KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Saulius Menkevičius

OBJEKTŲ SAVYBIŲ MODELIO GRAFINIS REDAKTORIUS

Magistro darbas

Vadovas
doc. dr. B. Paradauskas

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

OBJEKTŲ SAVYBIŲ MODELIO GRAFINIS REDAKTORIUS

Magistro darbas

Vadovas
doc. dr. B. Paradauskas

Recenzentas
doc. dr. A. Lenkevičius

Atliko
IFM 0/4 gr. stud.
S. Menkevičius

KAUNAS, 2006

SUMMARY

Graphical Editor for the Object Property Model

During the development of federated IS that make use of non-homogenous databases and data sources, XML documents are often used for data exchange among the local subsystems, while their corresponding XML Schemas are generated using the standard CASE tools for local systems. External data schemes of those systems must be specified in a unified common model. An assumption is given that OBJECT PROPERTY (OP) model is being used for the semantic integration of the local non-homogenous subsystems. A graphical editor was developed that can be used to specify relation objects, their identifiers, complex and multi-valued object attributes. As OP model's semantic expression capabilities can map those available in XML, additionally rules have been defined and implemented that can transform specific OP model structures into XML Schemas. Also algorithm is specified that can be used to extract tree-like structures from the model. Example transformations are performed that illustrate the process of generation of XML Schemas documents from sample OP models.

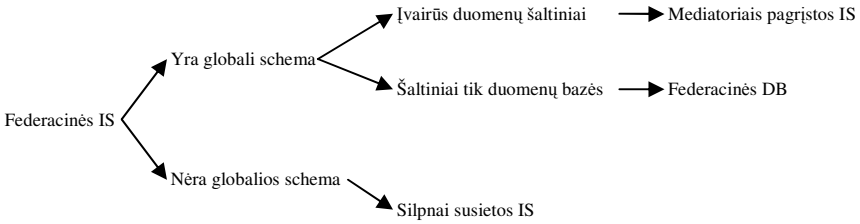
TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Federacinėse duomenų bazėse [1] naudojami autonomiški, heterogeniški duomenų šaltiniai, kurių integracija suteikia vartotojui universalų priėjimą prie duomenų bazių, kurios kitu atveju gali būti nesuderinamos techniškai, dėl skirtingo naudojamo duomenų modelio ar semantinio nesuderinamumo. Federacinės informacijos sistemos gali būti klasifikuojamos tokiu būdu:



1 pav. Federacinių informacijos sistemų klasifikacija

Darbo kontekstas yra federacinių duomenų bazių, turinčių globalią schemą, integracija. Vienas iš būdų semantinei integracijai atlikti – naudoti grafinius duomenų modelius. Kadangi jungiamos duomenų bazės gali būti nehomogeniškos duomenų modelio požiūriu, federacinės sistemos globalios schemos duomenų modelis privalo būti lankstus, jungiantis reliacines ir objektines savybes.

Integracijai užtikrinti pasiūlytas OBJEKTŲ SAVYBIŲ (OP) modelis [2], kuris:

- turi didesnes semantines išraiškos galimybes nei dauguma įprastų grafinių modelių (UML, EER diagramos, kt.);
- turi daug bendrų konstruktų su XML [3] žymių kalba, kuri populiari integruojant sistemas bei taikomąsias programas ir yra viena tinkamesnių kandidatų iliustruoti OP modelio tinkamumą sistemų modelių integracijai.

OP modelis yra vienas universalesnių duomenų modelių, galintis išreikšti modeliuojamų duomenų ne tik objektines, bet ir reliacines, hierarchines savybes. Šis modelis apibendrina EER ir objektinių modelių savybes – juo galima specifikuoti EER modeliui būdingas priklausomybes ir apribojimus, sudėtingus struktūrinius tipus bei elgseną, kas būdinga objektiniam modeliui. Toks modelis aktualus tiek teoriniu, tiek praktiniu požiūriu, kadangi šiuo metu paplitusios reliacinės duomenų bazės turi ir objektinių savybių.

OP duomenų modelio grafinio redaktoriaus programinis palaikymas nebuvo sudarytas, todėl vienas iš darbo tikslų yra suprojektuoti ir sudaryti OP modelio grafinį redaktorių, kurio galimybės būtų išplėstos XML schemų generavimo funkcijomis.

Duomenų modelis abstrakčiai apibrėžia, kaip duomenys yra saugomi informacinėse sistemose, duomenų bazių valdymo sistemose. Nors paprasti duomenų modeliai, susidedantys iš kelių objektų

gali būti sukuriami tiesiogiai, platesnio masto sistemoms sukurti yra reikalingi struktūrizuoti būdai duomenų apibrėžimui. Iš tokių metodų galima išskirti esybių ryšių diagramą ERD (*angl.* Entity-Relationship Diagram), kuri apibrėžia taikomosios srities duomenų modelį. Kiti metodai, tokie kaip ORM (*angl.* Object Role Modeling) ar UML (*angl.* Unified Modeling Language) turi galimybę išreikšti ne tik statines, bet ir dinamines savybes.

Heterogeninių duomenų bazių ir informacinių sistemų integracijai plačiai naudojamos XML technologijos. XML formato dokumentai gali būti griežtai apibrėžiami, yra lengvai perskaitomi ir apdorojami įvairių taikomųjų programų. Siekiant aprašyti tam tikro tipo XML dokumento semantiką bei sintaksę anksčiau buvo naudojamas DTD (*angl.* Document Type Definition) formatas, originaliai skirtas SGML [4] dokumentams apibrėžti, tačiau naudojamas ir su XML. Pastaruoju metu vis populiaresnės tampa XML schemos [6], kurios naudojamos patikrinti dokumento tinkamumą, ar jis teisingai suformuotas, taip pat su dokumento struktūroms susieti metaduomenis, apibrėžiančius šių struktūrų turinį bei paskirtį.

Nors XML kalba yra skirta duomenų perdavimui tarp įvairių, nesusietų informacinių sistemų, ji nebūtinai yra lengvai integruojama į reliacinėmis bazėmis paremtas sistemas. Pastaruoju metu tam skirta daug dėmesio ir daugelis SQL duomenų bazių gamintojų siūlo integruotas arba išorines priemones šiam procesui palengvinti [8]. Šiame darbe XML schemų dokumentų generavimui yra siūloma naudoti universalų duomenų OP modelį. Tokiu būdų taikomosios sistemos duomenų modelis yra apibrėžiamas griežčiau ir vaizdžiau, jis gali būti panaudojamas ir reliacinei IS duomenų bazei apibrėžti. Taip pat kur kas paprasčiau atsekamas XML dokumentas bendros federacinės duomenų schemos kontekste.

Kadangi OP modelis yra paremtas grafine notacija, praktiniam modelio išraiškos bei transformacijos galimybių tyrimui yra reikalinga OP modelio realizacija grafiniame redaktoriuje. Šiuo metu galima išskirti kelis grafinių redaktorių tipus: universalūs, leidžiantys sudaryti bet kokios sudėties, neapibrėžtos semantikos schemas, ir redaktoriai, realizuojantys griežtai apibrėžtą modelį bei notaciją. Darbe suprojektuotas ir realizuotas antrojo tipo redaktorius, skirtas OP modelio schemoms sudaryti.

Modelio universalumo iliustravimui sudaromas ir realizuojamas OP modelio transformavimo į XML schemą algoritmas. Šis algoritmas, integruotas redaktoriuje, leidžia iš OP grafe pažymėto medžio gauti jam izomorfiško XML schemos dokumento atitikmenį. Taip gauta schema yra tinkama apibrėžti duomenis, integruojamus į, arba ištraukiamus iš originaliu OP modeliu apibrėžtos duomenų bazės.

Darbo tikslai yra:

- Sudaryti grafinio OP modelio redaktoriaus projektą, jį realizuoti;
- Sudaryti OP į XML transformavimo taisykles;
- Realizuoti šias taisykles bei atlikti eksperimentinį jų tyrimą grafinio redaktoriaus aplinkoje.

Darbas sudarytas iš 5 skyrių, iš kurių pirmuose dviejuose išnagrinėti OP ir XML schemos modeliai; trečiajame sudarytos taisyklės ir algoritmas XML schemoms generuoti; ketvirtame skyriuje išnagrinėti reikalavimai kuriamai programinei įrangai, parinkta redaktoriaus architektūra; penktajame skyriuje aprašomas sudarytas programos projektas, charakterizuojamas ir pateikiamas redaktoriaus programinės įrangos eksperimentinio panaudojimo aprašymas.

Darbo naujumą sudaro galimybė grafiškai specifikuoti objektų visas standartizuotas struktūrines savybes, žinomas objektiniuose ir reliaciniuose duomenų modeliuose:

- Objektų atributus ir ryšius tarp objektų;
- Sudėtinius ir daugiareikšmius atributus;
- Objektų vaidmenis ryšiuose;
- Daugiareikšmes priklausomybes tarp sudėtinių atributų;
- Funkcines ir daugiareikšmes duomenų priklausomybes;
- Funkcines siurjektyviąsias ir bijektyviąsias, pilnąsias ir dalines funkcines priklausomybes tarp sudėtinių atributų.

Kitas darbo naujumas yra tai, kad pateiktose XML transformavimo taisyklėse yra galimybė perteikti objektų apibendrinimo ryšius:

- Išskirtinius;
- Laisvuosius;
- Pilnuosius;
- Išskirtinius–pilnuosius.

1. OBJEKTŲ SAVYBIŲ MODELIS

Tyrime dėmesys koncentruojamas į Objektų savybių modelį, todėl šiame skyriuje yra detaliau apibrėžiama šio modelio kilmė, notacija, apribojimai ir pateikti pavyzdžiai. OP modelis yra pakankamai lankstus, todėl naudingas tiek teoriniu, tiek praktiniu požiūriu.

Kaip jau minėta anksčiau, OP modelis apibrėžia reliacines ir objektines duomenų savybes, todėl padengia EER modeliui būdingas savybes ir apribojimus. Juo galima modeliuoti vidines ir išorines priklausomybės, apibrėžti veiklos taisykles, kurti naujus duomenų tipus. Tuo jis panašus, tačiau platesnis negu ORM [9], kadangi pastarajame negalima specifikuoti objektų elgsenos, taip pat sudėtingos struktūros dokumentų.

Dėl plačios OP modelio darinių įvairovės, jo darinius galima skirstyti į reliacinius ir objektinius, priklausomai nuo išreiškiamos schemos, tačiau bendru atveju jo darinių negalima skirstyti pagal šį kriterijų. Objektinis OP modelis apima struktūrinio tipo specifikaciją, papildytą tipinių objektinių duomenų modelių funkcijomis ir vartotojo nustatytomis operacijomis [2]. Pereinant prie realios realizacijos (programų, ar reliacinės duomenų bazės sudarymo metu) šios objektinės modelio savybės transformuojamos į procedūrinę formą.

1.1.1. Modelio apibrėžimas

OP modeliui sudaryti yra išskiriami objektai iš dalykinės srities objektų aibės D. Objektas yra konceptualiosios schemos objektas, jeigu pasižymi bent viena iš žemiau nurodytų savybių.

- Objektas *O* gali turėti atributą *A* iš aibės *aset(O)*. Jei objektas *O* turi egzempliorių *o*, tuomet atributas *o.A* turės reikšmę iš atributo *A* reikšmių aibės *vset(A)*. Atributai gali būti daugiareikšmiai ir neprivalomi. Daugiareikšmių atributų atveju objekto egzemplioriaus *o* atributas *A* turės poaibį reikšmių iš *vset(A)*. Tiek atributas *A*, tiek kiekviena iš atributo reikšmių *vset(A)* priklauso objekto savybių aibei *pset(O)*. Kadangi atributas vienu metu priklauso tiek objekto savybių aibei *pset(O)*, tiek jo atributų aibei *aset(O)* jo nebūtina specifikuoti kiekvienoje aibėje. Dėl atributų tipų įvairovės, jų reikšmės gali būti susietos su objekto egzemplioriumi įvairiais funkciniais ryšiais. Šių ryšių įvairovė gali būti sumažinama, įvedant naujus objektų tipus t.y. semantiškai normalizuojant.
- Objektas gali turėti vidinį ryšį, kuris dar vadinamas sudėtiniu atributu. Jis, kaip ir atributas gali būti daugiareikšmis arba neprivalomas (su neįvardinta reikšme). Vidinį ryšį sudaro pastovios sekos atributų aibė iš objekto *o* atributų aibės *aset(O)*, pats ryšys *R* priklauso objekto *O* savybių aibei *pset(O)*. Vidinis ryšys taip pat gali būti identifikacinis, t.y. $vset(R) \rightarrow vset(O)$. Tuo atveju jis yra rodomas kairėje, o atributų sąrašas nukreiptas į viršų. Identifikaciniai ryšiai gali būti keli. Objektas *O* taip pat gali būti suprojektuotas į savo vidinį ryšį; tokios projekcijos

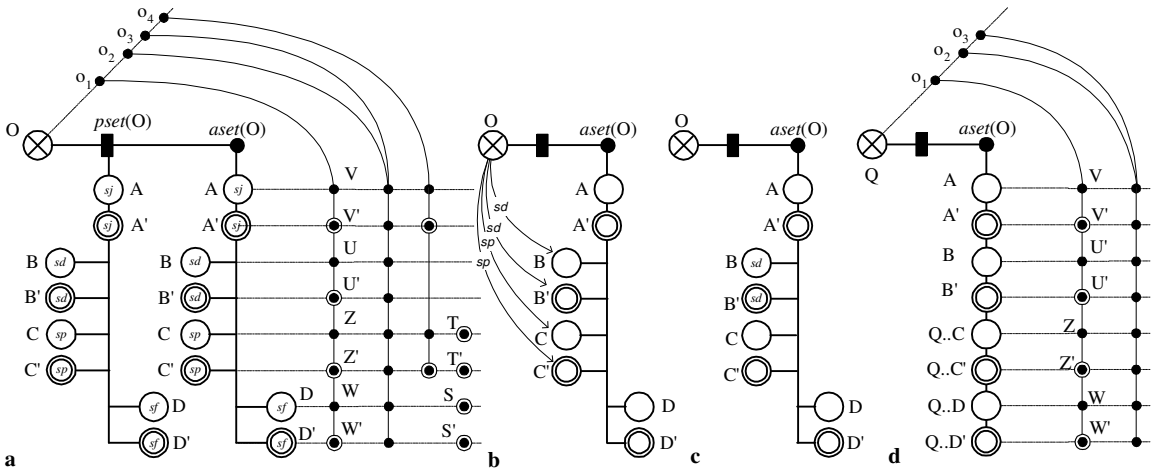
žymimos kryželiu. Jei kiekvienam objekto *O* egzemplioriui *o* galima priskirti vieną savybės *R* (vidinio ryšio) reikšmę, tuomet *O* ir *O.R* sieja siurjunktyvinio tipo (*sj*) ryšys: *vset(O)* → *vset(R)*. Jei ryšys identifikacinis, tai yra funkcija, atvaizduojanti objekto ryšio *R* reikšmę *r* į objekto egzempliorių aibės *O* elementą *o*. Jei atributas nepatenka į jokį vidinį ryšį, tuomet jis laikomas *laisvuoju*.

- Išorinė funkcinė priklausomybė taip pat gali būti atskiru (šakniniu) objektu. Turint objektus, kurie susideda iš dviejų ryšių *R* ir *S*, bei yra juos siejanti išorinė priklausomybė, vienareikšmiškai atvaizduojanti ryšio *R* reikšmių aibę į ryšio *S* reikšmių aibę. *vset(R)* → *vset(S)*, tuomet galima šią priklausomybę deklaruoti, kaip atskirą objektą, ryšį *R* įkeldami, kaip identifikuojantį vidinį ryšį jame, o ryšio *S* atributus, kaip laisvuosius to objekto atributus.
- Objektas gali turėti savo elgsenos metodą *m* iš objekto metodų aibės *mset(O)*. Metodas apibrėžia *sj* funkcinį ryšį tarp įėjimo vidinio ryšio *R*, ir metodo rezultatų vidinio ryšio *S* reikšmių aibių. *m* apibrėžiamas kaip funkcija: *m*: *vset(O.R)* → *vset(O.S)*.
- Objektas *O* taip pat gali būti kito objekto *P* savybe, t.y. jo:
  - Atributu: $P \in aset(O)$;
  - Atributo reikšme (daugiareikšmiu atributu): $P \in vset(O.A)$;
  - Vidiniu ryšiu (sudėtiniu atributu): $P \in pset(O)$;
  - Vidinio ryšio reikšme (sudėtinio atributo reikšme): $P \in vset(O.R)$.

1.2. Grafinė notacija, pavyzdžiai

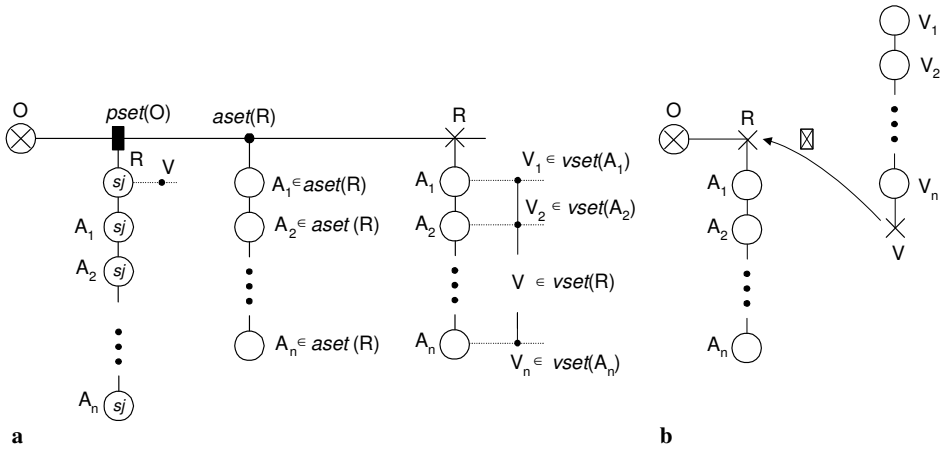
Žemiau pateiktame pavyzdyje nurodoma objekto *O* egzempliorių o_1, o_2, o_3, o_4 ryšys su jo atributų reikšmių aibėmis. Daugiareikšmiai atributai bei jų reikšmės nurodytos dvigubais apskritimais. Pateikti laisvieji atributai *aset(O)*, kurie gali būti (daugiareikšmiai atributų variantai pažymėti apostrofais (*A', B', C', D'*) (paveikslų šaltinis – [2]):

- Privalomi, apibūdinantys objektą – *A, A';*
- Neprivalomi, tačiau visos reikšmės iš *vset(B|B')* yra naudojamos apibūdinant objektus $o \in vset(O)$ – *B, B';*
- Privalomi, tačiau turi reikšmių *vset(C|C')*, kurios nėra naudojamos apibūdinant objektus $o \in vset(O)$ – *C, C';*
- Neprivalomi ir turi nenaudojamas reikšmes – *D, D'*.



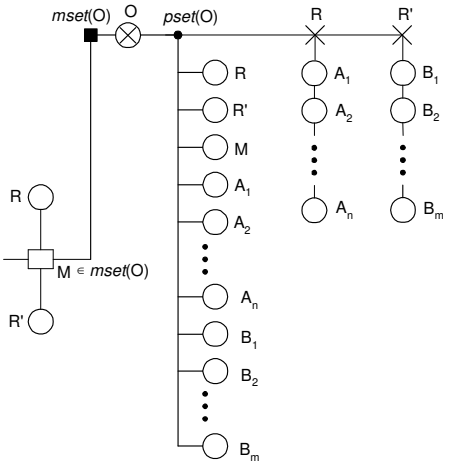
1.1 pav. OP objekto ir jo reikšmių, atributų, atributų reikšmių santykio įvairovė

Sekančiame paveiksle pateiktame pavyzdyje nurodytas vidinis ryšys *R*, jo projekcija (pažymėta kryžiuku), į kurį įeina atributai A_1-A_n. Taip pat galima nurodyti priklausomybę *vset(R)* ∈ *vset(V)* su išorinio ryšio jungtimi tarp projekcijų *O.R* ir *V*.



1.2 pav. OP objekto O vidinių ryšių išraiškos

Objektų elgsena apibrėžiama metodais (operacijomis), kurių grafinė išraiška nurodyta žemiau. Metodo įėjimo reikšmių aibė nurodoma virš metodo stačiakampio, o rezultatų reikšmių aibė – žemiau.

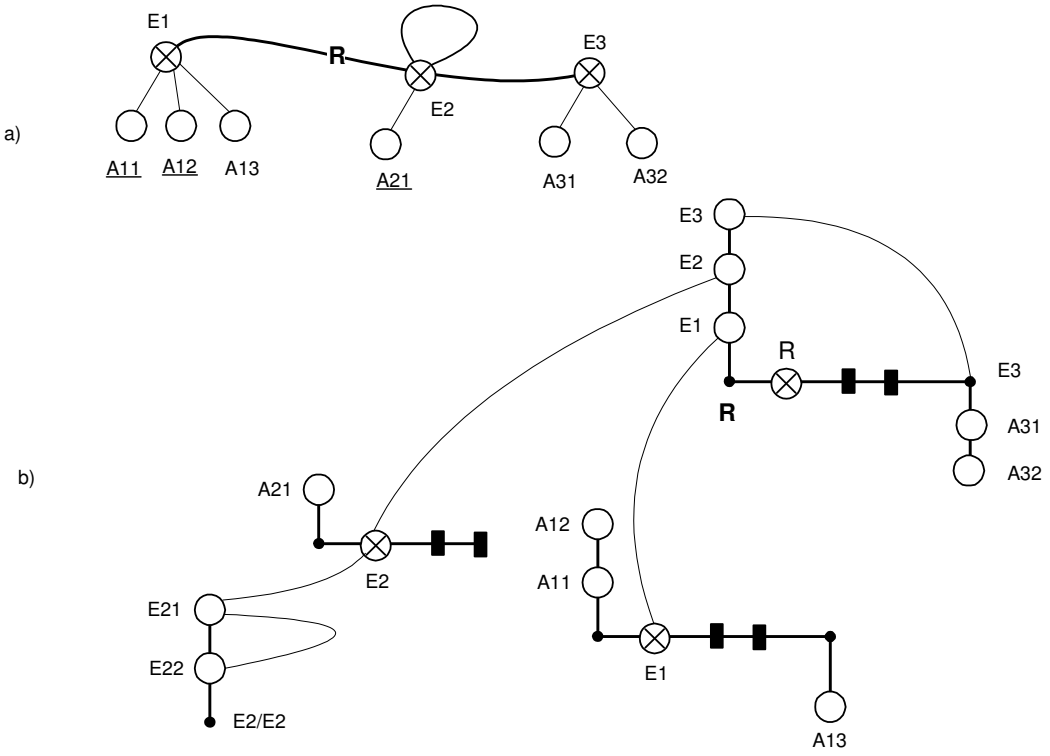


1.3 pav. Metodo M ∈ mset(O) signatūros išraiška

1.3. Grafinio OP duomenų modelio semantinės išraiškos galimybės.

Straipsnyje [25] duomenų schemų grafiniam vaizdavimui naudojami duomenų modelio hipergrafai DMHG (1.4 a pav.), kurių notacijoje esybių E1, E2 ir E3 raktiniai atributai pabraukiami ir jie neturi specialaus grafinio žymėjimo.

