzvārds (kolonna Uzvārds), x.uznemsanaslaiks (kolonna Uznem laiks), x.izrakstisanaslaiks (kolonna Izrakst laiks), x.atbarsts.vārds (kolonna AtbildVārds), x.atbarsts.uzvards (kolonna AtbildUzvārds) ",

„atrast visas Epizodes x, kurām izrIem.iemeslaKods<>3, izveidot tabulu x.vesturesNum (kol A), (skaits x.Kustibas) (kol B), atlasit rindas kuram B>2, sakārtot dilstoši pēc B, atstāt pirmās 10 rindas",


PVizite
+ vizitesDatums {{2013.05.25}}
+ sakumaLaiks {{2013.05.25 14:30}}
+ beiguLaiks
+ ilgumsMinutes {{135}}
+ ilgums {{2st15min}}
+ kabineta_id
+ maksasVizite {true, false}
+ atkartoti {true, false}
+ statuss
+ arsts :KBkusArsts
+ arstaSpecialitate :KSpecialitate
+ npk
Pacients
+ vards
+ uzvards
+ personasKods {{120912-2}}
+ dzimums {1-vīr, 2-siev}
+ apdzVieta
+ iela
+valsts {{LV}}
+ atvk {{0010094}}
+ dzimsanasDatums {{2012.08.25}}
+ vecumsTagad {{2g25d}}
+ vecumsGados {{14}}
+ vecumsMenesos {{5}}
+ vecumsDienas {{253}}
RtgNosutijums
+ nosArsts :KBkusArsts
+ nosSpec
+ nodala {08-55}}
+ nosutisanasLaiks {{2013.07.25 15:30}}
+ izmeklesanasLaiks {{2013.07.26 17:15}}
+ iemesls
+ nosInfo
+ veids
+ npk
RtgIzmeklejums
+ arsts :KBkusArsts
+ specKods
+ aprakstisanasLaiks {{2013.07.26 18:00}}
+ apraksts
+ sledziens
+ rekomendacijas
+ npk
RtgDiagnoze
+ diagnoze {{Apraksts}}
+ npk
KArsts
+ personasKods {{240145-10228}}
+ vards
+ uzvards
KBkusArsts
+ personasKods
+ vards
+ uzvards
+ irLigumarsts {true, false}
KSpecialitate
+ kods {{P2I}}
+ nosaukums {{neirologs}}
KMedicinaslestade
+ kods {{01001804}}
+ nosaukums
KUznemsanaslemesls
+ iemeslaKods {1-neatliek, 2-plan, 3-bez nosutijuma}
+ nosaukums
KIzrakstisanaslemesls
+ iemeslaKods {1-izrakst, 2-uzcilustac, 3-miris, 4-cits}
+ nosaukums
KDiagnoze
+ kods {{ "K59.9" }}
+ nosaukums
KManipVeids
+ manipKods
+ nosaukums
+ kodaTips {0,1-BKUS, 2-Pielik18, 3-Nomesco}
KDieta
+ kods {{ S }}
+ nosaukums {{Saudzējošā}}
PEpizode
+ datumsNo {{2013.05.25}}
+ datumsLidz {{2013.05.30}}
+ irNosutijums {true, false}
+ apm_veids
+ nos_uz
+ iem_kat
+ apr_veids
+ palidz_veids
+ strv_kods
+ petijumaEpizode {true, false}
+ unknown01 {true, false}
+ nosArsts :KArsts
+ noslestade :KMedicinaslestade
+ nosArstaSpecialitate :KSpecialitate
+ arsts :KBkusArsts
+ arstaSpecialitate :KSpecialitate
+ aizvArsts :KBkusArsts
+ npk
UznemsanasDieta
+ dietasKods
+ dieta :KDieta
Ja simbolu virknē iekšā "." (kā, piemēram, diagnozei "K59.9") vai cita līdzīga zime, tad šī simbolu virkne jāliek pēdiņās, kā "K59.9"
Epizode
+ uznemsanasLaiks {{2013.07.25 15:30}}
+ uznlem :KUznemsanaslemesls
+ vecumsUznemsanasBridi {{2men4d}}
+ nosArsts :KArsts
+ noslest :KMedicinaslestade
+ nosDiagn {{Smadzeņu satricinājums}}
+ atbArsts :KBkusArsts
+ izrakstisanasLaiks {{2013.09.27 17:00}}
+ izrlem :Klzrakstisanaslemesls
+ parvUz :KMedicinaslestade
+ ilgums {{2men2d1st30min}}
+ ilgumsDienas {{62}}
+ kartesVertibaBezPacientalemaksas
+ pacientaMaksajamaPacientalemaksa
+ valstsKompensetaPacientalemaksa
+ summaApmaksai {{79.25}}
+ vesturesNum
+ npk
UznemsanasDiagnoze
+ diagnTips {1-nosut, 2-uznem, 3-izrakst}
+ diagnoze :KDiagnoze
+ npkBkus
+ npk
Operacija
+ opZale
+ sakumaLaiks {{2013.07.25 20:00}}
+ beiguLaiks {{2013.07.25 22:30}}
+ operacijasIlgums {2st30min}}
+ operacijasIlgumsMinutes {{150}}
+ pozicija
+ maksas :boolean
+ status {0-plan, 1-opere, 2-pabeigt, 3-dzest}
+ akuts {true, false}
+ ievesanasLaiks {{2013.07.25 19:30}}
+ izvesanasLaiks {{2013.07.25 23:00}}
+ zalesIlgums {{3st30min}}
+ zalesIlgumsMinutes {{210}}
+ arsts :KBkusArsts
+ anesteziologs :BKusArsts
+ npk
Manipulacija
+ sakumaLaiks {{2013.07.29 18:00}}
+ beiguLaiks {{2013.07.29 19:45}}
+ ilgums {{1st15min}}
+ ilgumsMinutes {{75}}
+ skaits {{5}}
+ nodala {{05-23}}
+ izpilditajs :KBkusArsts
+ manipVeids :KManipVeids
+ npk
PApmeklejums
+ apmeklejumaDatums {{2013.05.27}}
+ apmekl_tips
+ apmekl_veids
+ status
+ npk
PDiagnoze
+ pamatDiagnoze :KDiagnoze
+ blakusDiagnoze1 :KDiagnoze
+ blakusDiagnoze2 :KDiagnoze
+ blakusDiagnoze3 :KDiagnoze
+ profApskatesDiagnoze :Kdiagnoze
Kustiba
+ sakumaLaiks {{2013.07.29 17:30}}
+ ilgums {2d9st15min}}
+ nodala {{12-69}}
+ palata {{16}}
+ arstArsts :KBkusArsts
+ aizvArsts :KBkusArsts
+ npk
KustibasDiagnoze
+ npkBkus
+ diagnoze :KDiagnoze
+ npk
PManipulacija
+ manipVeids :KManipVeids
+ skaits
+ petijumaManipulacija {true, false}
+ npk
Laiks, datums:
2012.08.25 15:30
2012.08.25 = 2012.08.25 00:00
Laika intervāls:
3g2men25d5st30min
2men25d = 2men25d0st0min
= 85d = 85d0st0min
0..1
0..1

"atrast visus KBkusārstus, kuriem personasKods.substring(1,4)=3001",

"atrast visus KBkusārstus, kuriem personasKods.substring(1,4)=3001, izveidot tabulu vārds, uzvārds",

"atrast visus Kbkusārstus, kurim vārds=Arta, izveidot tabulu vārds, uzvārds, personaskods.substring(1,4)",

"skaits kustības, kurām nodaļa=12-69",

"skaits Epizodes, kurām eksistē Kustiba, kurai nodala=12-69",

"skaits Pacienti, kuriem eksistē Kustiba, kurai nodala=12-69 un npk=1",

"skaits Pacienti, kuriem eksistē Kustiba, kurai nodala=12-69 un npk=*",

"skaits Pacienti kuriem eksistē Kustiba kurai (nodala=12-69 vai nodala=02-25) un (npk=1 vai npk=2)",

"skaits epizodēm, kurām nosArsts=nill",

"skaits epizodēm, kurām nosArsts<>nill",

"skaits epizodēm, kurām nosArsts.uzvārds=Bērziņa",

"skaits epizodēm, kurām (nosArsts.uzvārds=Bērziņa vai nosArsts.uzvards=Kalniņa) un uznemsanasLaiks.date>2014.06.25",

„— mirušo un izdzīvojušo pacientu skaits par 2014.g.:”,

"skaits Pacienti, kuriem eksistē Epizode, kurai izrIem.iemeslaKods=3",

"skaits Pacienti, kuriem neeksistē Epizode, kurai izrIem.iemeslaKods=3",

„— mirušo pacientu skaits nodaļā 12-69 (dažādi pieraksti):”,

"skaits Pacienti, kuriem eksiste Epizode, kurai izriem.iemeslakods=3 un eksiste Kustiba, kurai nodala=12-69 un npk=*",

"skaits Pacienti, kuriem eksiste Kustiba, kurai epizode.izriem.iemeslakods=3 un nodala=12-69 un npk=*",

„skaits Epizodēm, kurām izrIem.iemeslaKods=3 un eksiste Kustiba, kurai nodala=12-69 un npk=*”,

„skaits Epizodēm, kurām izrIem.iemeslaKods=3 un (Kustiba kurai npk=*).nodala=12-69”,

„max /ilgumsDienas/ Epizodēm, kurām izriem.iemeslakods=3”,

„max /izrakstisanasLaiks-uznemsanasLaiks/ Epizodēm, kurām izriem.iemeslakods=3”,

„min /ilgumsDienas/ Epizodēm, kurām izriem.iemeslakods=3”,

„vid /ilgumsDienas/ Epizodēm, kurām izriem.iemeslakods=3”,

„sum /ilgumsDienas/ Epizodēm, kurām izriem.iemeslakods=3”,

„biez /diagnoze.kods/ UznemsanasDiagnozēm, kurām diagntips=2 un epizode.pacients.dzimums=1”,

„biez /manipVeids.manipKods/ Manipulacijam, kurām manipVeids.kodatips=2 un epizode.izriem.iemeslakods=3”,

„— tabulas ar izrakstīšanas diagnozēm mirušajiem pacientiem nodaļā 12-69:”,

„atrast visas Epizodes x, kurām izrIem.iemeslaKods=3 un (Kustība, kurai npk=*).nodaļa=12-69, izveidot tabulu x.pacients.dzimsanasdatums (kol dzimdat), x.ilgums, x.izrakstisanaslaiks.date (kol izrakstdat), (x.UznemsanasDiagnoze kurai diagntips=3).diagnoze.kods (kol Diagn)”,

„—varēja ‘x’ arī nerakstīt, vēl jāuzmanās, lai kolonnu vārdi nesakristu ar atribūtu vārdiem, tad uzleks kļūdas ziņojums”,

„atrast visas Epizodes, kurām izrIem.iemeslaKods=3 un (Kustība, kurai npk=*).nodaļa=12-69, izveidot tabulu pacients.dzimsanasdatums (kol dzimdat), ilgums (kol Epizodilg), izrakstisanaslaiks.date (kol izrakstdat), (UznemsanasDiagnoze kurai diagntips=3).diagnoze.kods (kol Diagn)”,

„— apmēram tas pats, tikai pa visām nodaļām:”

„atrast visas Epizodes x kurām izriem.iemeslakods=3, izveidot tabulu x.pacients.personasKods (kol PK), x.vesturesnum (kol A), x.izrakstisanasLaiks.date (kol B), (x.Kustiba kurai npk=*).nodaļa (kol C), (x.UznemsanasDiagnoze kurai diagntips=3).diagnoze.kods (kol D), (x.UznemsanasDiagnoze kurai diagntips=3).diagnoze.nosaukums (kol E)”,

"skaits Pacienti, kuriem eksiste Epizode un neeksistē Pepizode",

„— vēl piemēri par skaitu:”,

"skaits Epizodēm, kurām izrakstīšanasLaiks.date-uzņemšanasLaiks.date>2men15d",

"skaits Epizodēm, kurām izrakstīšanasLaiks-uzņemšanasLaiks>2men15d",

"skaits Epizodēm, kurām ilgums>2men15d",

"skaits Pacienti, kuriem dzimums=1",

"skaits Pacienti, kuriem skaits Epizodēm >1",

"skaits Pacienti, kuriem (skaits Epizodēm, kurām nosArsts.vards=Edgars) >1",

"skaits operācijām kurām beiguLaiks-sakumaLaiks > 2st30min",

„skaits Pacienti, kuriem eksiste Epizode e un eksiste Pvizite p, kurām p.vizitesdatums<=e.uznemsanaslaiks.date un e.uznemsanaslaiks.date-p.vizitesdatums<5d”,

„skaits Epizodes e, kurām eksistē Pvizite, kurai vizitesdatums<=e.uznemsanaslaiks.date un e.uznemsanaslaiks.date-vizitesdatums<5d”,

„skaits Pacienti kuriem (skaits Epizodes e, kurām eksistē Pvizite, kurai vizitesdatums<=e.uznemsanaslaiks.date un e.uznemsanaslaiks.date-vizitesdatums<5d)>1”,

„-- vēl tabulu piemēri:”,

"atrast 2 operācijas kurām beiguLaiks-sakumaLaiks > 2st30min, izveidot tabulu sākumalaiks, beigulaiks, opZāle, anesteziologs.uzvārds",

"atrast Kustībam k, kurām next(k).nodala=12-69, atribūta nodala visas atšķirīgās vērtibas x, izveidot tabulu x (kolonna A), skaits Kustības k kurām k.nodala=x un next(k).nodala=12-69 (kolonna B), sakārtot dilstoši pēc B ",

"atrast Kustībam k, kurām next(k).nodala=12-69, atribūta nodala visas atšķirīgās vērtibas x, izveidot tabulu {x (kolonna A), skaits Kustības k kurām k.nodala=x un next(k).nodala=12-69 (kolonna B)}, sakārtot dilstoši pēc B ",

"atrast Kustībām atribūta nodala visas atšķirīgās vērtibas ",

"atrast Kustībām atribūta nodala visas atšķirīgās vērtibas x, izveidot tabulu x (kol Nodaļa), (skaits Kustības kurām nodala =x)(kol Skaits), sakārtot dilstoši pēc kol Skaits",

"atrast intervāla (1-12) visas vērtības x, izveidot tabulu x (kolonna Mēnesis), (skaits manipulacijas kurām sakumalaiks.month=x) (kolonna Skaits)",

"atrast intervāla (1-12) visas vērtības x, izveidot tabulu x (kolonna Mēnesis), (skaits manipulacijas, kurām manipveids.kodatips=3 un (manipveids.manipkods.substring(1,1)=P vai manipveids.manipkods.substring(1,1)=F) un sakumalaiks.month=x) (kolonna Skaits)",

„— par manipulācijām operācijas laikā:",

„atrast visas Epizodes x kuram eksiste Operacija, izveidot tabulu x.vesturesnum (kol A), (skaits x.manipulacijas m kuram eksiste Operacija o kurai m.sakumalaiks>o.sakumalaiks un m.sakumalaiks<o.beigulaiks) (kol B), sakārtot dilstoši pēc B",

„-- Izrādās, ka 'vesturesnum=1130814' ir 24 manipulacijas operācijas laikā. Varam gribēt paskatīties, kas tad tās ir par manipulacijām:",

„atrast visas Manipulacijas x kuram x.epizode.vesturesnum=1130814 un eksiste operacija o kurai x.sakumalaiks>o.sakumalaiks un x.sakumalaiks<o.beigulaiks, izveidot tabulu x.manipveids.manipkods (kol A), x.sakumalaiks (kol B), x.beigulaiks (kol C), (x.uznemsanasdiagnoze kurai diagntips=2 un npkbkus=1).diagnoze.kods (kol D), (x.uznemsanasdiagnoze kurai diagntips=2 un npkbkus=2).diagnoze.kods (kol E)",

„skaits operacijas kuram epizode.vesturesnum=1130814",

„— svarīgs jautājums -cik ilgi jāgaida ar uznem diagnozi K59.9 uz pirmo Rtg izmeklējumu:",

„atrast 10 Epizodes x, kuram eksiste Uznemsanasdiagnoze, kurai diagntips=2 un diagnoze.kods='K59.9', izveidot tabulu x.vesturesnum (kol VestNr), x.uznemsanaslaiks (kol Uznem), x.izrakstisanaslaiks (kol Izrakst), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).nosutisanaslaiks (kol Rtgnosut), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).izmeklesanaslaiks (kol Rtgizmekl)",

„atrast 10 Epizodes x, kuram eksiste Uznemsanasdiagnoze, kurai diagntips=2 un diagnoze.kods='K59.9', izveidot tabulu x.vesturesnum (kol VestNr), x.uznemsanaslaiks (kol Uznem), x.izrakstisanaslaiks (kol Izrakst), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).nosutisanaslaiks (kol Rtgnosut), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).izmeklesanaslaiks (kol Rtgizmekl), (rtgnosutijums

kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).izmeklesanaslaiks-(rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).nosutisanaslaiks (kol Starpiba)",

„atrast visas Epizodes x, kuram eksiste Uznemsanasdiagnoze, kurai diagntips=2 un diagnoze.kods='K59.9', izveidot tabulu x.vesturesnum (kol VestNr), x.uznemsanaslaiks (kol Uznem), x.izrakstisanaslaiks (kol Izrakst), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).nosutisanaslaiks (kol Rtgnosut), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).izmeklesanaslaiks (kol Rtgizmekl), (rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).izmeklesanaslaiks-(rtgnosutijums kuram nosutisanaslaiks>x.uznemsanaslaiks un nosutisanaslaiks<x.izrakstisanaslaiks).nosutisanaslaiks (kol Starpiba), sakārtot dilstoši pēc kolonnas Starpiba"

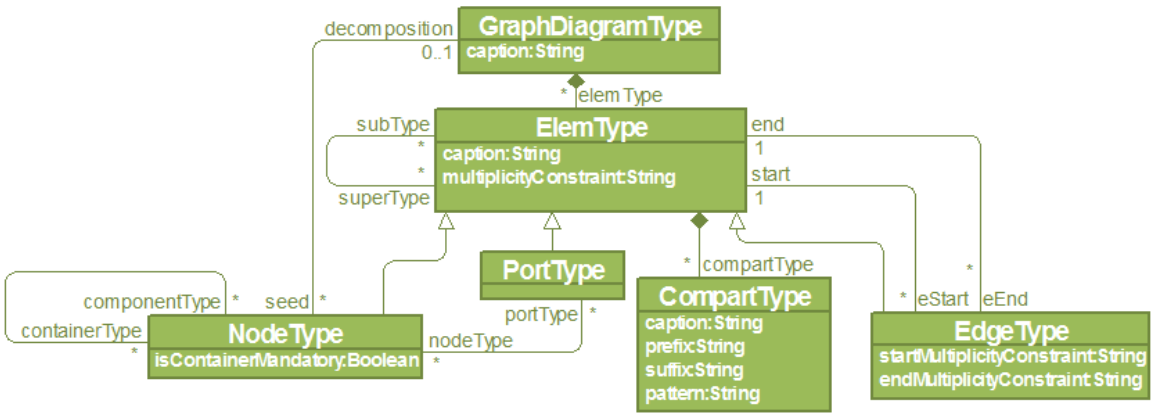
Pielikums 2.2

GRŪTI FORMALIZĒJAMU SISTĒMU MODELĒŠANAS METODIKA

Piedāvātā metodika balstās uz DSML rīku definēšanas metodi, kuru sauksim par konkretizācijas metodi. Atbilstoši šai metodei ir izstrādāts universāls DSML rīku definēšanas metamodelis. Tas nozīmē, ka katra rīku definēšanas metamodeļa instance definē vienu konkrētu DSML rīku. Tā kā rīku definīcijas sastāv no rīka valodas un uzvedības definīcijām, tad arī šī metamodeļa izklāstu sadalīsim attiecīgi divās daļās.

1.1 DSML metamodelis

Šajā nodaļā aprakstīsim valodas metamodeli, kas ļauj uzdot DSML specifikācijas. Šī metamodeļa izklāstu sadalīsim vairākās loģiskās daļās un sāksim ar valodas abstraktās sintakses metamodeli, kas ir redzams 1.1. attēlā.



1.1. att. DSML valodu abstraktās sintakses metamodelis

Šajā metamodelī valodas tiek specificētas ar klasi *GraphDiagramType*, un tās atribūts *caption* norāda valodas nosaukumu. Savukārt, valodas elementu specificēšanai ir paredzēta klase *ElemType*, kuras atribūts *caption* norāda elementa nosaukumu, bet to, kurai valodai elements pieder, norāda kompozīciju *elemType-graphDiagramType*. Klasei *ElemType* ir ieviestas apakšklases *NodeType, PortType* un *EdgeType*, kas attiecīgi norāda, ka elementa tips ir kaste, ports vai līnija. Klases *NodeType* asociācija *containerType-componentType* norāda, ka viena tipa

kastes var ievietot cita tipa kastē. Atribūts *isContainerMandatory* norāda, vai šī tipa kastei vienmēr ir jāatrodas ievietotai kādā citā kastē. Asociācija *portType-nodeType* (savieno klases *PortType* un *NodeType*) norāda, ka portam vienmēr ir jābūt piesaistītam noteikta tipa kastei.

Katra līnija savieno divus elementus, un tādēļ, lai definētu līniju, mums ir jānorāda, no kura elementa līnijai ir jāiziet un kurā jāieiet, un šo informāciju metamodelī norāda klases *EdgeType* asociācijas *start-eStart* un *end-eEnd*. Savukārt, atribūti *startMultiplicityConstraint* un *endMultiplicityConstraint* norāda līnijas sākuma un beigu kardinalitātes, proti, šīs kardinalitātes sāk darboties, kad tām ir norādīta skaitliskā vērtība (pretējā gadījumā maksimālā skaita ierobežojumi nepastāv), un to uzdevums ir norādīt attiecīgi maksimālo izejošo līniju skaitu no sākuma elementa un maksimālo ieejošo līniju skaitu beigu elementā.

Ar klases *ElemType* atribūtu *multiplicityConstraint* var norādīt maksimālo pieļaujamo elementu skaitu vienā diagrammā, un, līdzīgi kā ar līnijas kardinalitātēm, tas sāk darboties, kad ir norādīta skaitliskā vērtība.

Ar asociāciju *subType-superType* klasei *ElemType* var norādīt, ka kāds elements ir virstips kādam citam elementam, tādējādi ļaujot veidot elementu tipu hierarhiju, kurā apakštipi manto virstipu ierobežojumus. Piemēram, tipu metamodelis nosaka, ka katrai līnijai ir tieši viens sākuma un tieši viens beigu elements, respektīvi, ar šo konstrukciju var pateikt, ka „X" tipa līnijas savieno „A" tipa elementus ar „B" tipa elementiem, bet, ja mēs gribam, lai „X" tipa līnijas varētu savienot arī „A" tipa elementus ar „C" tipa elementiem, tad ir jāizveido jauns elementa tips „D" un jānorāda, pirmkārt, ka „X" tipa līnijas savieno „A" tipa elementus ar „D" tipa elementiem, un, otrkārt, ka „B" un „C" tipa elementi ir jāpadara par „D" tipa elementa apakštipiem (tas nozīmē, ka „B" un „C" mantos iespēju no „D" būt par „X" tipa līnijas galapunktu). Tā rezultātā tipu hierarhija ļauj norādīt, ka viena tipa līnijām ir iespējami vairāki sākuma un beigu elementu tipi. Tādējādi šī konstrukcija ļauj instanču līmenī simulēt UML klašu diagrammās izmantoto vispārināšanas attiecības ideju, kur apakšklases manto virsklases īpašības.

