Liet. matem. rink., **44**, *spec. nr.*, 2004, 369–376

Verslo taisyklių modeliavimas OCL kalba

Sergejus SOSUNOVAS, Olegas VASILECAS (VGTU)
el. paštas: sergejus@isl.vtu.lt, olegas@fm.vtu.lt

1. Įvadas

Šiuolaikinis verslo modeliavimas sudarytas iš trijų dalių: duomenų arba sandaros modeliavimo, procesų, atliekančių tam tikrus veiksmus su ištekliais, turinčių savo tikslus ir vykdytojus, procesai yra sukeliami įvykių ir generuoja įvykius, modeliavimo bei viską valdančių verslo taisyklių rinkinio. Sudedamosios dalys negali egzistuoti viena be kitos, nors praktikoje jas kartais būna sunku atskirti. Ypač sudėtinga išskirti verslo taisykles, kurios labai dažnai yra traktuojamos kaip sandaros ir procesų dalys. Nepaisant to, jų svarba yra akivaizdi, bet formalizavimo lygis organizacijose gali skirtis (klasifikacija [1]).

UML yra nauja verslo sistemų specifikavimo kryptis. Ši kalba į verslo sistemų modeliavimą atėjo iš programų sistemų modeliavimo. Verslo sistemų modeliavimo UML pavyzdžius galima rasti [3]. UML kalba vartojama tiek modeliuojant verslo sistemų sandarą (klasių diagramos), tiek jų elgseną (veiksmo, ansamblių, užduočių diagramos). Verslo taisykles galima specifikuoti ir tiesiogiai, užrašant jas OCL [2] kalba ir netiesiogiai skirtingose diagramose (pvz., per asociatyvaus ryšio kardinalumą klasių diagramoje).

UML kalba, į kurią šiuo metu yra integruota OCL, buvo sukurta kaip programų sistemų modeliavimo kalba ir vėliau jos taikymo sritis buvo išplėsta į verslo sistemų modeliavimą. Todėl atsiranda natūralus klausimas, ar įmanoma UML/OCL kalbos dabartine versija 1.4 efektyviai modeliuoti verslo sistemas. Šitame darbe daugiausia dėmesio bus skirta į verslo taisyklių modeliavimo OCL kalbą.

Vienas iš būdų patikrinti OCL kalbos tinkamumą verslo taisyklių modeliavimui – pateikti skirtingų verslo taisyklių rūšių pavyzdžius, užrašytus OCL kalba. Pagrindu bus paimtos verslo taisyklių šeimos iš [4]: egzempliorių tikrinimo (angl. instance verifiers), rūšies tikrinimo (angl. type verifiers), pozicijos tikrinimo (angl. position verifiers), funkcinio tikrinimo (angl. functional verifiers), palyginimo (angl. comparative evaluators), matematinio įvertinimo (angl. mathematical evaluators), projekcijos kontrolės (angl. projection controllers).