Pasiūlytame OP grafiniame modelyje (1.4, b pav.) esybių identifikatoriai nukreipiami į viršų. DMHG pavaizduotas R ryšys modelyje traktuojamas nauju objektu su identifikatoriumi (E1, E2, E3). Objekto R savybės E1, E2, E3 yra savo ruožtu ekvivalentūs struktūriniams tipams E1, E2 ir E3. Kaip parodyta b paveiksle refleksyvinis esybės E2 ryšys vaizduojamas binariniu ryšiu E21/E22.



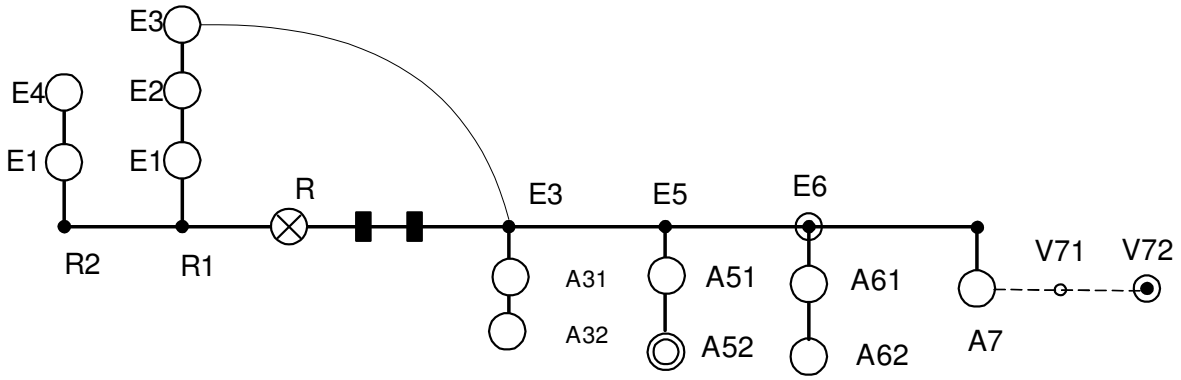
1.4 pav. ER duomenų modelio du grafinio vaizdavimo būdai: a) – naudojant Theodoratos hipergrafus; b) – naudojant OP grafus

Skirtingai nuo DMHG, kur esybių identifikatoriai neturi grafinio vaizdavimo OP grafinėse schemose:

1) Ta pati esybė gali turėti kelis raktus;
2) Atributai gali būti sudėtiniai, pvz. E3 turi du paprastuosius (terminalinius) atributus A31, A32; sudėtiniai traktuojami kaip objekto R vidiniai ryšiai;

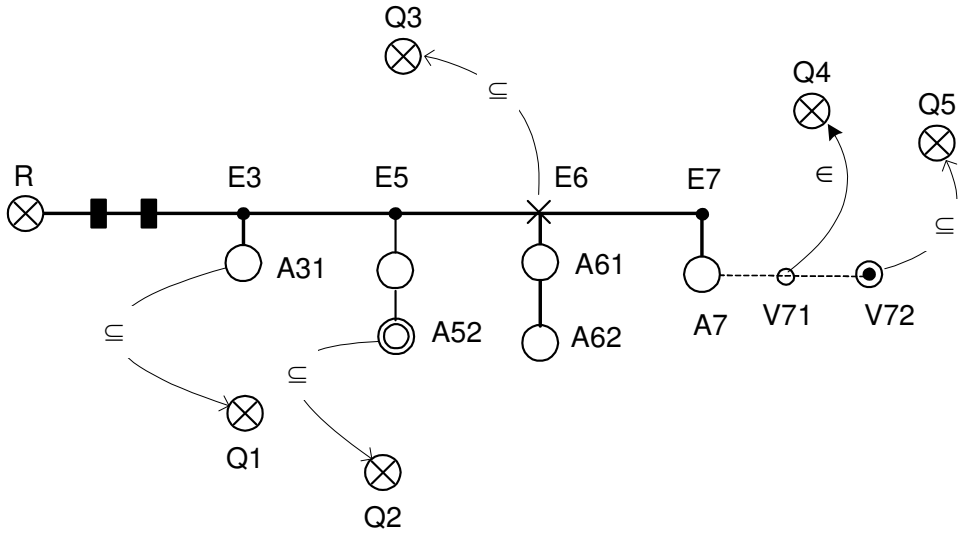
OP grafinėse schemose, be to:

3) Atributai gali būti daugiareikšmiai; jie žymimi dvigubu apskritimu;



1.5 pav. OP duomenų modelyje esybė R gali turėti: kelis identifikatorius (R1, R2); daugiareikšmius atributus (A52, E6); sudėtinius atributus (E5, E6); atributų paprastąją ir sudėtinę (V71, V72) reikšmes

Galimybė specifikuoti koncepcinėse duomenų schemose daugiareikšmius ir sudėtinius atributus, o taip pat paprastąsias ir sudėtines atributų reikšmes svarbi tuo, kad šie keturi koncepcinių schemų konstrukcijos turi tiesioginį pavaizdavimą XML schemose, t.y. jie pavaizduojami be semantinių nuostolių, ir todėl XML schemų sudarymo iš OP grafinio modelio taisyklės tampa paprastos.



1.6 pav. Esybė R integruotame duomenų OP modelyje su kitais objektų tipais gali jungtis išorinėmis poaibių ir aibės elemento priklausomybėmis

XML dokumentuose tokie išoriniai ryšiai realizuojami nuorodomis.

1.4. XML bei XML schemos

Kadangi numatoma redaktoriaus modelio panaudojimą iliustruoti OP modelio dokumentą transformuojant į XML schemos dokumentą, šiame skyriuje trumpai apibrėžiamos XML ir XML schemų savybės, struktūra bei semantika. Siūloma modeliuoti XML schemas naudojant OP modelį [2] duomenų savybių išraiškai. OP modelis sujungia reliacinį bei objektinį modelius ir tam naudojama

bendra notacija. Šis modelis ekspresyviai turtingas, leidžia išraiškingiau nei EER ar kitus objektinius modelius, nurodyti duomenų reliacines bei objektines savybes.

Dėl savo nepriklausomumo nuo DBVS ir platformos tipo, XML tapo ypatingai naudingas atviroms susisiekiančioms sistemoms, kuomet duomenys apsikeičiami tarp heterogeninių duomenų šaltinių. Atsiradus tokioms XML užklausų kalboms kaip XML-QL, XQL, Quilt, ir kt. [9], tapo įmanoma analizuoti XML duomenis nedetalizuojant juos realizuojančios DBVS.

1.4.1. XML

XML (*angl.* Extensible Markup Language) – išplečiama žymių kalba buvo pradėta kurti apie 1996-uosius metus pagrindine to priežastimi buvo tai, kad jos pirmtakė SGML [4] buvo pernelyg sudėtinga, kad būtų laisvai naudojama informacijos pasikeitimui Pasaulinio tinklo susietose informacinėse sistemose. Pirmos kalbos versijos standartizavimo procesas užtruko kelis metus ir buvo baigtas 1998, paskelbus W3C konsorciumo rekomendaciją XML kalbai [3].

XML yra duomenų aprašymo kalba, paremta medžio tipo struktūra. Tokia struktūra yra dažnai naudojama programinėse priemonėse ir tinka aprašyti daugumą duomenų tipų. Esminiai XML struktūros elementai yra *Elementai*, kurie gali būti tiek su turiniu (t.y. kitais elementais) (*angl.* container element), tiek be (*angl.* node element). Elementai taip gali turėti vieną ar daugiau *atributų*, kurie susiejami su tam tikru tipu bei jiems priskiriama reikšme ar reikšmėmis. Elementų turinys gali būti kiti elementai ir priskirto tipo simboliškai išreikšta reikšmė.

Privalumai

- XML dokumento formatas yra lengvai perskaitomas žmogaus ir skaičiavimo mašinose;
- Naudoja Unicode tarptautinį simbolių formatą, todėl pritaikomas praktiškai visų kalbų dokumentams;
- Gali išreikšti pagrindines informatikoje naudojamas struktūras: sąrašus, įrašus, medžius;
- Pats save aprašantis dokumentas, nurodantis reikšmių bei elementų prasmę;
- Griežtai apibrėžtas todėl gali būti perskaitomas ir patikrinama automatizuotomis programinėmis priemonėmis;
- Pagrindinė išraiška yra tekstinis failas, todėl prieinamas visose sistemose ir atsparus technologijų pasikeitimams;
- Yra kilęs ir dalinai suderinamas su SGML [4], todėl jau yra daug jį palaikančių taikomųjų programų.

Trūkumai:

- XML formato išraiška turi daug perteklinių elementų ir todėl gali būti sunkiai perskaitoma ir reikalauja daugiau vietos talpyklose. Iš dalies gali būti atsveriama panaudojus duomenų kompresiją, tačiau tai nėra visada įmanoma;
- Rekursyvi struktūra reikalauja daug skaičiavimų jos patikrinimo ir perskaitymo metu, kas gali sudaryti didesnį apkrovimą skaičiuojamam elementui negu pats dokumento turinio apdorojimo procesas;
- Kai kurie laiko sintaksę pernelyg sudėtinga ar paveldinčią pernelyg daug nereikalingų savybių iš pirmtakės SGML;
- Dažniausiai nuskaitymo procese negalima nustatyti perskaitomos informacijos įrašų tipo neturint dokumentą aprašančios duomenų schemos (DTD ar XML schemos);
- Sudėtinga išreikšti nehierarchiškas duomenų struktūras;
- Objektinių ar reliacinių duomenų perteikimas į XML dokumentą gali būti neefektyvus, sukeliantis pašalinių efektų, anomalijų.

XML dokumento tikrinimas

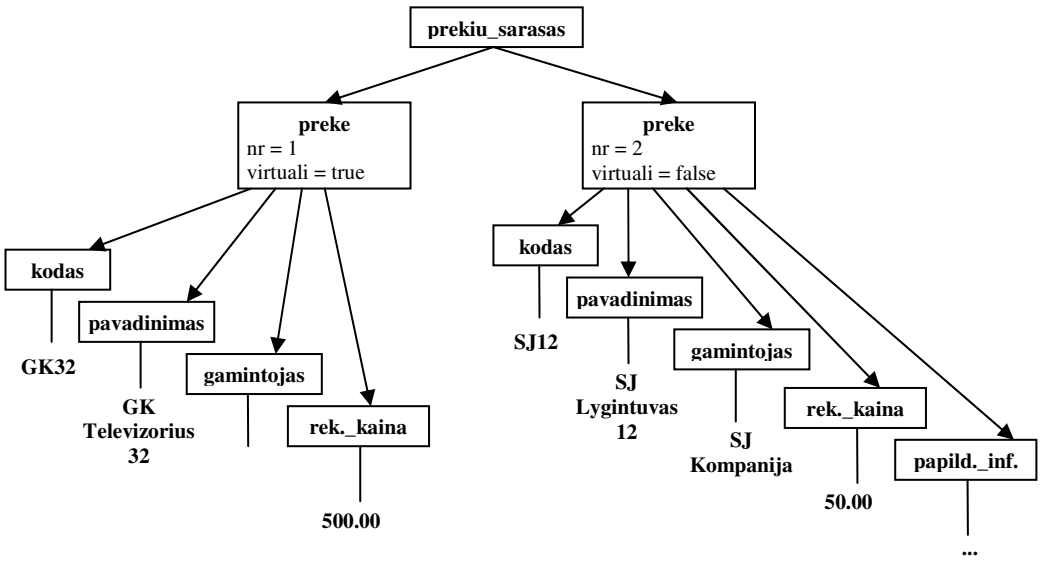
Dėl savo tekstinės prigimties ir medžio tipo struktūros prieš duomenis interpretuojant dažnai yra reikalinga formato patikra (*angl.* validation). XML dokumento tikrinimas gali būti įgyvendinamas dviem lygiais:

A) Ar dokumentas atitinka XML sudarymo taisykles.

Šią taisyklę tenkinantis dokumentas atitinka XML sintaksės taisykles, t.y. jis yra hierarchiškas dokumentas, kiekvienas elementas turi atidarymo ir uždarymo žymes (*angl.* tag) ir kt. Jeigu dokumentas nepatenkina šio lygio tikrinimo jis laikomas netinkamu ir nėra tikrinimas sekančio lygio (B) taisyklėmis.

B) Ar dokumentas atitinka vartotojo duomenų išraiškos (sintaksės) taisykles.

Jei XML dokumentas atitinka vartotojo nurodytas dokumento sudarymo taisykles, jis laikomas tinkamu (*angl.* valid). Tokių vartotojo taisyklių pavyzdys: nurodant asmens duomenų įrašą, jis turi turėti vardo bei pavardės vidinius elementus, saugančius po teksto tipo eilutę. Šio tipo patikrinimas dažnai yra atliekamas XML tikrintojais, kurie tikriną dokumentą pagal jo tipo aprašą (schemą).



1.7 pav. XML dokumento konceptualus medis

Šio dokumento XML tekstinis atitikmuo pateiktas 1 priede.

Formaliai XML medis apibrėžiamas kaip $T = (V, lab, ele, att, root)$, kur [7]:

- $V \subseteq$ Vert, V – viršūnių aibė;
- $Lab: V \rightarrow El$ ir $ele: V \rightarrow Str \cup V$, kur El – elemento vardai, Str – tekstinio tipo atributų galimos reikšmės;
- Att yra dalinė funkcija $V \times Att \rightarrow Str$, kur Att – atributo vardų aibė;
- $root \in V$ – medžio T šaknis;
- $paths(T)$ – XML medžio T kelių aibė.

1.4.2. XML schema

XML schema yra vienas iš XML dokumentus apibrėžiančių schemų tipų. Jo pirmtakas buvo DTD, tačiau ilgainiui pastarojo galimybių neužteko. Išleistas kaip W3C rekomendacija 2001 [6] šiuo metu tampa populiariausiu XML schemos formatu. Kūrimo metu stengtasi perimti geriausias DTD ir ankstyvųjų XML schemų formatų (DDML, SOX, XML-Data) savybes. Šiuo metu XML schema didžiausių programinės įrangos tiekėjų pripažinta kaip pagrindinė XML schema.

Ši schema apibrėžia dokumento struktūrą aukštesniu lygiu nei XML kalba, kuri užtikrina tik dokumento atitikimą tekstiniam XML formatui. XML schema suteikia XML dokumento elementams apibrėžtą sintaksę bei prasmę: elementų hierarchiją, atributų aibę, skaitinius, tekstinius ir kitus apribojimus. Jeigu XML dokumentas atitinka programos nurodytą XML schemą, jis galima būti priimamas ir interpretuojamas tos taikomosios programos. XML schemos dokumentas tuo pačiu metu yra XML dokumentas, t.y. XML schemos dokumento formatas taip pat gali būti apibrėžiamas atitinkama XML schema.

Esminė schemos paskirtis yra nustatyti duomenų formato susitarimą tarp duomenų šaltinio ir priėmėjo. Tokiu būdu dokumentai gali būti patikrinami prieš jų panaudojimą. Kadangi realūs duomenys, perduodami tarp informacinių sistemų turi sudėtingą, tarpusavyje susijusią struktūrą, šios struktūros apibrėžimas taip pat reikalauja pastangų. Todėl XML schema gali pasirodyti pernelyg sudėtinga savo paskirčiai, tačiau to reikalauja jos detalumas ir plati galimybių aibė.

Privalumai

- Išskiriami ir tiksliai apibrėžiami informacinės sistemos duomenų formatai, paskirtis tiek išsiunčiami, tiek priimami duomenys gali būti patikrinami
- Trivialūs, žmogaus parašyti dokumentai, gali būti integruojami į informacinę sistemą, nes yra galimybė juos patikrinti, jog jie teisingai suformuoti
- Centralizuotai saugomas duomenų apibrėžimas leidžia sukurti nepriklausomas programines priemones jų apdorojimui

Trūkumai

- XML schema yra sąlyginai sudėtinga ir reikalauja daug pastangų, kad būtų panaudojama, juo labiau efektyviai
- XML schemos, nors ir tiksliai apibrėžiančios dokumento gramatiką ir semantiką nenurodo tikslaus dokumento laukų panaudojimo ir/ar interpretavimo
- Jos panaudojimas reikalauja daug techninių resursų ir projektuotojų pastangų. XML schemų tikrintojai ir nagrinėtojai (*angl.* parser) privalo būti pilnai realizuoti, kad galima būtų nurodyti dokumento atitikimą schemai. Tokia programinė įranga yra sudėtinga ir gali būti lėtai vykdoma

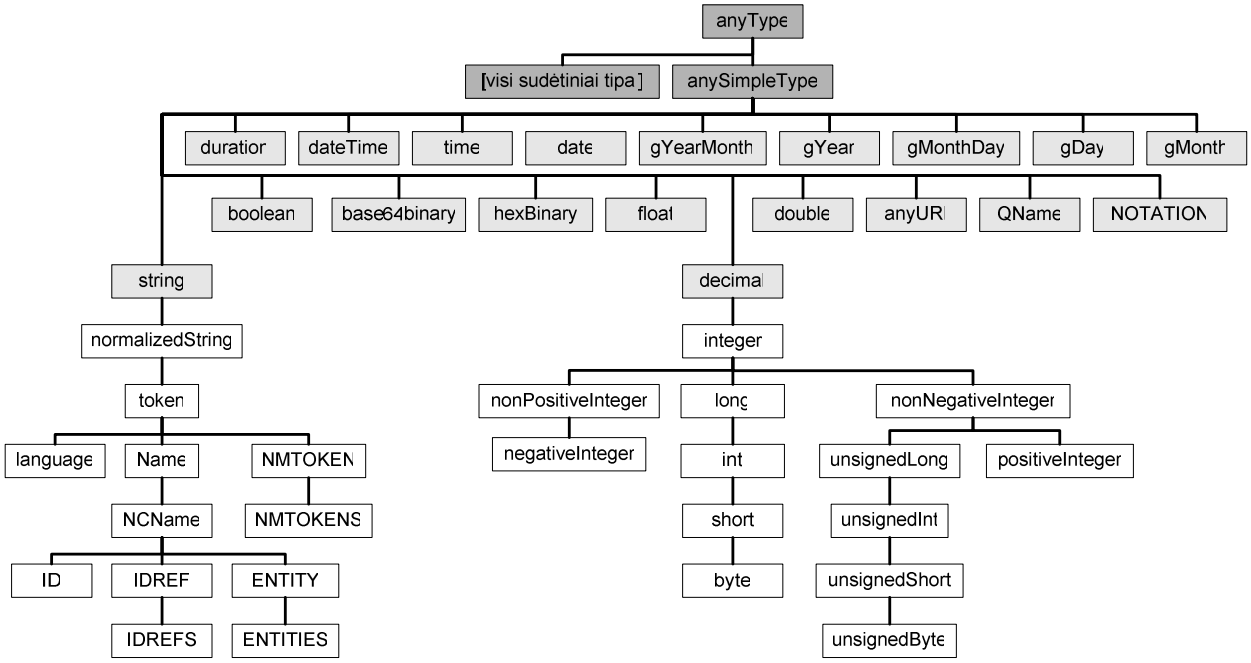
Struktūra

Pagrindinės schemos struktūrinės dalys yra duomenų, elementų, atributų tipų, vardų erdvės. XML schemos gali būti panaudotos viena kitos aprašui, kas leidžia lengvai išplėsti ir sukurti naujus schemų tipus. Keičiantis programinei įrangai (PĮ) išvengiamos suderinamumo problemos, pagerinamas PĮ moduliarizmas.

XML schemos duomenų tipai sudaro nepersidengiančią hierarchiją iš jau sudarytų, iš anksto nustatytų XML schemos ir vartotojo aprašytų duomenų tipų ir skirstomi į sudėtingus (angl. *complex*) bei paprastuosius (angl. *simple*). Žemiau pateikiama XML schemos duomenų tipų hierarchija [12]:

- Paprastieji tipai yra aprašomi `simpleType` elementu ir sudaromi patikslinant ar išplečiant jau egzistuojantį paprastąjį tipą. XML schema apibrėžia pagrindinius duomenų tipus kaip sveikasis skaičius, teksto eilutė ir kt.;

- Sudėtingieji tipai, priešingai nei paprastieji gali turėti vidinius elementus bei priskirtus atributus. Naudojant XML schemos elementą `complexType`, sudėtingieji tipai sukuriami, išplečiant jau egzistuojantį sudėtingąjį tipą;
- Duomenų tipai gali būti neįvardinti, jeigu naudojami tik vieno XML schemos elemento apibrėžimui, kas leidžia sutrumpinti ir supaprastinti XML schemos aprašą.



1.8 pav. XML schemos apibrėžtų duomenų tipų hierarchija

Elementai yra apibrėžiami nurodant jų turinio duomenų tipą, kuriame atitinkamai nurodyta kokius atributus elementas gali turėti, kokie galimi vidiniai jo elementai, ar galimas mišrus turinys (*angl.* mixed content), t.y. sumaišytas simbolinis-tekstinis bei elementų turinys – analogas HTML `<body>` elementas. Elementas taip pat gali neturėti vidinio turinio, tik atributus – tuomet jis laikomas tuščiuoju elementu, analogas – HTML `<br>` elementas.