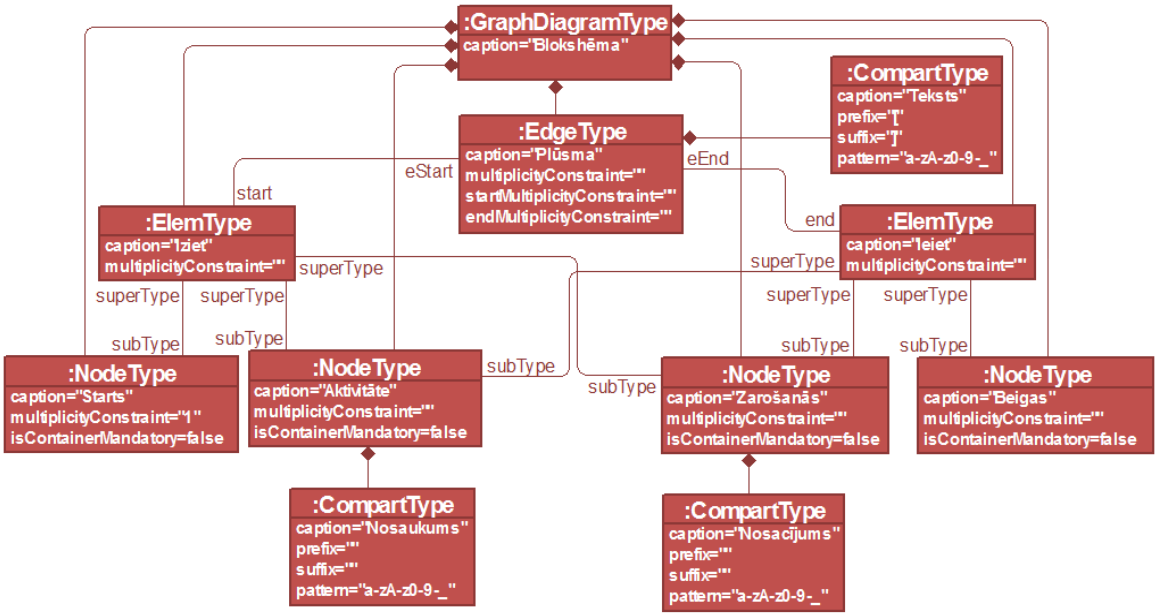
Jāpiezīmē, ka katram elementa tipam var būt ne tikai patvaļīgs skaits apakštipu, bet arī patvaļīgs skaits virstipu, kas atbilst UML klašu diagrammās izmantotajai daudzkāršās mantošanas idejai, ļaujot norādīt, ka viens elementa tips var mantot vairāku virstipu ierobežojumus.

Šis metamodelis paredz, ka ar klases *ElemType* tiešajām instancēm tiek definēti abstrakti elementi, jeb precīzāk sakot, tie ir tādi elementi, kuriem nav norādīts elementa tips (respektīvi, kaste, ports vai līnija). Tādējādi šādus elementus diagrammās nevar izveidot, bet tos var izmantot, lai veidotu elementu hierarhiju, kas ir līdzīgi kā tas ir ar abstraktajām klasēm UML klašu diagrammās, kuras izmanto klašu hierarhiju veidošanai, bet tām nav tiešo instanču (ir to apakšklasēm).

Metamodelī ir iestrādāta iespēja caur elementu detalizēt diagrammas. Tas tiek panāks ar asociāciju *seed-decomposition* starp *ElemType* un *GraphDiagramType*, kas norāda, ka viena tipa elementi detalizē noteikta tipa diagrammas.

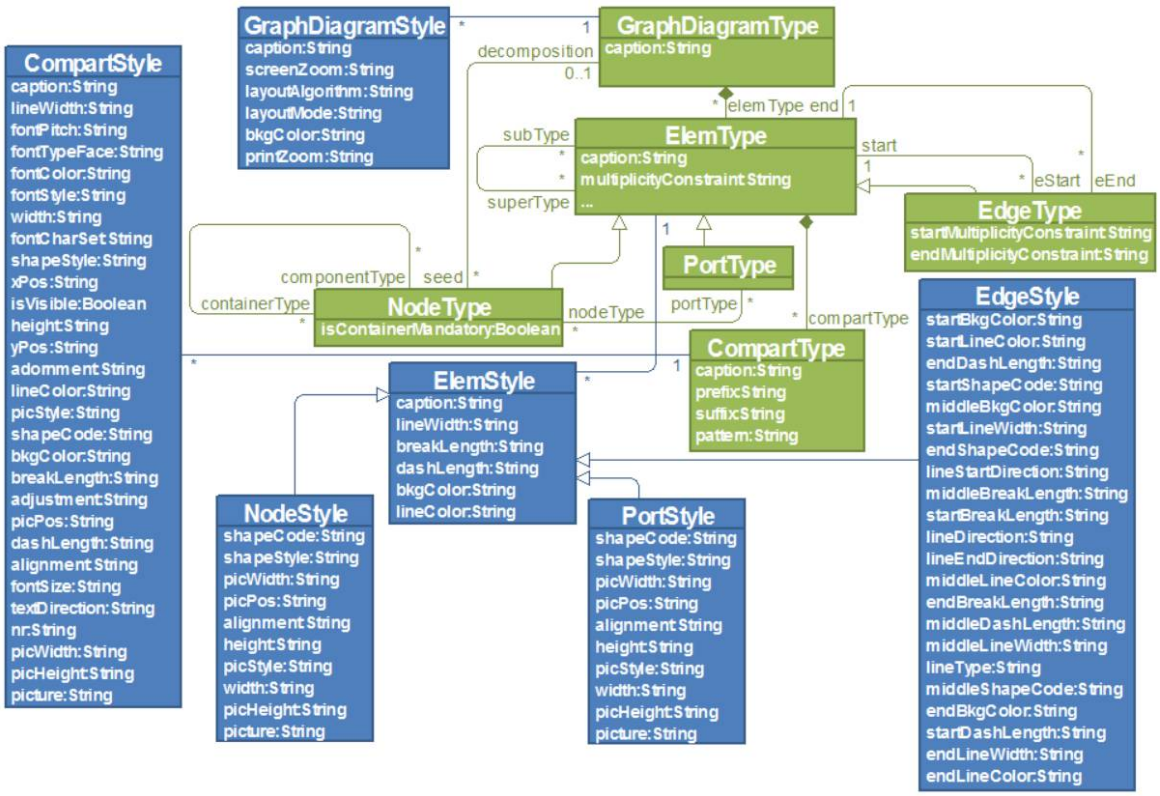
Klase *CompartType* ļauj specificēt elementu atribūtus, tās atribūts *caption* norāda atribūta nosaukumu, bet to, kuram elementam tie pieder, norāda kompozīciju *compartType-elemType*. Atribūti *prefix* un *suffix* norāda atribūta prefiksu un sufiksu, piemēram, ja prefikss ir „<<" un sufikss ir „>>", tad atribūta vērtība ir formā - „<<šī ir atribūta vērtība>>". Savukārt atribūts *pattern* ļauj norādīt atribūtu vērtībās pieļaujamos simbolus. Šīs vērtības ir jānorāda līdzīgi kā regulārās izteiksmes, piemēram, lai norādītu, ka atribūtā ir pieļaujami mazie burti, lielie burti, cipari, atribūtam *pattern* ir jāpiešķir „a-zA-Z0-9", vai, ja gribam pieļaut atribūtu vērtībās ciparus, domuzīmes, apakšsvītras un burtu „a", tad atribūtam *pattern* ir jāpiešķir „0-9-_a".

Lai nodemonstrētu šī metamodeļa lietojumu, 1.2. attēlā ir parādīts, kā ar šo metamodeli var definēt blokshēmu valodas abstrakto sintaksi.



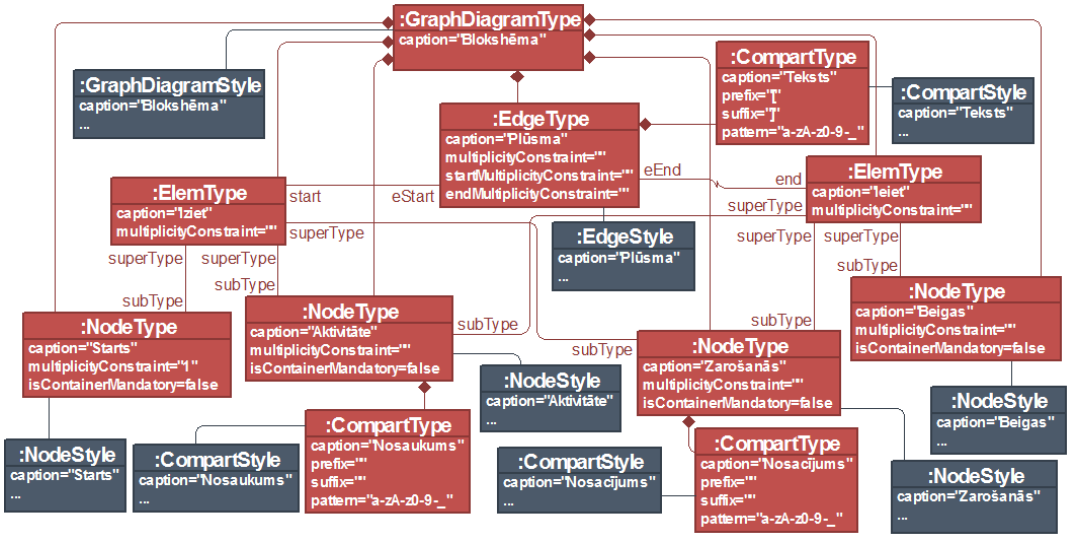
1.2. att. Blokshēmu valodas abstraktās sintakses specifikācija

Iepriekš aprakstījām, kā var specificēt valodas abstrakto sintaksi, bet, lai varētu specificēt valodas konkrēto sintaksi, ir jānorāda stili. Lai to izdarītu, abstraktās sintakses metamodelis ir papildināts ar stila klasēm. Klase *GraphDiagramStyle* norāda diagrammas stilu, klase *ElemStyle* un tās apakšklases *NodeStyle*, *PortStyle* un *EdgeStyle* norāda stilu attiecīgi kastēm, portiem un līnijām, bet *CompartStyle* norāda atribūtu stilus. Pilns valodas metamodelis ir redzams 1.3. attēlā.



1.3. att. DSML valodu metamodelis

Blokshēmu valodas konkrētās sintakses instanču diagramma ir redzama 1.4. attēlā.



1.4. att. Blokshēmu valodas specifikācija

1.2 DSML rīku definēšanas metamodelis

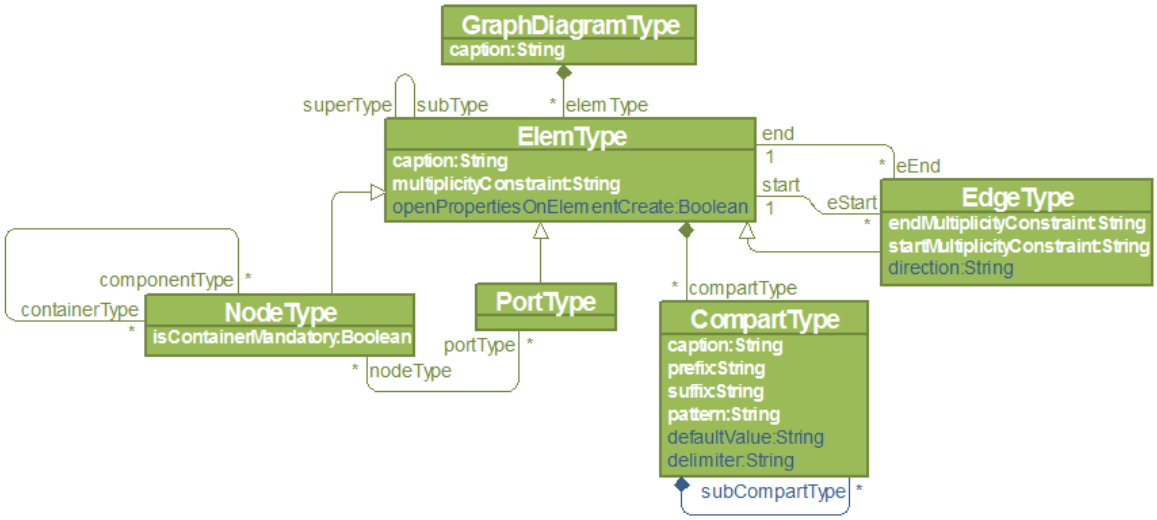
Lai specificētu DSML rīku, valodas specifikācija ir jāpapildina ar rīka uzvedības specifikāciju. Tas nozīmē, ka ir jāpapildina valodas metamodelis, un, lai šie papildinājumi būtu vieglāk saprotami, to izklāsts ir sadalīts pa fragmentiem, bet pilnais DSML rīku definēšanas metamodelis ir redzams 1.26. attēlā.

1.2.1 Valodas metamodeļa papildinājumi

Pirmajā solī valodas metamodelis ir papildināts ar četriem jauniem atribūtiem un vienu kompozīciju (skat. 1.5. attēlu). Klasei *ElemType* ir pievienots jauns atribūts *openPropertiesOnElementCreate*, kas norāda, vai jauna elementa veidošanas laikā ir jāatver dialogu logs. Klasei *EdgeType* ir pievienots jauns atribūts *direction*, kas norāda līnijas orientāciju un tam ir iespējamas trīs vērtības – „UniDirectional", „BiDirectional" un „ReverseBiDirectional". Noklusētā vērtība ir „UniDirectional" un tā norāda, ka līnija ir orientēta, t.i., līniju var novilkt no viena elementa uz otru, bet ne otrādāk. Vērtība „BiDirectional" norāda, ka līnija nav orientēta, t.i., līniju var vilkt no viena elementa uz otru un otrādāk. Vērtība „ReverseBiDirectional" norāda, ka līniju var vilkt abos virzienos, bet tās orientācija būs vienmēr nemainīga neatkarīgi no vilkšanas virziena, piemēram, ja ar bultu tiktu savienoti elementi A un B, tad bulta ietu no A uz B, bet, ja bultu vilktu no B uz A, tad bulta tiktu „pagriezta" pretējā virzienā un rezultātā tā ietu no A uz B.

Klasei *CompartType* ir pievienoti atribūti *defaultValue* un *delimiter*, kā arī kompozīcija *compartType-subCompartType*. Atribūts *defaultValue* norāda atribūta noklusēto vērtību, kas automātiski ir jāuzstāda uzreiz pēc atribūta izveidošanas. Kompozīcija *compartType-subCompartType* norāda, ka atribūtam var būt apakšatribūti „patvaļīgā dziļumā", kas veido atribūtu koku. Šāds koks tiek veidots, lai lietotājam, kā vienu veselu, varētu rādīt atribūtu vērtību, kas ir iegūta, saliekot kopā apakšatribūtu vērtības. Savukārt, atribūts *delimiter* kalpo kā atdalītājs starp apakšatribūtiem, kad no apakšatribūtu vērtībām tiek būvēta virsatribūta vērtība.

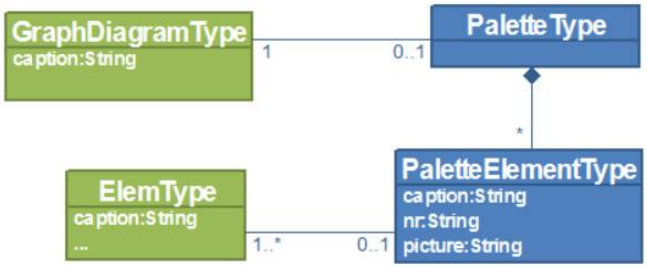
Vispārīgā gadījumā šīs iespējas tiek kombinētas. Gadījumā, ja ir vairāki apakšatribūti, tad tos atdala ar atdalītāju, un, ja kādam apakšatribūtam ir norādīts prefikss vai sufikss, tad šīs vērtības arī tiek uzstādītas. Piemēram, ja ir dots atribūts, kuram ir divi apakšatribūti un pirmajam apakšatribūtam prefikss ir „<<" un sufikss ir „>>", un atribūta atdalītājs ir „:", tad atribūta vērtība būs formā – „<<apakšatribūts1>>:apakšatribūts2", bet katra apakšatribūtu vērtības būs attiecīgi „<<apakšatribūts1>>" un „apakšatribūts2".



1.5. att. Papildināts valodas metamodelis

1.2.2 Palete

Jaunu elementu veidošana ir paredzēta ar paletes pogām, un tādēļ metamodelī ir ieviestas jaunas klases *PaletteType* un *PaletteElementType* (skat. 1.6. attēlu), kuras attiecīgi specificē paleti un tās pogas. To, kuram diagrammas tipam pieder palete, norāda asociācija *graphDiagramType-paletteType*, bet to, kuru elementu veidot ar paletes pogu norāda *elemType-paletteElementType*.

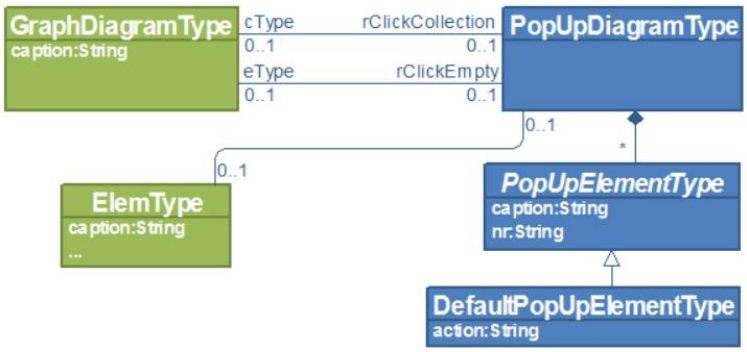


1.6. att. Paletes metamodeļa fragments

Klasei *PaletteElementType* ir atribūti – *caption, nr* un *picture.* Atribūts *caption* norāda paletes pogas nosaukumu, *nr* norāda tās kārtas numuru uz paletes un *picture* norāda ikonas adresi.

1.2.3 Uznirstošās izvēlnes

Lai realizētu uznirstošās izvēlnes, kuras parādās, kad lietotājs nospiež peles labo taustiņu, metamodelī ir ieviestas jaunas klases un asociācijas, kā tas ir redzams 1.7. attēlā.



1.7. att. Uznirstošo izvēļņu metamodeļa fragments

Izvēlnei metamodelī atbilst klase *PopUpDiagramType*, bet izvēlnes elementiem klase *PopUpElementType*, kuras atribūts *caption* norāda elementa nosaukumu, bet *nr* elementa kārtas numuru uz izvēlnes. Lai piedāvātu noklusētos izvēlnes elementus, metamodelī ir ieviesta klase *DefaultPopUpElementType*, kuras atribūts *action* norāda elementa darbību, un pieļaujamo darbību saraksts ir dots 1.1. tabulā.

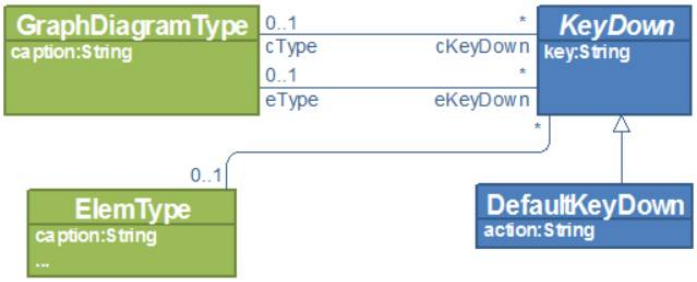
***1.1. tabula.* Noklusētās izvēļņu elementu darbības**

Darbība	Paskaidrojums
Cut	Realizē viena vai vairāku elementu izgriešanu (atkarībā no tā, vai elementu kolekcijā ir viens vai vairāki elementi).
Copy	Realizē viena vai vairāku elementu kopēšanu (atkarībā no tā, vai elementu kolekcijā ir viens vai vairāki elementi).
Paste	Realizē nokopēto elementu ielīmēšanu.
Delete	Realizē viena vai vairāku elementu dzēšanu (atkarībā no tā, vai elementu kolekcijā ir viens vai vairāki elementi).
Properties	Realizē elementa dialogu loga atvēršanu.
SymbolStyle	Realizē elementa stila nomaiņu.
Reroute	Realizē līnijas trajektorijas pārrēķināšanu.
Detail	Realizē detalizācijas diagrammas pievienošanu.

Lai parādītu uznirstošo izvēlni, atkarībā no konteksta tiek šķiroti trīs gadījumi. Viens gadījums ir, kad lietotājs ir nospiedis uz kāda elementa, un tādā gadījumā, lai uzbūvētu izvēlni, tiek izmantota asociācija *popUpDiagramType-elemType*. Otrs gadījums ir, kad lietotājs ir nospiedis uz tukšas vietas diagrammā, un tādā gadījumā ir jāizmanto asociācija *rClickEmpty-eType*. Trešais gadījums ir, kad lietotājs ir nospiedis uz elementu kolekcijas, un tādā gadījumā transformācija izmanto *rClickCollection-cType*.

1.2.4 Taustiņu kombinācijas

Lai realizētu lietotāju nospiesto taustiņu kombināciju apstrādi, metamodelī ir ieviestas jaunas klases un asociācijas, kā tas ir redzams 1.8. attēlā.



1.8. att. Taustiņu kombināciju metamodeļa fragments

Klase *KeyDown* un tās atribūts *key* specificē nospiesto taustiņu kombināciju. Analoģiski uznirstošajām izvēlnēm, lai piedāvātu noklusētās taustiņu kombināciju darbības, metamodelī ir ieviesta klase *DefaultKeyDown* un tās atribūts *action* norāda izpildāmo darbību un pieļaujamo darbību saraksts ir dots 1.1. tabulā.

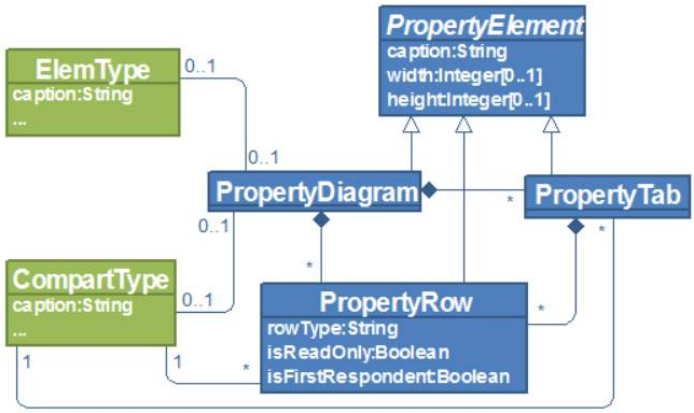
1.1. tabula. **Noklusētās taustiņu darbības**

Taustiņu kombinācija	Darbība	Paskaidrojums
Ctrl X	Cut	Realizē viena vai vairāku elementu izgriešanu (atkarībā no tā, vai elementu kolekcijā ir viens vai vairāki elementi).
Ctrl C	Copy	Realizē viena vai vairāku elementu kopēšanu (atkarībā no tā, vai elementu kolekcijā ir viens vai vairāki elementi).
Ctrl V	Paste	Realizē nokopēto elementu ielīmēšanu.
Delete	Delete	Realizē viena vai vairāku elementu dzēšanu (atkarībā no tā, vai elementu kolekcijā ir viens vai vairāki elementi).
Enter	Properties	Realizē elementa dialogu loga atvēršanu.

Tāpat kā uznirstošās izvēlnes gadījumā, atkarība no konteksta tiek izšķirti trīs gadījumi, kad taustiņu kombinācija tika nospiesta – uz elementa, tukšas vietas diagrammā vai elementu kolekcijas. Ja taustiņu kombinācija tika nospiesta uz elementa, tad izmanto asociāciju *keyDown-elemType*. Ja uz tukšas vietas, tad izmanto *eKeyDown-eType*, bet, ja uz kolekcijas, tad *cKeyDown-cType*.

1.2.5 Dialogu logi

Elementu atribūtu vērtību ievadīšana ir paredzēta caur dialogu logiem. Lai šos dialogu logus varētu uzbūvēt ģenēriski, metamodelis ir papildināts ar jaunām klasēm un asociācijām, kā tas ir redzams 1.9. attēlā.



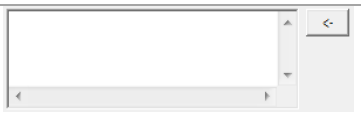
1.9. att. Metamodeļa fragments dialogu logu specifikācijai

Klase *PropertyElement* ir abstrakta klase, kas norāda, ka visām dialogu logu komponentēm ir atribūts *caption*, lai norādītu komponentes nosaukumu, un atribūti *height* un *width*, lai norādītu komponentes minimālo augstumu un platumu.