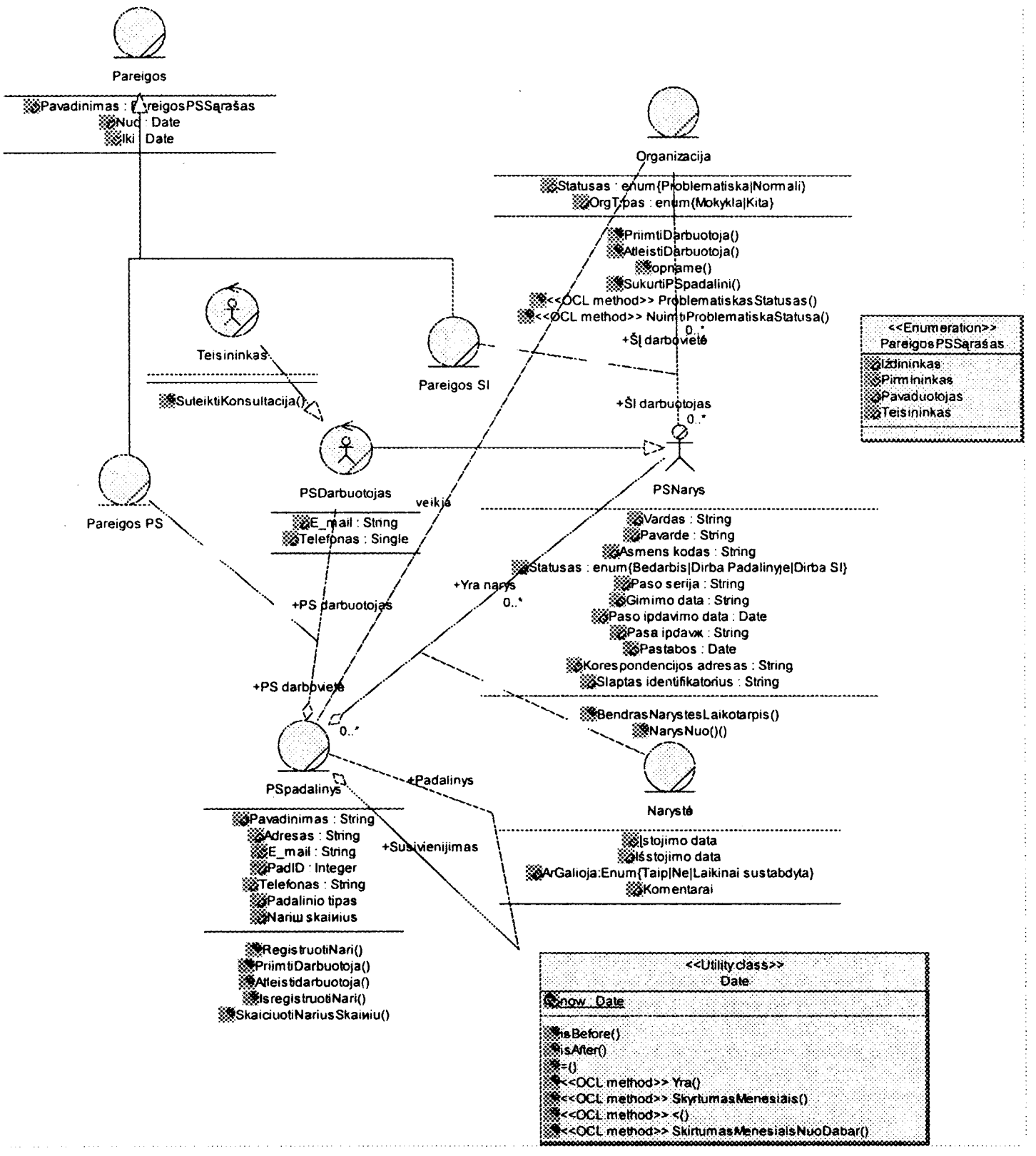
Dažnai tas pačias verslo situacijas galima išreikšti ir UML grafine notacija, ir OCL teksto komentarais. Viskas priklauso nuo modeliuotojo pasirinkto stiliaus, [5] rekomenduojama naudoti tokią priemonę, kuria pateikta specifikacija bus lengviau suprantama. Šio darbo tikslas – patikrinti OCL kalbos galią specifikuojant verslo taisykles, todėl visos taisyklės, kur tai įmanoma, bus pateikiamos OCL kalba.