Atributai apibrėžia XML elementams leidžiamus priskirti atributus. Atributams gali būti suteikiami tik paprastieji tipai, kadangi atributai negali agreguoti XML elementų. Kaip ir elementai, atributai gali būti apribojami pagal jų būtinybę (elemento apraše, sintaksę ir pan. Atributams, kaip ir elementams gali būti priskirti įvairūs apribojimai, tokie kaip pasikartojimo (*angl.* occurrence).

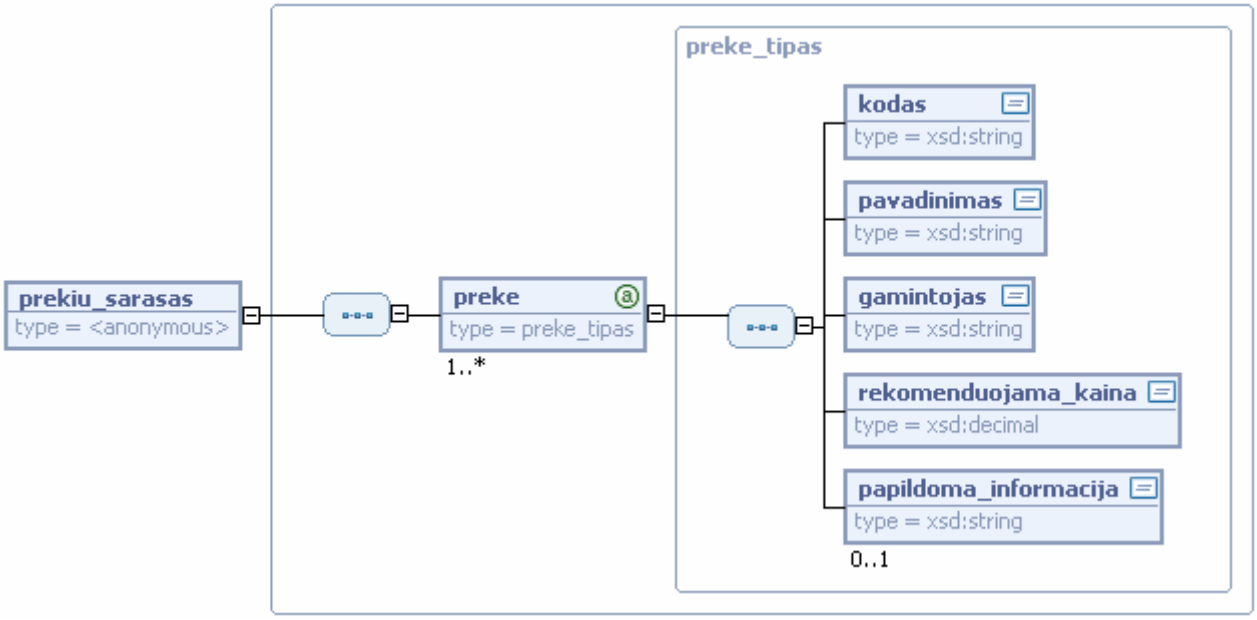
Vardų erdvės (*angl.* namespace) yra XML schemos elementas, apribojantis konkretaus XML schemos dalies matomumą tik tam tikroje vardų erdvėje. Tai leidžia moduliarizuoti XML schemų aprašus, turint omenyje, kad jos gali būti panaudojamos viena kitos aprašui sukurti.

Schemoje yra nustatyti dokumentacijos elementai `<annotation>` ir `<documentation>`, kurie leidžia paprasta kalba dokumentuoti schemos struktūrą, atkrenta būtinybė saugoti atskirus konkrečios XML schemos aprašo dokumentus.

Pateiksime trumpą XML schemos pavyzdį, apibrėždami anksčiau, 1.4.1 skyriuje pateikto XML dokumento tipą, nurodydami schemos tekstinį ir grafinį atitikmenis.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="prekiu_sarasas">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="preke" type="preke_tipas"
                              minOccurs="1" maxOccurs="unbounded"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>