Dialogu formām atbilst klase *PropertyDiagram*, kas sastāv no rindām vai cilnēm (metamodelī atbilst klases *PropertyRow* un *PropertyTab*). Katra dialogu logu rinda (*PropertyRow*) tiek attēlota kā rindas nosaukums un ievadlauks. Rindas nosaukumam atbilst *caption* vērtība (atribūts mantots no klases *PropertyElement*), bet ievadlauka tipu nosaka *rowType* vērtība (iespējamie ievadlauku veidi un to izskati ir aprakstīti 1.3. tabulā). Atribūts *isFirstRespondent* norāda to ievadlauku, uz kura jābūt uzstādītam fokusam formas atvēršanas brīdī (iespējams uzstādīt tikai vienam laukam), bet atribūts *isReadOnly* norāda, vai šīs rindas ievadlauks ir vai nav rediģējams.

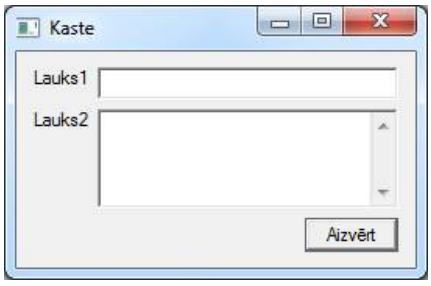
1.3. tabula. **Ievadlauku veidi**

Lauka tips	Izskats	Paskaidrojums
„InputField”		Vienkāršs teksta lauks.
„ComboBox”		Izkrītošās izvēlnes lauks.
„CheckBox”		Ķekša kastes lauks.

„TextArea”		Liels teksta lauks.
„Label”		Uzraksts.
„InputField+Button”		Vienkāršs teksta lauks ar pogu paredzēts, lai ievadītu saliktas atribūtu vērtības. Poga paredzēta, lai atvērtu papildus formu apakšatribūtu vērtību ievadei.
„TextArea+Button”		Teksta lauks ar pogu. Katra teksta lauka rinda atbilst saliktai apakšatribūtu vērtībai. Ar pogu tiek atvērta papildus forma apakšatribūtu ievadei.

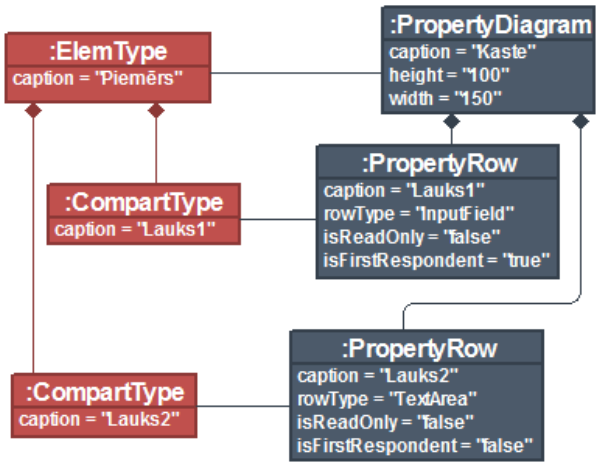
Klase *PropertyTab* ļauj uz dialogu loga formas būvēt cilnes, tādējādi rindas var atrasties tieši uz formas, ko norāda asociācija *propertyRow-propertyDiagram*, vai uz cilnes, ko norāda asociācija *propertyRow-propertyTab*.

Dialogu logu metamodeļa klases ar elementu un atribūtu specificējošajām klasēm ir sasaistīts caur asociācijām *compartType-propertyRow, compartType-propertyDiagram* un *elemType-propertyDiagram*. Asociācija *elemType-propertyDiagram* norāda elementa tipam atbilstošo dialogu logu formu, asociācija *compartType-propertyRow* norāda atribūtam piesaistīto dialogu logu rindu, bet ar *compartType-propertyDiagram* norāda atribūtam piesaistīto papildus dialogu logu formu apakšatribūtu vērtību ievadei.



1.10. att. Ģenerētā dialogu logu forma

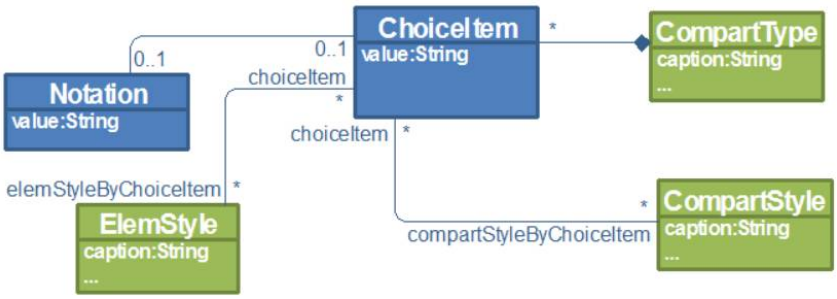
Kā piemēru aplūkosim 1.10. attēlā redzamo formu, kuras atbilstošā instanču diagramma ir redzama 1.11. attēlā. Formas nosaukums ir „Kaste” un tā sastāv no divām rindām ar šādiem ievadlaukiem - „InputField” un „TextArea”. Jāpiezīmē, ka uz formas ir arī poga „Aizvērt”, kā arī pogas formas minimizēšanai (Minimize), izmēru atjaunošanai (Restore Down) un aizvēršanai (Close), kuras metamodelī netiek attēlotas, jo ģenēriski veidotajām formām tās tiek pievienotas vienmēr.



1.11. att. Dialogu loga instanču diagramma

1.2.6 Atribūtu loģiskās un reprezentatīvās vērtības

Dažās situācijās vērtības, kuras ievada caur dialogu logu atšķiras no tām, kas tiek rādītas lietotājiem diagrammās. Piemēram, ja dialogu logā atribūta vērtību ievada caur ķekša kasti, tad ir iespējamas divas vērtības – „true" vai „false". Ja diagrammā rādīs kādu no šīm vērtībām, tad lietotājam būs grūti saprast, kura tipa atribūtam šī vērtība atbilst un ko tā apraksta. Lai lietotājiem diagrammās varētu rādīt atribūtu reprezentatīvās vērtības, bet dialogu logos lietotāju ievadītās (piemēram, „true" vai „false), metamodelis ir papildināts ar jaunām klasēm un asociācijām, kā tas ir redzams 1.12. attēlā.



1.12. att. Atribūtu reprezentatīvo vērtību un stilu metamodeļa fragments

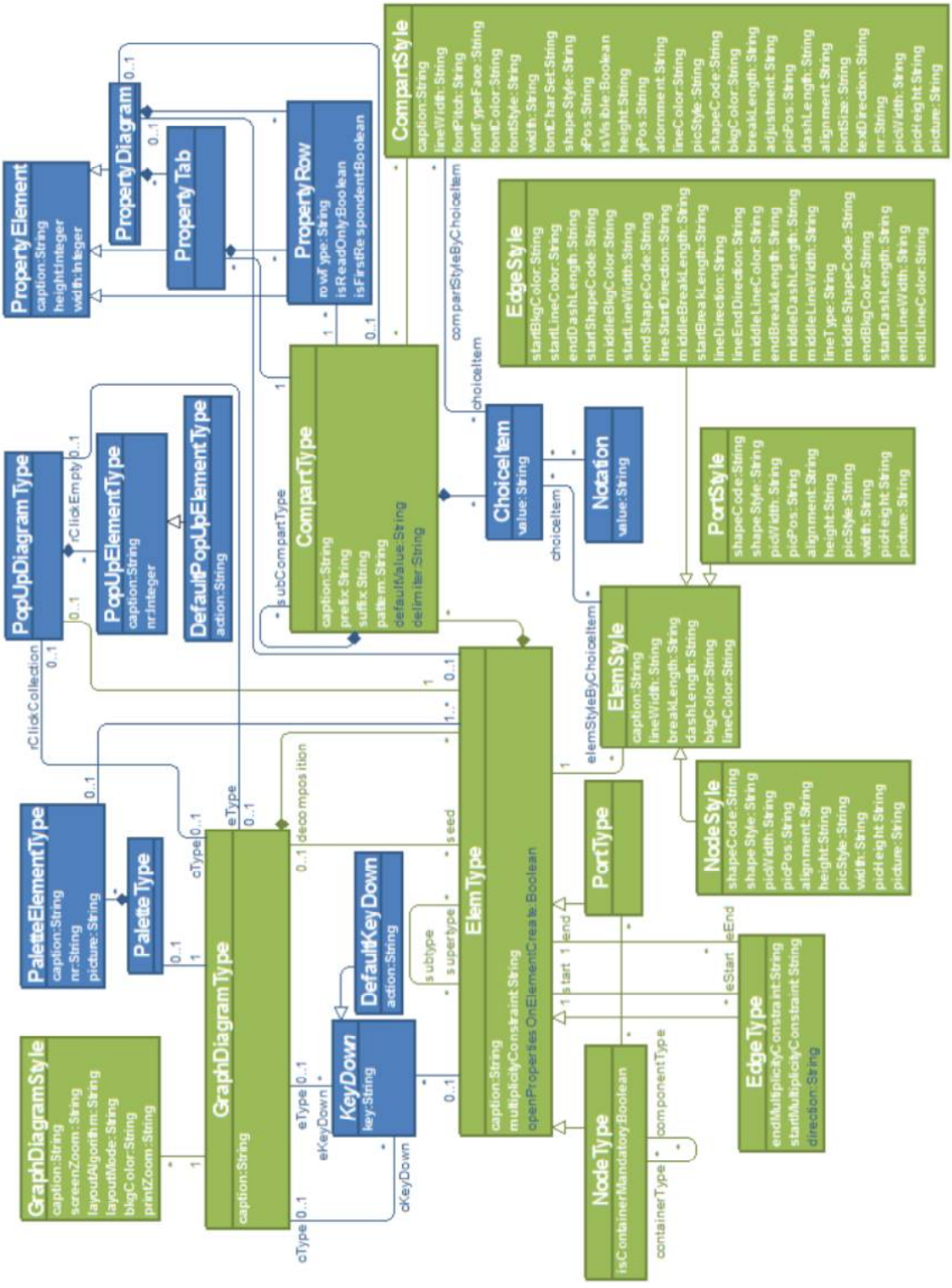
Klase *ChoiceItem* norāda vienu iespējamo atribūta vērtību dialogu logā, bet klase *Notation* norāda tā reprezentatīvo nosaukumu un abas šīs klases saista asociācija *choiceItem-notation*. Ar kompozīciju *compartType-choiceItem* norāda, kuram atribūta tipam vērtības ir piesaistītas.

Klases *ChoiceItem* vērtības tiek izmantotas arī priekš izkrītošo izvēļņu sarakstu specificēšanas, respektīvi, ja formas ievada lauks ir izkrītošā izvēlne, tad katra *ChoiceItem* instance tiek attēlota kā viens elements izkrītošās izvēlnes sarakstā.

Tomēr ir situācijas, kad ar reprezentatīvo vērtību uzstādīšanu nepietiek, un ir vajadzība mainīt atribūta vai pat elementa stilu. Piemēram, ja gribam, lai pie noteiktām vērtībām, atribūtu tekstu vai pašu elementu rāda citā krāsā. Metamodelī šī iespēja tiek realizēta ar asociācijām *choiceItem-compartStyleByChoiceItem* un *choiceItem-elemStyleByChoiceItem*. Tātad, kad lietotājs dialoga logā ievada kādu vērtību un, ja tā atbilst kādai no *ChoiceItem* vērtībām, tad kā atribūta vērtība ir jāuzstāda atbilstošā *Notation* vērtība un, ja ir norādīti stili, tad ir jāuzstāda arī *ChoiceItem* instancei piesaistītos atribūta un elementa stilus.

1.2.7 Pilnais DMSL rīku metamodelis

Pilnais DSML rīku metamodelis ir redzams 1.13. attēlā, un tas ir iegūts apvienojot valodu metamodeli un rīku uzvedību aprakstošās klases vienā metamodelī.



1.13. att. DSML rīku definēšana metamodelis

1.3 Blokshēmu redaktora specificēšana ar tipu instancēm

Lai nodemonstrētu DSML rīku definēšanas metamodeļa lietojumu, specificēsim iepriekš apskatīto blokshēmu redaktoru. Lai iegūtu blokshēmu redaktora specifikāciju, valodas specifikācija (skat. 1.13. attēlu) ir papildināta ar redaktora uzvedības specifikāciju, kā tas ir redzams 1.14. attēlā.

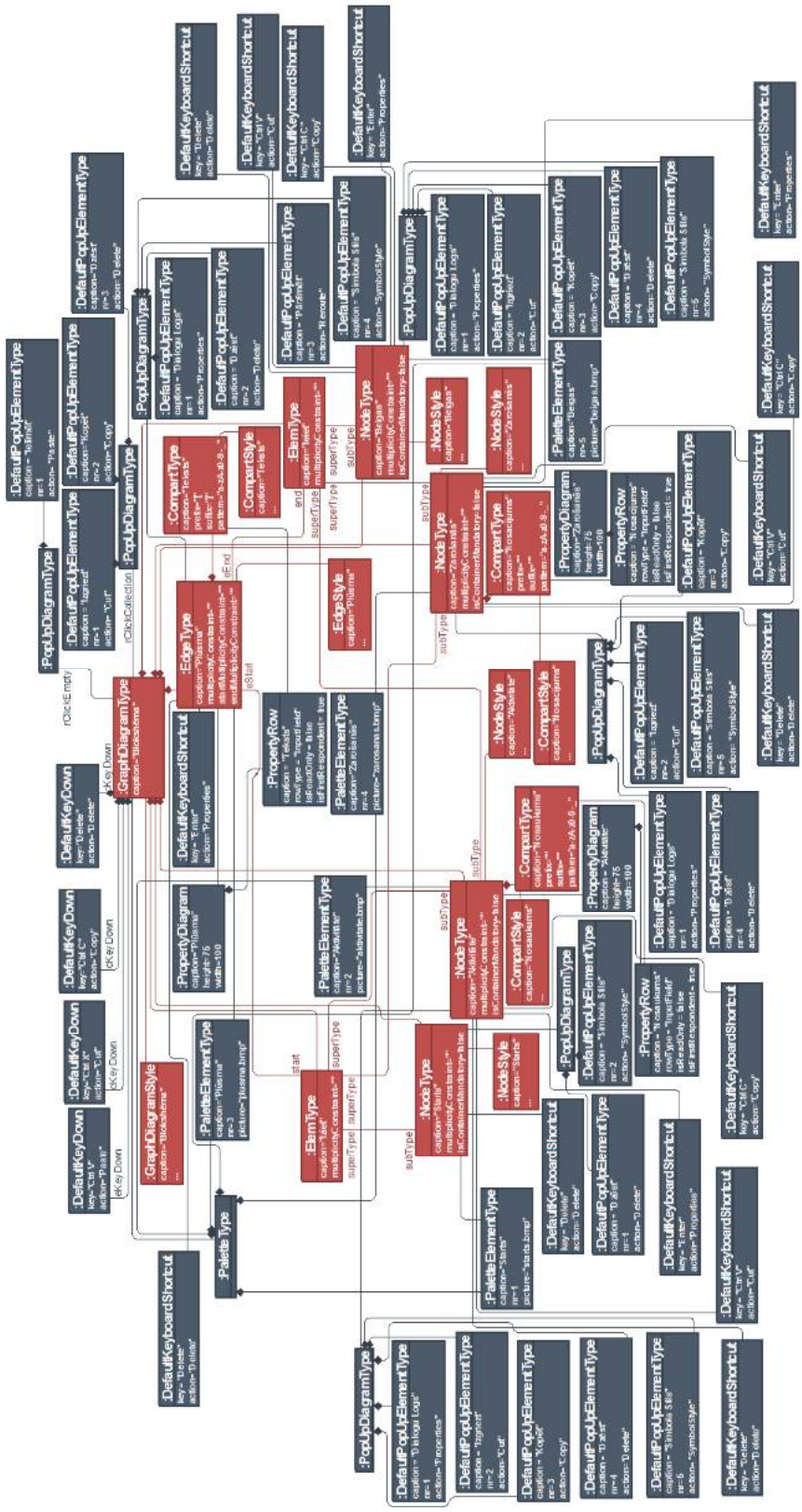
Šī specifikācija paredz, ka blokshēmu diagrammās, nospiežot peles labo taustiņu uz vairāku elementu kolekcijas, tiek piedāvāta uznirstošā izvēlne ar trīs operācijām – izgriezt, kopēt un dzēst, bet, ja peles labo taustiņu nospiež uz tukšas vietas diagrammā, tad tiek piedāvāta izvēlne ar vienu operāciju – ielīmēt. Taustiņu kombināciju apstrāde ir specificēta līdzīgi. Vairāku elementu kolekcijai ir specificētas trīs taustiņu kombinācijas – „Ctrl X", kas atbilst izgriešanas operācijai, „Ctrl C", kas atbilst kopēšanas operācijai, un „Delete", kas atbilst dzēšanas operācijai, bet tukšai vietai diagrammā ir specificēta viena taustiņu kombinācija - „Ctrl V", kas atbilst ielīmēšanas operācijai. Savukārt, lai varētu pievienot jaunus elementus, šī specifikācija paredz, ka diagrammās ir palete, ar kuras elementiem var izveidot jaunus diagrammas elementus.

Valodā ir specificēti šādi kastes tipa elementi – „Starts", „Aktivitāte", „Zarošanās" un „Beigas". Tā kā elements „Starts" diagrammā var būt tikai viens, un tam nav neviena atribūta, tad tam ir specificēta tikai uznirstošā izvēlne ar divām operācijām - dzēst un nomainīt stilu un viena taustiņu kombinācija – „Delete", kas atbilst dzēšanas operācijai.

Elementiem „Aktivitāte" un „Zarošanās" specificētās uznirstošās izvēlnes un taustiņu kombinācijas piedāvā plašāku funkcionalitāti. Uznirstošās izvēlnes šiem elementiem sastāv no piecām operācijām – atvērt dialogu logu, izgriezt, kopēt, dzēst un nomainīt stilu, un četrām taustiņu kombinācijām – „Delete", kas atbilst dzēšanas operācijai, „Ctrl X", kas atbilst izgriešanas operācijai, „Enter", kas atbilst dialogu loga atvēršanas operācijai un „Ctrl C", kas atbilst kopēšanas operācijai. Katram no šiem elementiem ir viens atribūts, un attiecīgi to vērtību ievadīšanai katram elementam ir specificēts dialogu logs ar vienu rindu.

Elementa „Beigas" specificētā funkcionalitāte ir līdzīga elementiem „Aktivitāte" un „Zarošanās", bet, tā kā šim elementam nav atribūtu, tad šim elementa nav nepieciešams specificēt dialogu logu, kā arī izvēlnes operāciju atvērti dialogu logu un taustiņu kombināciju „Enter" neeksistējošā dialogu loga atvēršanai.

Vienīgais līnijas tipa elements blokshēmu valodā ir „Plūsma". Šim elementam ir viens atribūts un tā vērtību ievadīšanai ir specificēts dialogu logs ar vienu rindu. Savukārt tā uznirstošā izvēlne sastāv no četrām operācijām - dzēst, atvērt dialogu logu, pārzīmēt līniju un nomainīt stilu, bet specificētās taustiņu kombinācijas ir „Delete", kas atbilst dzēšanas operācijai, un „Enter", kas atbilst dialogu loga atvēršanas operācijai.



1.14. att. Blokshēmu redaktora specifikācija

2 Konfigurators

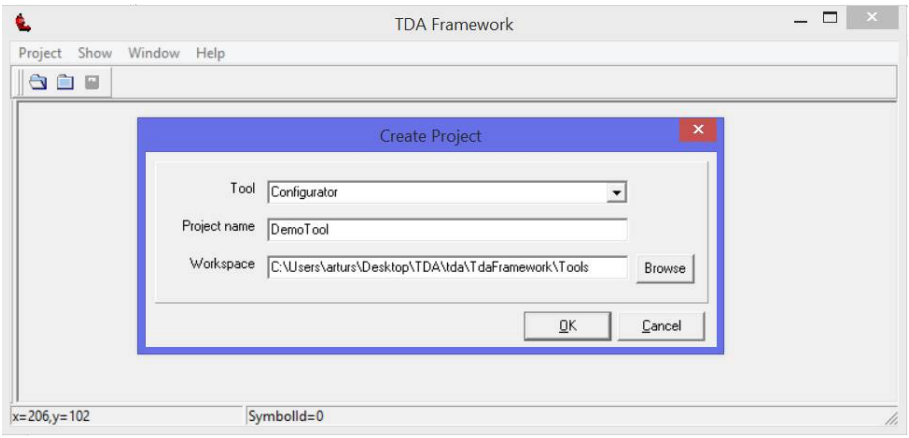
Kā iepriekš noskaidrojām, ar rīku definēšanas metamodeļa instancēm var specificēt DSML rīkus un pēc tam šo specifikāciju izmantot, lai to pārvērstu strādājošā rīkā. Tādējādi par aktuālu problēmu DSML rīku izstrādē kļūst rīku definēšanas metamodeļa instanču radīšana, un tas, cik ātri un ērti tās var radīt, ietekmē DSML rīku izstrādes laiku.

Šim nolūkam ir izstrādāts konfigurators jeb rīks, ar kuru tiek radītas un uzturētas rīku definēšanas metamodeļa instances. Konfigurators ir izstrādāts kā grafisks DSML rīks (kad lejupielādē platformu, konfigurators jau pēc noklusējuma ir iekļauts platformā), ar kuru jauni DSML rīki tiek specificēti kā konfiguratora grafiskie modeļi. Konfiguratora grafisko valodu var uzskatīt par augstāku rīku definēšanas metamodeļa abstrakciju, un tās galvenā priekšrocība ir daudz kompaktākie modeļi salīdzinājuma ar UML klašu diagrammām, jo elementu stili tiek definēti ar tā izskatu un daļa informācijas tiek ievadīta caur dialogu logu formām.

Šādas rīku izstrādes ieguvums ir tas, ka konfigurators automātiski savus modeļus saglabā kā rīku definēšanas metamodeļa instances, un tā rezultātā tas nepieprasa rīku izstrādātājiem pārzināt rīku definēšanas metamodeli, dažādas tā noklusētās vērtības, kā arī zināt kādi modeļi ir korekti un kādi nav. Tādējādi, specificējot rīkus ar konfiguratoru, tiek ievērojami samazināts rīku definēšanas metamodeļa instances izstrādes laiks, un tiek izslēgta iespējamība uzbūvēt tādu interpretāciju, kuru neatbalsta interpretators.

2.1 Konfiguratora valoda

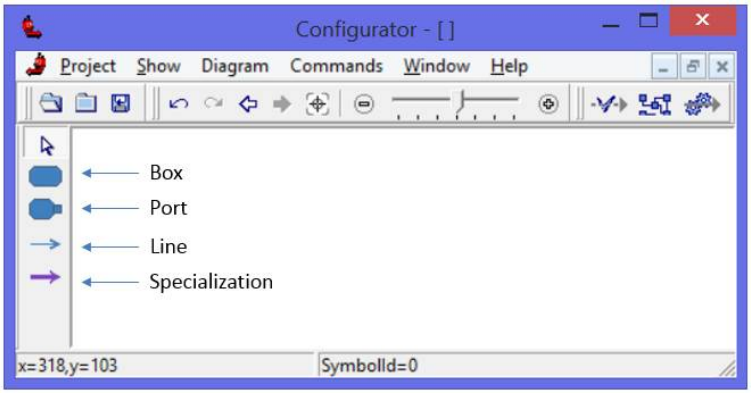
Jauna DSML rīka konfigurēšana sākas ar jauna projekta izveidošanu, ar 2.1. attēlā redzamo logu.