2. Verslo taisyklių modeliavimas OCL kalba

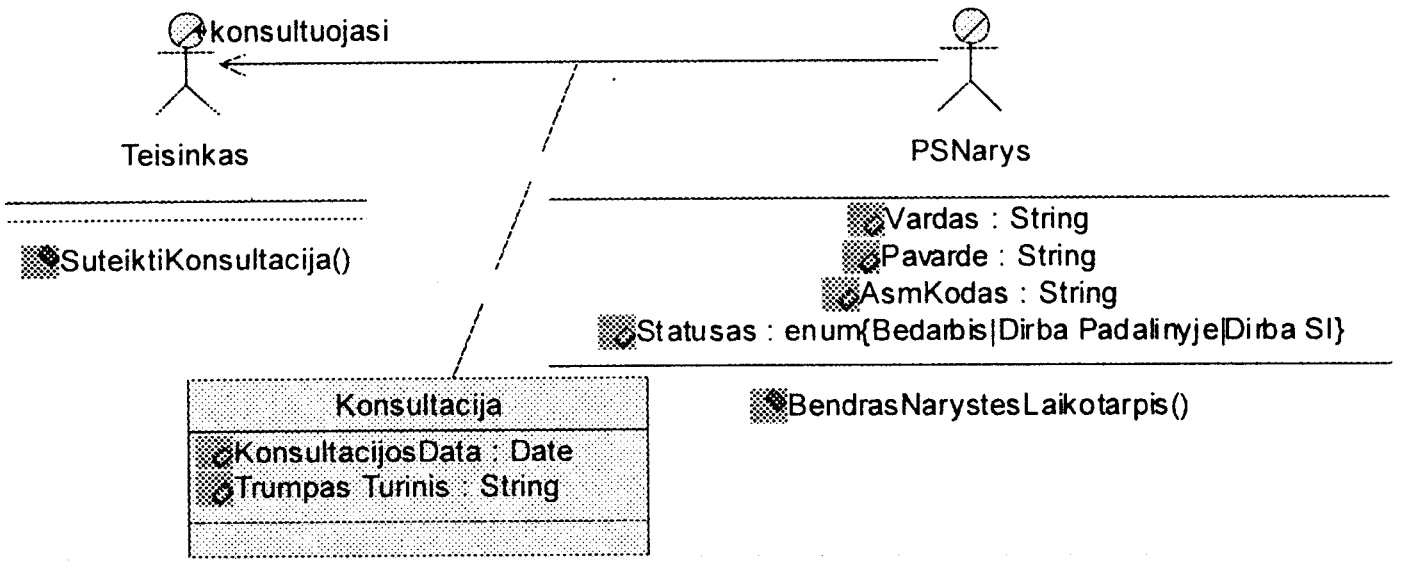
Tipinė verslo taisyklė sudaryta iš trijų dalių: inkaro (angl. anchor), korespondentų (angl. correspondent) bei sąlygos. Inkaras suteikia jungimosi tašką, nuo kurio reikia pradėti interpretuoti taisyklę. Visos taisyklės, turėdamos savo sąlygas, kreipiasi į modelio egzempliorius, kuriuos vadina korespondentais [4].

Taisyklių sandara gali būti atvaizduota OCL tokiu būdu: inkarai taps konteksto klasifikatoriais arba jų metodais, korespondentai, vartojami sąlygose, taps nuorodomis į UML klases, jų atributus, metodus (OCL kalboje kaip korespondentai gali būti vartojami tik metodai-užklausos, kurie nekeičia objektų reikšmių).