      <xsd:complexType name="preke_tipas">
            <xsd:sequence>
                  <xsd:element name="kodas" type="xsd:string"/>
                  <xsd:element name="pavadinimas" type="xsd:string"/>
                  <xsd:element name="gamintojas" type="xsd:string"/>
                  <xsd:element name="rekomenduojama_kaina" type="xsd:decimal"/>
                  <xsd:element name="papildoma_informacija" type="xsd:string"
                        minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="nr" type="xsd:positiveInteger" use="required"/>
            <xsd:attribute name="virtuali" type="xsd:boolean" use="required"/>
      </xsd:complexType>
</xsd:schema>
```



1.9 pav. XML schemos pavyzdžio grafinis atitikmuo

Formaliai XML schema apibrėžiama kaip $D = (E, A, P, R, R)$, kur [7]:

- $E \subseteq El$, kur E – baigtinė elementų aibė;
- $A \subseteq Att$, kur Att – baigtinė atributų aibė;
- P – E susiejimas su elemento tipo aprašais;
- R – E susiejimas su A, o $r \in E$ –šakninio elemento tipas;
- $paths(D)$ – visų XML schemos D kelių aibė.

2. OP MODELIO POSCHEMĖS TRANSFORMAVIMAS Į XML SCHEMĄ

Tyrimo metu numatyta sudaryti ir realizuoti algoritmą, skirtą XML schemos aprašo generavimui iš OP modelio. Kadangi abu modeliai nėra trivialiai suderinami tarpusavyje, jų transformavimas gali būti sudėtingas, o transformavimo taisyklės tampa neortogonaliomis.

Darbe yra siekiama išplėsti XML schemų generavimo, panaudojant OP modelį, taisykles, nurodytas darbe [14][2] bei jas realizuoti redaktoriaus PĮ. Išplėtimas yra reikalingas, kadangi pradinė šių taisyklių versija atsiremia į OP modelio sudarymo taisykles, sudarant kitas nurodančias, modelio transformavimą į XML schemą. Praktiniu požiūriu to nepakanka, kadangi sudarant schemas informacinės sistemos analizės metu yra reikalingas selektyvus schemos išrinkimas, transformuojant tik reikalingą OP schemos dalį. Be to šiose taisyklėse nėra padengti visi OP modelio elementai, kaip, pavyzdžiui, paveldėjimo (apibendrinimo) elementas.

Šiame skyriuje bus apibrėžtos papildomos transformavimo taisyklės, pateiktas neformalus transformavimo algoritmas.

2.1. Modelių suderinamumo analizė

Nagrinėjant transformavimo problemą, yra būtina atlikti bent paviršutinišką dviejų operuojamų duomenų modelių palyginimą. Reikia atkreipti dėmesį, kad skiriasi šių dviejų modelių formalaus apibrėžimo tikslumas ir apimtis, todėl skyriuje nebus stengiamasi atlikti detalų galimybių palyginimą, išskyrus esmines.

XML schema yra orientuota į XML dokumentų aprašą, t.y. dokumentų, kurie sudaryti griežtai hierarchiško medžio principu. Tuo metu OP modelio notacija stengiasi aprėpti reliacinių ir objektinių modelių galimybes, nors gali apibrėžti ir hierarchines struktūras.

XML dokumento medžio mazgų išraiškos galimybės griežtai apibrėžtos ir vienareikšmiškos, tam skirta W3C rekomendacija [3] ir yra realizuota daug programinių priemonių dokumentų apdorojimui bei tikrinimui. Apibrėžiant projekcijų ir pačio modelio objektų santykius OP modelio grafo mazgai gali dalyvauti įvairiuose ryšiuose su kitais, patys modelio mazgai gali būti kelių mazgų savybėmis.

OP modelis, priešingai nei XML schema, neturi griežtai apibrėžto fizinio dokumento formato. Tai natūralu, kadangi XML schemai realizuoti buvo įdėta kur kas daugiau pastangų, o šio formato galimybes stengtasi priderinti prie pramonėje naudojamų duomenų ir operacijų. Tuo tarpu OP yra abstraktesnis, teorinis modelis, kurio apimtis pernelyg didelė detaliai modelio reprezentacijai apibūdinti.

Kadangi nėra OP modelio tekstinės reprezentacijos, jo schemos dokumentai visuomet apibrėžiami grafine notacija. Tuo pačiu nėra apibrėžta standartinė duomenų, modeliuojamu OP, sintaksė, kokią turi XML. XML schemos dėl medžio tipo organizacinės struktūros ir ortogonalių

numatytų medžio mazgų savybių gali taip pat pavaizduoti grafiškai, nors ir nėra priimta vieningos notacijos XML dokumentų ar XML schemų grafinei reprezentacijai.

2.1 lentelė. OP bei XML semantinių galimybių palyginimas

Darinys	**OP modelis**	**XML schema**
Esybė	+	–
Atributas	+	+
Daugiareikšmis atributas	+	+
Sudėtinis atributas	+	+
Metodas	+	–
isa ryšys tarp objektų	+	+ (subtipas gali paveldėti tik vieną tėvinį tipą)
Identifikatoriai	+	+ (apribota išraiška ir panaudojimas)
Paprastoji bei sudėtinė atributo reikšmės kaip objektai	+	+ (realizuojama pakeitimo sąrašo *substitution group*, arba pasirinkimo *choice* elementais, įvedant tipams reikšmės apribojimą(-us))

Aukščiau pateiktoje lentelėje galima pastebėti, kad XML schema leidžia interpretuoti paprastąją ir sudėtinę reikšmes, kaip objekto tipą (esybę), nors ir netiesioginiu būdu.

2.2. OP modelio padengimas medžio tipo struktūra

Kadangi OP modelis turi grafu paremtą struktūrą, o XML – medžio, yra reikalinga numatyti metodą, leidžiantį padengti OP modelio darinius medžio tipo struktūra.

2.2.1. Grafo sąvokų apibrėžimai

Toliau pateikiami formalūs grafų bei susijusių sąvokų apibrėžimai, kurie bus naudojami darbe. Šaltinis: [16].

Kryptinis grafas G susideda iš baigtinės aibės V ir nerefleksyvaus dvejetainio sąryšio su aibe V. Aibė V dar vadinama mazgų aibe. Dvejetainis ryšis gali būti išreiškiamas kaip aibė E, susidedanti iš fiksuotos padėties porų (v, w), arba kaip funkcija vaizduojanti V į savo aukštesnio laipsnio aibę $\mathcal{P}(V)$:

$$\text{Adj: } V \rightarrow \mathcal{P}(V).$$

Adj(v) yra vadinama v mazgo gretimumo aibe, t.y. rinkiniu kraštinių, kurios jungia mazgą v su kitu mazgu. $\text{Adj}(v) \in E$.

Taip pat: $(v, w) \in E$, jei ir tik jei $w \in \text{Adj}(v)$.

Sakoma, kad mazgai v ir w yra gretimi, jei jie turi bendrą kraštinę $(v, w) \in E$.

$N(v) = \{v\} + \text{Adj}(v)$ yra šalutinių mazgui v mazgų aibė, įskaitant v, ir dar vadinama v mazgo aplinka.

Be formalaus apibrėžimo, dažnai yra patogu grafus vaizduoti grafiškai, kas įmanoma daugybe būdų. Jeigu kraštinė (x, y) priklauso grafui tuomet brėžiama linija su rodyklė į y viršūnę; jei yra kraštinė (x, y), tiek (y, x), tuomet linija brėžiama be rodyklių į kurią nors viršūnę. Dėl vaizdavimo

galimybių įvairovės keli grafiniai to pačio grafo atvaizdai gali atrodyti nepanašūs. Panašumui apibrėžti yra *izomorfizmo* sąvoka:

Du grafai G = (V, E) ir G' = (V', E') vadinami izomorfiniais jei egzistuoja vienareikšmė funkcinė priklausomybė f: V $\rightarrow$ V', patenkinanti sąlygą, kad visiems mazgams x, y $\in$ V:

(x, y) $\in$ E $\Leftrightarrow$ (f(x), f(y)) $\in$ E'.

Dvi kraštinės laikomos gretutinėmis jei jos turi vieną bendrą mazgą.

Atvirkštiniu grafui G = (V, E) grafu vadinamas grafas G^{-1} = (V, E^{-1}), kurio kraštinės turi priešingą kryptį nei G:

E^{-1} = { (x, y) | (y, x) $\in$ E }, arba, (x, y) $\in E^{-1} \Leftrightarrow$ (y, x) $\in$ E.

Grafo G subgrafu gali būti bet kuris grafas H = (V', E'), jei tenkinamos sąlygos:

V' $\subseteq$ V, ir E' $\subseteq$ E. Subgrafas H *padengia* medį G.

*Kelias g*rafe G = (V, E) yra mazgų [v_0, v_1, ... v_n] seka, kur $v_{1-n} \in$ V.

Ciklu grafe G yra vadinamas kelias sudarytas iš mazgų [v_0, v_1, ... v_0], $v_{1-n} \in$ V.

Susietame grafe visuomet egzistuoja kelias tarp dviejų duoto grafo mazgų.

Medis yra susietas grafas, kuriame egzistuoja tik vienas kelias tarp dviejų duotų grafo mazgų.

Hipergrafas yra grafas kuriame kraštinė gali jungti daugiau nei du mazgus.

2.2.2. Grafo padengimas

Padengiant grafą yra siekiama gauti dengiantį medį, kuriam tolesniame transformavimo etape bus sudaromas izomorfiškas XML schemos medis. Šis padengiantis medis yra rūšiuotas, t.y. bet kurio medžio mazgo žemesnieji (lapo) mazgai turi nekintančią tvarką tėviniame mazge.

Šis padengiantis medis apibrėžiamas šiomis taisyklėmis:

- Medžio šakninis mazgas gali būti bet kuris OP objektas *O* iš dalykinės srities *D*, bet kuris objekto *O* atributas *O.A* ar vidinis ryšis *O.R*;
- OP grafo mazgas (objektas *O*, atributas *A*, vidinis ryšys *R*, arba išorinis *C*) gali būti jungiamas prie medžio, jeigu jis yra gretutinis kuriam nors iš grafo mazgų, jau įjungtų į dengiantį medį;
- Prijungiant grafo mazgą prie dengiančio medžio yra natūraliai sudaroma medžio grafo kraštinė, nukreipta nuo prijungiančio mazgo iki prijungiamojo;
- Grafo mazgas negali būti prijungiamas prie medžio kelis kartus toje pačioje ar skirtingose vietose;

- Tiek tie mazgai gali būti prijungti prie dengiančio medžio, kurių prijungimo metu nesusidaro ciklai padengiančiame medyje.

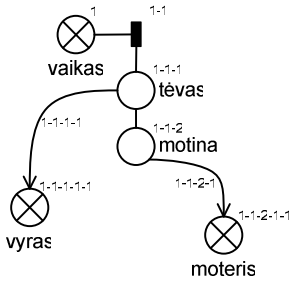
2.2.3. Padengiančio mazgo žymėjimas

Siekiant pavaizduoti modelį padengiantį medį, tame pačiame vaizde kaip ir OP modelis, naudojamas specialus padengiančio mazgo žymėjimas. Šalia mazgo vaizdinio elemento yra nurodoma n ilgio seka [N_1-N_2-..N_{n-1}-N_n]; čia [N_n] yra mazgo vieta tarp kitų tėvinio mazgo „lapų“, o [N_1-N_2-..N_{n-1}] – tėvinio mazgo žymėjimas. Šakninis mazgas turi žymėjimą [1]. Tokio žymėjimo pavyzdžiai: „1-3-1-2“, „1“, „1-5“.

Jeigu tikslinga tam tikruose kontekstuose šis žymėjimas gali būti sutrumpinamas išimant centrinius skaičius: „1-2-3-4-5“ ⇔ „1-..-4-5“. Pasikartojantys elementai žymėjime gali būti įterpiami į skliaustus, nurodant jų pasikartojimo skaičių: „1-1-1-1-1-1-1“ ⇔ „1(-1){5}”.

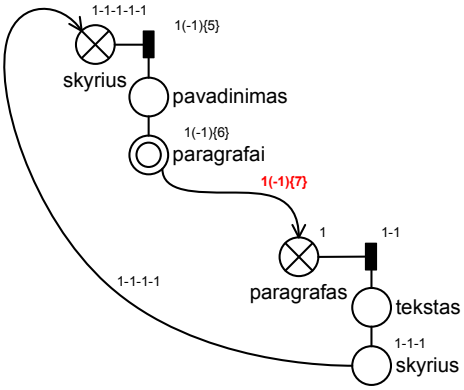
2.2.4. Padengimo pavyzdžiai

Medžio padengimo taisyklių iliustravimui naudojami žemiau pavyzdžiai. Šiuose modeliuose OP mazgo dalyvavimas padengiančiame medyje žymimas skaitine notacija, nurodyta ankstesniame 2.2.3 skyriuje.



2.1 pav. Viso OP modelio grafo padengimo medžiu pavyzdys

Žemiau pateikiamame pavyzdyje parodoma, kaip jungties įkėlimas į grafą dengiantį medį sukeltų ciklą. Jeigu prie dengiančio medžio būtų prijungiamas išorinis ryšys, pažymėtas „1(-1){7}“, susidarytų ciklas.



2.2 pav. Draudžiamo padengimo medžiu pavyzdys

2.3. Transformavimo taisyklės

Ankstesniame skyriuje aprašyto OP modelį padengiančio medžio mazgus galima transformuoti į XML elementus, transformuojant ir pačius medžio mazgus siejančias kraštines, t.y. keičiant jų interpretavimą. Pagal šias taisykles yra sukuriamas neformalus algoritmas OP poschemės transformacijai į XML schemą.

Šios taisyklės neaprėpia visų įmanomų OP grafo darinių padengimo bei nepanaudoja visų XML schemos mechanizmų, tokių kaip XPath [15], kurie gali būti panaudojami identifikatorių reikšmių apribojimui XML dokumente.

2.3.1. Originalios taisyklės

Šiame tyrime, sudarant transformavimo algoritmą yra remiamasi anksčiau minėtame tyrime [14] nurodytomis OP modelio transformavimo į XML schemą taisyklėmis. Jos nepadengia visų OP modelio darinių ir yra išplečiamos sekančiame, 2.3.2 skyriuje.

Žemiau pateikiamos transformavimo taisyklės, kurios remiasi pagrindiniu OP darinių apibrėžimu, susidedančių iš 5 taisyklių.

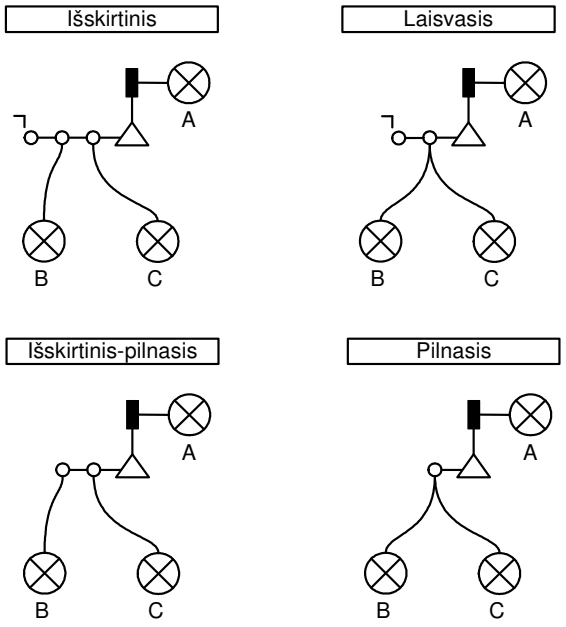
- OP objektas *O* transformuojamas į XML kompleksinio tipo elementą: *OP(O)* → XML(kompleksinio tipo elementas);
    - Objekto *O* atributas *A*, priklausomai nuo to ar turi identifikuojantį vaidmenį objekte O, yra transformuojami:
        - Į paprasto tipo elementą: *OP(O.A)* → XML(paprasto tipo elementas);
        - Į objekto *O* elemento XML atributą: *OP(O.A)* → XML(*O* kompleksinio elemento atributas);
- Objekto *O* vidinis sąryšis (dar vadinamas sudėtiniu atributu) yra transformuojamas skirtingai, priklausomai nuo to ar turi identifikuojantį vaidmenį:
    - Jei vidinis sąryšis *R* neturi identifikacinio vaidmens:
        - Ryšys *R* transformuojamas į XML kompleksinį elementą: *OP(O.R)* → XML (kompleksinis elementas);
        - Ryšio atributai *R.A* į vidinio ryšio elemento subelementus: *OP(O.R.A)* → XML (*R* kompleksinio elemento subelementas);
    - Jei vidinis R turi identifikacinį vaidmenį:
        - Ryšys *R* transformuojamas į atributų grupę: *OP(O.R)* → XML (*O* kompleksinio elemento atributų grupė);
        - Ryšio atributai *R.A* į atributų grupės atributus: *OP(O.R.A)* → XML (*O* kompleksinio elemento atributų grupės atributas);

- Funkcinė priklausomybė OP apibrėžime gali būti interpretuojama kaip atskiras OP objektas, kuriame vaizduojamos egzempliorių aibės objektų projekcija sudaro identifikacinį, o atvaizduojamosios aibės objektai – neidentifikuojantį vidinį ryšius. Taip specifikuotas OP objektas gali būti transformuojamas į XML schema pagal aukščiau aprašytas taisyklės
- Objekto metodai neturi atitikmens XML schemoje, kadangi joje galima aprašyti tik duomenų struktūrą, o ne elgseną, todėl jie gali būti į ją perkelti tik komentarų ar aprašo pavidalu
- OP modelyje vienas objektas gali būti kito objekto savybė, todėl reikalingas XML schemos pagrindu sudarytas agregacijos ir nuorodų mechanizmas. XML schema yra hierarchiškas dokumentas, dėl to natūraliausia yra objektus, kaip kitų objektų savybes, agreguoti tėvinių objektų XML atitikmenų elementuose

2.3.2. Išplėstinės taisyklės

Kaip anksčiau minėta, 2.3.1 skyriuje nurodytos taisyklės nepadengia visų OP modelio darinių. Viena pagrindinių savybių yra paveldimumo (apibendrinimo ryšys). Taip pat nėra nurodytas OP modelio objektų ir vidinių ryšių savybių poaibių priklausomybės ryšio transformavimas. Šiame skyriuje siūlomos išplėstinės taisyklės šiems dariniams generuoti į XML schemos elementų ir tipų darinius.

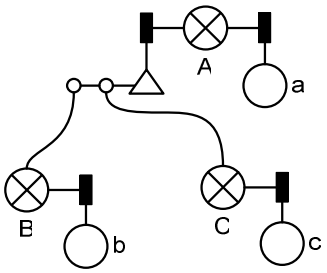
OP modelio apibendrinimo elementai, žymimi „Δ“ simboliu, nurodo išorinio apibendrinimo (arba *isa*) ryšio grupę, kurios elementai yra kiti objektai paveldintys pradinio objekto savybes. Paveldinčių objektų aibės tarpusavyje gali sudaryti įvairias kombinacijas [2]. Paveiksle pateikiami *isa* ryšio grupės galimos kombinacijos ir jų vaizdavimas OP modelyje:



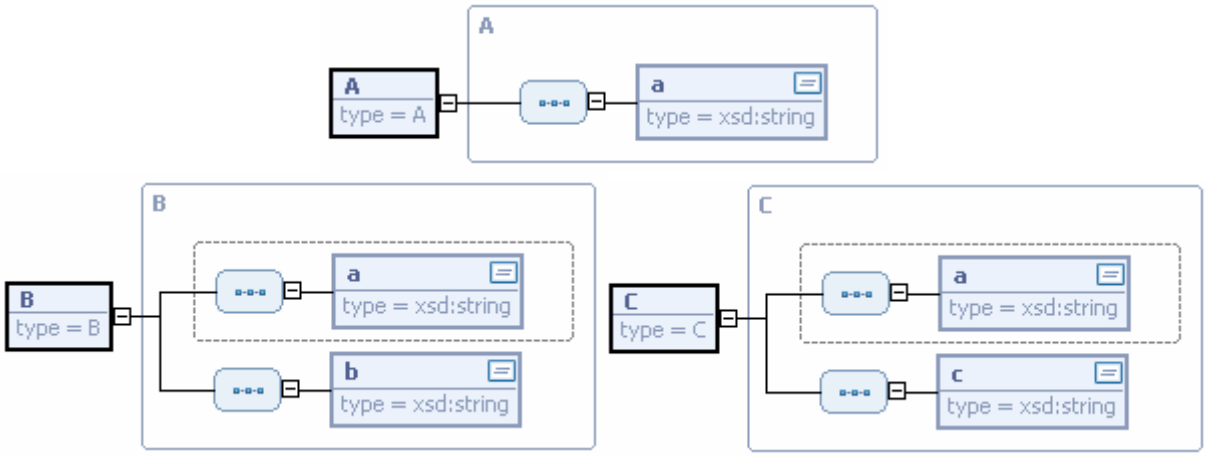
2.3 pav. *isa* ryšio kombinacijos ir jų OP modelio vaizdavimas

XML schemos turi išplečiamą ir sudėtingą tipų sistemą [12], kurioje galima tiesiogiai atvaizduoti šias paveldimumo kombinacijas. Formaliai:

- OP modelio objektas *O* esantis *isa* ryšyje su objektu *P*, yra transformuojamas jo elemento tipą kildinant iš objekto *O* elemento tipo XML schemos tipo sukūrimo aprašu `<extension>`. Pateikiamas transformavimo iš OP modelio pavyzdys, kuriame OP objektai *B* ir *C* dalyvauja išskirtinio tipo *isa* ryšyje su *A* objektu. Šio pavyzdžio tekstinis XML schemos dokumentas pateiktas 1 priede.

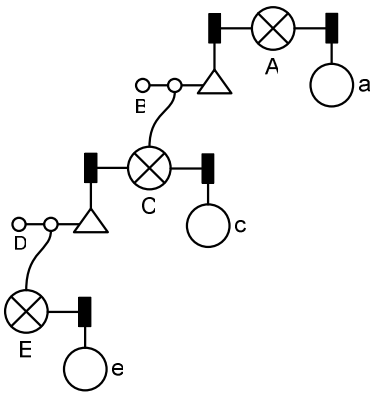


2.4 pav. Išskirtinio-pilnojo *isa* ryšio pavyzdys

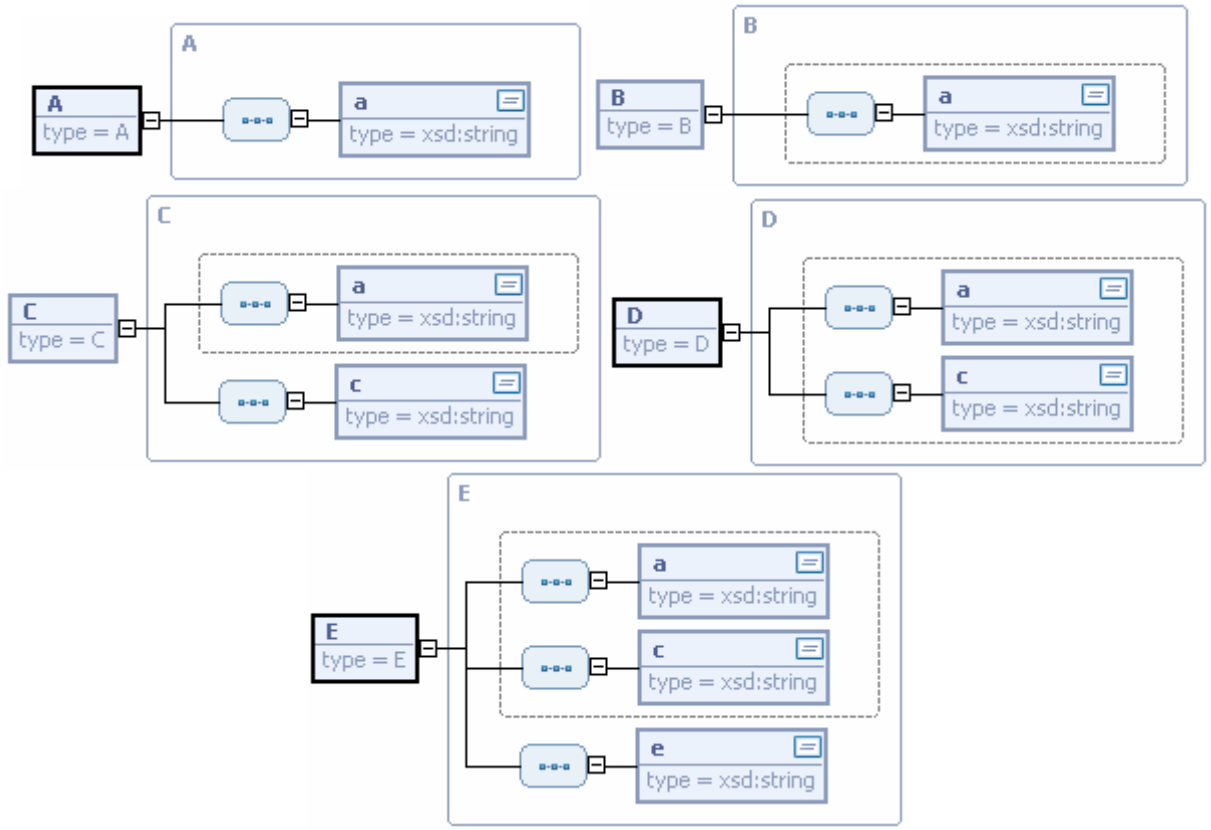


2.5 pav. Išskirtinio *isa* ryšio tipu susietų esybių XML schemos atitikmuo

- Taip pat galimas tranzityvus paveldėjimas. Žemiau pateikiamame pavyzdyje objektai *B* ir *C* yra *isa* ryšyje su objektu *A*; *D* ir *E* – *isa* ryšyje su *C*. Šio pavyzdžio tekstinis XML schemos dokumentas pateiktas 1 priede.



2.6 pav. Tranzityviu *isa* ryšiu susietų esybių OP grafo pavyzdys

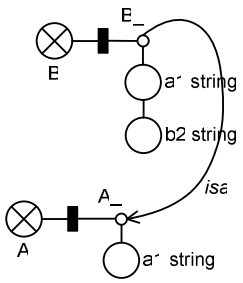


2.7 pav. Tranzityviu *isa* ryšiu susietų esybių XML schemos atitikmuo

- Šaltinyje [17] nurodyta, jog XML schemoje galima nurodyti tik vieną tėvinį tipą, paveldint `<extend>` būdu, tačiau nėra apribojimu paveldėjimo medžio gyliui
- Tačiau nėra galimybės XML tipų sistemoje trivialiai perteikti *isa* paveldinčių tipų egzempliorių aibės persidengimo apribojimus – jie yra ignoruojami

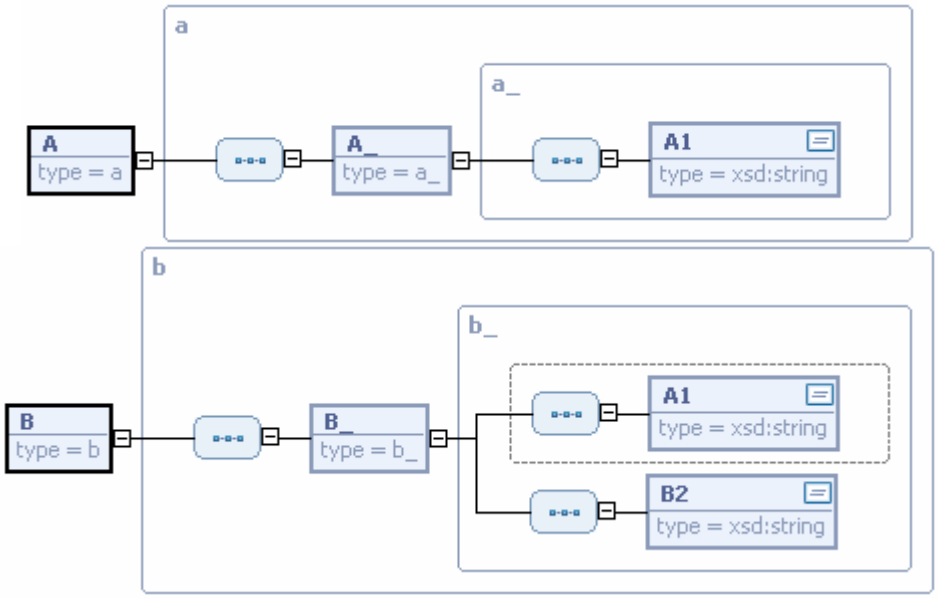
Objekto vidinių ryšių išoriniai poaibio ryšiai

OP modelyje yra galimybė nurodyti objektų bei objektų vidinių ryšių savybių tarpusavio priklausomybę. Ši priklausomybė OP schemoje vaizduojama *isa* ryšiu. Toks ryšys interpretuojamas kaip paveldimumas, kadangi vienas iš ryšio dalyvių perima kito dalyvio savybių poaibį, o vidinio ryšio atveju – jo atributus. Jį galima realizuoti XML schemos `extension` elementu. Iliustruojant šią taisyklę pateikiamas pavyzdys, kai dviejų OP objektų vidiniai ryšiai specifikuojami *isa* sąryšiu. Objektai *A* ir *B* turi vidinius ryšius *A_* ir *A_* atitinkamai. Ryšiu nurodoma, kad *_A* paveldi *_A* savybes (atributus). Formaliai, *savybės(A_) ⊆ savybės(A_)*.



2.8 pav. OP objektų vidinių ryšių priklausomybė

Šiam pavyzdžiui gaunama XML schema, pasitelkus aukščiau nurodytas taisykles. Šio pavyzdžio tekstinis XML schemos dokumentas pateiktas 1 priede.



2.9 pav. OP objektų vidinių ryšių priklausomybės XML schemos atitikmuo

2.3.3. Neformalus transformavimo algoritmas

Naudojant anksčiau sudarytas transformavimo taisykles, neformaliai apibrėžiamas XML schemos sudarymo iš OP modelio schemos algoritmas. Algoritmo aprašymas susideda iš išankstinių sąlygų, nurodytų operuojamoms schemoms, algoritmo sekos ir sąlygų transformavimo rezultatui. Algoritmas remiasi paieškos į plotį metodu, ją pradedant nuo to medžio šaknies. Detaliau:

Išankstinės sąlygos.

Transformuojamas tik teisingai sudarytas OP modelis. Turi būti parinktas duoto OP modelio grafo padengimas medžiu pagal taisykles aprašytas skyriuje 2.2.

Algoritmo seka.

1) Analizė pradedama nuo padengiančio medžio šaknies.
2) Ieškomas nagrinėjamo OP modelio mazgo tėvinis objektas, jeigu pats mazgas nėra objektas.
3) Rastas objektas transformuojamas į XML sudėtinio tipo elementą, pagal jam tinkamas transformavimo taisykles. Į objekto XML tipo aprašą įkeliami tik padengiančiuoju medžiu pažymėti OP grafo mazgai.
4) Nagrinėjami objekto *isa* apibendrinimo elementai ir paeiliui nagrinėjami šio objekto savybes per šiuos elementus paveldintys objektai, rekursyviai vykdoma 3) algoritmo šaka.
5) Nagrinėjami visi išoriniai OP objekto *isa* ryšių elementai, kurie seka iš objekto vaikų mazgų į kitus objektus ar tų objektų vaikų mazgus. Kiekvienam iš kitos ryšio dalies mazgų vykdoma 2) algoritmo šaka, t.y. sukuriamas to mazgo tėvinio OP objekto XML schemos atitikmuo.

Rezultatai.

Atlikus algoritmą yra gaunamas hierarchiški XML schemos tipų medžiai, kurie sudarantys vieną schemą. Tokia schema yra tinkamai suformuota, t.y. gali būti patikrinama pagal XML schemos taisykles. Joje nurodomas tik vienas šakninis elementas, naudojantis šakninį schemos tipą.

3. GRAFINIO OP MODELIO REDAKTORIAUS GALIMYBIŲ TYRIMAS

Įprasti XML schemų redaktoriai naudoja tekstinį vaizdavimą, klasių (kaip EER), medžio tipo grafines diagramas. Šiame darbe aprašomas duomenų modelio redaktorius, naudojantis OP grafinę notaciją ir galintis generuoti XML schemas iš vartotojo sudarytų OP modelių.

Sukurtas schemų projektavimo įrankis privalo būti patogus vartoti IS projektavimo aplinkoje. Pasirinktas Objektų savybių modelis yra universalus, tinkantis lanksčiai specifikuoti sudėtingus ryšius bei struktūras. Dėl modelio įvairiapusiškumo ir apimties, realizacijai redaktoriuje daugiasia atrenkami tie elementai, kurie gali būti transformuojami į XML schemos darinius.

Galima išskirti kelias duomenų modelių (schemų) redaktorių rūšis:

1. Specializuotų duomenų modelių (pvz. EER), tinkamų tam tikroje panaudojimo sferoje, redaktoriai. Jie dažniausiai visiškai padengia modelio notacijos galimybes, griežtai užtikrina modelio semantiką, tačiau turi ribotą darinių įvairovę, kadangi apsiriboja tam tikra panaudojimo sritimi ir požiūriu (reliacinis, objektinis ar hierarchinis);
2. Universalių (įvairiapusiškų) duomenų modelių redaktoriai. Jie yra sudėtingesni negu specializuotų modelių redaktoriai, realizacijos apimtis kur kas didesnė, kadangi notacija turtingesnė ir įvairiapusiška. Paprastai tokių duomenų modelių schemos nėra naudojamos realiose sistemose, prieš panaudojimą originalūs modeliai yra transformuojami į reikalingą duomenų schemą (pvz. reliacinės DB).

Šiame darbe yra kuriamas universalaus OP modelio grafinis redaktorius (t.y. antrojo tipo), tačiau dėl tyrimo apimties yra apsiribojama tam tikrais jo elementais, konkrečiau – orientuojamasi į OP modelio elementus, kurie yra nesudėtingai realizuojami XML schemos dariniais.

Šis skyrius yra skirtas apibrėžti redaktoriaus programinės įrangos apimties, funkcionalumo ir kitus reikalavimus.

3.1. Reikalavimai operuojamiems duomenims

Redaktoriaus realizacijoje privalo būti galimybė sukurti ir išsaugoti redaguojamus OP modelio schemų failus, taip pat išsaugoti sugeneruotų schemų failus. OP modelio failų formatas neapibrėžtas, tuo tarpu yra būtina, kad sugeneruotos XML schemos atitiktų W3C rekomendacijos standartą [6].

OP modelio informacija, kuri negali būti transformuojama į XML schemų darinius, gali būti perteikiama schemų dokumentuose kaip komentarai, tačiau tai neprivaloma.

3.2. Nefunkciniai reikalavimai ir apribojimai

3.2.1. Reikalavimai standartams

XML schemos generavimo algoritmas turi generuoti XML schemas, atitinkančias W3C konsorciumo XML schema formatą versijai 1.0. XML schemos dokumento schema yra pasiekiama Interneto adresu http://www.w3.org/2001/XMLSchema.xsd; XML dokumente naudojamos schema pasiekiama Interneto adresu http://www.w3.org/2001/XMLSchema-instance. Tokiu būdu užtikrinamas suderinamumas su įrankiais ir programomis, kurios palaiko šį standartą.

3.2.2. Reikalavimai kokybei

Programos kokybės reikalavimai:

3.1 lentelė. Programos kokybės reikalavimai

Reikalavimas	Detaliau
Programos stabilumas	Redaktoriaus programa privalo būti stabili, neprarasti informacijos išsaugant ir užkraunant failus.
Išsaugomų failų turinio kokybė	Sugeneruotas schemos failas negali turėti nuorodų į kitus, išorinius failus (išskyrus XML NameSpace nuorodas).
Užkraunamų failų apdorojimas	Programa turi pranešti kuomet nurodytas failas yra neperskaitomas, taip pat neapdoroti failų, kurių duomenys viršytų programai prieinamus resursus, taip išvengiant neteisingo programos veikimo.

3.2.3. Reikalavimai sąsajai

Numatyti šie reikalavimai vartotojo sąsajai:

3.2 lentelė. Reikalavimai vartotojo sąsajai

Reikalavimas	Detaliau
Numatytų funkcijų realizavimas	Grafinėje sąsajoje turi būti realizuotos visos funkcijos, kurių pagalba vartotojas galėtų sudaryti norimo tipo schemą, jei tai įmanoma OP modelio pagalba.
Sąsajos prieinamumas	Grafinė sąsaja gerai integruota, realizuotos funkcijos lengvai pasiekiamos. Tos funkcijos, kurios dažnai naudojamos, turi būti pasiekiamos ir per pagrindinį programos meniu ir per kontekstinius meniu. Visi meniu punktai, žymės ir kiti užrašai turi būti suprantami vartotojui ir naudojami tinkami terminai.
Atlaidumas vartotojo įvesties klaidoms	Turi būti numatyta programos reakciją į nekorektišką vartotojo duomenų įvestį. Vartotojui neturi būti leidžiama sudaryti semantiškai ar logiškai neteisingų dokumentų. Jei to neįmanoma, apie tai turi būti pranešama vartotojui prieš veiksmą, kurio pasekmių negalima vėliau atšaukti.

3.2.4. Kiti reikalavimai

Reikalavimai programai, kurie nėra klasifikuoti:

3.3 lentelė. Neklasifikuoti reikalavimai

Reikalavimas	Detaliau
Sistemos išplėtimas	Turi būti galimybė išplėsti ir papildyti sistema naujomis funkcijomis.
Operacijų vykdymo laikas	Reikalinga, kad visos programos funkcijos nereikalautu neprotingai ilgo laiko įvykdymui, o jeigu neišvengiama – praneštų apie tai vartotojui.
Suderinamumas su OS bei technine įranga.	Programa turi vienodai gerai veikti įvairios komplektacijos kompiuteriuose ir operacinėse sistemose, taip pat nepriklausytų nuo papildomų, nebūtinų modulių. Ji neturi konfliktuoti su kitomis sistemoje esančiomis programomis.
Įdiegimas	Programą turi būti paprasta įdiegti bei išdiegti.

3.3. Rizikos faktorių analizė

Kadangi projektuojama programa nėra triviali, egzistuoja rizika, kad programa nebus įgyvendinta. Faktorius, įtakojančius programos riziką, juo identifikavus galima bent dalinai sumažinti.

3.4 lentelė. Projekto rizikos faktoriai

Rizikos faktorius	Rizikos faktoriaus eliminavimo būdas
Pasikeitę reikalavimai	Prieš įvedant pakeitimus privalo būti nustatyta ar nauji reikalavimai pagrįsti.
Neprieinamos projektavimo ar kūrimo priemonės, arba nustatytos reikšmingos problemos jas naudojant.	Nustatyti ar įmanoma naudojamas priemones/bibliotekas pakeisti analogiškomis.
Galutinė grafinė sąsaja pernelyg sudėtinga arba primityvi	Sudaryti grafinės sąsajos prototipą, įvertinant įgyvendinimo galimybes.
Programa lėtai veikia ar reaguoja į vartotojo veiksmus.	Nustatyti kritines programos dalis, funkcijas, kurias galima būtų optimizuoti ar net atsisakyti.
Realizuotos funkcijos realiai nepanaudojamos ar pernelyg sudėtingos.	Suskaldyti programos funkcijas į mažesnes, arba atsisakyti dalies.
Viršytas programos kūrimo laikas	Atsisakyti dalies numatytų funkcijų arba jas supaprastinti.

3.4. Rezultato kokybės kriterijai

Užbaigta programa turi atitikti jai iškeltus reikalavimus.

3.5 lentelė. Rezultato kokybės kriterijai

Reikalavimas	Specifikavimas
Numatytų funkcijų realizacija	Galimi neesminiai neatitikimai, ypač tų funkcijų, kurios liečia vartotojo sąsają, kadangi su ja susiję pakeitimai dažnai paaiškėja tik pradėjus testuoti realizuotą programą ir neturi didelės įtakos programos funkcionalumui.
Užbaigta vartotojo dokumentacija	Turi būti aprašytos redaktoriaus funkcijos, vartotojo sąsaja, kt.
Užbaigta sisteminė dokumentacija	Dokumentacijos turi pakakti norint numatyti su programos išplėtimu ar tobulinimu susijusių darbų mastą ir tokio išplėtimo reikalavimus.
Vartotojo sąsajos patogumas	Vartotojo sąsaja turi būti patogi ir nepatyrusiems vartotojams.

3.5. Reikalavimų specifikacija

Šiame skyriuje pateikiami panaudos atvejai, jų specifikacijos, vartotojo ir sistemos sekų, veiklos diagramos taip pat nefunkciniai reikalavimai. Nustatomos projektuojamos programinės įrangos galimybės, apribojimai. Vartotojui prieinamos programos ir procesų valdymo galimybės.

Sudarant reikalavimų specifikaciją programinei įrangai buvo peržiūrėti ir detalizuoti panaudos atvejai.

3.5.1. Projekto tikslas

Tyrimo eigoje kuriamo redaktoriaus projekto tikslas – suprojektuoti ir realizuoti taikomąją programinę įrangą, kuri leistų:

1. Grafiškai sudaryti OP modelio schemas;
2. Jas išsaugoti bei užkrauti iš/į išorinius failus;
3. Suteikti lanksčią vartotojo sąsają redagavimui;
4. Parinkti OP schemos elementus kaip medžio šakas;
5. Sugeneruoti ir išsaugoti XML schemą sudarytą nurodytą šaką, panaudojant parinktas OP modelio šakas (poschemes).

3.5.2. Projektavimo aplinka

Numatoma, kad projektavimas ir programavimas bus atliekamas IBM PC tipo kompiuteriais, naudojant JAVA J2SE 1.5 [18] programavimo kalbą, Eclipse platformą, MagicDraw UML [19] modeliavimo priemones.

Sąsaja su vartotoju bus sudaroma naudojantis Eclipse/GEF platformos vartotojo sąsajos priemonėmis.

3.5.3. Eksploatavimo aplinka

Programos veikimui bus reikalingas IBM PC tipo kompiuteris. Operacinė sistema, veikianti jame, gali būti Windows arba Linux, kadangi Java yra pernešama tarp platformų. Taip pat reikalingas įskiepis Java virtualiai mašinai, rekomenduojamas standartinis J2SE 5.0.

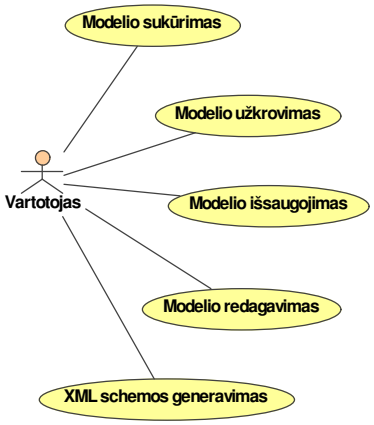
Preliminarūs reikalavimai kompiuterio galimybėms:

3.6 lentelė. Reikalavimai eksploatavimo aplinkai

Parametrai	**Minimalūs reikalavimai**	**Rekomenduojami reikalavimai**
Procesoriaus dažnis	> 300 Mhz	> 800 Mhz
Operatyvinės atminties dydis	> 64 MB	> 256 MB
Sistemos laisvos kieto disko laikmenos dydis	> 50 MB	> 50 MB

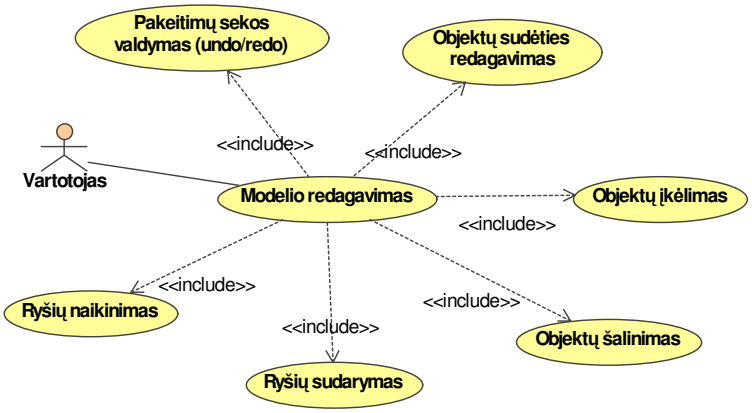
3.5.4. Panaudojimo atvejai

Redaktoriaus vartotojų sąsaja su redaktoriumi apibendrinama šiais panaudos programos atvejais. Taikomoji programa turi tik vieną vartotojo tipą.

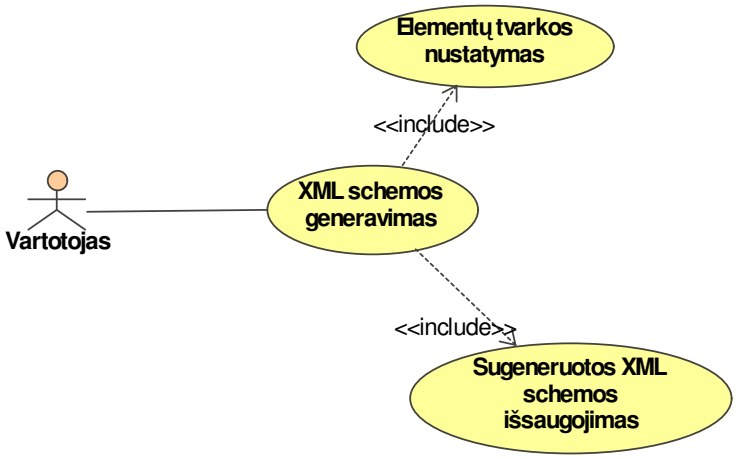


3.1 pav. Pagrindiniai redaktoriaus panaudos atvejai

Modelio redagavimo bei XML schemos generavimo panaudos atvejai yra detalizuojami toliau.



3.2 pav. Modelio redagavimo vidiniai panaudos atvejai



3.3 pav. XML schemos generavimo vidiniai panaudos atvejai

3.5.5. Panaudojimo atvejų specifikacijos

Šiame skyriuje pateikiamos panaudojimo atvejų specifikacijos.