2.1. att. Jauna projekta izveidošana

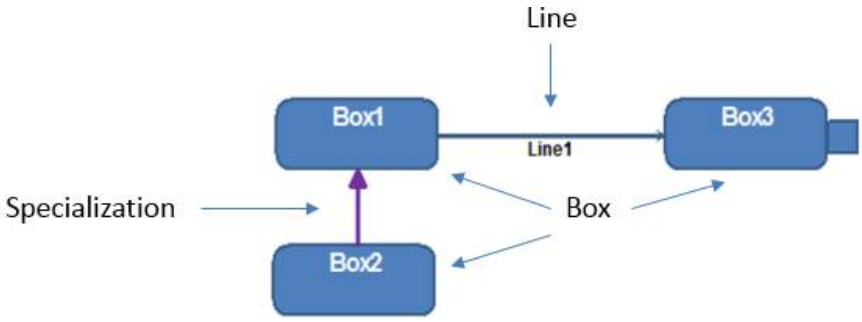
Lai izvēlētos konfiguratoru, laukā *Tool* lietotājam ir jānorāda „Configurator”, laukā *Project Name* ir jāievada projekta nosaukums, kas sakritīs ar izstrādājamā rīka nosaukumu, piemēram,

„DemoTool”, bet laukā *Workspace* ir jānorāda projekta atrašanās vieta platformas *Tools* direktorijā. Pēc jauna projekta izveides atveras projektu diagramma, bet, nospiežot taustiņu „C”, atveras 2.2. attēlā redzamā konfiguratora diagramma.



2.2. att. Konfiguratora diagramma

Jaunus elementus konfiguratora diagrammā veido ar paleti, kura sastāv no četrām pogām, un ar katru pogu var izveidot kādu no 2.3. attēlā redzamajiem konfiguratora grafiskajiem elementiem – *Box, Line, Port* vai *Specialization.*

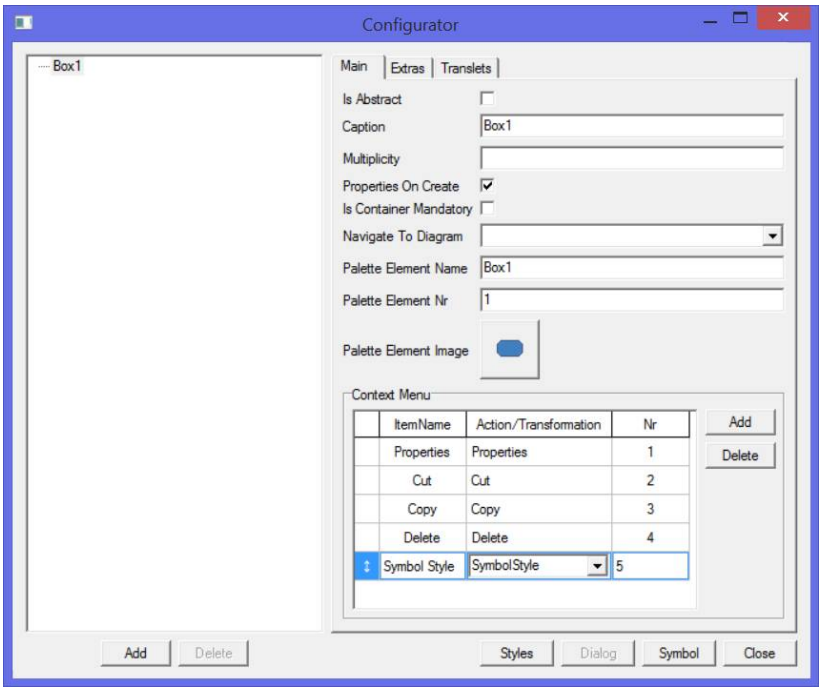


2.3. att. Konfiguratora grafiskās valodas elementi

Katrs valodas elements, izņemot *Specialization*, specificē kādu izstrādājamā DSML rīka grafisko elementu, respektīvi, elements *Box* specificē kastes tipa elementus, *Line* specificē līnijas tipa elementus un *Port* specificē porta tipa elementus.

2.1.1 Box

Box elementu veidošana notiek ar paletes elementu *Box*, un tā vērtības tiek ievadītas caur dialogu logu, kas ir redzams 2.4. attēlā.



2.4. att. *Box* dialogu loga cilne *Main*

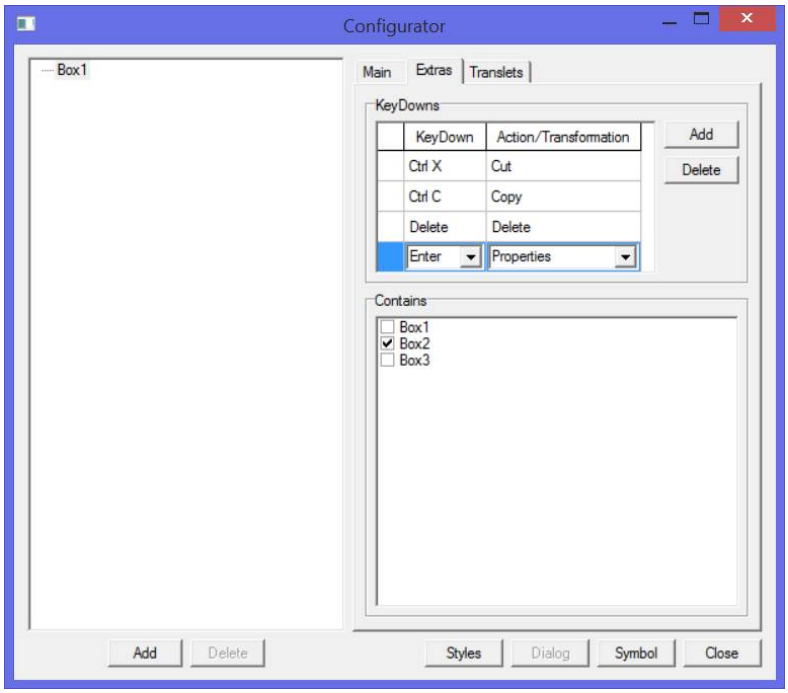
Dialogu loga kreisajā pusē atrodas koks ar elementa un tā atribūtu nosaukumiem (pagaidām neviena atribūta nav), bet labajā pusē atrodas trīs cilnes – *Main, Extras* un *Translets*. Cilnes *Main* laukā *Caption* ir jānorāda elementa nosaukums (jābūt diagrammā unikālam). Laukos *Palette Element Name, Palette Element Nr* un pogu *Palette Element Image* var ievadīt atbilstošā paletes elementa īpašības – nosaukumu, numuru paletē un paletes ikonas adresi. Ja kādu iemeslu dēļ jaunu elementu nevajag veidot ar paletes elementu, tad lauks *Palette Element Name* ir jāatstāj tukšs un atbilstošais paletes elements netiks izveidots.

Laukā *Open Properties On Element Create* var norādīt, vai pēc jauna elementa izveidošanas nepieciešams atvērt dialogu logu, bet laukā *Is Container Mandatory* var norādīt, ka šī tipa kastēm ir vienmēr jāatrodas ievietotām kādā kastē. Lai norādītu maksimāli pieļaujamo viena tipa kastu skaitu diagrammā, laukā *Multiplicity* ir jānorāda to ierobežojošais skaits. Ja šis lauks ir atstāts tukšs, tad maksimālais ierobežojums nepastāv. Lauks *Navigate To Diagram* ļauj norādīt, ka, veidojot jaunu kasti, kastei vēl nepieciešams piesaistīt jaunu diagrammu. Piemēram, diagrammas piesaistīšana ir nepieciešama elementu detalizēšanai ar diagrammu.

Tabulā *Context Menu* var definēt kastes uznirstošās izvēlnes operācijas. Katra tabulas rindiņa atbilst vienai uznirstošās izvēlnes operācijai, un katra operācija tiek specificēta, ievadot operācijas nosaukumu, norādot vai nu izpildāmo darbību (izpildāmās darbībās tiek piedāvātas no saraksta), vai transformācijas nosaukumu (tā, kas realizēs šo operāciju), kā arī operācijas kārtas numuru uz izvēlnes.

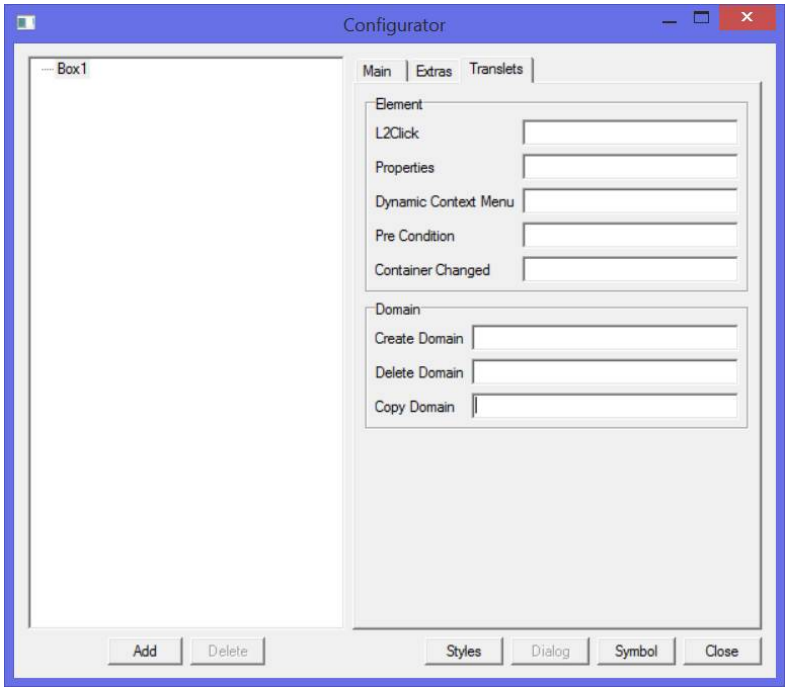
Cilnē *Extras*, kas ir redzama 2.5. attēlā, rīka izstrādātājs var ievadīt īpašības, kuras tiek reti lietotas, vai arī reti mainītas. Cilne sastāv no divām daļām - tabulas *KeyDowns* un saraksta *Contains*. Tabulā *KeyDowns* var norādīt elementam piekārtotās taustiņu kombinācijas un tām atbilstošās transformācijas, kas tiek izpildītas pēc attiecīgās taustiņu kombinācijas ievades. Katra taustiņu kombinācija atbilst vienai tabulas rindiņai un pēc noklusējuma katrai kastei ir piesaistītas četras taustiņu kombinācijas „Ctrl X", „Ctrl C", „Delete" un „Enter" ar tām atbilstošajām noklusētajām darbībām - izgriezt, kopēt, dzēst un atvērt dialogu logu.

Sarakstā *Contains* ir norādīti visi diagrammā esošie kastes tipa elementi. Ja kādu saraksta elementu ieķeksē, tad ar to tiek norādīts, ka specificējamā tipa kaste drīkst saturēt ieķeksētā tipa kastes. Piemēram, šajā logā ir norādīts, ka *Box1* tipa kastes drīkst saturēt *Box2* tipa kastes.



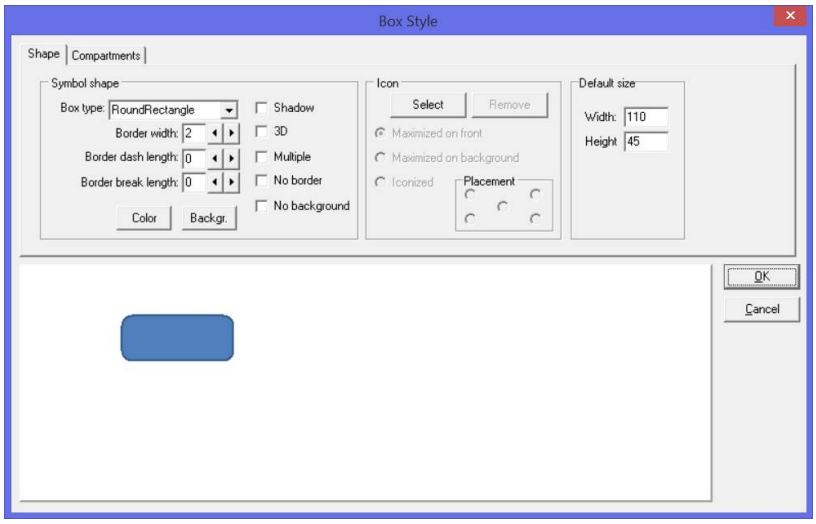
2.5. att. *Box* dialogu loga cilne *Extras*

Cilne *Translets* ir redzama 2.6. attēlā. Šīs cilnes laukos var norādīt elementa paplašinājuma punktu transformācijas, kur lauks „L2Click" atbilst paplašinājuma punktam *DynamicDoubleClick*, „Properties" atbilst *Properties*, „Dynamic Context Menu" atbilst *DynamicPopUp*, „Pre Condition" atbilst *PreCondition*, „Container Changed" atbilst *ContainerChanged*, „Create Domain" atbilst *CreateElementDomain*, „Delete Domain" atbilst *DeleteElementDomain* un „Copy Domain" atbilst *CopyDomain*.



2.6. att. *Box* dialogu loga cilne *Translets*

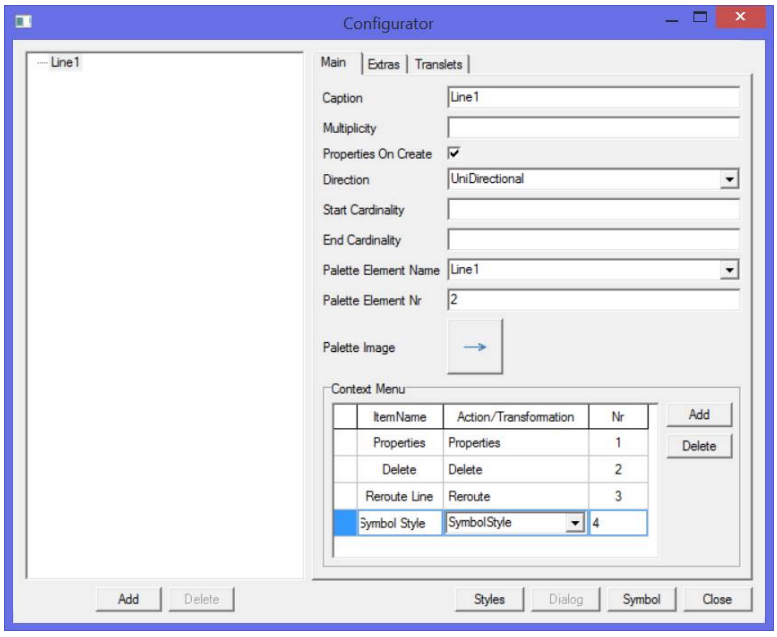
Pēc kastes īpašību ievadīšanas ir jāspecificē kastes izskats. To var izdarīt nospiežot pogu *Symbol*, kas atver 2.7. attēlā redzamo dialoga logu. Cilnē *Shape* ir paredzēta kastes stila definēšana un tā ir sadalīta četrās grupās – *Symbol shape, Icon, Default size* un lauks, kurā var redzēt kastes izskatu. Svarīgākā grupa kastes stila definēšanā ir *Symbol shape*. Šīs grupas lauks *Box type* ļauj norādīt kastes formu, piedāvājot sarakstu ar iepriekš definētām formām, piemēram, taisnstūri, elipsi, trapeci utt. Laukā *Border width* ir jānorāda kasti ietverošā rāmja līnijas biezums, kas pēc noklusējuma ir „1”, bet vispārīgā gadījumā – jo lielākā vērtība, jo platāka līnija. Lauki *Border dash length* un *Border break length* ļauj norādīt, ka ietverošā rāmja līnija ir raustīta. Pēc noklusējuma abu lauku vērtība ir „0”, kas norāda, ka līnija nav raustīta. Ja kādu no šīm vērtībām palielina, tad līnija kļūst raustīta. *Border dash length* ļauj norādīt, cik gara ir raustītā līnijas, bet *Border break length* ļauj norādīt, cik garš ir tukšums uz līnijas. Ar pogu *Color* var uzstādīt ietverošā rāmja līnijas krāsu, bet ar pogu *Backgr.* var uzstādīt kastes krāsu. Pārējie lauki ļauj uzstādīt dažādus papildus specefektus. Grupa *Icon* ļauj kasei piesaistīt bildi. Ar pogu *Select* var bildi pievienot, bet ar *Remove* noņemt. Piesaistītās bildes pozīciju diagrammā var norādīt *Placement* apakšgrupā kopā ar iezīmētu *Iconized* lauku. Ja ir iezīmēts *Maximized on front*, tad bilde pārklāj visu kasti ar visiem atribūtiem, bet, ja ir iezīmēts *Maximized on background*, tad bilde pārklāj visu kasti, atribūtus atstājot virspusē. Savukārt grupa *Default size* ir paredzēta noklusētā kastes platuma un augstuma uzstādīšanai.



2.7. att. Box stila dialogu logs

2.1.2 Line

Ja izvēlas paletes elementu *Line* un pēc tam izveido līniju, kas savieno divus citus diagrammas elementus, tad ar to tiek pateikts, ka par šī tipa līnijas galapunktiem drīkst būt norādītā tipa elementi, bet līnijas īpašības ievada caur 2.8. attēlā redzamo dialogu logu.

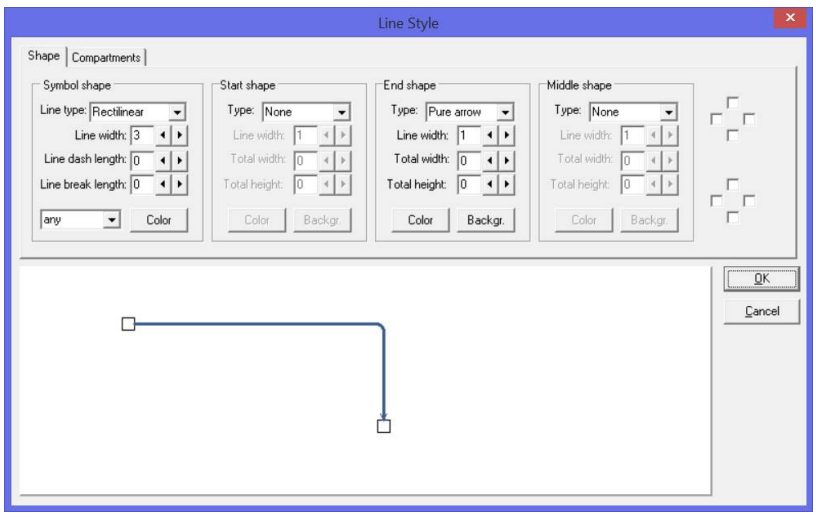


2.8. att. Line dialogu loga cilne *Main*

Lai arī *Line* dialogu loga *Main* cilne ir līdzīga ar *Main* cilni *Box* elementam, tām ir vairākas atšķirības. Pirmkārt, nav lauku *Is Container Mandatory* un *Navigate To Diagram*. Otrkārt, ir jauns lauks *Direction*. Šis lauks ļauj norādīt līnijas orientāciju, un pieļaujamās vērtības ir – „UniDirectional", „BiDirectional" un „ReverseBiDirectional" (tās atbilst iepriekš skaidrotajām klases *EdgeTpye* atribūta *direction* vērtībām). Treškārt, ir jauni lauki *Start Cardinality* un *End Cardinality*, kas norāda līnijas galu elementiem maksimālo izejošo un ieejošo viena tipa līniju skaitu. Ceturtkārt, *Palette Element Name* lauka tips ir izkrītošā izvēlne, nevis vienkāršs teksta lauks. Izvēlne sastāv no visu valodā esošo līniju paletes elementiem, un, izvēloties kādu no tiem, tiek norādīts, ka ar izvēlēto paletes elementu tiks veidotas specificējamā tipa līnijas, respektīvi, ar šo mehānismu ir iespējams norādīt, ka dažādu tipu līnijas tiek veidotas ar vienu paletes elementu. Savukārt, ja laukā ievada izvēlnē neesošu nosaukumu, tad tiek izveidots jauns paletes elements.

Cilnēs *Extras* un *Translets* ievadāmā informācija ir līdzīga *Box* dialogu logos ievadāmajai. Cilnē *Extras* var norādīt elementam piekārtotās taustiņu kombinācijas, bet cilnē *Translets* var norādīt transformācijas, kas attiecas uz līnijas paplašinājuma punktiem.

Line stilu definēšanas logs ir redzams 2.9. attēlā. Tā cilne *Shape* sastāv no grupām *Symbol shape, Start shape, End shape, Middle shape* un lauka, kurā var redzēt līnijas izskatu. Grupa *Symbol shape* attiecas tieši uz līnijas stila definēšanu. Laukā *Line type* var norādīt līnijas tipu, laukā *Line width* var norādīt līnijas biezumu, bet laukos *Line dash length* un *Line break length* var norādīt, ka līnija ir raustīta, bet ar pogu *Color* var norādīt līnijas krāsu. Grupās *Start shape, End shape* un *Middle shape* ir lauks *Type*, kas ļauj norādīt līnijas sākuma, beigu un vidus simbolu, piemēram, bultu, trijstūri, riņķi utt. *Line width* ļauj norādīt simbolu ierobežojošās līnijas biezumu, bet *Total width* un *Total height* norāda simbola platumu un garumu. Poga *Color* ļauj norādīt simbolu ierobežojošās līnijas krāsu, bet *Backgr.* ļauj norādīt simbola krāsu.



2.9. att. Line stila dialogu logs

2.1.3 Port

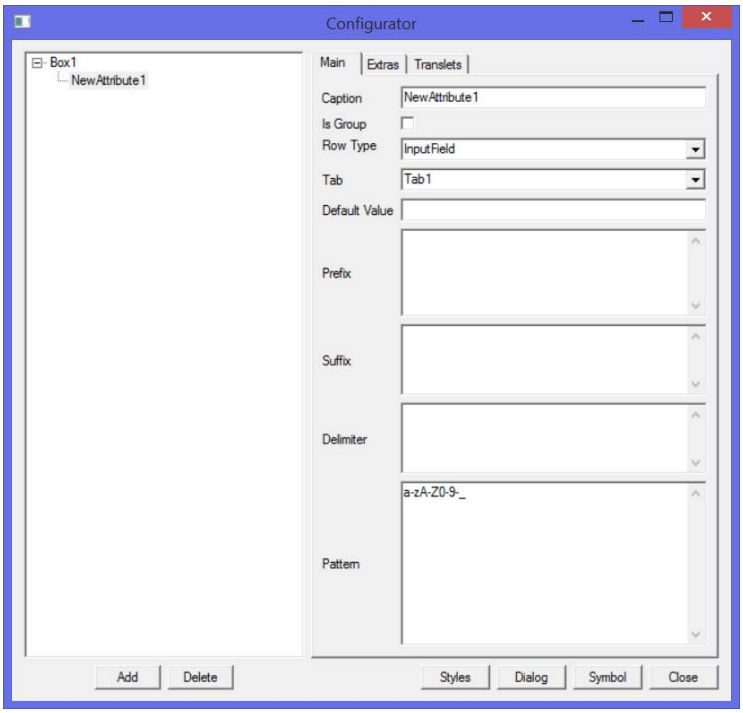
Ja nospiež paletes pogu *Port* un pēc tam nospiež uz kāda *Box* elementa, tad tiek izveidots jauns *Port* elements (*port* tipa elementiem ir vienmēr jābūt piesaistītiem kādam kastes tipa elementam), un ar to tiek pateikts, ka jaunajā rīkā tiek izveidots jauns porta tipa elements, kuru drīkst pievienot šī tipa kastei. Tā kā šī elementa īpašībās ievada caur dialogu logu, kura struktūra sakrīt ar citu elementu dialogu logu struktūru, un stilu definēšanas logs ir ļoti līdzīgs ar *Box* stila logu, tad šos logus detalizētāk neaplūkosim.

2.1.4 Specialization

Ja nospiež paletes pogu *Specialization* un pēc tam izveido līniju, kas savieno divus diagrammas elementus, tad tiek izveidots jauns *Specialization* elements. Ar šo elementu tiek norādīts, ka viens diagrammas elements ir apakštips otram.