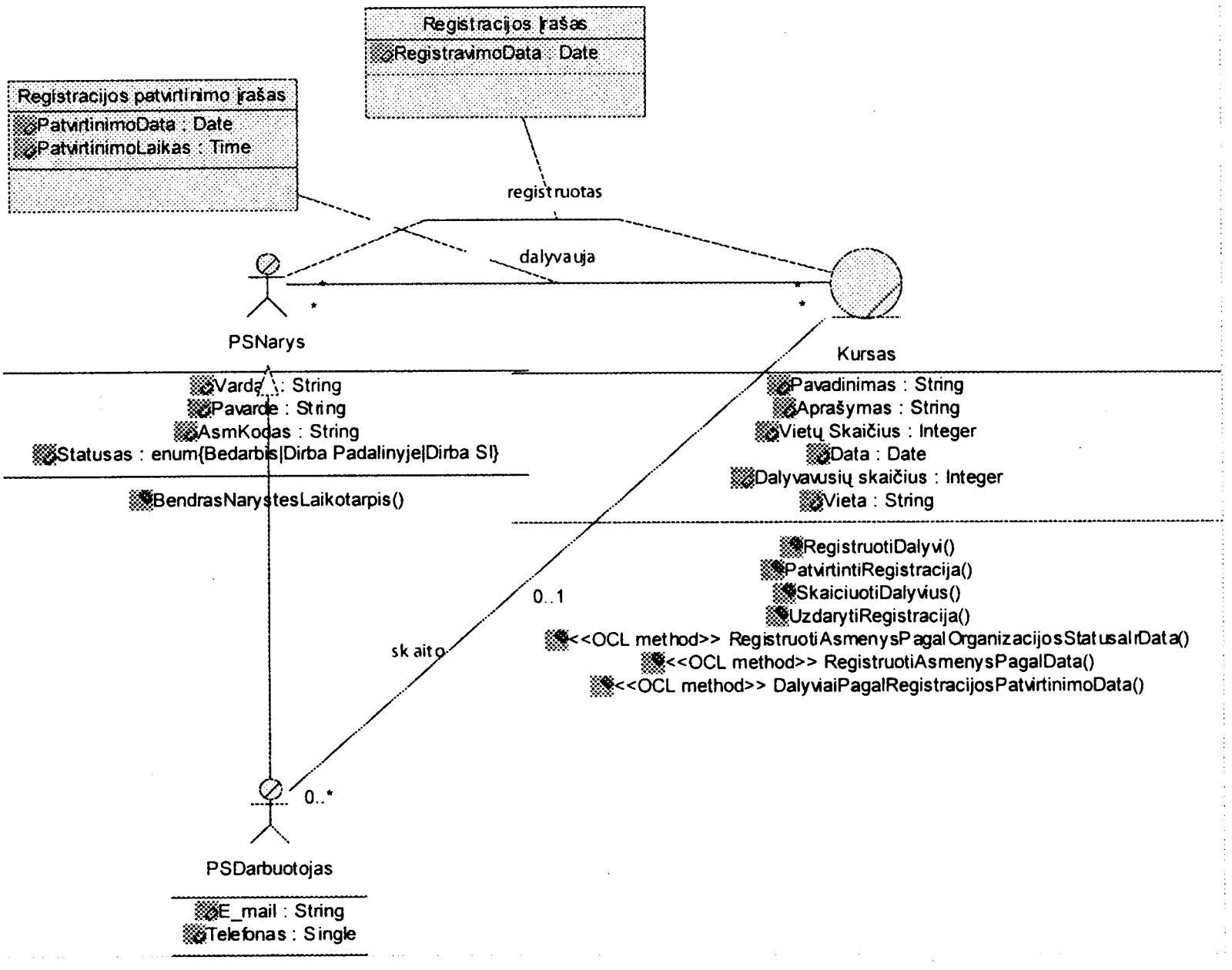
Bandymams panaudota dirbtinė dalykinė sritis panaši į Švietimo darbuotojų profesinės sąjungos dalykinę sritį. Tipinėje profesinėje sąjungoje galima išskirti kelias veiklos sritis: administravimo, mokymo, teisės. Dalykinės srities modelio fragmentai pateikti 1, 2 ir 3 paveikslėliuose. Kadangi vienas iš darbo tikslų – parodyti verslo taisyklių specifikavimo OCL galimybes, todėl dalykinės srities ypatumai bus nagrinėjami tik tiek, kiek to riekia siekiant tikslą.



1 pav. Dalykinės srities UML modelio pirmas fragmentas.



2 pav. Dalykinės srities UML modelio antras fragmentas.



3 pav. Dalykinės srities UML modelio trečias fragmentas.

3. Egzempliorių tikrinimo verslo taisyklių šeimos modeliavimas OCL

Kiekvienas inkaro egzempliorius šitoje taisyklių šeimoje tikrinamas pagal korespondento egzemplioriaus (egzempliorių) buvimą [4]. Taisyklių šeimą sudaro dviejų tipų taisyklės – privalomosios (angl. mandatory), ribotosios (angl. limited). Privalomosios taisyklės reikalauja atlikti patikrinimą, ar egzistuoja bent vienas korespondento egzempliorius. Ribotosios taisyklės reikalauja atlikti patikrinimą, ar egzistuoja tam tikras skaičius korespondento egzempliorių.