3.7 lentelė. Modelio sukūrimo panaudos atvejo specifikacija

Prieš sąlyga	<Nėra>
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka modelio sukūrimo veiksmą (meniu, įrankių juostoje) arba programa yra paleidžiama.	1.1. Jei šiuo metu programoje jau yra redaguojamas modelis, užklausiama ar galima jį naikinti.
2. Vartotojas patvirtina, kad dabartinis modelis gali būti naikinamas.	2.1. Sistema sunaikina dabartinį modelį ir sukuria naują.
Po sąlyga	PĮ sunaikino senąjį modelį ir sudarė naują.

3.8 lentelė. Modelio užkrovimo panaudos atvejo specifikacija

Prieš sąlyga	<Nėra>
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka modelio užkrovimo veiksmą (meniu, įrankių juostoje).	1.1. Jei šiuo metu programoje jau yra redaguojamas modelis, užklausiama ar galima jį naikinti.
2. Vartotojas patvirtina, kad dabartinis modelis gali būti naikinamas.	2.1. Sistema naikina dabartinį modelį. 2.2. Išveda failo pasirinkimo dialogą vartotojui.
3. Vartotojas pasirenka norimą užkrauti failą failų sistemoje.	3.1. Sistema patikrina ar failas gali būti užkrautas 3.2. Sistema užkrauna bei apdoroja modelio failą sukurdama ir pavaizduodama grafinį modelį vartotojui.
Po sąlyga	Redaktoriuje senas modelis ištrinamas ir vaizduojamas vartotojo pasirinktas užkrautas modelis.

3.9 lentelė. Modelio išsaugojimo panaudos atvejo specifikacija

Prieš sąlyga	Redaktoriuje yra sudarytas netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka modelio išsaugojimo veiksmą (menių, įrankių juostoje).	1.1. Sistema vartotojui parodo failo pasirinkimo dialogą.
2. Vartotojas nurodo failą (failo pavadinimą) kuriame turėtų būti išsaugotas modelis.	2.1. Jei toks failas jau egzistuoja, PĮ paklausia dialogu ar vartotojas nori užrašyti ant seno failo turinio.
3. Vartotojas patvirtina išsaugojimą nurodytu pavadinimu.	3.1. Redaktorius išsaugo modelį (serializuoja) į nurodytą išorinį failą.
Po sąlyga	Redaktorius išsaugojęs dabartinį modelį į išorinį failą.

3.10 lentelė. Modelio redagavimo panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra užkrautas arba tuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Sistema pateikia vartotojui grafinius modelio redagavimo įrankius.	1.1. Vartotojas nurodytais įrankiais atlieka modelio pakeitimus.
Po sąlyga	Atlikti modelio pakeitimai.

3.11 lentelė. Objektų įkėlimo panaudos atvejo specifikacija

Prieš sąlyga	Programoje sudarytas modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka objekto įkėlimą nunešdamas objekto grafinį elementą iš įrankių juostos ant redaguojamosios srities.	1.1. Programa pateikia dialogą objekto savybėms suvesti (pavadinimui, ir pan.)
2. Vartotojas suveda objekto savybes į objekto savybių redagavimo langą.	2.1. Programa sukuria objektą su vartotojo nurodytomis savybėmis ir jį patalpina nurodytoje vietoje, kurioje buvo nuneštas elementas.
Po sąlyga	Į modelį įkeltas naujas objektas.

3.12 lentelė. Objektų šalinimo panaudos atvejo specifikacija

Prieš sąlyga	Pažymėtas kokio nors objekto grafinis elementas.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka objekto pašalinimo veiksmą iš įrankių juostos arba iškylančio meniu.	1.1. Sistema užklausia vartotojo dialogu ar tikrai norima naikinti objektą kartu su jo ryšiais su kitais.
2. Vartotojas patvirtina objekto naikinimo veiksmą.	2.1. Šalinami objekto ryšiai su kitais objektais. 2.2. Objektas naikinamas iš modelio.
Po sąlyga	Iš modelio pašalintas objektas ir jo ryšiai su kitais objektais.

3.13 lentelė. Objektų sudėties redagavimo panaudos atvejo specifikacija

Prieš sąlyga	Pažymėtas (pasirinktas) koks nors modelio objektas.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka objekto turinio redagavimo veiksmą iš meniu, arba iššokančio meniu.	1.1. Sistema parodo objekto sudėties (atributų, savybių) dialogą.
2. Iššokusiame dialoge vartotojas papildo/šalina objekto atributus, keičia objekto savybes (pavadinimą ir pan.)	
3. Vartotojas pasirenka pakeitimų atlikimo veiksmą (mygtuką).	3.1. Sistema atlieka objekto pakeitimus modelyje pagal vartotojo suvestus pakeitimus.
Po sąlyga	Pakeista modelio objekto sudėtis.

3.14 lentelė. Ryšių sudarymo panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka ryšio įkėlimo įrankį iš įrankių juostos.	
2. Vartotojas „nutempia“ ryšio liniją tarp dviejų skirtingų objektų sąsajos (*angl.* port) taškų.	2.1. Sistema patikrina ar ryšys gali būti sukurtas (ar toks jau egzistuoja, ar galimas). 2.2. Jei ryšys gali būti sudarytas, jis įkeliamas į modelį. 2.3. Jei ryšys negalėjo būti sukurtas, apie tai pranešama vartotojui.
Po sąlyga	Į modelį įkeltas ryšys tarp dviejų vartotojo pasirinktų objektų arba vartotojui nurodoma, kad ryšys tarp jo nurodytų objektų negalimas.

3.15 lentelė. Ryšių naikinimo panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra pasirinktas (pažymėtas) ryšys tarp objektų.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka ryšio naikinimo veiksmą iš įrankių juostos arba naudojantis iššokančiu meniu.	1.1. PĮ dialogu paklausia ar tikrai norima naikinti nurodytą ryšį tarp objektų.
2. Vartotojas pasirenka patvirtinti ryšio naikinimo veiksmą.	2.1. PĮ šalina vartotojo nurodytą ryšį tarp dviejų objektų iš programoje užkrauto modelio.
Po sąlyga	Ryšys panaikintas iš modelio.

3.16 lentelė. Pakeitimų sekos valdymo (*angl.* undo/redo) panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra užkrautas modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka ankstesnio jo atlikto veiksmo grąžinimą.	1.1. PĮ pažiūri ar yra grąžinamų veiksmų undo/redo steke. 1.2. Jei dar yra grąžinamų veiksmų, gražinamas sekantis steke.
ARBA 2. Vartotojas pasirenka sekančio atliko veiksmo atlikimą.	1.1. PĮ pažiūri ar yra iš naujo atliekamų veiksmų undo/redo steke. 1.2. Jei yra iš naujo atliekamų veiksmų, jie yra iš naujo atliekami.
Po sąlyga	Gražintas arba iš naujo atliktas vartotojo anksčiau atliktas modelio pakeitimo veiksmas.

3.17 lentelė. XML schemos redagavimo panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra užkrautas netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka XML schemos generavimo veiksmą iš įrankių juostos arba programos meniu.	1.1. PĮ patikrina ar yra pažymėtas OP modelio medis. 1.2. Jei nėra pažymėto medžio, pranešama apie tai vartotojui dialogu.
	2. PĮ generuoja schemą ir ją išsaugo vartotojo nurodytame faile.
Po sąlyga	PĮ sugeneravo ir išsaugojo vartotojo pasirinktą OP modelio medį kaip XML schemą į išorinį failą.

3.18 lentelė. Elementų tvarkos nustatymo panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra užkrautas netuščias modelis.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pažymi modelio objektus ir sąryšius įtraukdamas juos į OP modelio medį.	1.1. Sistema pažiūri ar vartotojo pasirinkti į medį įtraukiami ryšiai ir objektai nesudaro ciklų, taip pat kitus apribojimus. 1.2. Jei modelio elementas (objektas arba ryšys) gali būti įtrauktas į medį, jis pažymimas kaip įtrauktas grafinėje aplinkoje bei užregistruojamas medyje.
Po sąlyga	OP modelyje yra pažymėtas objektų medis , kuriuo galima generuoti XML schemą.

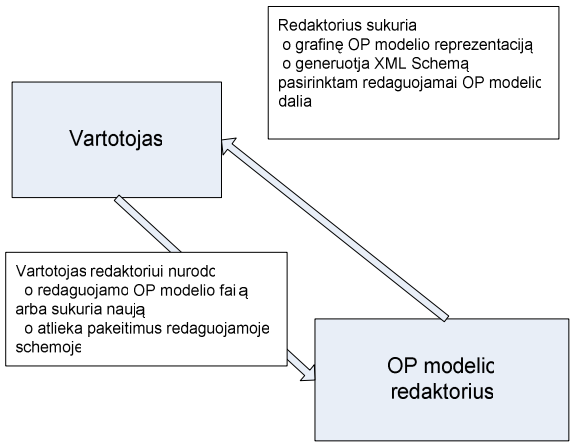
3.19 lentelė. Sugeneruotos XML schemos išsaugojimo panaudos atvejo specifikacija

Prieš sąlyga	Programoje yra užkrautas netuščias modelis bei vartotojas yra pažymėjęs jo medį schemos generavimui.
Įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka XML schemos generavimo veiksmą iš meniu arba įrankių juostos.	1.1. Sistema išveda vartotojui dialogą nurodyti XML schemos išsaugojimo failą.
2. Vartotojas pasirenka failą kuriuo turėtų būti išsaugojama generuota schemą.	2.1. Sistema generuoja XML schemą pagal programoje esantį OP modelį bei pasirinktą jo medį. 2.2. PĮ išsaugo sugeneruotą schemą į išorinį vartotojo nurodytą failą.
Po sąlyga	Pagal vartotojo nurodytą OP modelio medį sugeneruota schema išsaugota į išorinį failą.

3.6. Siekiamos sistemos apibrėžimas

Redaktoriaus, kaip taikomosios programa, yra izoliuota, nepriklausoma nuo išorinių sąlygų, sistema. Duomenys apsikeičiami su šiai duomenų failų tipais:

- Serializuotas OP modelis (naudojantis `java.io.Serializable` priemonėmis);
- XML schema (tik išsaugojimas sugeneravus).



3.4 pav. Redaktoriaus programos kontekstas

3.7. Vartotojų analizė

3.7.1. Vartotojų aibė, tipai ir savybės

Produkto vartotojais numatoma keletas vartotojų tipų. Orientuojamasi į informacinių sistemų sudarymą, kuriame daugiausia dalyvauja tų sistemų projektuotojai bei programuotojai.

- Sistemų projektuotojai naudosis redaktoriumi sudaryti OP schemas iš duomenų modelių, kurie nustatyti sistemos analizės fazėje, todėl jiems reikalinga, kad nebūtų didelio atotrūkio

tarp pradinio duomenų modelio bei schemos sąvokų bei sudarymo principo. Tai pasiekiama naudojant OP modelį bei realizavus jo transformaciją į XML schemą.

- Taikomųjų bei informacijos sistemų programuotojai siekia, kad grafinis schemos modelis atitiktų hierarchinį XML DOM ar kitą modelį naudojamą tų programų naudojamose bibliotekose. Šie vartotojai gali būti pripratę prie skirtingų XML schemų grafinų notacijų. OP modelis nėra tinkamas šiai reprezentacijai, tačiau sugeneruotai iš OP modelio XML schemai reprezentuoti yra daug jau egzistuojančių programinių priemonių.

3.7.2. Vartotojų tikslai ir problemos

Modeliuodami informacines sistemas ir jų sąveiką, numatomi programos vartotojai siektų sparčiai modeliuoti duomenų schemas OP modelio notacija, jas išsaugoti, modifikuoti.

3.8. Redaktoriaus realizavimo platformos bei architektūros parinkimas

Siekiant parinkti optimaliausią programos architektūrą, buvo nagrinėjami keli sekančiuose skyriuose nurodyti variantai. Pagrindiniai kriterijai parinkimui yra šie:

- Naudojamos programavimo kalbos patogumas;
- Platformos bibliotekų apimtis, XML schemų apdorojimo priemonių galimybės, prieinamumas bei dokumentacijos kokybė;
- Numanoma realizavimo sparta, platformos taikomųjų programų kūrimo priemonių įvairovė ir kokybė;
- Numatomos programos architektūros paprastumas, redaktoriaus funkcijų realizavimo būdų įvairovė, išplečiamumas.

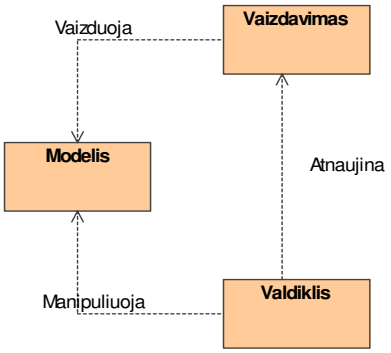
Prieš projektavimo etapą reikalinga atrinkti PĮ realizavimo platformą bei jai pritaikytas XML DOM arba, jei galima rasti, XML schemų programinio manipuliavimo bibliotekas.

3.8.1. MVC architektūra, Windows .NET platforma

Naudojant šią architektūrą bei platforma, GUI priemonės (meniu, dialogai, kt.) būtų naudojamos iš .NET bibliotekų rinkinio, o esminės OP modelio redagavimo priemones MVC architektūroje būtų rašomos nuo pradžių. XML schemų dokumentų apdorojimui naudojama `System.Xml.Schema` klasių biblioteka.

Programos modeliavimui naudojama MagicDraw įrankis, o programavimas būtų atliekamas Microsoft Visual Studio .NET aplinkoje.

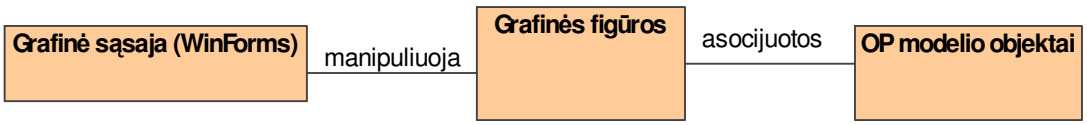
Naudojamos MVC (Modelis-Vaizdas-Valdiklis) architektūros privalumas yra tai, kad OP schemos modelis yra atskirtas nuo pristatymo logikos. Pakeitimai atlikti vartotojo sąsajoje neverčia keisti modelio logikos. Gali iškilti problemų su programos apimtimi, jeigu būtų sukuriama daug objektų su sudėtinga struktūra, o PĮ realizacija ir derinimas užtruktų pernelyg ilgai.



3.5 pav. Modelio-Vaizdavimo-Valdiklio architektūra

3.8.2. Tiesioginis schemos redagavimas, Windows.NET platforma

Naudojant šį realizavimo metodą, schemos dokumentas būtų dinamiškai tvarkomas tiesiogiai iš .NET GUI elementų procedūrų. Šis metodas yra analogiškas anksčiau aprašytam, tačiau atsisakius MVC. Pagrindinis tokio varianto trūkumas: naudojant struktūrinio programavimo metodiką, vartotojo sąsajos kodas būtų neatsiejamai surištas su dokumento manipuliacijos kodu. Programa gali būti sparčiau realizuojama ir reikalautų mažiau išeities kodo tekstų, tačiau apsunkintų jos plėtimą ir modifikavimą.



3.6 pav. Architektūra, kurioje schemos modelio valdymas integruotas į vartotojo aplinkos logiką

3.8.3. MVC architektūra su GEF

GEF grafų redagavimo aplinka (*angl.* Graph Editing Framework) yra Java klasių biblioteka, leidžianti paprasčiau sukurti taikomąsias programas, skirtas diagramų ir susietų grafų redagavimui. Redaguojamos diagramos gali būti verslo procesų modelis, organizacinės lentelės ir kt. Ši priemonė papildo standartinę Java Swing grafinės sąsajos sukūrimo biblioteką. Be to GEF gali būti naudojamas tiek įprastinėse taikomosiose programose, tiek apletuose (*angl.* applet).

GEF biblioteka yra sukurta kaip pakartotinio naudojimo platforma, ir pagrindinės išskiriamosios savybės yra: a) API paprastumas; b) vartotojo sąsajos panaudojamumas; c) pritaikomumas įvairiems modeliams; d) išplečiamumas naudojimui su dideliais modeliais.

Pagrindiniai GEF objektai figūros, grupės, pažymėjimai, komandos ir sluoksniai. GEF pagrindu sukurti redaktoriai remiasi standartiniu MVC modeliu.

3.8.4. MVC architektūra su Eclipse GEF priemonėmis, Eclipse platformoje

Eclipse platforma, paremta Java programavimo kalba, pateikia platų bibliotekų rinkinį ir yra lengvai išplečiama įskiepių (*angl.* plug-in) pagalba. Vienas iš įskiepių – Eclipse GEF [21] (Grafinio redagavimo platforma – *angl.* Graphical Editing Framework) yra MVC architektūros grafinio redagavimo klasių rinkinys, kuris gali būti išplečiamas sukuriant plataus profilio grafinius redaktorius.

Eclipse architektūra turi išplečiamą XML schemų bibliotekų rinkinį XSD (XML Schema Infoset Model) [22], kuris naudojamas XML schemų apdorojimui.

Redaktorius veiktų kaip įskiepis Eclipse platformoje. Jis būtų modeliuojamas MagicDraw įrankiu, programavimas atliekamas toje pačioje Eclipse Java IDE aplinkoje.

3.8.5. Architektūros pasirinkimas

Įvertinant šiuos architektūros modelius, realizacijai buvo pasirinktas paskutinis architektūros variantas, kadangi Eclipse GEF priemonės suteikia daug paruoštų klasių, kurios sutrumpintų modeliavimo ir realizavimo laiką naudojant MVC modelį. Eclipse platforma turi patogias XML schemos programinio modeliavimo priemones, be to sudarant redaktorių nereikės sukurti redaktoriaus vartotojo sąsajos realizacijos, kas atimtų nemažai laiko.