2.1.5 Atribūtu specificēšana

Atribūtu specificēšana visiem elementiem neatkarīgi no to tipa ir vienāda un tādēļ apskatīsim tikai atribūta specificēšanu kastes gadījumā. Jauni atribūti tiek pievienoti ar pogu *Add*, un tā rezultātā zem iepriekš aktīvās koka virsotnes tiek izveidota jauna koka virsotne, kura automātiski aktivizējas, un pēc tam parādās 2.10. attēlā redzamais dialogu logs atribūta īpašību ievadīšanai.

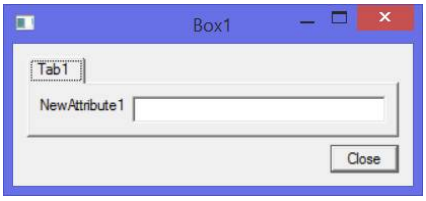


2.10. att. Atribūta dialogu loga cilne *Main*

Līdzīgi kā elementiem, atribūtiem laukā *Caption* ir jānorāda tā nosaukums (viena līmeņa atribūtiem jābūt unikāliem). Atribūta dialogu logā tiek norādītas divu veidu īpašības. Vienas specificē atribūta vērtību veidošanu, piemēram, noklusēto vērtību vai prefiksu, bet otras specificē informāciju dialogu logu veidošanai. Lauki *Row Type* un *Tab* attiecas uz dialogu logu veidošanu. Laukā *Row Type* ir jānorāda ievadlauka tips atribūta vērtību ievadīšanai. Šim laukam ir izkrītošās izvēlnes tips, kur izvēlne sastāv no 1.1. tabulā uzskaitītajiem ievadlauku nosaukumiem.

Savukārt, ja nepieciešams dialoga loga komponentes grupēt, tad to var izdarīt ar cilnēm. Laukā *Tab* ir jānorāda cilnes nosaukums. Gadījumā, ja norādītais cilnes nosaukums jau eksistē, tad atribūta lauks tiek piesaistīts jau esošajai cilnei. Gadījumā, ja norādītais cilnes nosaukums neeksistē, tad tiek izveidota jauna cilne un atribūta lauks tiek piesaistīts no jauna izveidotajai cilnei.

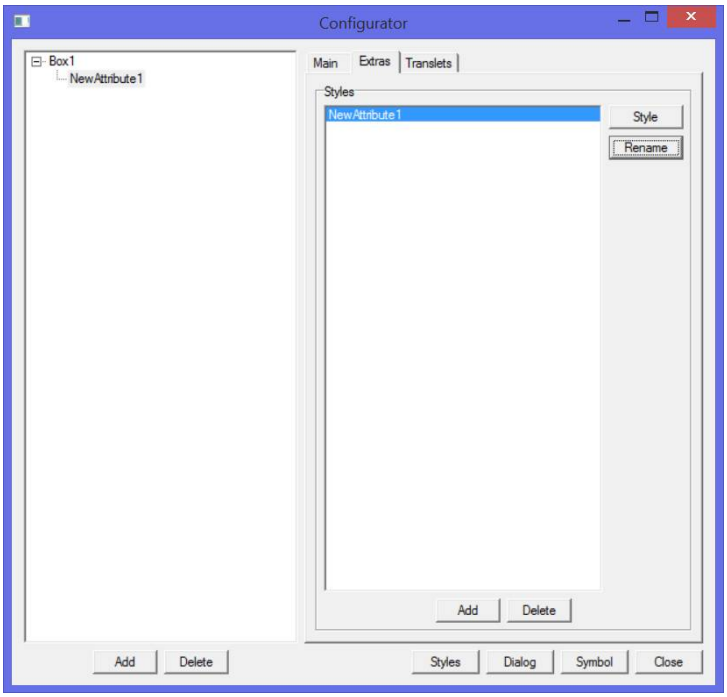
Piemēram, 2.10. attēlā redzamajā dialogu logā ir norādīts, ka atribūta „NewAttribute1" ievadlauka tips ir „InputField" un cilnes nosaukums ir „Tab1". Šai specifikācijai atbilstošais dialogu logs ir redzams 2.11. attēlā.



2.11. att. Specificētais dialogu logs

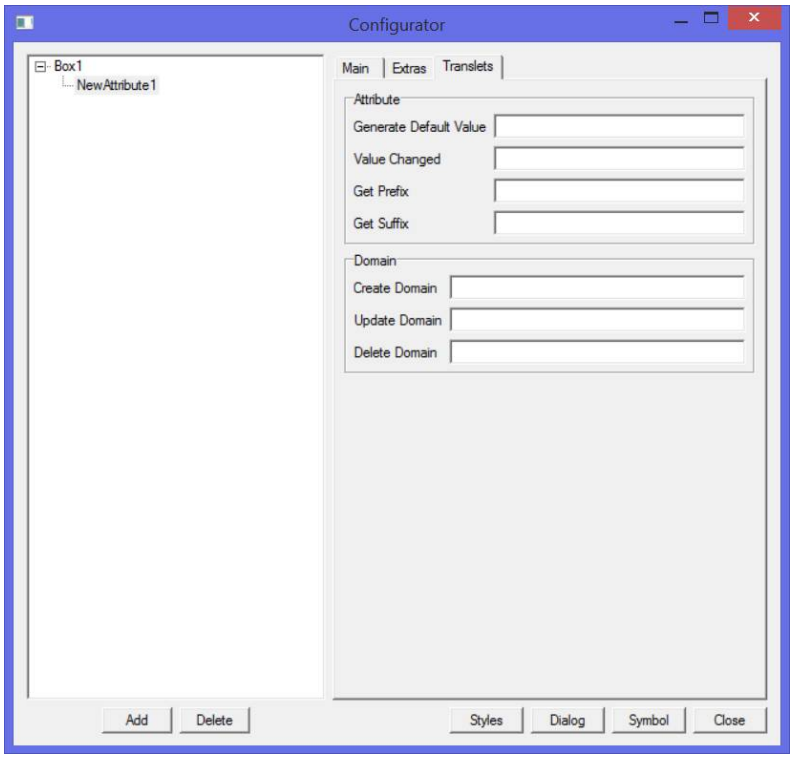
Atlikušo lauku vērtības atbilst klases *CompartType* atribūtu vērtībām. Laukā *Default Value* ir iespējams norādīt atribūta noklusēto vērtību, kas metamodelī atbilst klases *CompartType* atribūta *defaultValue* vērtībai, laukos *Prefix* un *Suffix* var norādīt atribūta prefiksa un sufiksa vērtības un tās atbilst atribūtu *prefix* un *suffix* vērtībām, laukā *Delimiter* var norādīt apakšatribūtu atdalītāja vērtību un tā atbilst atribūta *delimiter* vērtībai, laukā *Pattern* var norādīt atribūta vērtībā pieļaujamos simbolus un tas atbilst atribūta *pattern* vērtībai.

Cilne *Extras* ir redzama 2.12. attēlā. Šajā cilnē atribūtam var piesaistīt jaunus stilus.



2.12. att. Atribūta dialogu loga cilne *Extras*

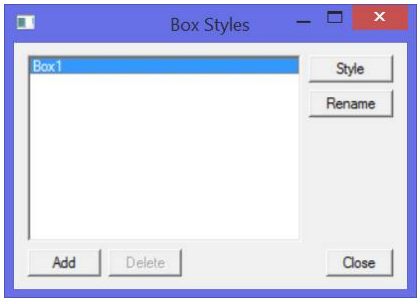
Cilne *Translets* ir redzama 2.13. attēlā. Šīs cilnes laukos var norādīt atribūta paplašinājuma punktu transformācijas, kur lauks „Generate Default Value" atbilst *GenerateDefaultValue*, „Value Changed" atbilst *ValueChanged*, „Get Prefix" atbilst *GetPrefix*, „Get Suffix" atbilst *GetSuffix*, „Create Domain" atbilst *CreateCompartmentDomain*, „Update Domain" atbilst *UpdateCompartmentDomain* un „Delete Domain" atbilst *DeleteCompartmentDomain* paplašinājuma punktam.



2.13. att. Atribūta dialogu loga cilne *Translets*

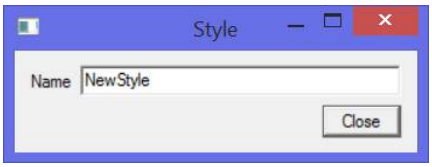
2.1.6 Stilu pievienošana

Katram elementam un atribūtam pēc noklusējuma ir piesaistīts tieši viens stils. Gadījumā, ja ir vajadzība pēc vairākiem stiliem, tad elementa dialogu logā ir jānospiež *Styles* poga, bet atribūta gadījumā jāpārslēdzas uz cilni *Extras*. Tā rezultātā priekš elementa parādīsies 2.14. attēlā redzamais logs, bet priekš atribūta 2.11. attēlā redzamais logs.



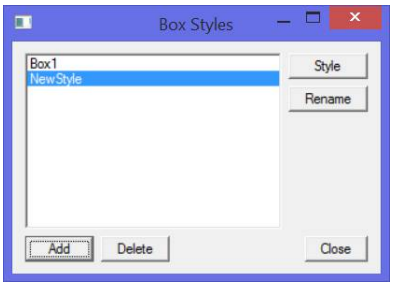
2.14. att. Stila specificēšanas logs

Tā kā gan elementu, gan atribūtu stilu pievienošana notiek pēc viena principa, tad apskatīsim tikai elementa stila pievienošanu. Lai pievienotu jaunu stilu, ir jāspiež poga *Add* un tad parādīsies 2.15 attēlā redzamais logs, kurā ir jānorāda stila nosaukums.



2.15. att. Stila nosaukuma ievade

Stāvoklis pēc stila pievienošanas ir redzams 2.16. attēlā.



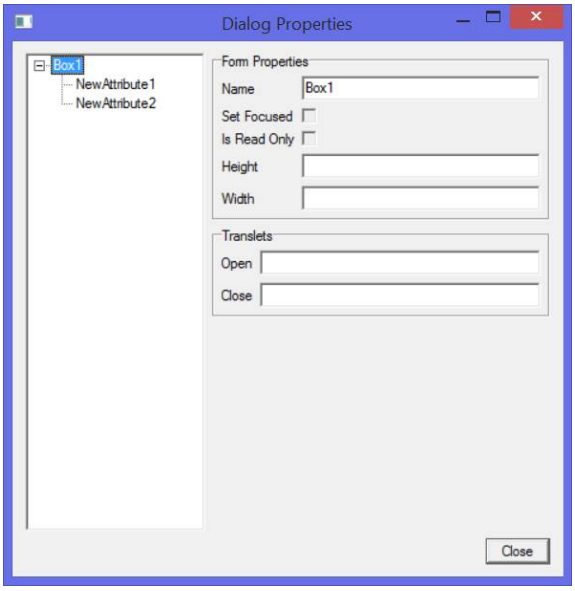
2.16. att. Stila specificēšanas logs pēc stila pievienošanas

Lai uzstādītu stila parametrus, ir jānospiež poga *Style* un tad parādīsies elementa tipam atbilstošais dialogu logs, piemēram, ja elements ir *Box*, tad parādīsies 2.7. attēlā redzamais logs, vai, ja elements ir *Line*, tad parādīsies 2.9. attēlā redzamais logs.

2.1.7 Dialogu logu konfigurēšana

Ja elementu dialogu logā nospiež pogu *Dialog*, parādās 2.17. attēlā redzamais dialogu logs. Dialogu loga kreisajā pusē ir koks, kura saturs atbilst specificētā elementa dialogu loga struktūrai. Piemēram, attēlā redzamā dialogu loga koks reprezentē formu ar diviem laukiem „NewAttribute1" un „NewAttribute2".

Sākotnējā dialogu logu komponenšu secība atbilst to veidošanas secībai, bet, ja nepieciešams secību mainīt, to var izdarīt, ar peli mainot koka virsotnes pozīciju viena vecāka ietvaros.

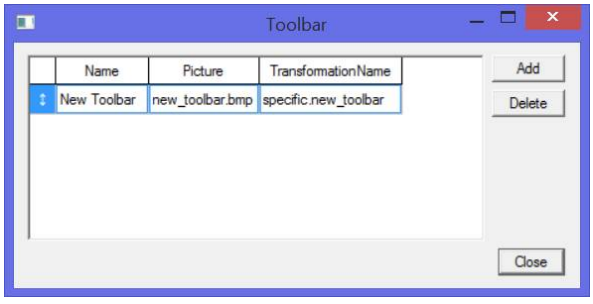


2.17. att. Dialogu loga konfigurēšanas logs

Dialogu loga labajā pusē var norādīt aktīvās dialogu loga komponentes īpašības. Piemēram, attēlā redzamajā logā aktīvā komponente ir forma. Grupā *Form Properties* var mainīt dažādus komponentes parametrus, bet grupā *Translets* var norādīt komponentes paplašinājuma punktu transformācijas, kas atbilst 3.1. tabulā aprakstītajiem.

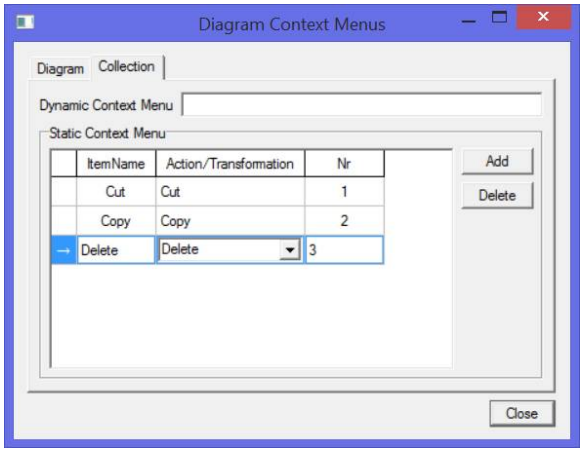
2.1.8 Diagrammas funkcionalitāte

Nospiežot peles labo pogu uz tukšas vietas konfiguratora diagrammā, parādās uznirstošā izvēlne, kuras operācijas – *Add Toolbar, Add Context Menu* un *Add Key Downs* specificē diagrammas funkcionalitāti. Ja no izvēlnes izvēlas operāciju *Add Toolbar*, tad parādās 2.18. attēlā redzamais dialogu logs. Šajā logā katra tabulas rinda specificē vienu rīku joslas pogu. Piemēram, attēlā redzamajā logā ir specificēta rīku josla ar vienu pogu „New Toolbar”.



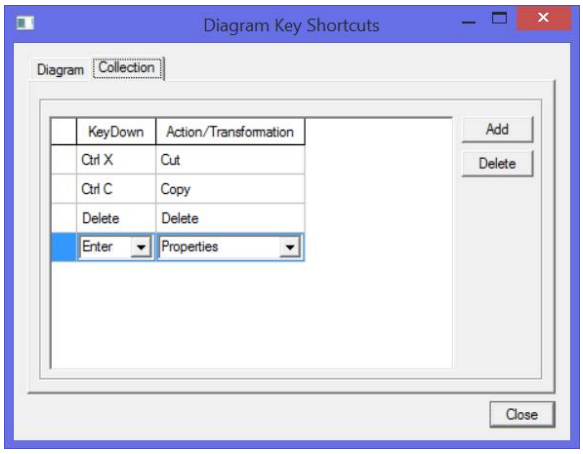
2.18. att. Rīku joslas specificēšanas logs

Savukārt, ja no sākotnējās uznirstošās izvēlnes izvēlas operāciju *Add Context Menu*, tad parādās 2.19. attēlā redzamais logs. Šajā logā var specificēt diagrammas tipa uznirstošās izvēlnes un tiek izšķirtas divas situācijas. Vienā situācijā uznirstošā izvēlne tiek specificēta tukšai kolekcijai un šai situācijai atbilst cilne *Diagram*, otrā situācijā uznirstošā izvēlne tiek specificēt vairāku elementu kolekcijai un šai situācijai atbilst cilne *Collection*. Attēlā redzamajā logā ir specificēta statiskā uznirstošā izvēlne ar trīs operācijām vairāku elementu kolekcijai, bet, lai specificētu dinamisku uznirstošo izvēlni, laukā *Dynamic Context Menu* ir jānorāda atbilstošās transformācijas nosaukums.



2.19. att. Uznirstošās izvēlnes specificēšanas logs

Visbeidzot, ja no izvēlnes izvēlas operāciju *Add Key Downs*, tad parādās 2.20. attēlā redzamais dialogu logs. Šajā logā var specificēt diagrammai piesaistītās taustiņu kombinācijas un arī šajā gadījumā tiek izšķirtas divas situācijas. Vienā gadījumā taustiņu kombinācijas tiek specificētas tukšai kolekcijai un šai situācijai atbilst cilne *Diagram*, otrā gadījumā taustiņu kombinācijas tiek specificētas vairāku elementu kolekcijai un šai situācijai atbilst attēlā redzamā cilne *Collection*, kura specificē trīs taustiņu kombināciju apstrādi.



2.20. att. Taustiņu kombināciju specificēšanas logs

2.1.9 Blokshēmu redaktora konfigurēšanas apraksts

Šajā nodaļā aplūkosim, kā ar konfiguratoru var specificēt to pašu blokshēmu redaktoru, kurš tika specificēts ar rīku definēšanas metamodeļa instancēm 1. nodaļā. Blokshēmu redaktoru specificēšana sākas ar jauna platformas projekta izveidi, kas tiek izdarīts norādot attiecīgo informāciju 2.1. attēlā redzamajā dialogu logā. Pēc tam, nospiežot taustiņu „C", atveras konfiguratora diagramma.

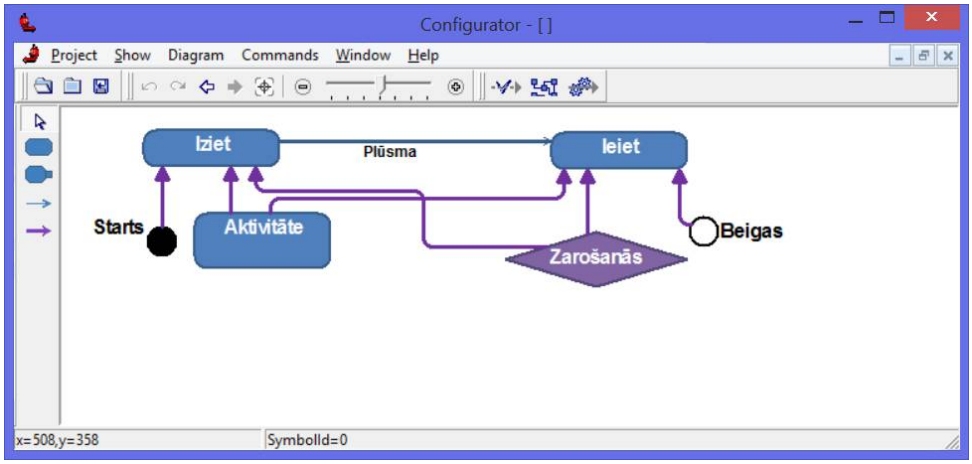
Šajā diagrammā vispirms specificēsim kastes tipa elementu, kas atbilst aizmetnim. Lai to izdarītu, ir jāizveido jauns *Box* tipa elements. Tā rezultātā atvērsies *Box* dialogu logs (2.4. attēls), kurā ir jāievada aizmetņa elementu specificējošās vērtības. Laukos *Caption* un *Palete Element Name* ir jāievada elementa nosaukums - „Blokshēma". Laukā *Palette Element Nr* ir jānorāda paletes elementa numurs (piemēram, „1"), laukā *Palette Element Image* paletes ikonas adrese, bet laukā *Navigate To Diagram* ir jāievada nosaukums „Blokshēma", kas norāda, ka aizmetnis detalizē blokshēmu diagrammas, proti, veidojot aizmetni, interpretators automātiski izveidos aizmetnim atbilstošo blokshēmas diagrammu. Kad ir specificēts aizmetņa elements, tad, uz šī elementa veicot peles kreisās pogas dubultklikšķi, atvērsies jauna konfiguratora diagramma, kurā ir jāspecificē blokshēmas elementi (t.i., šī diagramma ir blokshēmas specifikācija).

Tā kā visu blokshēmas elementu specificēšanas ir līdzīga, detalizētāk apskatīsim „Aktivitāte" specificēšanu. Šī un citu elementu specificēšana ir tāda pati kā aizmetņa specificēšana, t.i., lai specificētu „Aktivitāte", pirmajā solī ir jāizveido *Box* elements. Tā rezultātā atvērsies *Box* dialogu logs (2.4. attēls), kurā ir jāievada elementu „Aktivitāte" specificējošās vērtības. Laukos *Caption* un *Palete Element Name* ir jāievada elementa nosaukums - „Aktivitāte", bet laukos *Palette Element Nr* un *Palette Element Image* attiecīgi paletes elementa numurs un paletes ikonas adrese.

Savukārt, ja gribam izveidot transformāciju, kas, piemēram, pārbauda, vai starta simbols jau eksistē, mums ir jāpārslēdzas uz cilni *Translets* (2.5. attēls) un laukā „Pre Condition" ir jāievada vērtība, piemēram, „Demo.CheckExistingElements".

Lai specificētu „Aktivitāte” noklusēto stilu, jānospiež poga *Symbol*. Tā rezultātā parādīsies 2.7. attēlā redzamais dialogu logs, un tajā jānorāda attiecīgie stila parametri. Pēc „Aktivitāte” funkcionalitātes un stilu uzstādīšanas, ir jāspecificē elementa „Aktivitāte” atribūts „Nosaukums”. Lai specificētu „Nosaukums”, ir jānospiež poga *Add*, un tā rezultātā parādīsies atribūta dialogu logs (2.10. attēls), kura laukā *Caption* jāievada vērtība „Nosaukums”, bet laukā *Row Type* jāizvēlas vērtība „InputField”.

Pārējo blokshēmas redaktora elementu specificēšana ir līdzīga, atšķirīgāka vien ir abstrakto elementu „Iziet” un „Ieiet” veidošana. Lai šos elementus padarītu abstraktus, ir jāieķeksē to dialogu logu lauks *Is Abstract*. Savukārt, lai norādītu, ka abstraktie elementi ir blokshēmas „īsto” elementu virstipi vēl papildus, ir jānovelk *Specialization* līnijas starp abstraktajiem un blokshēmas „īstajiem” elementiem. Rezultātā iegūtā blokshēmu redaktora konfigurācijas grafiskais modelis ir redzams 2.21. attēlā.



2.21. att. Blokshēmu redaktora konfigurācija

Kad konfigurācija ir pabeigta, ir jānospiež rīkjoslas poga *New Version Created* un pēc tam projekts ir jāsaglabā un jāaizver. Tā rezultātā ir izstrādāts blokshēmu redaktors, un, lai ar šo redaktoru varētu izveidot blokshēmu modeļus, ir jāizveido jauns blokshēmas redaktora projekts, t.i., veidojot jaunu projektu parādīsies 2.1. attēlā redzamais dialogu logs, kura laukā *Tools* ir jānorāda, ka rīks būs „BloskhēmuRedaktors”, bet laukā *Workspace* jānorāda projekta atrašanās vietā, un tad atvērsies blokshēmu redaktors, ar kuru var zīmēt blokshēmas.