Privalomųjų verslo taisyklių pavyzdys. Profesinės sąjungos darbuotojas privalo turėti pareigų vykdymo pradžios datą.

Kadangi modelyje pareigų vykdymo pradžios datai yra iš klasės „Pareigos PS“ paveldėtas atributas „Nuo“, kontekstu gali būti klasė „Pareigos PS“ su atitinkamu ribojimu

context: Pareigos PS inv: self.Nuo.Yra()

OCL yra apibrėžtos egzistavimo reikšmės baziniams tipams. Data nėra bazinis tipas, todėl yra vartojama pagalbinė (angl. utility) klasė, kurioje yra metodas, turintis stereotipą „OCL method“ ir grąžinantis loginę reikšmę „tiesa“, kai data egzistuoja, ir loginę reikšmę „netiesa“, kai data neegzistuoja. Aišku, egzistavimo sąlyga datai turi būti apibrėžta papildomai modelio dokumentuose. Taip pat įmanoma užrašyti ribojimą, kai kontekstas – PSDarbuotojas

context: PSDarbuotojas inv: self.Pareigos PS.Nuo.Yra()

Dabar OCL invarianto sakinys panašesnis į neformalizuotai pateiktą taisyklę, bet tuo pat metu sakinys tapo sudėtingesnis, nes atsirado papildomas navigacijos mazgas.

Ribotųjų verslo taisyklių pavyzdys. Profesinės sąjungos padalinys organizacijoje gali būti sukurtas tik tada, kai organizacijoje dirba ne mažiau kaip trys profesinės sąjungos nariai:

context: PSpadalinys inv: self.Yra narys->count()>=3

Metodas count() priklauso OCL kalbai ir taikomas, kai argumento tipas yra rinkinis. Būtina pastebėti, kad navigacijos per asociatyvų ryšį išraiška šiuo atveju visada grąžina rinkinį.

4. Rūšies tikrinimo verslo taisyklių šeimos modeliavimas OCL

Rūšies tikrinimo taisyklių šeimoje reikalaujama atlikti kelių korespondentų egzempliorių patikrinimą siekiant nustatyti, koks jų skaičius yra teisingas (angl. true): jeigu korespondentas yra duomenų tipas, tai teisingas bus tada, kai bent vienas egzempliorius egzistuoja; jeigu korespondentas – kita taisyklė, tai teisingas reiškia, kad mažiausiai vienas taisyklės egzempliorius yra tenkinamas.

Šeimai priklauso tokių tipų taisyklės: abipusės (angl. mutual), abipusėsišimtinės (angl. mutual-exclusive), abipusėsįskaitančios (angl. mutual-inclusive), abipusėsdraudžiančios (angl. mutual-prohibited). Abipusio tipo taisyklės reikalauja patikrinti ar tam tikras skaičius korespondento egzempliorių yra teisingas duotu laiko momentu. Abipusėsišimtinės taisyklės reikalauja, kad ne daugiau kaip vienas iš dviejų arba daugiau korespondento tipų būtų teisingas. Abipusėsįskaitančios taisyklės reikalauja patikrinti, ar vienas arba daugiau korespondentų teisingi vienu metu. Abipusėsdraudžiančios taisyklės reikalauja, kad bent vienas iš dviejų arba daugiau korespondentų tipų nebūtų teisingai vienu metu [4].

Šitai šeimai užrašyti OCL kalba geriausia naudoti kolekcijų apdorojimo operatorius: count(), exists(), forAll(), notEmpty(), isEmpty(). OCL neturi priemonių kreiptis į atskirų taisyklių egzempliorius, išskyrus atvejus, kai taisyklė yra kitos taisyklės dalis ir yra sujungta loginiais operatoriais. Kreipimosi į atskiras taisykles reikalingumas turi būti papildomai ištirtas. Abipusės taisyklės gali būti suskaldytos į kelias privalomas, jeigu jos yra vartojamos kaip ribojimai.