3.9. Tyrimo apribojimai

Reikia pažymėti, kad grafiniame redaktoriuje nėra numatyta realizuoti visų galimų OP darinių dėl apribojimų tyrimo apimčiai. Stengiamasi sudaryti notaciją, galinčią pavaizduoti esmines OP modelio savybės, kurią taip pat galima lengvai transformuoti į XML schemą. Todėl atsisakyta kai kurių OP darinių, ryšių – jie bus supaprastinti.

4. GRAFINIO REDAKTORIAUS PROJEKTAS IR REALIZACIJA

Darbe siekiama sukurti OP modelio schemų grafinį redaktorių, kuriuo būtų galima modeliuoti federacinių sistemų schemas, taip pat panaudoti XML duomenų srautų, naudojamų informacinėse sistemose apibrėžimui XML schema. Kadangi XML turi medžio tipo struktūrinį formatą, skirtingai nuo reliacinių ar objektinių DB, redaktorius atlieka suderinimo tarp duomenų bazėse naudojamų esybių ir sąryšių bei XML medžio struktūros funkciją, aprašytą 2.2 skyriuje. Vartotojas grafiškai redaguoja esybes, nurodo ryšius tarp objektų.

4.1. Eclipse platformos architektūra

Šiame poskyryje bus detalizuojama Eclipse platformos architektūrą, į kurią bus integruojamas projektuojamas redaktorius.

Eclipse yra atviro kodo bendrija, siekianti sukurti atvirą, nepriklausomą nuo gamintojų, taikomųjų programų, kūrimo platformą. Eclipse suformavo nepriklausomą, išplečiamą infrastruktūrą, paremtą atviromis technologijoms ir priemonėmis. Eclipse platforma realizuoja įskiepių (*angl.* plug-in) sąsają, kuri leidžia paprasčiau sukurti, integruoti ir panaudoti programinius įrankius, taupant laiką ir išlaidas. Eclipse platforma parašyta Java kalba ir prieinama kartu su plačia įskiepių kūrimo įrankių aibe ir pavyzdžiais. Eclipse veikia įvairiose kompiuterių architektūrose ir OS: Linux, HP-UX, AIX, Solaris, QNX, Mac OS ir Windows.

Platforma kuriama šiais pagrindiniais tikslais:

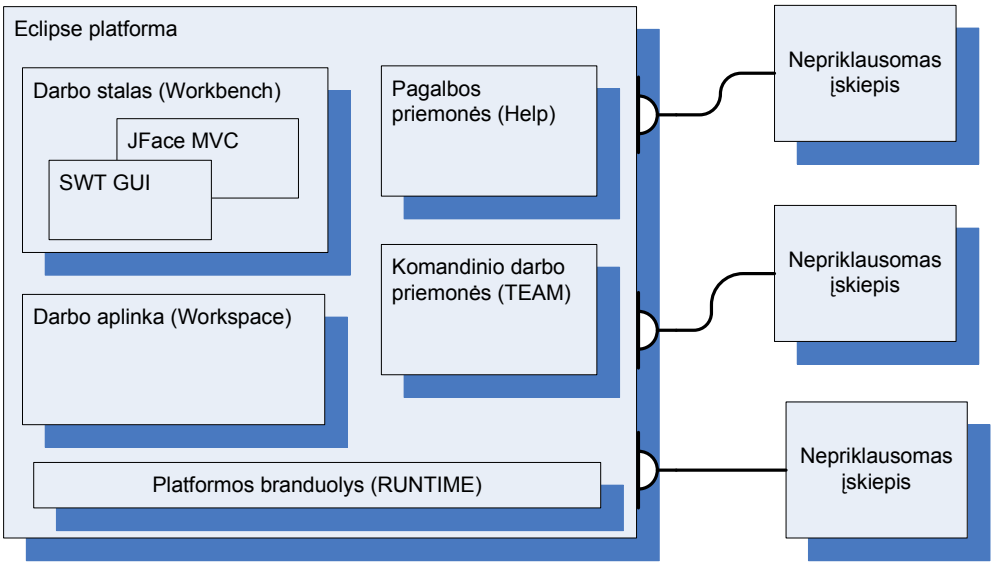
- Suteikti galimybę integruoti įvairiausias priemones PĮ kūrimui;
- Leisti integruoti įvairių gamintojų plačių tipų įskiepius į vieną platformą;
- Suteikti galimybę redaguoti įvairių tipų duomenis ir failus (HTML, Java, C, kt.);
- Pagerinti pakartotinį kodo naudojimą;
- Palaikyti tiek grafinį tiek konsolės lygio įrankių kūrimą;
- Veikimas įvairiose platformose;
- Panaudoti Java programavimo kalbos populiarumą skatinant nepriklausomų įskiepių kūrimą.

Platformos aprašyme naudoti šaltiniai: [20], [23].

4.1.1. Branduolys ir įskiepiai

Eclipse platformos centre yra mažas branduolys skirtas modulių (įskiepių) autorizavimui, jų tarpusavio priklausomybių tikrinimui ir paleidimui. Moduliai susiejami priklausomybėmis, t.y. įskiepis negali būti užkrautas, kol neužkrauti jam reikalingi kiti įskiepiai. Programinio įrankio tiekėjas produktą apiforminą kaip įskiepį, kuris integruojasi į Eclipse platformą. Paleidus ją ir užkrovus visus įskiepius vartotojas gali naudotis tų įskiepių galimybėmis.

Platforma suteikia daug paruoštų bibliotekų, kurias sėkmingai integravus galima išvengti pakartotinio kodo bei panaudoti šių bibliotekų galimybes, kurios tobulėja atskirai nuo įskiepio.



4.1 pav. Eclipse platformos architektūra ir įskiepiai

Eclipse platformos programiniai objektai yra maksimaliai universalūs, stengiamasi išplėsti jų panaudojimo sritis. Daugelis esminių objektų gali prisitaikyti (*angl.* adaptable) priklausomai nuo naudojimo.

Įskiepiai platformoje charakterizuojami pagal jų *manifest* faile paskelbtas naudojamas kitų įskiepių paslaugas bei pačio įskiepio suteikiamos paslaugos. Tokios paslaugos yra naudojamos pačioje platformoje ar kitų įskiepių. Remiantis šia informacija platforma nustato įskiepių užkrovimo tvarką. Paslaugų pavyzdžiai gali būti: „HTML failų redaktorius“, „Grafikos braižybos biblioteka“ ir pan.

4.1.2. Darbo aplinka (Workbench)

Eclipse vartotojo darbo aplinka remiasi keliomis esminėmis sąvokomis.

Projektai Eclipse aplinkoje yra resursų rinkinys realizuojantis vieną programinį sprendimą. Projektai gali būti tipizuoti, t.y. skirti tam tikro tipo (pvz. Java) sprendimams realizuoti.

Resursai yra duomenų vienetas (dažniausiai failas) platformoje, kurio apdorojimui/redagavimui gali būti priskirtas konkretus užkrautas įskiepis. Resursai gali būti ir vietiniai, ir nutolę (per ftp, http protokolą ir pan.).

Žymės (*angl.* marker) yra platformos mechanizmas apipavidalinti resursus. Žymės gali būti naudojamos nurodyti kompiliavimo klaidas, darbų sąrašą, paieškos rezultatus ir kita. Žymės gali būti išplečiamos įskiepių, pritaikant tam tikram resursų tipui.

Pakeitimų istorija platformoje yra papildoma apsaugos nuo vartotojo klaidų priemonė leidžianti realizuoti vykdymo ir grąžinimo (*angl.* undo/redo) mechanizmą visiems vartotojo atliekamiems veiksmams.

4.1.3. Vartotojo sąsajos priemonės

Eclipse platformos vartotojo sąsajos realizacija remiasi darbo aplinka, ir gali būti išplečiama įskiepių. Realizacija remiasi dviem pagrindiniais įrankiais:

SWT, vartotojo grafinių komponentų (*angl.* widget) biblioteka, abstrahuojanti realios vartotojo OS aplinkos komponentus Java objektais. Šia biblioteka realizuotas Eclipse vartotojo sąsaja. SWT sujungia skirtingų OS grafinės aplinkos komponentus į vieną sąsają, vienu metu pagerinant sistemos perkeliamumą bei sudarant natūralios konkrečiai OS taikomosios programos vaizdą.

JFace yra programinių priemonių biblioteka, pagreitinant vartotojo aplinkos programavimo užduotis. JFace objektais abstrahuoja modernios vartotojo aplinkos sudarymo elementus: veiksmus (*angl.* action), MVC elementus: rodytojus (*angl.* viewers) ir valdiklius.

4.1.4. Kitos priemonės

Bazinėje platformoje yra sukurta vartotojo pagalbos priemonių infrastruktūra. Paremta HTML failais, dokumentai gali būti indeksuojami, pasiekiami centralizuotai ir pagal kontekstą.

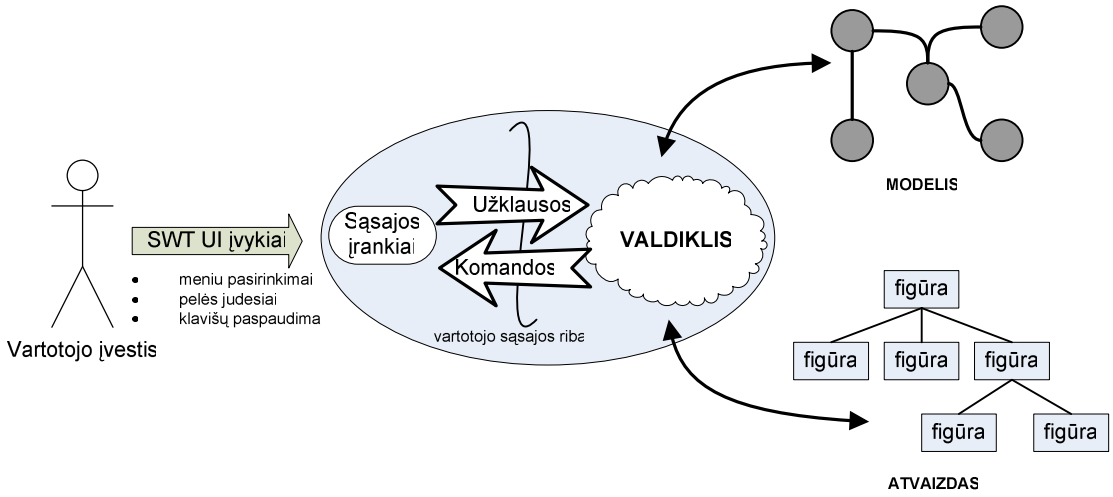
Taip pat yra realizuotos pagrindinės komandinio darbo priemonės, kurios gali būti išplečiamos priklausomai nuo naudojamos infrastruktūros. Kartu su bazine platforma yra gaunamas CVS komandinio darbo įskiepis, išplečiantis platformos komandinio darbo priemones.

4.2. Eclipse GEF įskiepis ir grafinio redaktoriaus architektūra

GEF [21] (Grafinio redagavimo platforma) leidžia programinės įrangos kūrėjams sukurti grafinius redaktorius, turtingus grafiniu vaizdavimu bei elgsena. GEF platforma susideda iš dviejų įskiepių: Draw2D bei pačio GEF.

Draw2D yra išplečiama grafinių figūrų sudarymo, išdėstymo ir brėžimo biblioteka. Gali būti naudojama atskirai nuo GEF.

GEF paremtas MVC (Modelis-Vaizdas-Valdiklis) šablonu. Toks šablonas leidžia patogiai atrišti modelio objektus nuo vaizduojamųjų. GEF gali būti pritaikomas įvairiems modeliams ir grafinių notacijų, tokių kaip programų vartotojo sąsajos, klasių, EER diagramos, būsenų mašinoms ir pan. modelių redaktoriams.

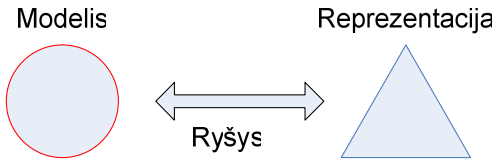


4.2 pav. GEF platformoje realizuoto redaktoriaus architektūra

Įskiepiai norintys sukurti savo redaktorių nurodo priklausantys nuo GEF ir realizuoja atitinkamus objektus.

4.2.1. Modelio – Vaizdo susiejimas, valdiklių objektai

Pagrindinė grafinio redagavimo problema yra susieti redaguojamo modelio objektus, saugančius tam tikrus duomenis su grafine šių objektų reprezentacija, realizuota tiesėmis, apskritimais ir kitomis figūromis. Taip pat yra būtina, kad modelio pakeitimus būtų galima dinamiškai atvaizduoti reprezentacijoje, be to modelio grafinės reprezentacijos pakeitimai atsispindėtų modelyje:



4.3 pav. Modelio ir ją reprezentuojančios figūros sąryšis

MVC architektūroje šis modelį ir reprezentaciją siejantis ryšys yra abstrahuojamas į valdiklio (*angl.* controller) objektą, kurio bazinė GEF realizacija yra `AbstractGraphicalEditPart` klasė. Vienas iš esminių MVC apribojimų yra tai, kad modelio objektas neturi nieko žinoti apie reprezentacinį objektą ir atvirkščiai. Taip pasiekiamas geras kodo moduliarizacijos lygis, leidžiantis nepriklausomai pakeisti modelį ar vaizdo figūrą.

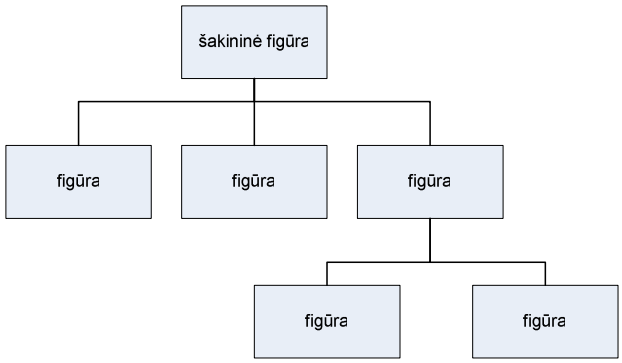
Tam, kad modelis galėtų būti prijungtas prie valdiklio objektų reikalingas įvykių mechanizmas, valdikliui signalizuojantis apie modelio pasikeitimą. Gavęs tokį pranešimą valdiklis keičia arba sukuria modelio objektą reprezentuojančią figūrą(-as).

Vartotojo pakeitimai grafinėms figūroms yra modeliui perduodami užklausų (*angl.* request) pagalba, kurios yra priimamos atitinkamų figūrų valdiklių ir yra transformuojamos į modelio objektų pakeitimus inkapsuliuotus į komandų objektus (*angl.* command). Komandų vykdymo metu yra

pakeičiamas modelis ir komanda užregistruojama Eclipse „undo/redo“ steke, kad būtų galima atstatyti originalią modelio būseną prieš pakeitimą.

4.2.2. Vaizdavimo figūros, Draw2D

Redaktorių modelių grafinės notacijos gali būti labai įvairios ir sudėtingos. Draw2D realizuoja hierarchišką sistemą grafinių figūrų sudarymui ir pavaizdavimui. Figūros gali būti tiek sudarytos iš kitų, tiek bazinės, kurios pateikiamos su Draw2D (apskritimas, tiesė, ir t.t.).



4.4 pav. Draw2D figūrų hierarchijos pavyzdys

Sudėtingos figūros, susidedančios iš kitų yra formuojamos pasitelkiant išdėstymo (*angl.* layout) objektus. Išdėstymo objektai sudeda figūros sudedamąsias figūras tam tikra geometrine tvarka. Draw2D pateikia standartinius išdėstymo objektus, skirtus, pavyzdžiui, vertikaliam subfigūrų išdėstymui tėvinėje figūroje. Sudėtingesnioms grafinėms notacijoms pavaizduoti dažnai prireikia sudaryti specialius išdėstymo objektus.

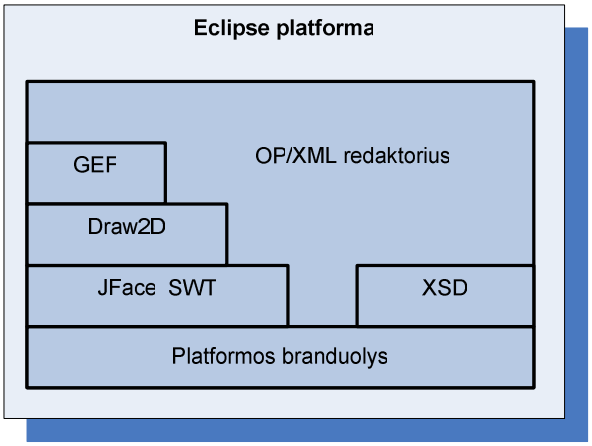
4.3. XML schemų redaktoriaus įskiepio architektūra

Kuriamas redaktorius turi integruotis į Eclipse aplinką, todėl natūraliai pasirinkta Eclipse įskiepio architektūra bei Java programavimo kalba. Įskiepis suteikia (išplečia) šiuos platformos servisus:

- Redaktoriaus, papildant galimybę redaguoti „.opmodel“ tipo resursus/failus;
- Darbo aplinkos veiksmus (meniu, įrankių juostos elementai);
- Naujo failo sukūrimo galimybė „.opmodel“ failams.

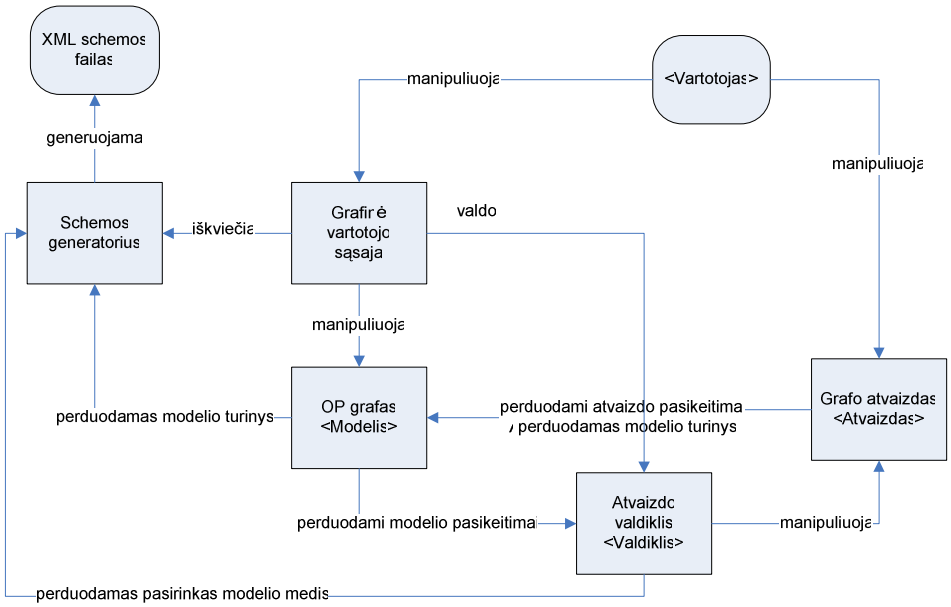
4.3.1. Loginė architektūra

Redaktoriaus architektūros modelis realizuoja MVC šabloną, išplėsdamas Eclipse GEF įskiepį. Taip pat integruojasi į kitus Eclipse platformos įskiepius:



4.5 pav. Redaktoriaus įskiepio priklausomybės

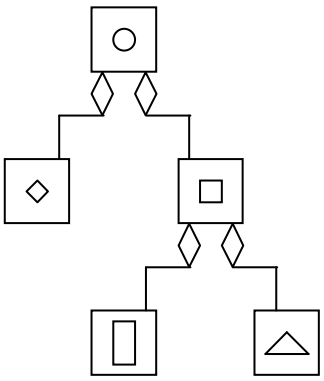
Pateikiama loginė redaktoriaus architektūra, pavaizduojant failų mainus, sąsają su vartotoju ir vidinius duomenų srautus.



4.6 pav. Loginė redaktoriaus architektūra

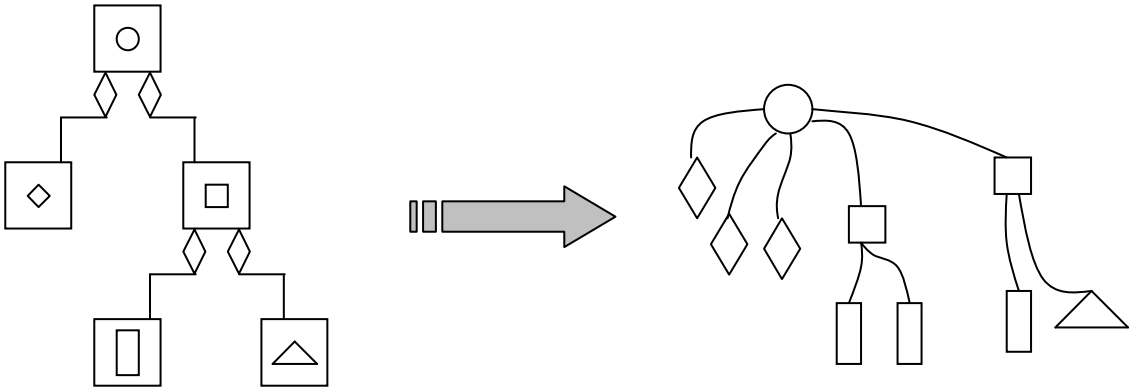
4.3.2. Modelio-Vaizdo-Valdiklio šablono realizacija

Redaktoriaus sąsajai sukurti naudojamas MVC objektiniu šablonas, kurio pagrindas yra Eclipse GEF [21] platforma. OP grafo modelį sudarantys elementai agreguojami (sudarantys) vienas į kitą, taip pat nustato galimus agreguoti modelio objektus, ryšius kuriuose jie gali dalyvauti ir kitą elgseną.



4.7 pav. Modelio elementų agregavimo hierarchijos pavyzdys

Valdiklis, rekursyviai apeidamas modelio elementus sukuria atitinkamas atvaizdo celes, kurios yra susiejamos su modelio elementų objektais. Taip pat valdiklis seka modelio elementų pasikeitimus, atitinkamai keisdamas atvaizdo celių būseną, jas ištrindamas, sukurdamas ar modifikuodamas.



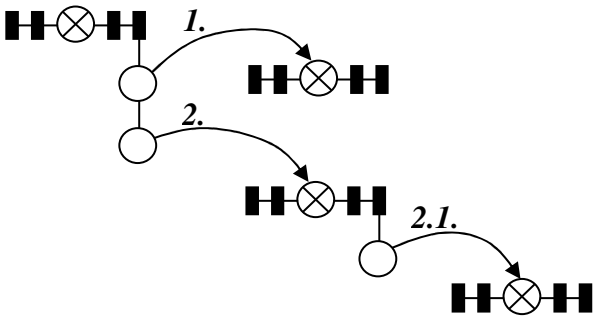
4.8 pav. Modelių elementų vaizdo atitikmens generavimas

Vartotojas manipuliuoja grafiniais elementais, kurių poveikio metu yra generuojamos užklausos (*angl.* request), o jie, apdoroti to grafinio elemento valdiklio, perteikiami komandos objektais (*angl.* command) keičiančiais realius OP modelio schemą realizuojančius objektus.

4.3.3. XML schemos generavimo iš OP poschemės procesas

XML schemos generavimui iš OP modelio poschemės redaktoriaus įskiepyje realizuojamas algoritmas, aprašytas darbo „2.3 Transformavimo taisyklės“ skyriuje. Šio proceso duomenų srautai simbolizuoti paveiksle loginės redaktoriaus architektūros paveiksle, ankstesniuose skyriuose.

Kaip nurodyta paveiksle, XML schemos sudarymui yra reikalingas OP grafo medžio padengimas, nurodantis aprašomo XML medžio struktūrą. Medis padengiamas vartotojui nurodžius esybes, atributus bei ryšius, kurie įeina į medžio hierarchiją. Naudodamiesi OP modelio sudedamųjų klasių diagrama analizuojame medį nuo šaknies ir kiekvieną vis tolimesnę nuo šaknies modelio elementą transformuojame į atitinkamą XML schemos elementą.



4.9 pav. OP modelio grafo padengimas medžio tipo struktūra

Turint OP modelio padengimą yra generuojamas jį atitinkančios XML schemos dokumentas, naudojantis taisyklėmis, apibrėžtomis 2.3 skyriuje.

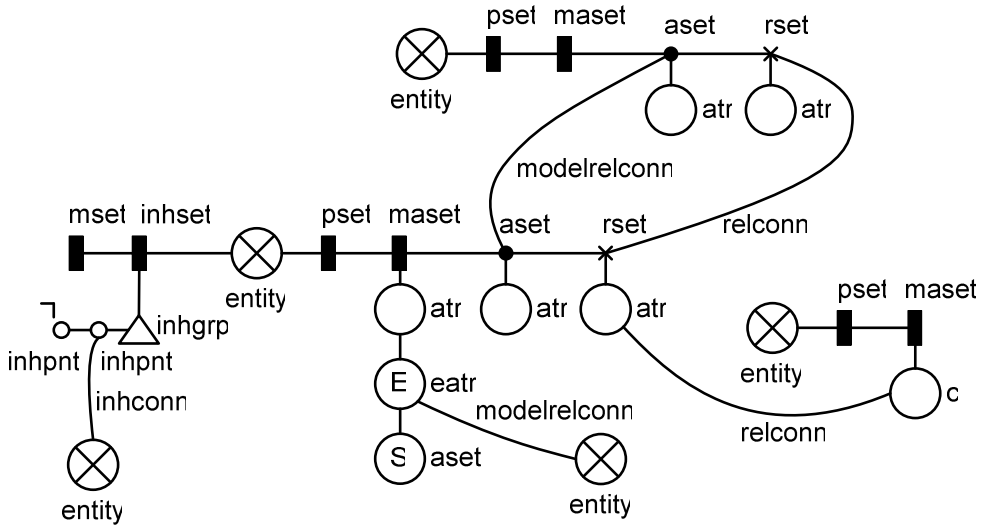
4.4. OP bei XML schemų modelių realizacija

Pereinant prie redaktoriaus projekto reikalinga realizuoti jame manipuliuojamų duomenų modelių objektus. Šioje PĮ naudojami OP bei XML schemų modeliai, kurių realizacija aprašyta sekančiuose skyriuose.

4.4.1. OP modelio darinių programinė reprezentacija

Kadangi redaktoriuje realizuojama OP modelio grafinė notacija, visų pirma yra reikalinga apibrėžti objektų klasių sistemą, atitinkančią OP modelio grafo darinių savybes. Į šių objektų aibę taip pat įeis ne tik modelio esminė informacija (OP objektų pavadinimai, tipai, agregavimas), bet ir grafinė informacija (OP elemento geometrinė pozicija bei dydis).

OP dariniai skirstomi į dvi grupes: mazgų bei ryšių objektai. Toks padalinimas yra naudingas atskiriant šias dvi funkcionaliai skirtingas grupes ir moduliarizuojant kodą, tačiau taip pat yra patogus realizuoti GEF infrastruktūroje, kurioje šie objektų yra atskirti. Modelio analizės metu buvo gautos šios OP darinių Java klasės, pateikiant darinius kuriuos jos atitinka.



4.10 pav. Įvairių OP modelio darinių reprezentacija redaktoriuje