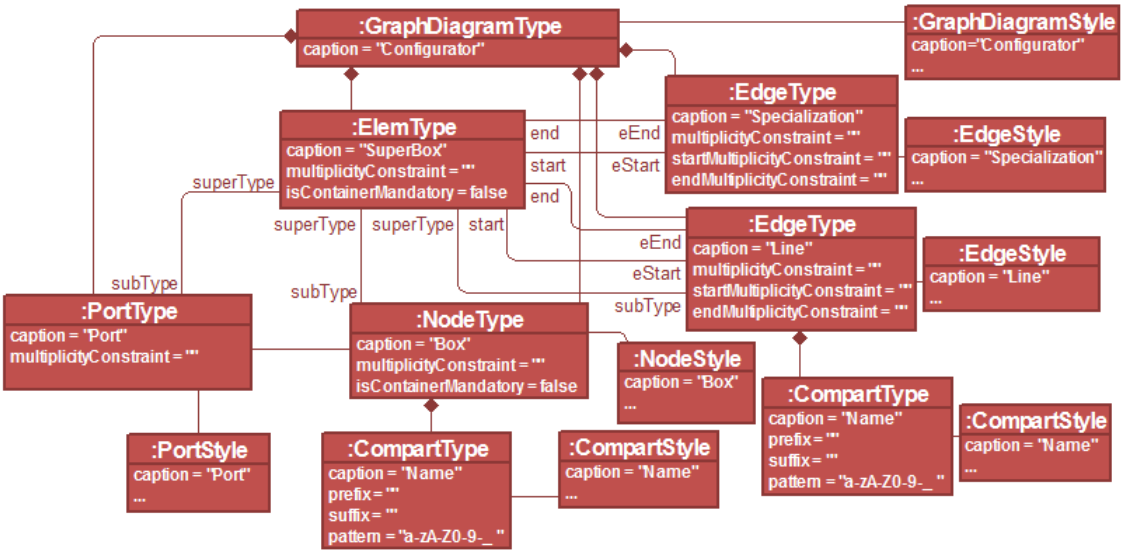
2.2 Konfiguratora realizācija

Konfigurators ir realizēts kā DSML rīks, un tā realizācijai ir izmantotas TDA komponentes, lietojot to pašu mehānismu, ko citu DSML rīku būvei. Proti, rīka specifikācijas uzdošanai ir izmantots rīku definēšanas metamodelis, kuru par strādājošu rīku pārvērš interpretators ar papildinātu funkcionalitāti (caur paplašinājuma punktiem).

2.2.1 Konfiguratora specifikācija

Kā 2.1. nodaļā tika minēts, konfiguratora valoda sastāv no četriem elementiem – *Box*, *Line*, *Port* un *Specialization*. Tādējādi, lai specificētu šos elementus, mums ir jāizveido tiem atbilstošās specifikācijas ar rīku definēšanas metamodeļa instancēm. Specifikācijas uzdošanu, līdzīgi kā iepriekš, sadalīsim divās daļās. Vispirms specificēsim konfiguratora valodu, un pēc tam rīka uzvedību.

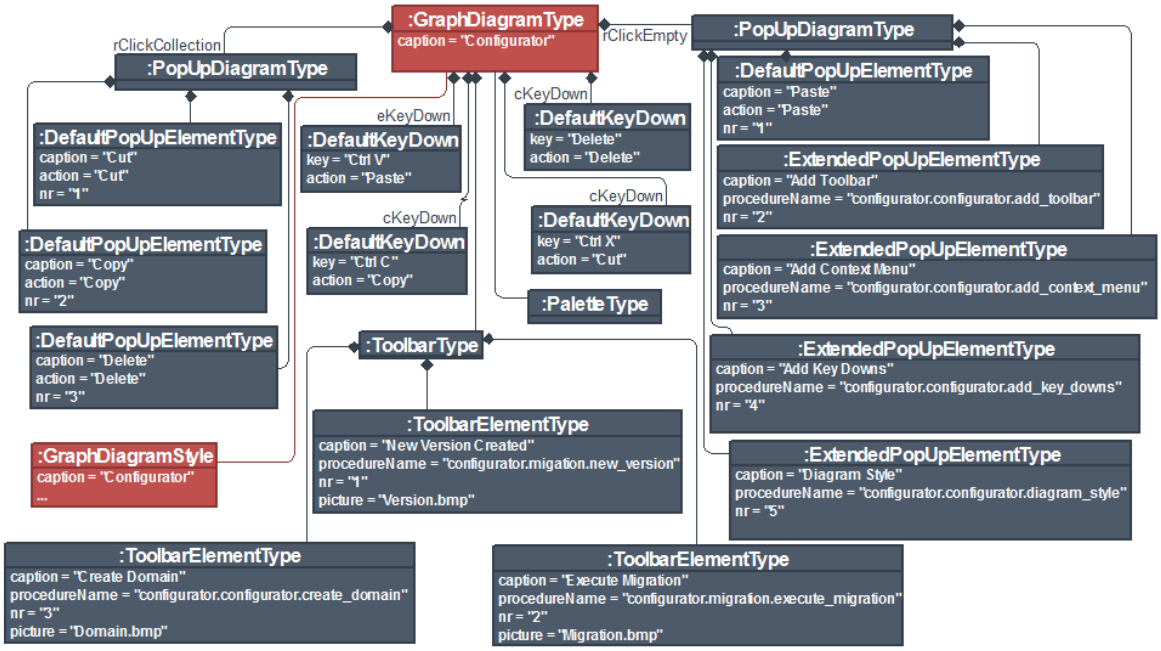
Konfiguratora valodu specificējošā instanču diagramma ir redzama 2.22. attēlā. Jāpiezīmē, ka bez minētajiem valodas elementiem šī specifikācija satur vienu abstraktu elementu „SuperBox" (klases *ElemType* instance), lai norādītu, ka *Line* un *Specialization* drīkst patvaļīgi savienot *Box*, *Line* un *Port* tipa elementus.



2.22. att. Konfiguratora valodas specifikācija

Nākamais solis ir specificēt katra valodas elementa un to diagrammas uzvedību. Sāksim ar diagrammu (skat. 2.23. attēlu). Konfiguratora diagrammās ir realizētas divu veidu uznirstošās izvēlnes. Viena izvēlne ir gadījumiem, ja lietotājs ar peles labo pogu nospiež uz vairāku elementu kolekcijas, un tā sastāv no operācijām – *Cut* (izgriezt), *Copy* (kopēt) un *Delete* (dzēst). Otra ir gadījumiem, kad lietotājs ar peles labo pogu nospiež uz tukšas vietas diagrammā, un tā sastāv no operācijām – *Paste* (ielīmēt), *Add Toolbar* (ļauj pievienot rīkjoslas pogas), *Add Context Menu* (ļauj pievienot uznirstošās izvēlnes), *Add Key Downs* (ļauj pievienot taustiņu kombināciju apstrādes) un *Diagram Style* (ļauj nomainīt diagrammas stilu). Savukārt, elementu kolekcijai ir piesaistītas šādas taustiņu kombinācijas - „Ctrl X", „Ctrl C" un „Delete" (izpilda izgriešanas, kopēšanas un dzēšanas operācijas), bet gadījumam, kad neviens diagrammas elements nav aktīvs, ir piesaistīta „Ctrl V" (izpilda ielīmēšanas operāciju).

Konfiguratora diagrammu rīkjoslā ir specificētas trīs pogas. Pogas „New Version Created" un „Execute Migration" attiecas uz modeļu datu migrēšanu (par šīm pogām detalizētāk 2.1.3. nodaļā), bet poga „Create Domain" atver dialogu logu, kurā var norādīt transformācijas nosaukumu, kas pievienos domēna metamodeli.

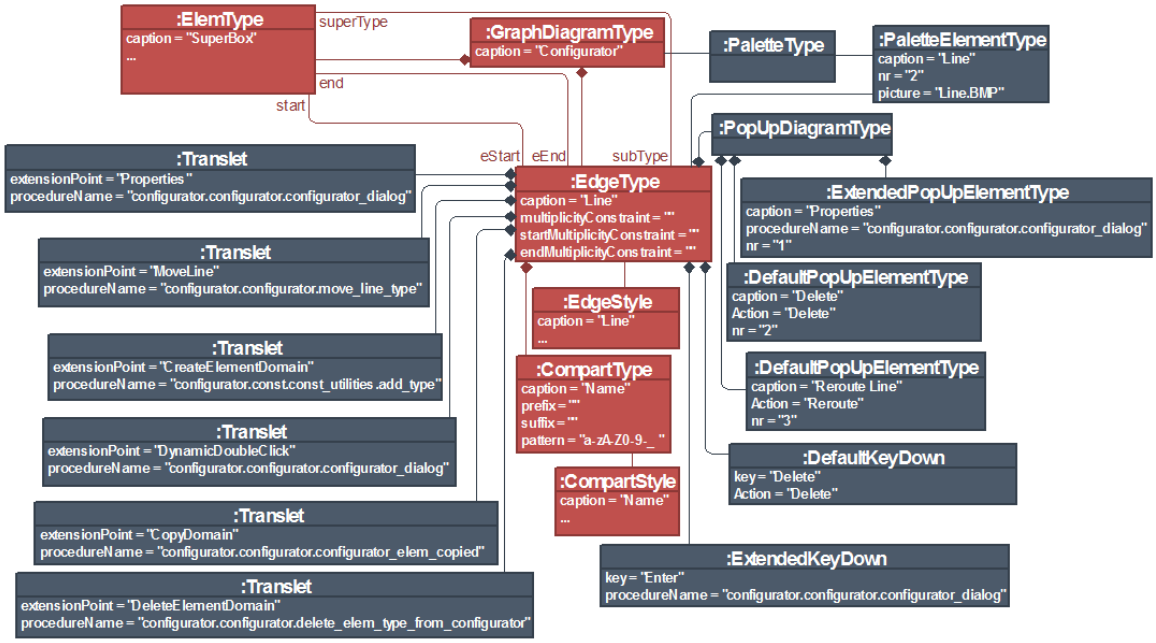


2.23. att. Konfiguratora diagrammas tipa specifikācija

Elementa *Line* specifikācija ir redzama 2.24. attēlā. Konfiguratora specifiskās funkcionalitātes nodrošināšanai *Line* ir piesaistītas sešas paplašinājumu punktu transformācijas. Paplašinājuma punktu *Properties* un *DynamicDoubleClick* transformācijas nodrošina dialogu loga (skat. 2.8. attēlu) izveidošanu, *CreateElementDomain* un *DeleteElementDomain* transformācijas attiecīgi nodrošina *Line* atbilstošo rīku definēšanas metamodeļa instanču izveidošanu un izdzēšanu, *MoveLine* transformācijas sinhronizē *Line* instanču stāvokli pēc līnijas gala pārcelšanas un *CopyDomain* transformācija nodrošina *Line* atbilstošo instanču kopēšanu.

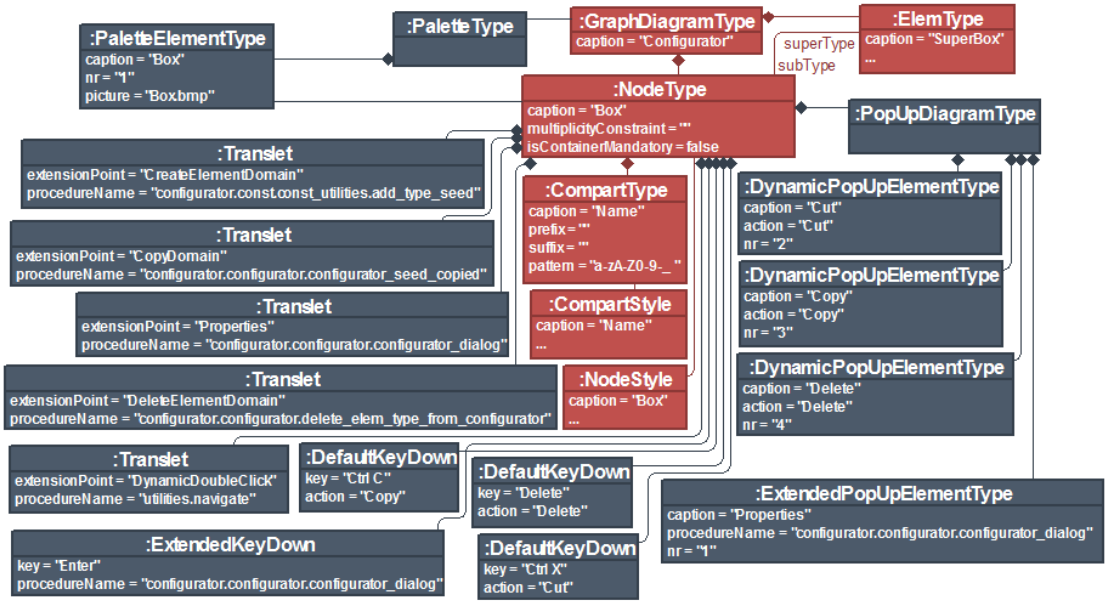
Line uznirstošā izvēlne sastāv no trīs operācijām *Delete* (dzēst), *Properties* (atvērt dialogu logu) un *Reroute Line* (pārzīmēt līniju), bet līnijai piesaistītās taustiņu kombinācijas ir „Delete" un „Enter" (izpilda dzēšanas un dialogu loga atvēršanas operācijas).

Par *Line* sākuma un beigu elementu ir norādīts abstraktais elements *SuperBox*, tādējādi nodrošinot, ka elementu *Line* ir iespējams visās kombinācijās savienot ar jebkuriem diviem *SuperBox* apakštipiem. Savukārt, lai konfiguratora diagrammā parādītu *Line* nosaukumu, elementam *Line* ir piekārtots atribūts *Name*.



2.24. att. Line specifikācija

Elementa *Box* specifikācija ir redzama 2.25. attēlā. Atskaitot paplašinājuma punkta *MoveLine* transformācijas, atlikušās piecas *Box* piesaistītās transformācijas ir tās pašas, kas elementam *Line*, un nodrošina to pašu funkcionalitāti.

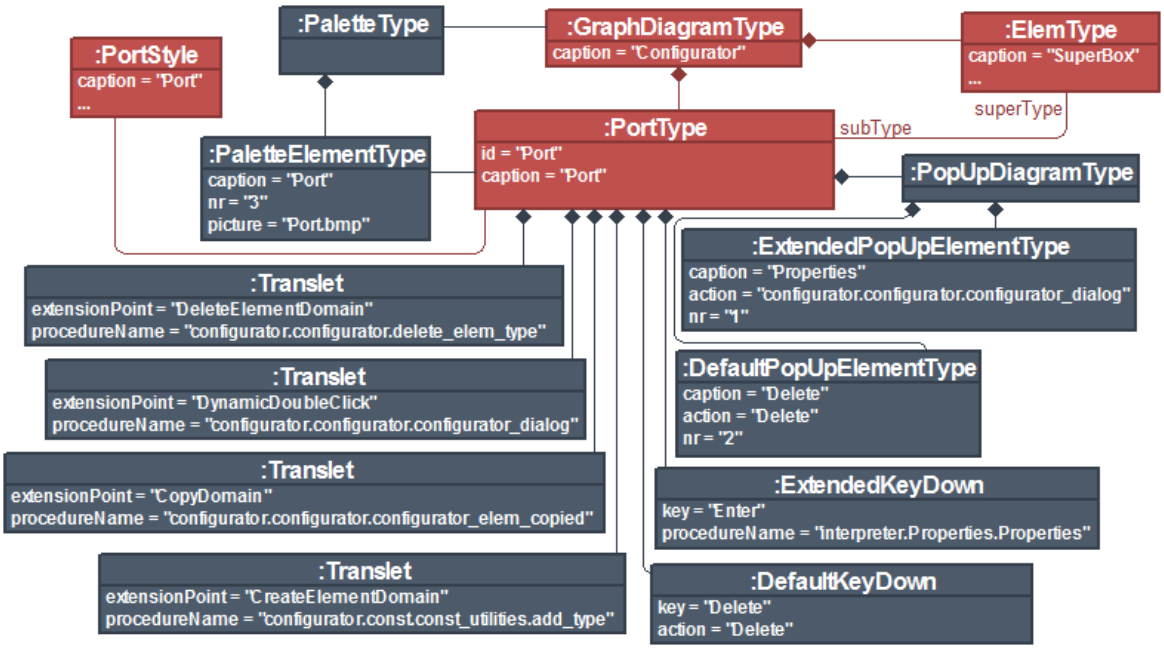


2.25. att. Box specifikācija

Box uznirstošā izvēlne sastāv no operācijām *Properties* (atvērt dialogu logu), *Cut* (izgriezt), *Copy* (kopēt) un *Delete* (dzēst), bet elementam piesaistītās taustiņu kombinācijas ir „Enter", „Ctrl X", „Ctrl C" un „Delete" (attiecīgi izpilda atvērt dialogu logu, izgriezt, kopēt un dzēst operācijas).

Box ir norādīts par *SuperBox* apakštipu, tādējādi tas manto *SuperBox* ieejošās un izejošās līnijas (t.i., iespēju būt pievienotam *Line*). Līdzīgi kā *Line*, *Box* arī ir atribūts *Name* elementa nosaukuma rādīšanai.

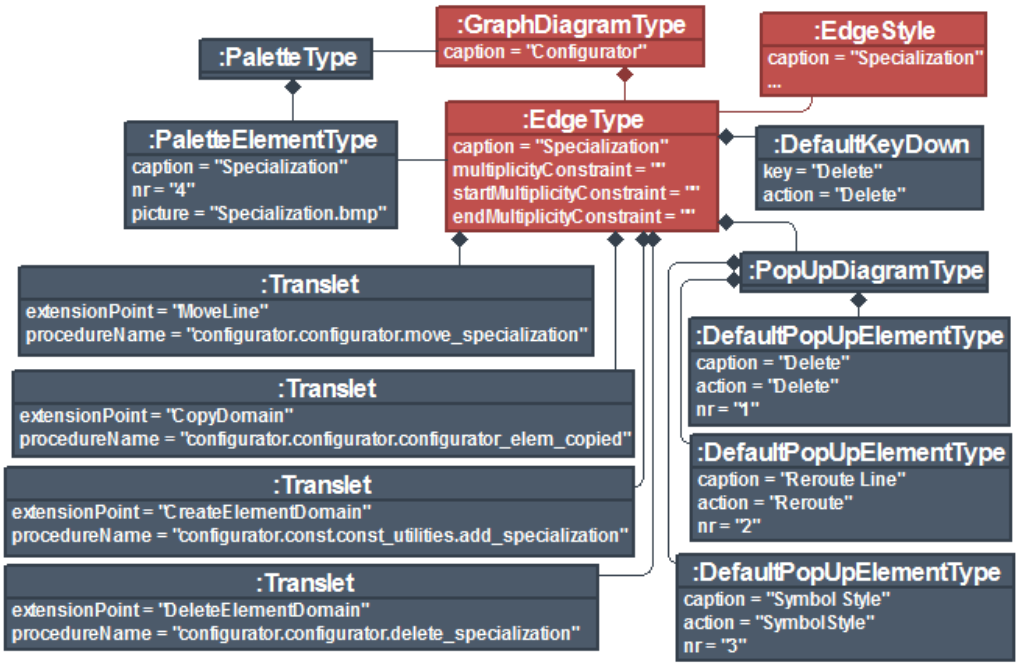
Elementa *Port* specifikācija ir redzama 2.26. attēlā. Šim elementam ir piesaistītas tās pašas paplašinājuma punktu transformācijas, kas elementam *Box*. *Port* uznirstošā izvēlne sastāv no operācijām *Properties* (atvērt dialogu logu) un *Delete* (dzēst), bet elementam piesaistītās taustiņu kombinācijas ir - „Enter" un „Delete" (attiecīgi izpilda atvērt dialogu logu un dzēst operācijas). Līdzīgi kā *Box*, *Port* ir *SuperBox* apakštips, lai varētu mantot *SuperBox* ieejošās un izejošās līnijas.



2.26. att. Port specifikācija

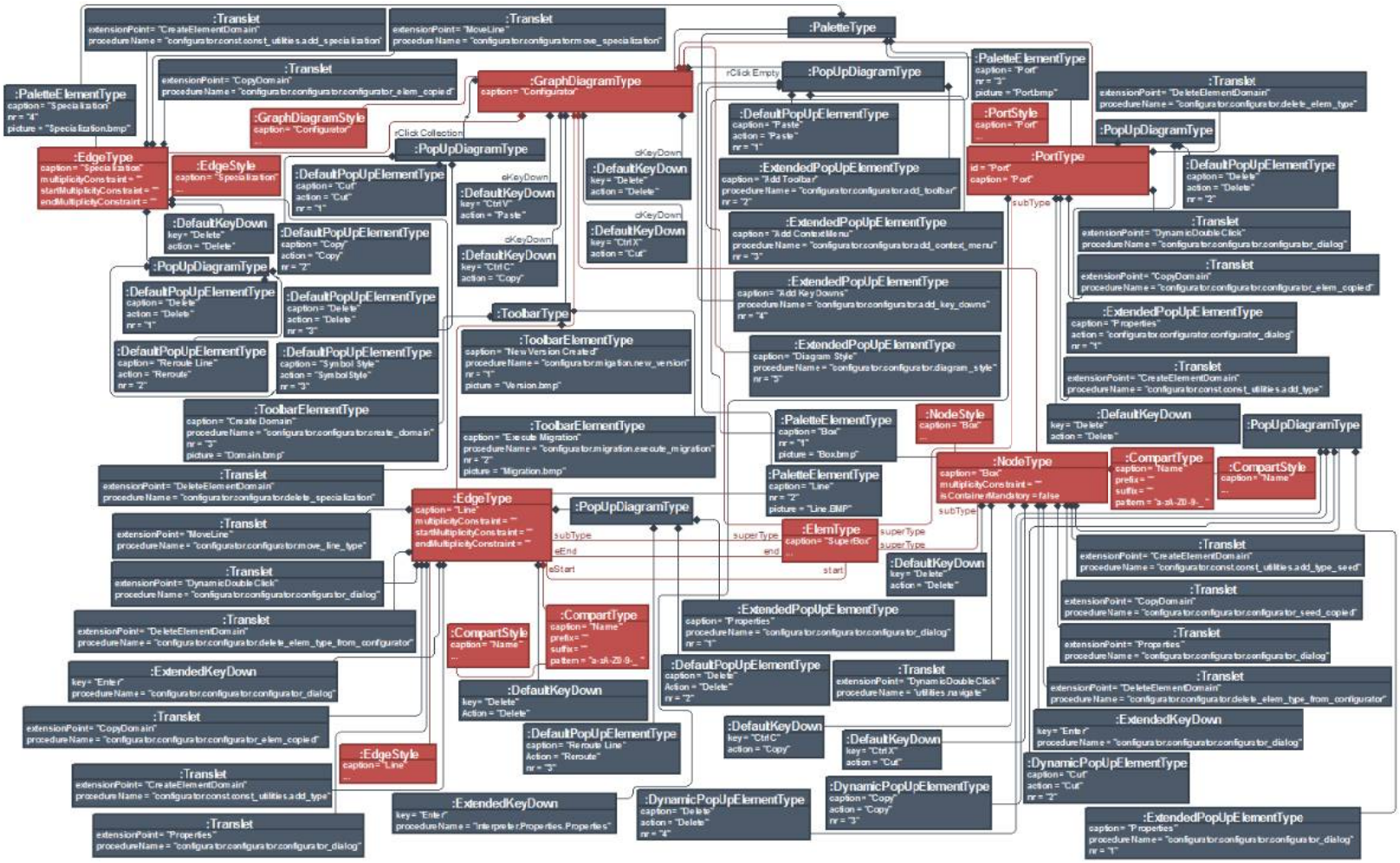
Elementa *Specialization* specifikācija ir redzama 2.27. attēlā. Šim elementam ir piesaistītas tikai četras no minētajām sešām *Line* paplašinājuma punktu transformācijām (nav *Properties* un *DynamicDoubleClick*, jo *Specialization* elementam nav dialogu logu).

Specialization uznirstošā izvēlne sastāv no trīs operācijām *Delete* (dzēst), *Reroute Line* (pārzīmēt līniju) un *Symbol Style* (uzstādīt simbola stilu), bet elementam piesaistīta ir tikai viena taustiņu kombinācija „Delete", kas atbilst dzēšanas operācijai.