Abipusės verslo taisyklės pavyzdys. Kandidatas prašyme į profesinės sąjungos narius turi pateikti asmens kodą ir dabartinę darbovietę:

context:PSNarys inv: self.AsmKodas and self.ŠIdarbuovietė ->notEmpty()

Abipusėsišimtinės verslo taisyklės pavyzdys. Profesinės sąjungos padalinio darbuotojas gali arba skaityti arba klausyti tą patį kursą:

context: PSDarbuotojas inv: not self.skaito->exists(elem : Kursas | elem = self.dalyvauja)

Panašiai transformuojasi į OCL ir kitų tipų šios šeimos verslo taisyklės.

5. Pozicijos tikrinimo verslo taisyklių šeimos modeliavimas OCL

Duomenų tipų egzemplioriai dažnai yra surūšiuoti tam tikru būdu. Rūšiavimas gali būti pagristas egzemplioriaus reikšme arba egzemplioriaus amžiumi – chronologiškas eiliškumas. Pozicijos tikrinimo verslo taisyklių šeimai priklauso taisyklės, nusakančios egzemplioriaus poziciją to paties tipo egzempliorių sekoje. Išskiriamos taisyklių tipai, kai rūšiavimas pagal reikšmę: pozicija, didžiausias, mažiausias. Kai rūšiavimas yra chronologinis, vartojami tokie tipai: seniausias, naujausias [4].

OCL nėra duomenų manipuliavimo kalba, bet rūšiavimą galima traktuoti kaip objektų reikšmių keitimą, todėl OCL kalboje neįmanoma rūšiuoti duomenis. Tačiau OCL yra apibrėžtas duomenų tipas seka (angl. sequence) su atitinkamais metodais: first(), last(), at() ir kitais. Tuo tarpu metodo, grąžinančio egzemplioriaus poziciją sekoje nėra. Tai sunkina pozicijos tikrinimo taisyklių modeliavimą OCL.

Rūšiavimo pagal reikšmę ir datą pavyzdžiai. Profesinių sąjungų nariai iš „problemi-nių“ organizacijų turi prioritetą dalyvaudami mokymo kursuose; kiti nariai dalyvauja pagal registravimo datą.

Skirtingai nuo aukščiau aprašytų ribojimų, kai kontekstas buvo klasė, dabar tikslinga naudoti kontekstą metodą. Tai yra ne dėl taisyklių šeimos, bet dėl dalykinės srities.

```
context:Kursas::PatvirtintiRegistracija() post:
if self.RegistruotiAsmenysPagalOrganizacijosStatusaIrData()-> count()>0
self.DalyviaiPagalRegistracijosPatvirtinimoData()-> subSequence(1,
self.RegistruotiAsmenysPagalOrganizacijosStatusaIrData()->count()) =
self.RegistruotiAsmenysPagalOrganizacijosStatusaIrData() else
self.DalyviaiPagalRegistracijosPatvirtinimoData()=
self.RegistruotiAsmenysPagalData()endif
```

Šitas OCL sakinys nors riboja ne visas įmanomas situacijas, jau yra pakankamai komplikuotas.

6. Funkcinio tikrinimo verslo taisyklių šeimos modeliavimas OCL

Funkcinio tikrinimo šeimos taisyklės tikrina korespondentų reikšmes kaip inkaro egzemplioriaus funkcijas. Naudojant šitos šeimos taisyklės, reikia turėti omenyje: eiliškumą, tvarka pagal kuria yra surūšiuoti inkaro egzemplioriai; taisyklės tipą,

unikalumą (angl. unique) – reikalauja reikšmės unikalumo, svyruojanti (angl. fluctuating) – reikalaujanti tam tikro žingsnio tarp reikšmių, didėjanti (angl. ascending) – reikalauja reikšmių didėjimo nuo vienos reikšmės iki kitos, mažėjanti (angl. descending) – reikalauja reikšmių mažėjimo nuo vienos reikšmės iki kitos, ne-atstatoma (angl. non-renewable) – riboja reikšmės pakartotiną naudojimą.