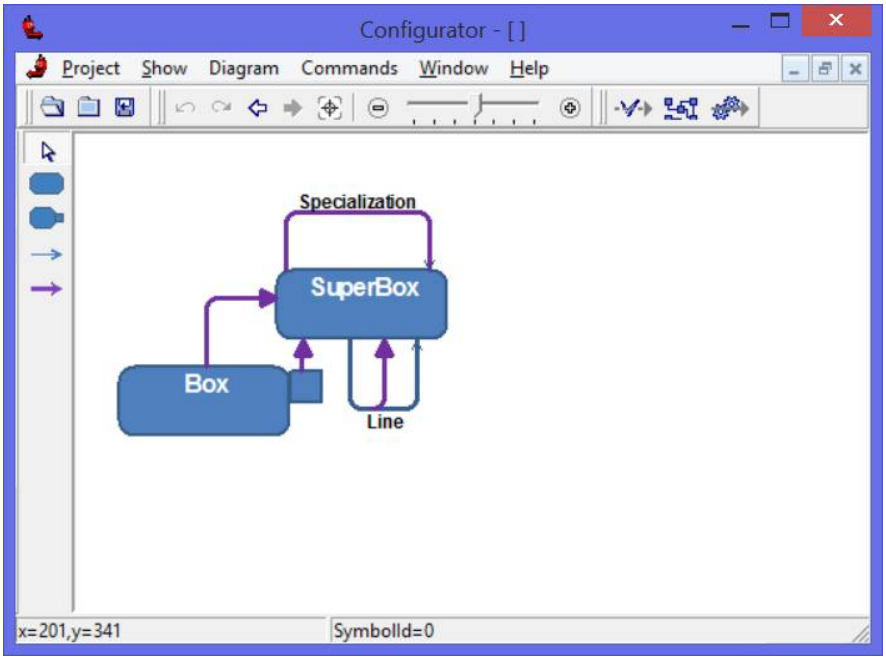
2.27. att. Specialization specifikācija

Pilna konfiguratora specifikācija ar rīku definēšanas metamodeļa instancēm ir redzama 2.28. attēlā.



2.28. *att.* Pilna konfiguratora specifikācija

Tā kā konfigurators ir specificēts ar rīku definēšanas metamodeļa instanci, šo pašu specifikāciju var uzdod ar konfiguratoru, respektīvi, ar konfiguratoru ir iespējams specificēt pašu konfiguratoru (sāknēšanas metode) un šādas specifikācijas atbilstošais grafiskais modelis ir redzams 2.29. attēlā.



2.29. att. Konfiguratora specifikācijas grafiskais modelis ar konfiguratoru

Pielikums 2.2.1

GRŪTI FORMALIZĒJAMU SISTĒMU MODELĒŠANAS METODIKA

Pārnese uz tīmekļa vidi

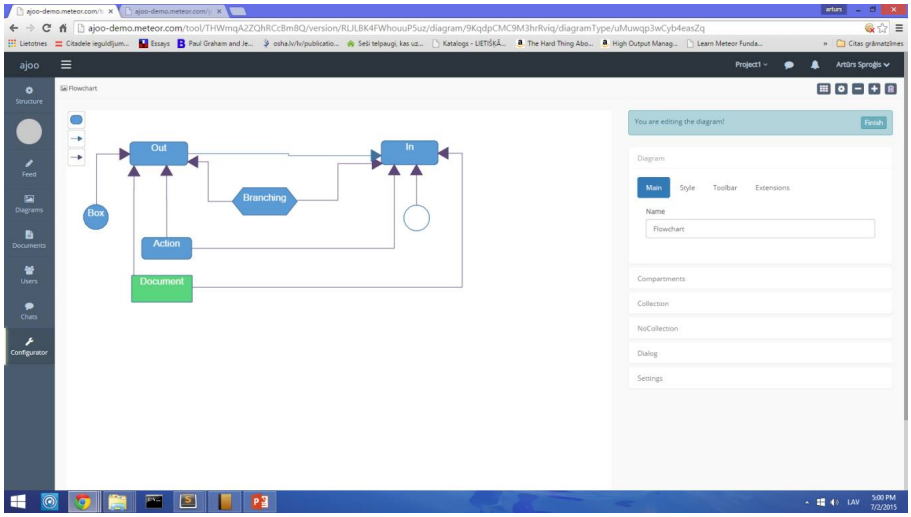
Izmantotās tehnoloģijas

- Pielikumā 2.3.1 aprakstītā DSML rīku būves metode balstīta uz metamodeļiem
- Meteor (web aplikāciju izstrādes platforma) un MongoDB (datubāze)
- KonvaJS (diagrammu zīmēšana)
- Bootstrap (HTML, CSS, JavaScript)

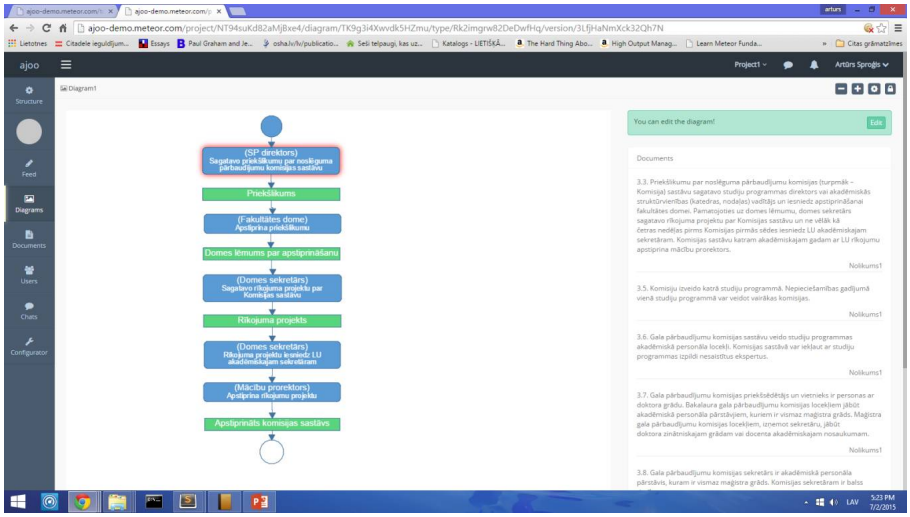
Ko nodrošina tīmekļa platforma

- Interaktivitāte
- Daudzlietotāju režīms (collaboration)
- Dažādas iekārtas – datori, planšetes, telefoni...
- Reālā laikā (reactivity and live HTML)
- Viegli pieejams un ieviešams

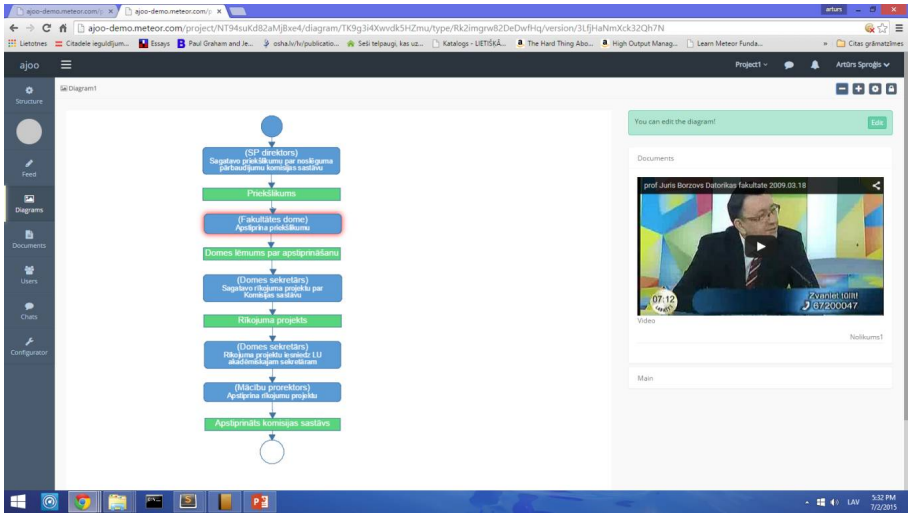
Konfigurācija



Rīka piemērs I



Rīka piemērs II



Pielikums 2.3

Saistīto datu izmantošana dažādos priekšmetu apgabalos: labākā prakse

J.Bārzdiņš
LU DF maģistr. stud. Ņ. Kumarins
2015

Pētījuma sākuma punkts

- Raksts: Adoptation of the Linked Data Best Practices in Different Topical Domains
- Autori: Max Schmachtenberg, Christian Bizer, and Heiko Paulheim.
- University of Mannheim Research Group Data and Web Science
- Projekts: Planet-Data (No. 257641, EC Seventh Framework Programme)
- Publicēts: ISWC 2014, LNCS 8796, 2014

Svarīgākie informācijas avoti

- http://linkeddatabook.com/
- Government Linked Data Current Status,

  http://www.w3.org/standards/techs/gld#w3c_all
- Health Care and Life Science (HCLS) Linked Data Guide

  http://www.w3.org/2001/sw/hcls/notes/hcls-rdf-guide/

Svarīgākie informācijas avoti

Linked Open Government Data: Lessons from Data.gov.uk ,

Nigel Shadbolt, Kieron O'Hara, Tim Berners-Lee, Nicholas Gibbins, Hugh Glaser, Wendy Hall and M.C. Shraefel, University of Southampton

IEEE Intelligent Systems, 2014

Svarīgākie informācijas avoti

Linked Data in Government

Issue No.04 - July-Aug. (2013 vol.17)

pp: 72-77

Published by the IEEE Computer Society

Nigel Shadbolt , University of SouthamptonKieron O'Hara , University of Southampton

DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/MIC.2013.72

Svarīgākie informācijas avoti

- Linked Data in Healthcare and Life Sciences.

  James G. Kim

  http://www.slideshare.net/echo4ngel/linked-data-in-healthcare-and-life-sciences-16926052
- Special issue on Linked Data for Health Care and the Life Sciences.

  Editorial: Mikel Egaña Aranguren, Jesualdo Tomás FernándezBreis , Michel Dumontier

  www.semantic-web-journal.net/system/files/swj475_0.pdf
- Privacy | HealthCare.gov

  https://www.**healthcare**.**gov**/privacy/

Svarīgākie informācijas avoti

http://eprints.soton.ac.uk/340564/

http://healthcaredelivery.cancer.gov/seermedicare/

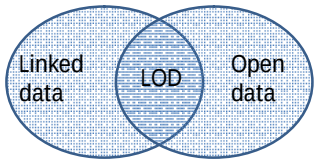
Saistīto datu (Linked data) definīcija

- Termins "Linked data" šobrīd apzīmē labāko praksi datu publicēšanai un strukturēto datu sasaistīšanai tīmeklī.
- Šo terminu ieviesa Tim Berners-Lee rakstā-piezīmēs "Linked data"
- "Linked data" ir veids kā pareizi būvēt semantisko tīmekli (*Tom Heath)*

Datu kopu var saukt par Linked data, ja

- Tā dod URI vārdu jebkurai lietai/jēdzienam, kuru tā glabā
- Dod iespēju iegūt informāciju par lietu izmantojot HTTP URI
- Lietas ir aprakstītas strukturētā veidā izmantojot RDF standartu vai SPARQL pieprasījumus
- Tā ir savienota ar lietām ārpus datu kopas izmantojot HTTP URI

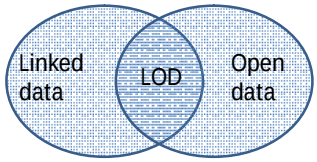
Linked Open Data (LOD)



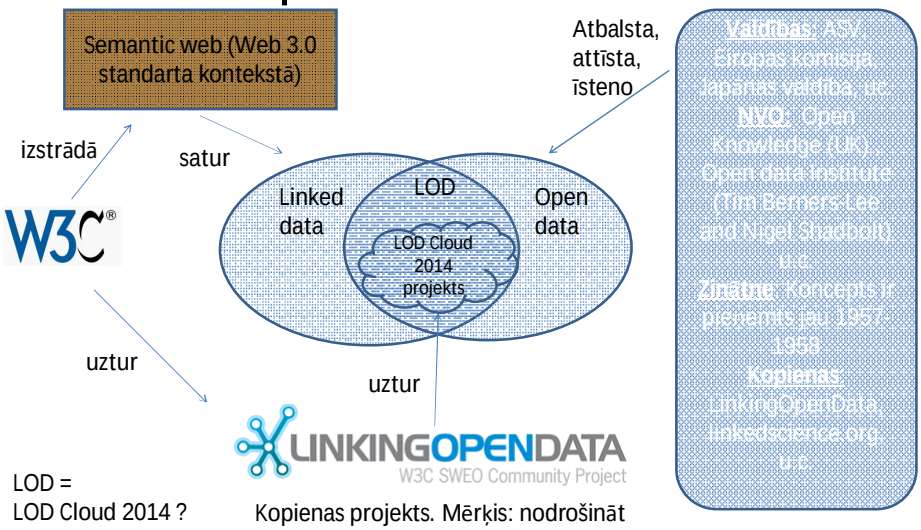
Linked Open Data (LOD)

- Vai visiem “Linked data” ir jābūt “Linked open data” ?

  Nē, un visdrīzāk tā nebūs. Termins "Linked Open Data" ir plaši izmantots, bet bieži ar to apzīmē “Linked Data”, nevis datus, kuri ir publicēti izmantojot publisku licenci. Ne visi “Linked data” būs publiski pieejami un ne visi publiski pieejamie dati atbilst Linked data principiem (Tom Heath)



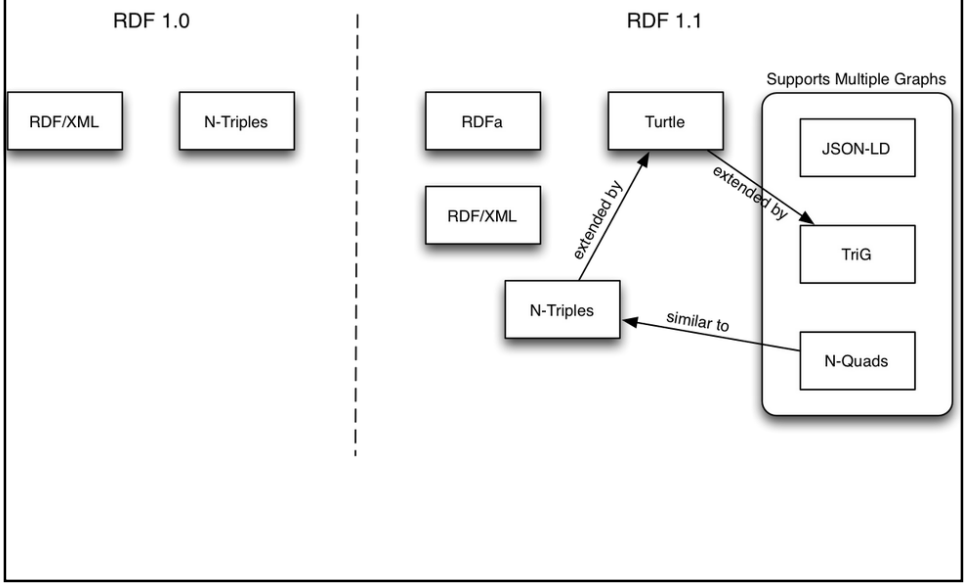
Linked Open Data – kas to uztur



LOD =
LOD Cloud 2014 ?

Kopienas projekts. Mērķis: nodrošināt publiski pieejamo datu atbilstību Linked data publicēšanas praksēm

Linked data definīcija – RDF



Linked data definīcija – SPARQL loma

- Vai SPAQL var būt vienīgā metode kā piekļūt "Linked data" datu kopai?
- SPARQL ir papildus metode, lai piekļūtu datiem, avota dati tāpat ir jānodrošina RDF formātā (*Tim Berners-Lee)*
- Linked data apstaigāšanas robots LDSpider, kas ir izmantots šajā projektā var nolasīt RDF, bet ne SPARQL

Datu kopu katalogu piemēri - I

Kataloga adrese un saturs	Datu kopu skaits (uz 27.05.2015)	Organizācija
http://datahub.io/ Var publicēties jebkurš ja atbilst noteikumiem	9518 (RDF un tā paveidi <3000)	OPEN KNOWLEDGE starptautiskā bezpeļņas nevalstiskā organizācija https://okfn.org/ England & Wales
http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets Datu kopas, kuras pieteiktas uz public-lod@w3.org	1014	LINKINGOPENDATA W3C SWEO Community Project Kopienas projekts (community project) Saite
http://publicdata.eu/ ES valstu institūciju publiskie dati	47863 (2023 RDF)	Creating Knowledge out of Interlinked Data LOD2 Daļa no Eiropas komisijas LOD2 projekta 2010-2014. Koordinē Leipcigas universitāte Saite

Datu kopu katalogu piemēri - II

Kataloga adrese, saturs	Datu kopu skaits (uz 27.05.2015)	Organizācija
Billion Triple Challenge 2012 dataset project http://km.aifb.kit.edu/projects/btc-2012/ katalogs izveidots tīkla apstaigāšanas rezultātā (DBPedia, datahub.io, Tim Berners-Lee's FOAF fails)	-	Karlsrūe, Vācija
http://www.data.gov/ "Mājas" ASV valdības publiskajiem datiem	132088 (RDF 5873)	
enigma.io Publisko datu katalogs no valdībām, universitātēm, kompānijām, privātpersonām"	87010 (piekļuve datiem izmantojot RESTful APIs)	enigma.io Privāts projekts, bezmaksas vai komerciāls

http://dataportals.org/

«Open data» katalogi un datu kopas Latvijā

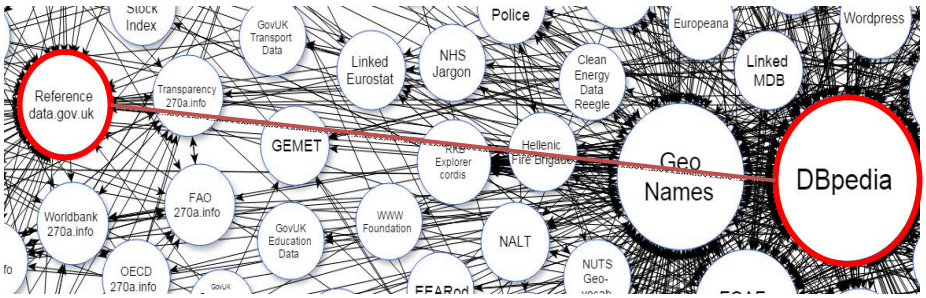
- Data.opendata.lv – katalogs par publiski pieejamām datu kopām Latvijā. Uztur entuziasti, ir sponsori. Datu kopām nav “tag”, līdz ar to grūti atrast kādu RDF datu kopu. Šī referāta autors to neatrada.
- https://opendata.riga.lv/ - Rīgas pašvaldības datu kopas. Tie ir dump faili : csv, txt, xlsx. Dati ir publiski, bet neatbilst “Linked data”

Katalogu pārklājumi

- Daudzi katalogi savā starpā nepārklājas, jo publicē datu kopas no dažādiem domēniem pēc savas būtības. Piem. ASV data.gov un Eiropas komisijas dati publicdata.eu
- Citi pārklājas daļēji. Piem. ASV data.gov un datahub.io
  - Statistika par Čikāgas noziegumiem (RDF formāts):
    - ✓ http://catalog.data.gov/dataset/crimes-2001-to-present-398a4
    - ✓ http://datahub.io/dataset/chicago-homicides
    - ✓ datahub.io katalogā meta informācija par šo datu kopu nav atjaunota kopš 2014
    - ✓ data.gov informācija ir svaiga
  - Energy usage 2010:
    - ✓ http://catalog.data.gov/dataset/crimes-2001-to-present-398a4
    - ✓ datahub.io neekstistē
- Kāpēc dati par Čikāgas noziegumiem ir datahub.io bet Energy usage 2010 nav?

Ko nozīmē katalogu nepārklāšanas

- Dažādu katalogu datu kopas joprojām var būt saistītas viena ar otru ar URI. Tas netraucē veidot globālo Linked data tīklu
- Bet veidojot vienotu LOD mākoni un sākot apstaigāšanu no dažiem katalogiem var nenonākt līdz citiem nesaistītiem vai vāji saistītiem LOD "mākoņiem"



Linked data publicēšanas labākas prakses - I

- **Saites**

  Datu kopa ir jāsaista ar esošo globālo saistīto datu kopu grafu izmantojot RDF

- **Vārdnīcu izmantošana**

- *Ir jāizmanto plaši izplatītas vārdnīcas (RDF, FOAF, DCTERM, utt)*
- *Ja datu kopai ir sava vārdnīca, tās vārdiem ir jābūt interpretējamiem: RDF shēma vai OWL*

Linked data publicēšanas labākas prakses - II

- **Meta dati**
- sevi aprakstoša datu kopa
- satur informāciju par izcelsmi un datu kvalitāti
- satur informāciju par licencēšanu, satur informāciju par pieprasījumu veidiem (SPARQL)

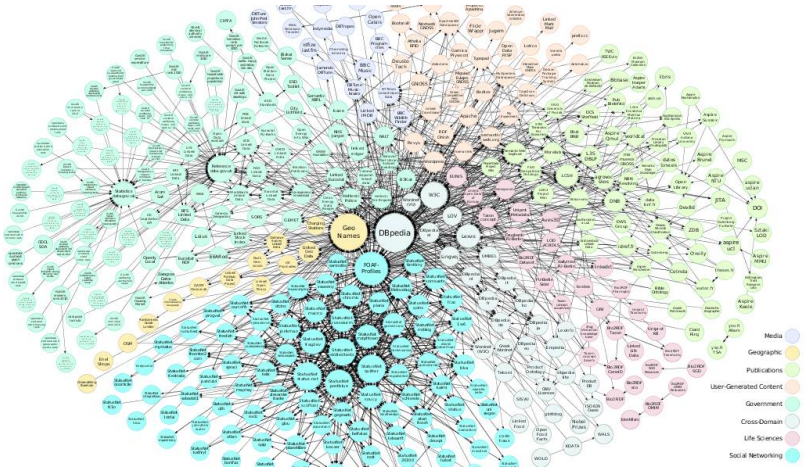
Labākas prakses principu pielietošanas analīze

- **State of the LOD Cloud 2011**

  - datu kopas tika analizētas balstoties uz informāciju, ko sniedza paši autori (specifikācijas)
- **State of the LOD Cloud 2014** **<- projekta tēma**

  - datu kopas tika analizētas balstoties uz informāciju, ko izdevās iegūt/sasniegt tīmekļī izmantojot crawling “robotu”.

Linked Open Data Cloud Apr.2014

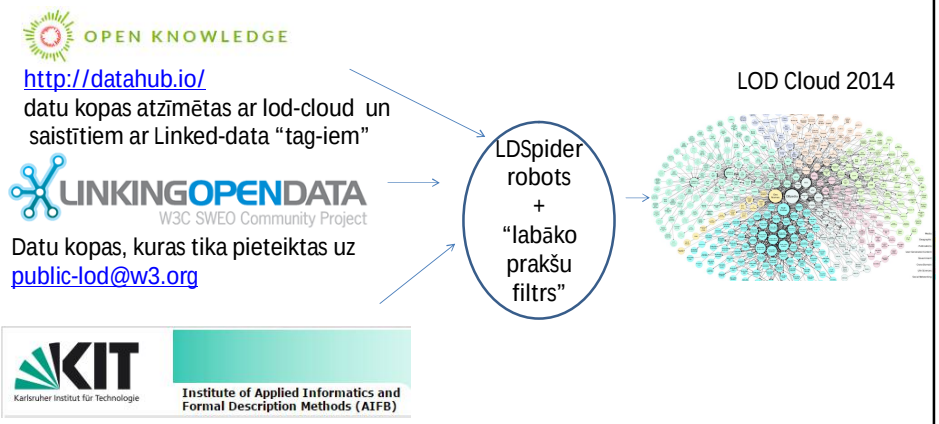


LOD Cloud 2014 izveidošanas algoritms - I

- Tīmekļa apstaigāšanai tika izmantots LDSpider robots
  - Apstaiga RDF saites Linked Data datu kopas
  - Nodrošina dažādu RDF serializāciju lasīšanu (RDF/XML, N-TRIPLES, N-QUADS, Turtles)
  - Apstaigāšanas stratēģijas : dziļumā, plašumā
  - Var ierobežot apstaigāšanu domēna ietvaros

LOD Cloud 2014 izveidošanas algoritms - II

- 560t. starta URI no 3 katalogu datu kopām:



katalogs izveidots tīkla apstaigāšanas rezultātā (sākot no DBPedia, datahub.io, Tim Berners-Lee's FOAF faila)

LOD Cloud 2014 izveidošanas algoritms - III

- Vai tīmeklī var pastāvēt cits LOD mākonis, kas atbilst visiem 4 linked data publicēšanas principiem?
- Teorētiski var, ja tam būtu cita "sakne"

LOD Cloud 2014 izveidošanas algoritms - III

- Datu kopas, kuras saturēja tikai vārdnīcas, tika izslēgtas
- Dati no viena domēna (PLD – pay level domain) reprezentē vienu datu kopu, ja vien nav norādīts pretēji datu kopu katalogā datahub.io.