Unikalios verslo taisyklės pavyzdys. Nario asmens kodas turi būti unikalus:

```
context: PSnarys inv:PSnarys.allInstances->forAll(p1, p2|
p1 <> p2 implies p1.AsmKodas <> p2.AsmKodas)
```

Svyruojančios verslo taisyklės pavyzdys. Laiko tarpas tarp vienodų mokymo kursų turi būti ne mažesnis kaip trys mėnesiai:

```
context: Kursas inv: Kursas.allInstances->forAll(p1, p2 |
(p1 <> p2 and p1.Pavadinimas= p2.Pavadinimas) implies
p1.Data.SkyrtumasMenesiais(p2.Data) > 3
```

Didėjančios verslo taisyklės pavyzdys. Nario įstojimo data turi didėti:

```
context: PSNarys inv: self.Yra narys->forAll(p1| p1.
Įstojimo data <(self.Yra narys->append(p1)->first().
Įstojimo data))
```

Ne-atstatomos verslo taisyklės pavyzdys. Narys gali dalyvauto tos pačios temos kursuose tik vieną kartą:

```
context: PSNarys inv: self.dalyvauja->forAll(p1,p2| not p1=p2)
```

7. Palyginimo verslo taisyklių šeimos modeliavimas OCL

Verslo taisyklės vykdo palyginimą tarp panašių korespondentų tipų. Šitai šeimai priklauso sekantys taisyklių tipai: lygu (equal-to), ne lygu (not-equal-to), daugiau-kaip (greater-than), daugiau-kaip-arba-lygu (greater-than-or-equal-to), mažiau-negu (less-than), mažiau-negu-arba-lygu (less-than-or-equal-to) [4].

Kiekvienas iš aukščiau paminėtų taisyklių tipų gali būti tiesiogiai atvaizduotas į atitinkanti OCL loginį operatorių arba jų derinį.

Palyginimo verslo taisyklės pavyzdys. Vienas padalinys negali turėti daugiau kaip 100 narių, jeigu tai mokykla:

```
context: Organizacija inv: self.OrgTipas=#mokykla implies
self.veikia.Narių skaičius<100
```

8. Matematinio įvertinimo verslo taisyklių šeimos modeliavimas OCL

Matematinio įvertinimo šeimos taisyklės yra skirtos atlikti paprasčiausiais matematines operacijas su korespondentais. Numatomos operacijos: suma, atimtis, daugyba, dalyba, maksimumo radimas, minimumo radimas, vidurkio skaičiavimas, medianos radimas, modos skaičiavimas, dažnio skaičiavimas, procentų radimas, procentilio radimas [4]. Kiekvienas iš aukščiau paminėtų taisyklių tipų gali būti tiesiogiai atvaizduotas į atitinkantį OCL operatorių arba jų derinį.

Matematinio įvertinimo verslo taisyklės pavyzdys. Susivienijimo narių skaičius turi būti lygus padalinių narių skaičių sumai:

```
context: PSPadalinys inv: self.padalinys.Narių skaičius->
Sum() =self.Narių Skaičius
```

9. Projekcijos kontrolės verslo taisyklių šeimos modeliavimas OCL

Ši taisyklių šeima skiriasi nuo prieš tai aptartų variantų tuo, kad skirtingai nuo jų – tai ne atmetimo (angl. rejector) taisyklės o leidimo taisyklės (angl. projector). Vykdant šitos šeimos taisykles visada yra vykdomas perėjimas iš vienos modelio būsenos į kitą. Šeimai priklauso: suteikiančios (angl. enabled), kopijuojančios (angl. copied), įvykdančios (angl. executed) taisyklės. Suteikiančios taisyklės reikalauja korespondento egzemplioriaus(ių) buvimo, gali automatiškai sukurti/sunaikinti korespondento egzempliorių. Kopijuojančios taisyklės nurodo reikšmes kurių kopijos turi būti įrašytos korespondento egzemplioriuose. Įvykdančios taisyklės reikalauja įvykdyti korespondento egzempliorių (pvz., įvykdyti veiksmą arba iššaukti taisyklę) [4].

Specifikuojant leidimo taisykles OCL atsiranda akivaizdus semantinės problemos. OCL kalba yra ribojimų kalba, todėl tiesiogiai išreikšti tokias taisykles neįmanoma. Be to OCL buvo kuriama kaip kalba be „pašalinio efekto“, tai reiškia OCL negali keisti modelio egzempliorių, šiuo atveju to ir reikalaujama. Sprendimas yra spefikuoti šias taisykles, vartojant veiksmo sakinius.

Suteikiančios verslo taisyklės pavyzdys. Jeigu organizacijos darbuotojams per mėnesį buvo suteikta daugiau kaip dešimt teisinių konsultacijų, organizacijos statusas automatiškai tampa problematiškas:

```
context: Organizacija action: if self.ŠI
darbuotojas.konsultuojasi->
select(p1|p1.KonsultacijosData.SkirtumasMenesiaisNuoDabar()<1)->
count()>10 to self send ProblematiskasStatusas() end if
```

ir prie jo pridedame papildomai metodo ribojimą

```
context: Organizacija:: ProblematiskasStatusas() pre:
post: statusas=#problematiskas
```

Įvykdančios verslo taisyklės pavyzdys

Verslo taisyklė: Dalyvavusių mokymo kurse narių skaičius turi būti skaičiuojamas po kursų pabaigos.

```
context: Kursas::UzdarytiRegistracija()
action: to self send SkaiciuotiDalyvius()
```

10. Išvados

Darbe pateiktos skirtingos verslo taisyklių šeimos su attinkamais dalykinės srities pavyzdžiais. Verslo taisyklių pavyzdžiai yra suformuluoti neformaliai ir formaliai naudojant OCL. Parodyta kad daugelis verslo taisyklių šeimų gali būti specifikuotos naudojant OCL. Tai pat pastebėta, kad skirtingos verslo taisyklės gali būti išreikštos tomis

pačiomis OCL konstrukcijomis. Prie kiekvienos taisyklių šeimos pavyzdžių yra išanalizuotos pateikimo OCL kalba problemų aprašymas. Apibendrinant, galima teigti kad OCL kalboje sunku išreikšti pozicijos tikrinimo verslo taisyklių šeimą. Eiliškumas ir rūšiavimas nėra OCL stiprioji pusė, ir dėl to atsiranda tam tikras skaičius problemų. Tai pat neįmanoma specifikuoti projekcijos kontrolės taisyklių šeimą naudojant tik [2] pateiktą OCL variantą, jį būtiną išplėsti naudojant veiksmų sakinius iš [6, 7].

Modeliuojant pozicijos tikrinimo verslo taisyklių šeimą pastebėta, kad gali būti tikslinga papildyti OCL kalbą metodu position(), kuris butų taikomas „seka“ (sequence) tipo elementams ir grąžintų elemento poziciją sekoje. Naudojimo tikslingumas ir metodo semantika turi būti išanalizuota.

Literatūra

1. H. Herbst, *Business Rule-Oriented Conceptual Modeling*, Physica-Verlag Heidelberg (1997).
2. *OMG Unified Modelling Language*, Specification, Version 1.4, September (2001). `http://www.omg.org`
3. H.-E. Eriksson, M. Penker, *Business Modeling with UML, Business Patterns at Work*, John Wiley & Sons, Inc. (2000).
4. R.G. Ross, *The Business Rules Book. Classifying, Defining and Modelling Rules*, second edition, Business rule solutions, LLC (1997).
5. J. Warmer, A. Kleppe, *The Object Constraint Language. Precise Modeling with UML*, Addison Wesley Longman (1999).
6. A. Kleppe, J. Warmer, Extending OCL to include actions, *LNCS*, **1939**, 440–450 (2000).
7. A. Kleppe, J. Warner, The semantics of the OCL action clause, *LNCS*, **2263**, 213–227 (2002).

SUMMARY

S. Sosunovas, O. Vasilecas. Business rules modeling using OCL

Business rules are important components of every business system. Therefore creating information systems aimed to support business it is important to specify business rules unambiguously. Formal specification languages have different power specifying business rules of different types. This work argues that formal language OCL powerful enough to specify business rules of main types.

Keywords: business rules, OCL, information systems, business systems, formal specification languages.