  Piemēram example.com ir PLD priekš top level domain (TLD) www.example.com
- Apstaigāšanas ierobežojumi, kuri ir norādīti robots.txt failos tika ievēroti. 77 datu kopas to neļāva

Apstaigāšanas rezultāti

- 1014 datu kopas
- 900,129 dokumenti
- 8,038,396 resursi

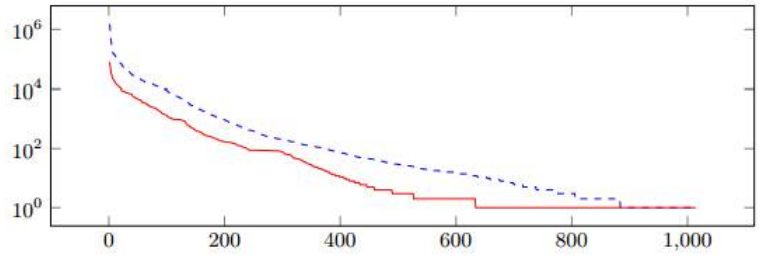


Fig. 1: Distribution of the number of resources (- - -) and documents (——) per dataset contained in the crawl (log scale).

Datu kopu kategorijas/Domēni

Table 1: Number of datasets in each category and growth compared to 2011.

Category	Datasets 2014	Percentage	Datasets 2011	Growth
Media	24 (-2)	2%	25	-4%
Government	199 (-16)	18%	49	306%
Publications	138 (-42)	13%	87	59%
Geographic	27 (-6)	2%	31	-13%
Life Sciences	85 (-2)	8%	41	107%
Cross-domain	47 (-6)	4%	41	15%
User-generated Content	51 (-3)	5%	20	155%
Social Networking	520 (-0)	48%	-	-
Total	1091 (-77)		294	271%

LOD 2014 mākoņa datu kopas - piemēri

- DBPedia – vispārīgu zināšanu datu kopa
  - Strukturēti dati no Wikipedia (parasti daļa pa labi)
  - 4.58 miljons lietu angļu versijā
  - 4.22 miljoni ir klasificēti ontoloģijā
  - > 3 miljardi RDF-trijnieku
  - Saistīta ar citām "Linked data" datu kopām ar 50 miljoniem RDF saišu

- statistics.data.gov.uk
  - Lielbritānijas valdības oficiāla statistika
  - Ģeogrāfijas un ģeotelpiskā statistika (nosaukumi, kodi, klasifikatori, teritoriju robežas/izmēri, utt.)
  - 343733 trijnieku (datahub.io 03.06.2015)

LOD 2014 mākoņa datu kopas - piemēri

- Ļoti daudzas StatusNet servera (mikroblogu serveris) instances – kategorija sociālie tīkli

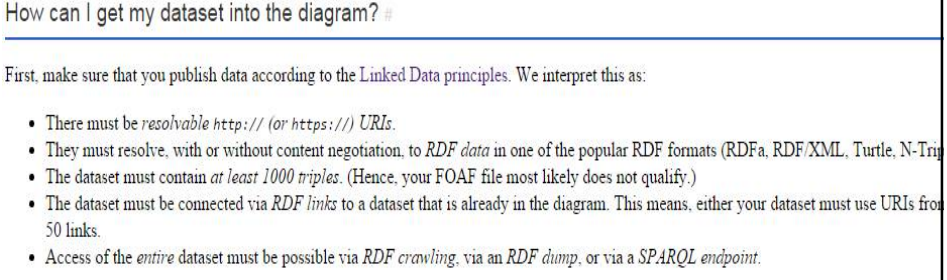
http://sweetiebelle.net/

Field	Value
links:personal-homepages	5
links:statusnet-rainbowdash-net	120
links:statusnet-status-net	2

http://johndrinkwater.name/

LOD 2014 mākoņa datu kopas - piemēri

- Kā Sweetiebelle un johndrinkwater nokļuva starp 1024 mākoņa datu kopām?
- http://lod-cloud.net/ rekomendācijas



LOD 2014 mākoņa datu kopas - piemēri

- Turpretī Tim Berners-Lee (Sematiskā weba "tēva") FOAF ieraksts mākonī neparādās
- Jo Tim to publicēja ne savā, bet w3 domēnā http://www.w3.org/People/Berners-Lee/card#i, kas ir atsevišķa datu kopā
- 1024 datu kopas? DBPedia vs Sweetiebelle ?

Datu kopu sasaistīšanas stāvoklis LOD mākonī I

- Ja starp datu kopām ir atrasta viena saite (RDF), viņas mākonī ir savienotas
- 56% no datu kopām norāda uz kādu citu datu kopu no mākoņa
- Atlikušie 46% ir vai datu kopas uz kurām atsaucās, vai ir izolētas

Datu kopu sasaistīšanas stāvoklis LOD mākonī II

Categorization by number of linked datasets

Number of linked datasets	Number of datasets
more than 10	79 (7.79%)
6 to 10	81 (7.99%)
5	31 (3.06%)
4	42 (4.14%)
3	54 (5.33%)
2	106 (10.45%)
1	176 (17.36%)
0	445 (43.89%)

71.99% no visām datu kopām ir vāji saistītas

Datu kopu sasaistīšanas stāvoklis LOD mākonī III

Table 2: Datasets with the highest in- and outdegrees.

Dataset	Category	In	Dataset	Category	Out
dbpedia.org	cross-domain	207	bibsonomy.org	publications	91
geonames.org	geographic	141	semanlink.net	user-gen. cnt.	88
w3.org	cross-domain	117	deri.org	social netw.	70
quitter.se	social netw.	64	harth.org	social netw.	68
status.net	social netw.	63	quitter.se	social netw.	67
postblue.info	social netw.	56	semanticweb.org	user-gen. cnt.	64
skilledtest.com	social netw.	55	skilledtests.com	social netw.	60
reference.data.gov.uk	government	45	postblue.info	social netw.	59
data.semanticweb.org	publications	44	status.net	social netw.	47
fragdev.com	social netw.	41	w3.org	crossdomain	45
lexvo.org	cross-domain	37	data.semanticweb.org	publications	45

Datu kopu sasaistīšanas stāvoklis LOD mākonī IV – predikātu lietošana

Table 3: Top three linking predicates per category. The percentages are relative to number of datasets within the category which set outgoing links.

Category	Predicate	Usage	Category	Predicate	Usage
social networking	foaf:knows	60.27%	life sciences	owl:sameAs	52.17%
social networking	foaf:based_near	35.69%	life sciences	rdfs:seeAlso	43.48%
social networking	sioc:follows	34.34%	life sciences	dct:creator	21.74%
publications	owl:sameAs	32.20%	government	dct:publisher	47.57%
publications	dct:language	25.42%	government	dct:spatial	30.10%
publications	rdfs:seeAlso	23.73%	government	owl:sameAs	24.27%
user-generated content	owl:sameAs	53.13%	geographic	owl:sameAs	64.29%
user-generated content	rdfs:seeAlso	21.88%	geographic	skos:exactMatch	21.43%
user-generated content	dct:source	18.75%	geographic	skos:closeMatch	21.43%
media	owl:sameAs	81.25%	crossdomain	owl:sameAs	80.00%
media	rdfs:seeAlso	18.75%	crossdomain	rdfs:seeAlso	52.00%
media	foaf:based_near	18.75%	crossdomain	dct:creator	20.00%

Datu kopu sasaistīšanas stāvoklis LOD mākonī V – predikātu lietošana

Table 4: Top 10 datasets regarding in- and outdegree for `owl:sameAs` links by category.

Dataset	Category	In	Dataset	Category	Out
dbpedia.org	crossdomain	89	bibsonomy.org	publications	91
geonames.org	geographic	29	data.semanticweb.org	publications	31
data.semanticweb.org	publications	24	myopenlink.net	user-gen. cnt.	25
l3s.de	publications	24	dbpedia.org	crossdomain	23
semanticweb.org	user-gen. cnt.	18	semanticweb.org	user-gen. cnt.	18
nytimes.com	media	11	revyu.com	user-gen. cnt.	16
dbtune.org	social networking	11	advogato.org	social networking	16
kit.edu	social networking	9	el.dbpedia.org	crossdomain	13
revyu.com	user-gen. cnt.	8	nl.dbpedia.org	crossdomain	11
w3.org	crossdomain	8	harth.org	social networking	11
it.dbpedia.org	crossdomain	8			

Izplatīto vārdnīcu izmantošana LOD 2014 mākonī -I

- Datu kopa izmanto vārdnīcu ja tās vārds parādās kā predikāts vai kā objekts rdf:type trijniekā

Table 5: Vocabularies used by more than 5% of all datasets.

Prefix	Occurrence	Quota	Prefix	Occurrence	Quota
rdf	996	98.22%	void	137	13.51%
rdfs	736	72.58%	bio	125	12.32%
foaf	701	69.13%	cube	114	11.24%
dcterm	568	56.01%	rss	99	9.76%
owl	370	36.49%	odc	86	8.48%
wgs84	254	25.05%	w3con	77	7.60%
sioc	179	17.65%	doap	65	6.41%
admin	157	15.48%	bibo	62	6.11%
skos	143	14.11%	dcat	59	5.82%

Izplatīto vārdnīcu izmantošana LOD 2014 mākonī - II

- Mākonis izmanto 638 vārdnīcas
- No tām 58.77% ir plaši izplatītas vārdnīcas (rdf, foaf, utt)
- 41.23% ir datu kopas “lokālas” vārdnīcas
- Gandrīz visas datu kopas izmanto izplatītās vārdnīcas
- Tikai 23.08% izmanto lokālās
- “Standarta” vārdnīcu pielietošana uzlabojas, jo 2011 izveidota mākonī 64% datu kopu izmantoja “savas” vārdnīcas

Izplatīto vārdnīcu izmantošana LOD 2014 mākonī - III

- Vārdu definīciju esamība lokālās vārdnīcās (ar HTTP GET pēc URI)

Category	Different prop. vocabs. used (% of all prop. vocabs.)	# of datasets using prop. vocab. (% of all datasets)	Dereferencability		
			full	partial	none
Social networking	126 (33.60%)	81 (15.57%)	19.47%	8.8%	77.78%
Publications	59 (15.73%)	33 (34.38%)	22.03%	8.47%	69.49%
Government	47 (12.53%)	34 (18.58%)	21.28%	12.77%	65.96%
Cross-domain	56 (14.93%)	17 (41.46%)	26.79%	10.71%	62.50%
Geographic	13 (3.47%)	8 (38.10%)	15.38%	7.69%	76.92%
Life sciences	36 (9.60%)	27 (32.53%)	27.78%	5.56%	66.67%
Media	12 (3.20%)	12 (54.55%)	0.00%	16.67%	83.33%
User-gen. cnt.	26 (6.93%)	22 (45.83%)	11.54%	11.54%	76.92%
Total	375 (58.77%)	234 (23.08%)	19.47%	8.80%	71.73%

Izplatīto vārdnīcu izmantošana LOD 2014 mākonī - IV

- Lokālo vārdnīcu "tulkošana" uz izplatītām vai citām) vārdnīcām nav izplatīta

Table 7: Predicates used to link terms between different vocabularies.

Term	% of vocabularies	Term	% of vocabularies
rdfs:range	9.87%	rdfs:seeAlso	1.60%
rdfs:subClassOf	8.80%	owl:equivalentClass	1.60%
rdfs:subPropertyOf	6.93%	owl:inverseOf	1.33%
rdfs:domain	5.60%	swivt:type	1.07%
rdfs:isDefinedBy	3.73%	owl:equivalentProperty	0.80%

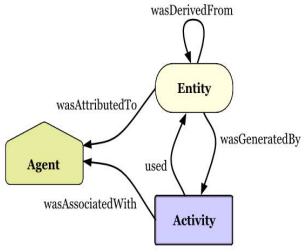
Metadata labākas prakses – datu izcelsme I

- Informācija par datu izcelsmi (provenance) apraksta, kā "lietas" tika izveidotas. Satur informāciju par aktivitātēm, "aktieriem" un enītijām, kas izveidoja tīmekļa resursu. Kā arī ar to saistītiem jēdzieniem: "datu izmantošana", laiks, plāns, alternatīvas entītīju prezentācijas
- PROV-O – OWL2 ontoloģija, kas apraksta attiecīgo datu modeli
- PROV-XML – XML shēma datu modelim
- Namespace http://www.w3.org/ns/prov#
- Piem. http://www.w3.org/ns/prov#actedOnBehalfOf

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix :     <http://example.com/> .

:derek
   a prov:Agent;
   foaf:givenName "Derek"^^xsd:string;
   foaf:mbox      <mailto:derek@example.org>;
   prov:actedOnBehalfOf :national_newspaper_inc;
.

:national_newspaper_inc
   a prov:Agent, prov:Organization;
   foaf:name "National Newspaper, Inc.";
.
```



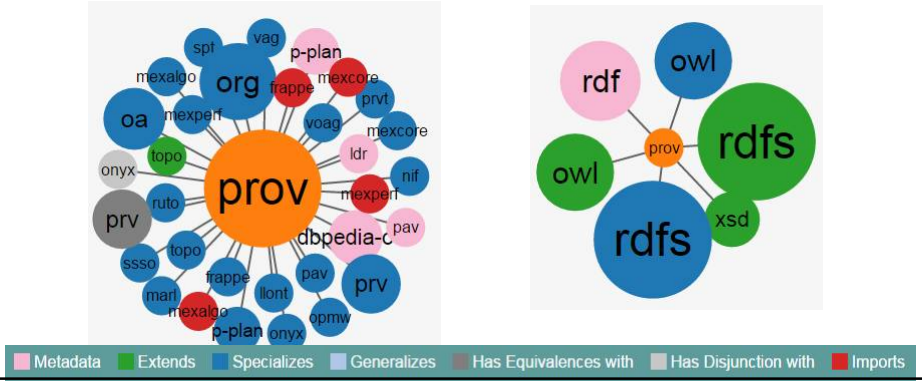
Metadata labākas prakses – datu izcelsme II

- Dublin Core Schema – vārdnīca, kura apraksta tīmekļa resursus (web lapas, datus, bildes, utt)

Title: A name given to the resource.
Creator: An entity primarily responsible for making the content of the resource.
Subject: The topic of the content of the resource.
Description: An account of the content of the resource.
Publisher: An entity responsible for making the resource available
Contributor: An entity responsible for making contributions to the content of the resource.
Date: A date associated with an event in the life cycle of the resource.
Type: The nature or genre of the content of the resource.
Format: The physical or digital manifestation of the resource.
Identifier: An unambiguous reference to the resource within a given context.
Source: A reference to a resource from which the present resource is derived.
Language: A language of the intellectual content of the resource.
Relation: A reference to a related resource.
Coverage: The extent or scope of the content of the resource.
Rights: Information about rights held in and over the resource.

Metadata labākas prakses – datu izcelsme III

- Šajā projektā tika izanalizēta datu kopu saistība ar 26 datu vārdnīcām, kuras apraksta datu izcelsmi vai līdzīgus jēdzienus:
  - W3C Provenance Working Group
  - LOV vārdnīcu katalogs
  - Pašu autoru pieredze



Metadata labākas prakses – datu izcelsme IV

Table 8: Provenance vocabulary usage and license vocabulary usage by category.

Category	Any prov vocab		Dublin Core	Admin	Prv/Prov	Any license vocab
social networking	169	(32.5%)	57.39%	57.39%	1.18%	5.38%
publications	39	(40.63%)	94.87%	5.13%	2.56%	4.17%
government	76	(41.54%)	100.00%	0.00%	1.32%	30.05%
life sciences	20	(24.10%)	100.00%	0.00%	0.5%	3.61%
cross-domain	7	(17.07%)	100.00%	14.29%	0.00%	9.76%
geographic	3	(14.29%)	100.00%	0.00%	33.34%	0.00%
user-gen. content	9	(18.75%)	88.89%	66.67%	0.00%	10.42%
media	4	(18.18%)	100%	0.00%	0.00%	5.41%
Total	372	(36.69%)	29.09%	11.05%	0.79%	9.96%

2011 gadā mākoņa versija 36.63% datu kopu izmantoja datu izcelsmi definējošas vārdnīcas

Metadata labākas prakses – licences

Table 8: Provenance vocabulary usage and license vocabulary usage by category.

Category	Any prov vocab		Dublin Core	Admin	Prv/Prov	Any license vocab
social networking	169	(32.5%)	57.39%	57.39%	1.18%	5.38%
publications	39	(40.63%)	94.87%	5.13%	2.56%	4.17%
government	76	(41.54%)	100.00%	0.00%	1.32%	30.05%
life sciences	20	(24.10%)	100.00%	0.00%	0.5%	3.61%
cross-domain	7	(17.07%)	100.00%	14.29%	0.00%	9.76%
geographic	3	(14.29%)	100.00%	0.00%	33.34%	0.00%
user-gen. content	9	(18.75%)	88.89%	66.67%	0.00%	10.42%
media	4	(18.18%)	100%	0.00%	0.00%	5.41%
Total	372	(36.69%)	29.09%	11.05%	0.79%	9.96%

- Tika meklēti RDF trijnieki, kuros dokuments tika minēts kā subjekts, un predikāts satur "licen" vai "rights" un "waiver" vārdnīcu, kopā 47 termini
- 2011 gada mākoņa versijā 17.84% no datu kopām saturēja info par licencēšanu
- Populārākie RDF predikāti dc/dct:license (7.39%), cc:license (2.07%) and dc/dct:rights (1.68%)

Kopsavilkums: galvenās saistīto datu kategorijas (citāts no iepriekš pieminētā M.Schmachtenberg et.al. raksta):

The **media category** contains datasets providing information about films, music, TV and radio programmes, as well as print media. Prominent datasets within this category are the dbtune.org music datasets, the New York Times dataset, and the BBC radio and television program datasets.

The government category contains Linked Data published by federal or local governments, including a lot of statistical datasets. Prominent examples include the data.gov.uk and opendatacommunities.org datasets.

The **category publications** holds library datasets, information about scientific publications and conferences, reading lists from universities, and citation databases. Well known datasets includ ethe German National Library dataset, the L3S DBLP dataset, and the Open Library dataset.

The **category geographic** contains datasets like geonames.org and linkedgeodata.org comprising information about geographic entities, geopolitical divisions, and points of interest.

The life sciences category comprises biological and biochemical information, drug-related data, and information about species and their habitats.

The **cross-domain category** includes general knowledge bases such as DBpedia or UMBEL, linguistic resources such as Word- Net or Lexvo, as well as product data.

The **category user-generated** content contains data from portals that collect content generated by larger user communities. Examples include metadata about blogposts published as Linked Data by wordpress.com, data about open source software projects published by apache.org, scientific workflows published by myexperiment.org, and reviews published by goodreads.com or revyu.com.

The **category social networking** contains people proles as well as data describing the social ties amongst people. We include into this category individual FOAF proles, as well as data about the interconnections amongst users of the distributed microblogging platform StatusNet.

Statistika atbilstoši saistīto datu kategorijām

Table 9: Percentage of datasets using the VoID vocabulary and percentage of datasets offering alternative access methods.

Category	VoID	Link	Well-known	Inline	Alt. access	SPARQL	Dump
social networking	5 (0.96%)	0.19%	0.77%	0.00%	4 (0.77%)	0.77%	0.19%
publications	13 (13.54%)	6.25%	3.13%	7.29%	13 (13.54%)	12.50%	4.17%
life sciences	30 (36.14%)	28.92%	2.41%	4.82%	20 (24.10%)	24.10%	15.66%
government	72 (42.08%)	2.73%	2.73%	36.61%	63 (34.43%)	31.15%	31.15%
user-gen. content	6 (11.76%)	11.76%	0.00%	0.00%	3 (6.25%)	6.25%	2.08%
geographic	6 (38.10%)	14.29%	9.52%	14.29%	5 (23.81%)	14.29%	19.05%
cross-domain	5 (12.20%)	7.32%	2.44%	4.88%	4 (9.76%)	4.88%	4.88%
media	2 (9.09%)	0.00%	0.00%	9.09%	1 (4.55%)	0.00%	4.55%
Total	149 (14.69%)	4.14%	1.28%	9.27%	113 (11.14%)	9.96%	8.19%

Secinājums par saistīto datu lietojumiem e-pārvaldē un e-medicīnā

- Saistīto datu būves tehnoloģijas uz šo brīdi ir jau pietiekoši labi izstrādātas un tās var praktiski lietot.
- Uz ontoloģijām balstīto saistīto datu lietojumi pasaulē samērā strauji pieaug (aptuveni 30% gadā). Šie lietojumi pamatā balstās uz atvērtajiem datiem RDF formā, tie galvenokārt attiecas uz dažādu (iepriekš neparedzētu) pētījumu, statistiku, izziņu nodrošināšanu.
- Tieši e-pārvaldes un e-medicīnas laukā saistīto datu tehnoloģijas netiek plaši lietotas – galvenā problēma: e-pārvaldes un e-medicīnas dati ir ar likumdošanu strikti reglamentēti un ar ierobežotu pieejamību, kas ir pretrunā ar uz ontoloģijām balstītajām saistīto datu tehnoloģijām. Taču valstiskā līmenī paplašinot atvērto datu apjomu, saistīto datu tehnoloģijas vismaz ierobežotā veidā var tikt lietotas arī pārvaldes un medicīnas jomā – galvenokārt statistikas un pētniecības vajadzībām.

Papildliteratūra - I

- Līdzīgs projekts http://stats.lod2.eu/ (Feb. 2014)

- Analīze izmantotās vārdnīcas
- Neanalizē labākās prakses un nekategorizē
- Ir atrastas 928 "sasniedzamas" datu kopas, kas saskan ar iepriekš analizētā projekta rezultātiem. Tā ir atbilde (autora neanalizēta) uz šī referāta jautājumu "vai var būt vēl viens cits LOD cloud"

Papildliteratūra - II

- Christian Bizer, Freie Universität Berlin, Germany, Tom Heath, Talis Information Ltd, United Kingdom, Tim Berners-Lee, Massachusetts Institute of Technology, USA, "Linked Data - The Story So Far"
- Tom Heath , Talis Christian Bizer , Freie Universität Berlin, Linked Data: Evolving the Web into a Global Data Space http://linkeddatabook.com/
- http://en.wikipedia.org/wiki/Linked_data
- Tim Berners-Lee (2006-07-27). "Linked Data—Design Issues". W3C. Retrieved 2010-12-18
- http://linkeddata.org/
- http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

Papildliteratūra - III

- http://www.w3.org/standards/semanticweb/
- http://www.w3.org/2001/sw/wiki/Main_Page
- http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/
- http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/
- Robert Isele , Jurgen Umbrich2 , Christian Bizer , and Andreas Harth: LDSpider: An open-source crawling framework for the Web of Linked Data, http://ceur-ws.org/Vol-658/paper495.pdf
- http://en.wikipedia.org/wiki/Web_crawler
- Top level domain, pay level domain http://en.wikipedia.org/wiki/Domain_registration
- http://webdatacommons.org/structureddata/vocabulary-usage-analysis/

Papildliteratūra - IV

- http://lod2.eu/
- Lod2 project http://cordis.europa.eu/project/rcn/95562_en.html
- Ahmad Assaf, Raphael Troncy, Aline Senart: Roomba: An extensible Framework to Validate and Build Datasets http://ceur-ws.org/Vol-1362/PROFILES2015_paper1.pdf

Papildliteratūra - V

- http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/#deref
- http://www.w3.org/TR/swbp-vocab-pub/
- http://www.w3.org/People/Ivan/CorePresentations/State_of_SW/Slides.pdf
- http://vocab.org/waiver/terms/.html