Aukščiau pateiktame paveiksle pateikta OP darinių grafinė reprezentacija bus realizuota redaktoriuje. Sutrumpinimai šalia darinių nurodo į realios klasės pavadinimą, kurios detalizuojamos sekančioje lentelėje. Joje neaprašomos abstrakčios darinių klasės ir sąsajos, tokios kaip `ContainerElement` – OP elementas agreguojantis kitus elementus.

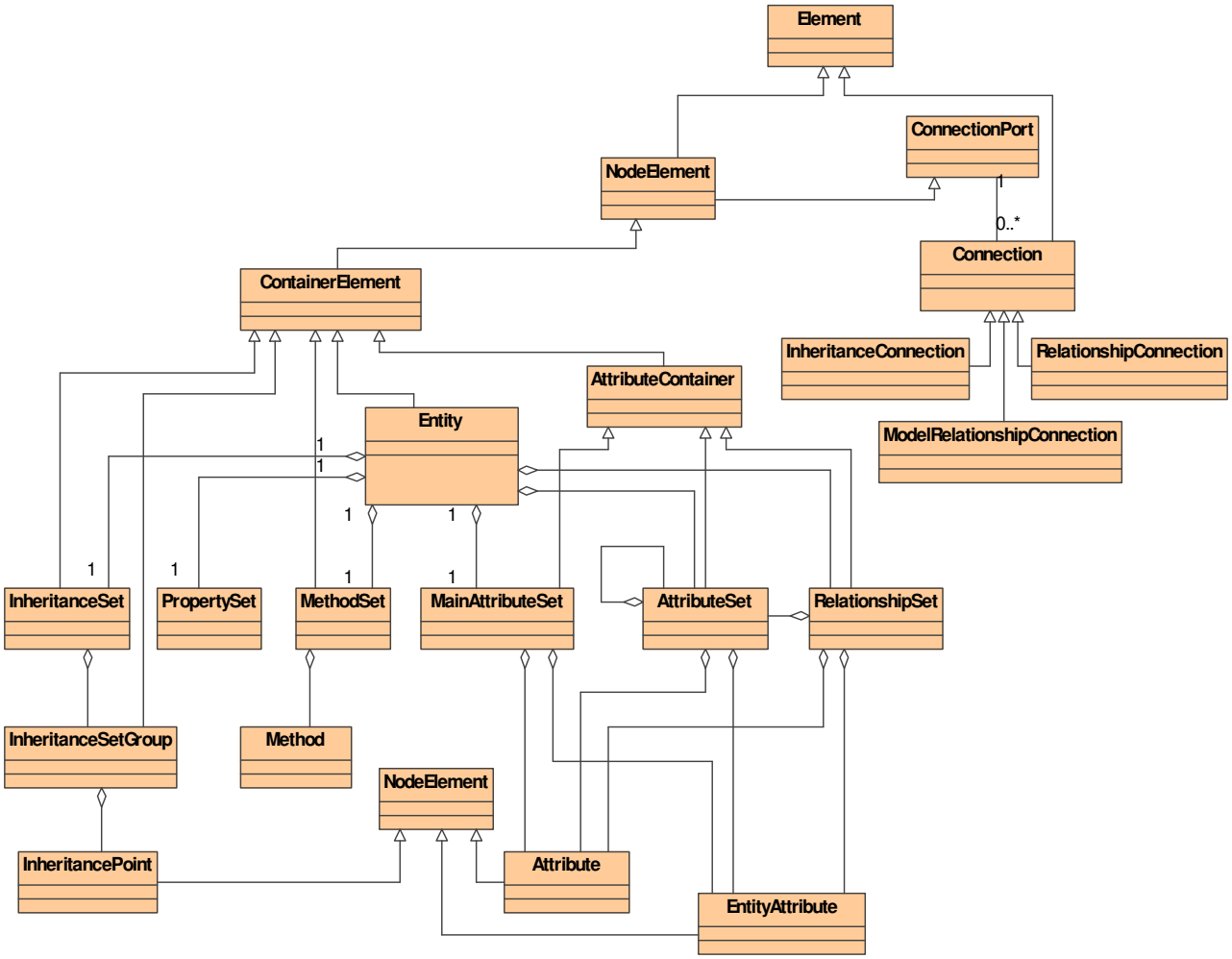
4.1 lentelė. OP modelio elementų atitikmenys realizacijoje Java klasėmis

Sutrumpinimas	Klasės pavadinimas	Tėvinė klasė	Aprašymas
entity	`Entity`	`ContainerElement`	`Entity` inkapsuliuoja OP modelio objektą. Taip pat saugoma geometrinė padėtis.
pset	`PropertySet`	`ContainerElement`	Savybių rinkinys, saugantis visas objekto savybes (atributus, vid. ryšius, metodus ir kt.)
maset	`MainAttributeSet`	`AttributeContainer`	Pagrindinis objekto atributų bei vidinių ryšių rinkinys. Agreguoja visus objekto vidinius ryšius ir atributus.
aset	`AttributeSet`	`AttributeContainer`	Vidinis atributų rinkinys (dar vadinamas vidiniu ryšiu). Gali agreguoti kitus vidinius ryšius, atributus.
rset	`RelationshipSet`	`AttributeContainer`	Projekcijos atributų rinkinys. Gali agreguoti vidinius ryšius bei atributus.
inhset	`InheritanceSet`	`ContainerElement`	Objekto paveldimumo rinkinys. Agreguoja paveldimumo grupes.
mset	`MethodSet`	`ContainerElement`	Metodų rinkinys.
atr	`Attribute`	`NodeElement`	Atributas, turintis pavadinimą bei priskirtą tipą. Tipai parenkami iš aprašytų standartinių XML schemos tipų [6].
eatr	`EntityAttribute`	`NodeElement`	Objekto, esančio kaip atributas kitame, klasė. Susietas modelio sąryšio jungtimi (`ModelRelationshipConnection`), nurodančia konkretų OP objektą.
inhgrp	`InheritanceSetGroup`	`ContainerElement`	Paveldimumo grupė agreguoja paveldimumo taškus. Priklausomai nuo agreguojamų taškų grupė atvaizduoti *isa* ryšio visus 4 variantus.
inhpnt	`InheritancePoint`	`NodeElement`	Paveldimumo taškas susiejamas su vienu išoriniu objektu paveldimumo ryšiu (`InheritanceConnection`), nurodant paveldintįjį objektą.

4.1 lentelės tęsinys

modelrelconn	`ModelRelationship-Connection`	`Connection`	Modelio ryšio objektas. Naudojamas apibrėžti OP modelio objektų bei atributų santykius.
relconn	`RelationshipConnection`	`Connection`	Ryšio objektas naudojamas pavaizduoti objektų egzempliorių sąryšiams (projekcijoms).
inhconn	`InheritanceConnection`	`Connection`	Paveldimumo ryšio objektas, naudojamas nurodyti OP objektų paveldimumo (*isa*) hierarchiją bei tipą.

Šių išnagrinėtų klasių hierarchijos diagrama pateikiama sekančioje diagramoje, nepateikiant abstrakčių klasių ar sąsajų.



4.11 pav. OP modelio darinių klasių diagrama

Diagramoje pateiktos bazinės klasės (tokios kaip `ContainerElement`, `NodeElement`, `Connection`) realizuoja esmines OP modelio elementų sąsajas bei jų elgseną. Siekiant sudaryti patogų modelio objektų išsaugojimo ir užkrovimo iš failų mechanizmą, visi OP darinių Java objektai paveldi `java.io.Serializable` sąsają.

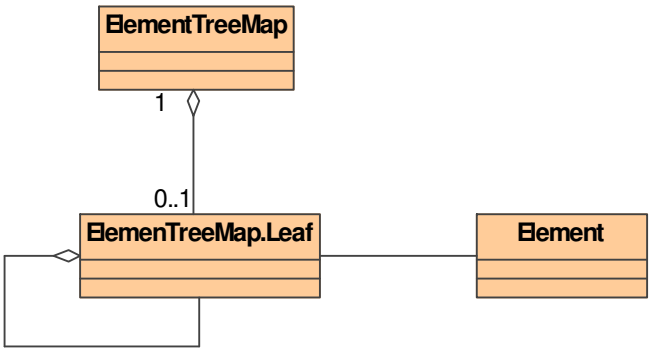
Išoriniai ryšiai OP realizacijos objektuose realizuoti išplečiant `Connection` bazinę abstrakčią klasę ir sieja du ryšio elementus, realizuojančius `ConnectionPort` sąsają. Kiekvienas konkretus OP modelio objektas turi savo unikalią elgseną, nurodančią kokiuose ryšiuose ir jų kombinacijose gali dalyvauti.

Reikia pažymėti, kad įvesti `ModelRelationshipConnection` ryšio objektai nurodantys atributo–esybės ir atributo–atributų rinkinio tipus, nėra būtini visais atvejais, tačiau redaktoriaus OP modelio realizacijos atveju jie yra privalomi, prieš sudarant projekcijos ryšius `RelationshipConnection`, vardan paprastesnės realizacijos.

Visi `Entity` objektai esantys OP grafe yra agreguoti `Graph` klasėje, kuri nenurodyta klasių diagramoje pav. 4.11.

4.4.2. OP modelio padengimo medžiu realizacija

XML poschemės generavimo iš OP modelio algoritmui reikalingas skyriuje 2.2 aprašytas padengimo metodas redaktoriuje realizuojamas `ElementTreeMap` bei jos vidinėje klasėje `ElementTreeMap.Leaf`.



4.12 pav. OP modelio padengimo medžiu realizacijos klasių diagrama

Tokia klasių struktūra leidžia sudaryti nepriklausomą nuo OP modelio padengimo medį (`ElementTreeMap`), nurodant padengiamus elementus (`Element`). `ElementTreeMap` klasė tarp kitų funkcijų taip pat realizuoja sub-medžio kopijos ištraukimą pagal dabartinį medžio padengimą.

Natūraliai, `ElementTreeMap` saugomas medžio padengimas yra naudojamas generuojant jį atitinkančios XML schemos medį.

4.4.3. XML schemos objektų programinė reprezentacija

Redaktoriuje naudojamas XML schemų objektinis modelis yra realizuotas Eclipse įskiepyje XSD [22] (XML Schema Infoset Model).

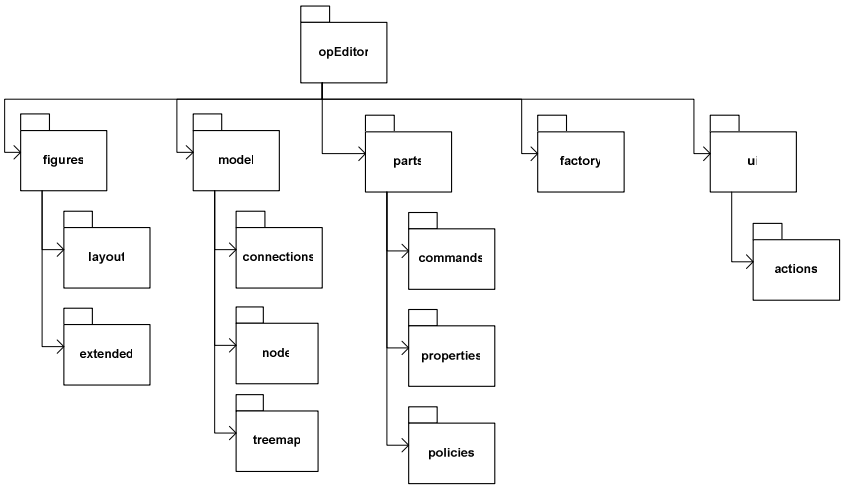
XSD yra JAVA biblioteka, paremta Eclipse platformos EMF įskiepiu bei XML schemos modeliu, kuri gali būti naudojama ten, kur yra reikia analizuoti, sukurti ar modifikuoti XML schemos

dokumentus. XSD API yra paremta DOM API [24] modeliu, kuo labiau jį priartinant prie XML schemos sąvokų. Taip pat yra galimybė nuskaityti ir išsaugoti serializuotą modeliuojamos schemos modelį į ir iš failo, patikrinti schemos dokumentų integralumą. Bibliotekos kūrėjų yra siekiamas 100% suderinamumas su XML schemos modeliu.

4.5. Projekto realizacijos komponentų detalizacija

Poskyris skirtas detalizuoti redaktoriaus programinės įrangos struktūrą ir sudėti. Identifikuojami paketai bei komponentų paskirtis.

Programinės įrangos išeities tekstai yra suskirstyti į Java paketus, kurie pateikti ir aprašyti žemiau. Paketų hierarchizacija atspindi jų vietą realizacijoje.



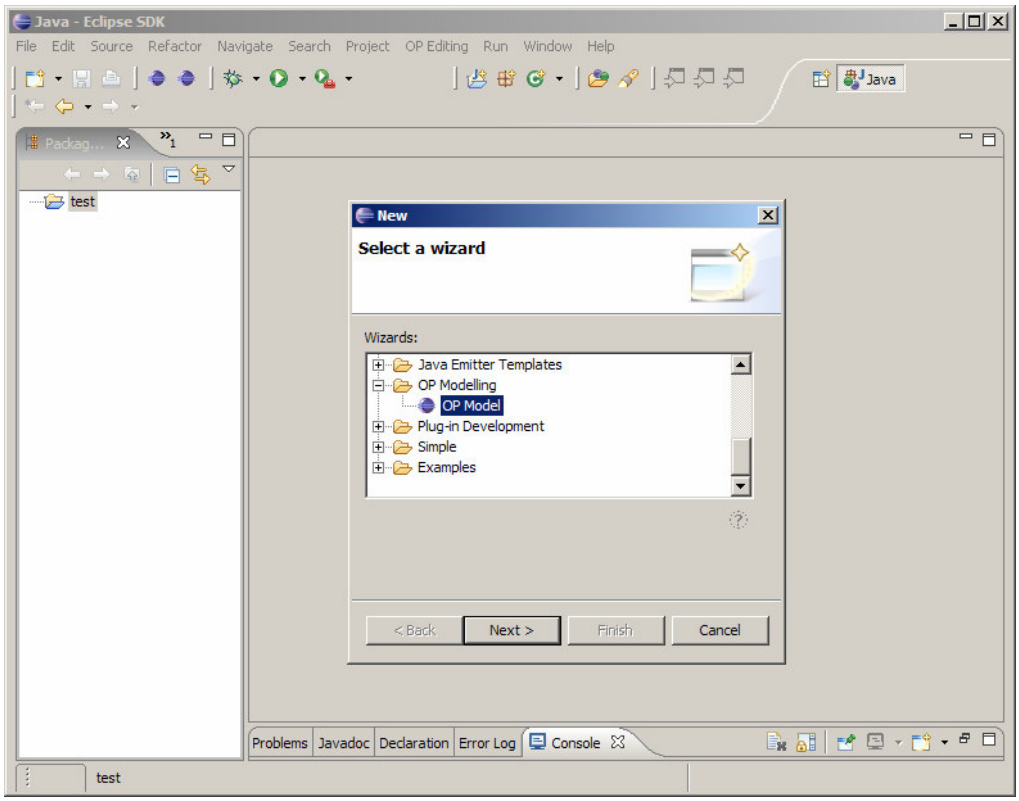
4.13 pav. PĮ realizacijos komponentų diagrama

4.2 lentelė. Paketų funkcionalumo aprašymas

Pavadinimas	Funkcionalumas
`opEditor`	Šakninis redaktoriaus įskiepio paketas.
`opEditor.figures`	Redaktoriaus Draw2D OP notacijos figūrų realizacijos.
`opEditor.figures.layouts`	Figūrų išdėstymo klasių realizacijos.
`opEditor.figures.extended`	Išplėstos bazinių figūrų realizacijos (elipsė, laužtinė linija ir kt.)
`opEditor.model`	OP modelio darinių atitikmenų realizacijos bazinės klasės.
`opEditor.model.connection`	Ryšių realizacijos klasės bei sąsajų aprašai.
`opEditor.model.node`	Konkrečių OP darinių realizacijos.
`opEditor.model.treemap`	Medžio padengimo bei OP grafo transformavimo į XML schemą klasių paketas.
`opEditor.parts`	Redaktoriaus valdiklio objektų klasės.
`opEditor.factory`	OP valdiklių bei modelio objektų gamyklų realizacijos.
`opEditor.parts.model.commands`	Komandų, modifikuojančių modelį realizacijos.
`opEditor.parts.model.properties`	Savybių pateikimo/redagavimo realizacijos (integravimui į Eclipse aplinką).
`opEditor.parts.policies`	Valdiklių elgsenos taisyklių, inkapsuliuojančių tam tikrą valdiklio elgseną, realizacijos.
`opEditor.ui`	Eclipse vartotojo sąsajos integravimo klasės.
`opEditor.ui.actions`	Veiksmų klasės, abstrahuojančios išorinius vartotojo veiksmus, susijusius su redaktoriaus funkcionalumu.

4.6. Vartotojo dokumentacija

Įdiegus Eclipse bei OP redaktoriaus įskiepį (opEditor) Eclipse atsiranda papildoma galimybe sudaryti ir redaguoti OP modelio (.opmodel) failus. Paleidus Eclipse yra sukuriamas naujas arba panaudojamas jau egzistuojantis projektas ir sukuriamas naujas OP modelio failas. Taip pat galima užkrauti jau egzistuojantį modelį, pasinaudojus Eclipse *File⇒Load* funkcija.



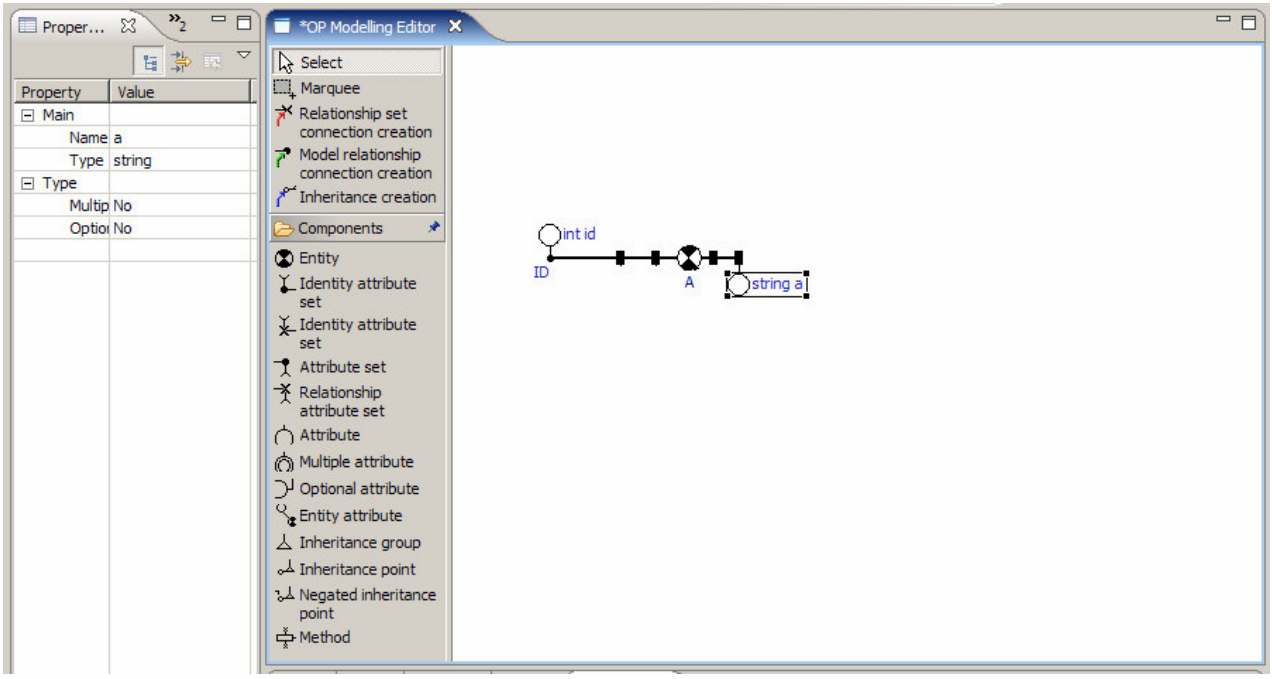
4.14 pav. OP modelio failo sukūrimas

4.6.1. Redagavimo galimybės

Užkrovus ar sukūrus naują failą vartotojo redagavimo lauke atsiranda OP modelio darinių paletė, kurios elementus galima „užtempti“ ant darbo lauko į tinkamą vietą. Kursoriaus pavidalu vartotojui yra nurodoma ar veiksmas (ne tik įkėlimo) galimas.

Ryšys tarp tinkamos rūšies darinių yra sudaromas pasirinkus jo ikoną paletėje bei nuspaudus du kartus po vieną kartą ant atitinkamų elementų. Jei ryšys nėra galimas sudaryti dėl apribojimų, jis nesudaromas, o vartotojui tokia situacija nurodoma kitokio tipo kursoriumi.

Modelio realizacijoje yra numatyti įvairūs apribojimai, todėl, pvz. negalima įkelti kelių atributų (*angl.* Attribute) su vienodais vardais į tą pačią esybę (*angl.* Entity).



4.15 pav. OP modelio redagavimo paletė bei darbo laukas

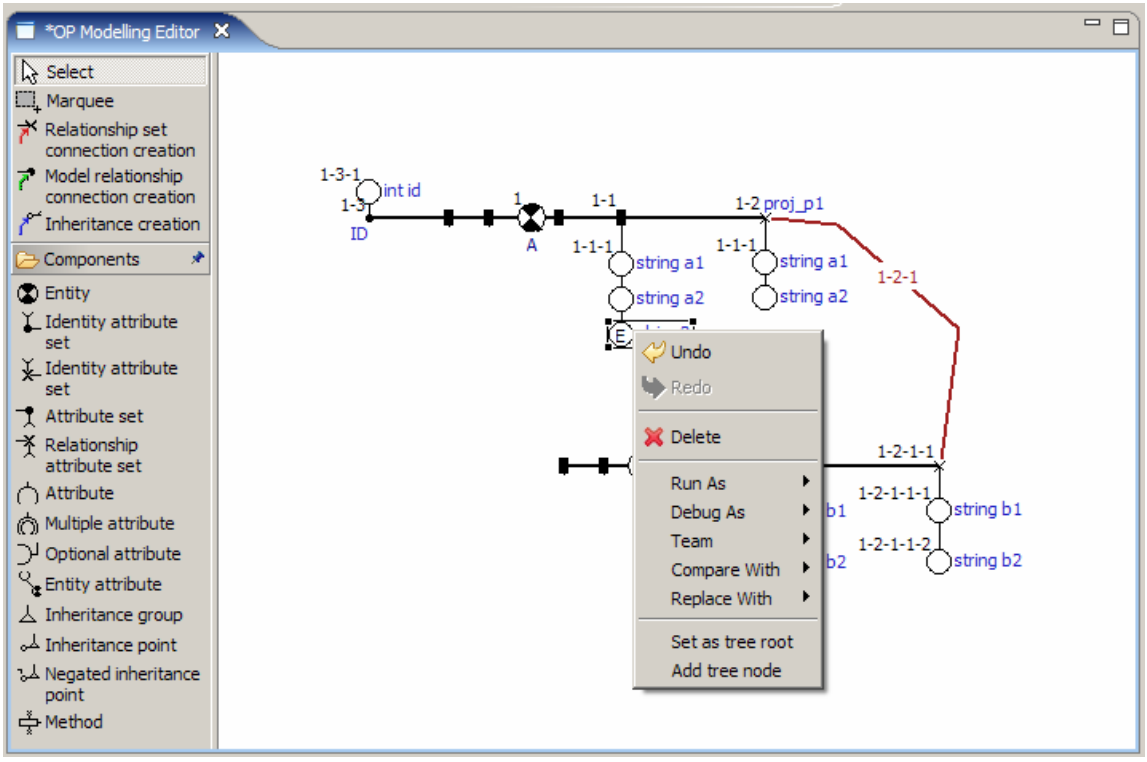
Modelio redagavimas vykdomas įvairiais būdais. OP darinio, kuris gali būti įvardintas, pavadinimą galimą keisti vietoje, pasinaudojus F2 klavišu, arba du kart spustelėjus ant pavadinimo. Taip pat galima naudoti savybių (*angl.* properties) langą, įjungiamą pasinaudojant meniu *Window⇒ Show⇒ View⇒ Other⇒ Basic⇒ Properties*. *Properties* lange taip pat galima keisti kitas OP darinio savybes (išskyrus pavadinimą) priklausomai nuo jo tipo: atributo reikšmės tipą, geometrinę poziciją lange ir kt. Taip pat galimas darbas klaviatūra, naudojant kitus funkcinius klavišus, kaip: elemento trynimas su *Delete*, navigacija tarp elementų su *Up*, *Down*, *Left*, *Right* klavišais, kontekstinio meniu iškvietimas *Menu* klavišu.

Taip pat realizuotas veiksmų sugražinimas ir pakartotinis vykdymas (*angl.* undo/redo). Veiksmą galimą gražinti bei vėl įvykdyti klavišų kombinacijom *Ctrl-Z* bei *Ctrl-Y* atitinkamai.

Redaguojamas modelis gali būti išsaugotas pasinaudojus *File⇒ Save/Save As* meniu punktais.

4.6.2. OP modelio padengimas medžiu, subgrafo iškėlimas

Redaktoriuje realizuota galimybė OP modelio grafą padengti medžiu, kuris naudojamas apibrėžiant OP modelio subgrafą kurio kopiją norima gauti arba transformuoti į XML schemos formatą. Šis funkcionalumas pasiekiamas naudojant OP darinio kontekstinio meniu punktus „*Set as tree root*“, „*Add tree node*“, „*Remove tree node*“ punktus.



4.16 pav. OP grafo padengimas medžiu

Sudarius OP padengimą, galimas subgrafo generavimas pasirinkus meniu *OP Editing⇒ Extract Graph* meniu punktą. Vartotojui yra suteikiamas langas nurodyti kopijos išsaugojimo failą, redaktorius sugeneruoja pažymėto grafo kopiją, į ją įtraukdamas ir būtinus elementus (pvz., jei medyje padengtas tik OP objekto atributas, tai kopijoje privaloma įtraukti ir OP objektą bei atributų rinkinį kuriam padengtas atributas priklauso.

4.6.3. XML schemos generavimas

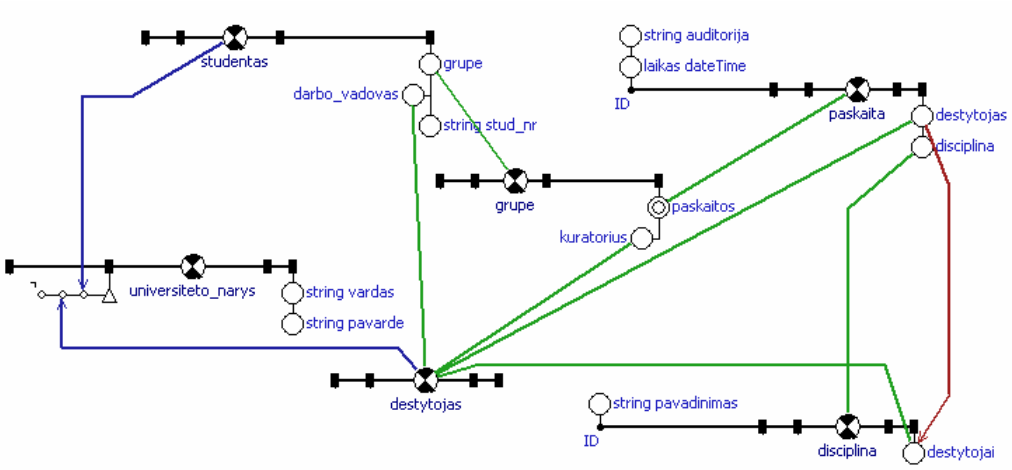
Nurodžius medžio padengimą galima OP medžio subgrafą transformuoti į XML schemos dokumentą. Transformavimas vykdomas meniu punkto *OP Editing⇒ Extract Graph* pagalba. Vartotojui pateikiamas sąsajos langas, skirtas nurodyti išsaugomos XML schemos failo pavadinimą, XML schemos elementų vardų erdvę (*angl.* namespace). Transformavus ir išsaugojus OP modelio subgrafą į schemos failą, atidaromas tos schemos redagavimo XSD redaktoriaus langas.

4.7. Eksperimentinis siūlomo XML schemų generavimo pritaikymas

Sukurtą redaktorių naudosime sudarinėjant pavyzdinės srities OP modelį, kurį vėliau naudosime generuodami dokumento XML schemas. Šiame skyriuje pateikiamas eksperimentinis OP modelis, analizuojama jo įvairius subgrafus atitinkančios, gautos po transformacijos XML schemos.

4.7.1. Eksperimentinis modelis

Pradinis duomenų modelis pateiktas OP modelio schema (pav. 4.17), paremta studijų planų schema. Eksperimentiniame modelyje vaizduojama OP darinių įvairovę bei kaip ji perteikiama iš šio OP modelio subgrafo transformuotoje XML schemoje.

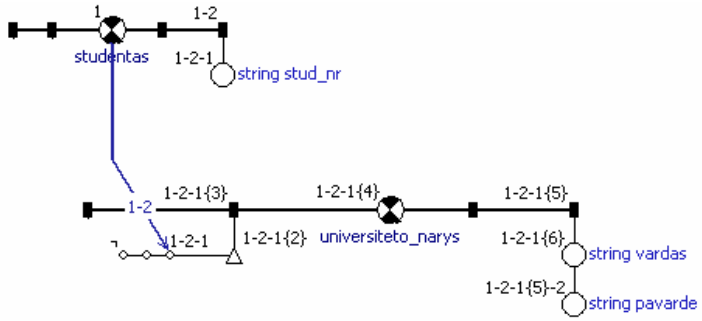


4.17 pav. Eksperimentinio modelio duomenų schema

4.7.2. Modelio padengimų transformacijos į XML schemą

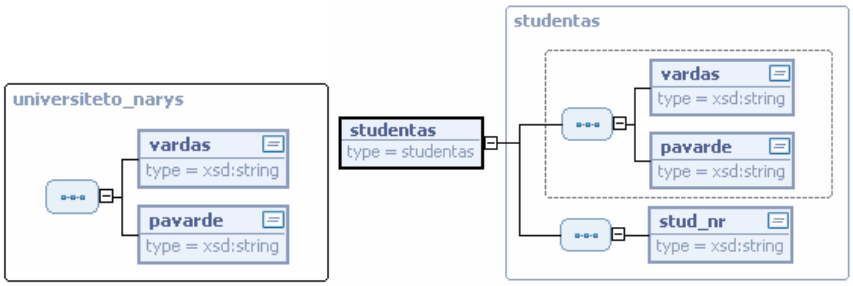
Padengdami nagrinėjamo modelio OP grafą įvairiais medžiais galima generuojamos eksperimentinės XML schemos. Atliekamos kelios padengimų transformacijos, nagrinėjant konkrečias transformavimo savybes:

Pirmas padengimas. *Isa* ryšių, OP objektų kaip kito objekto savybių transformavimas. Iliustruojamas paveldimų savybių perdavimas paveldinčiajame tipe. Paveldimumas realizuojamas XML schemos `<extension>` elementu.



4.18 pav. Pirmas eksperimentinio modelio padengimas

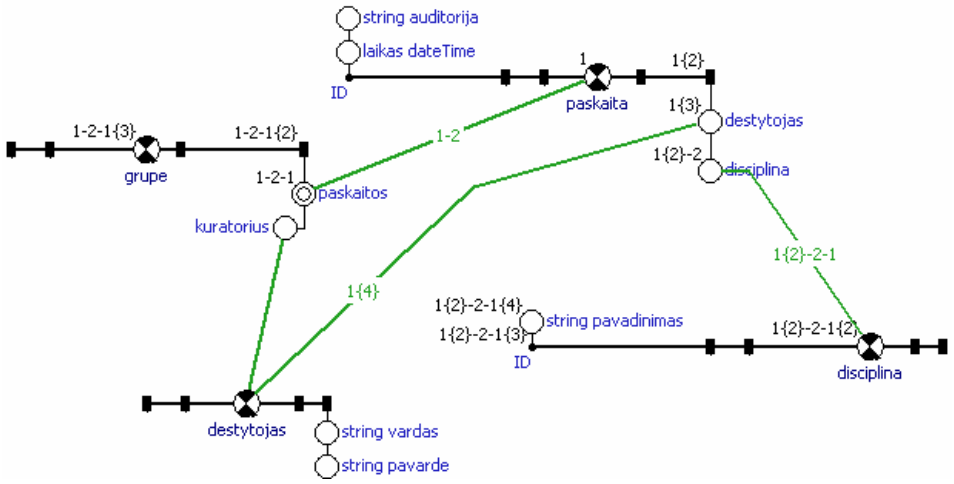
Transformavus šį padengimą gautos XML schemos tekstas yra priede [1].



4.19 pav. Eksperimentinio modelio antrojo padengimo XML schemos grafinis atitikmuo

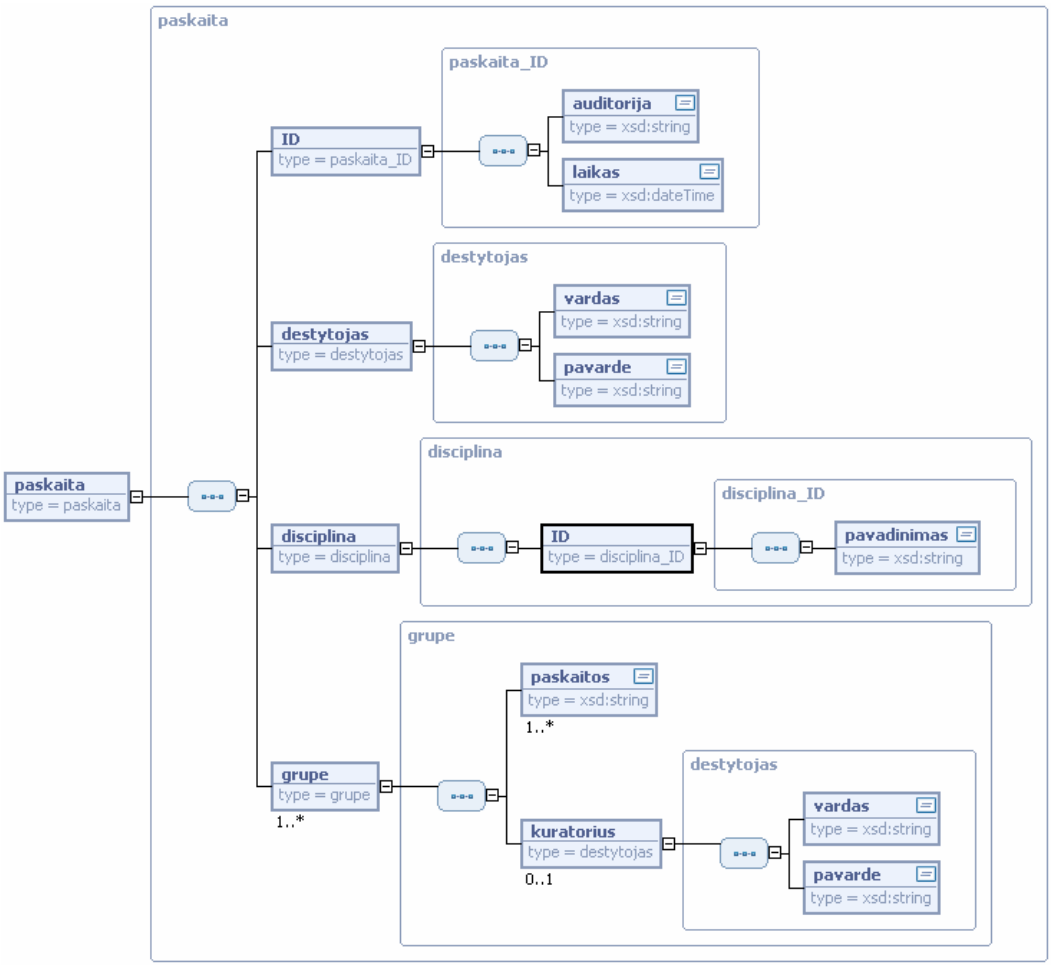
Antras padengimas. Naudojami daugiareikšmiai, neprivalomi atributai, „gilus“ padengimo medis. Šio padengimo transformacija panaudoja neprivalomų bei daugiareikšmių atributų

transformaciją. Šiame pavyzdyje pateikiama išorinio sąryšio transformacija (pavyzdyje sąryšis „1–2“ transformuojamas į XML tipo *paskaita* `<grupe>` elementą). Taip pat iliustruojama vidinių ryšių konversija į duomenų tipus, naudojamus jų „tėviniame“ objekte (pavyzdyje `<paskaita>` elemento vidinis identifikacinis ryšys *ID* transformuojamas į tipą *paskaita_ID* ir panaudojamas `<paskaita>` elemente kaip *ID* vaiko elemento tipas).



4.20 pav. Antras eksperimentinio modelio padengimas

Transformavus šį padengimą gautos XML schemos tekstas yra priede [1].



4.21 pav. Eksperimentinio modelio antrojo padengimo XML schemos grafinis atitikmuo

IŠVADOS

- Išnagrinėti federacinių IS sudarymo principai iš kurių seka nehomogeninių DB schemų integracijos aktualumas panaudojant grafinius modelius.
- Darbe išnagrinėtas OP modelis, jo paskirtis bei notacija. Apibrėžtos išraiškos galimybės:
  - atributai gali būti paprastieji ir sudėtiniai;
  - vienareikšmiai ir daugiareikšmiai;
  - privalomi ir neprivalomi;
  - objektų identifikatoriai gali būti paprastieji ir sudėtiniai;
  - vieno objekto tipai gali turėti kelis identifikatorius;
  - objektų tipų integracijai naudojamos išorinės poaibių priklausomybės.
- Išnagrinėta XML bei XML schemos žymių kalbų specifika, duomenų tipai, sandaros komponentai. Detalizuoti privalumai bei apribojimai. Nurodyti pavyzdžiai, iliustruojantys specifines XML savybes. Trumpai išnagrinėtos XML schemos bendros savybės su OP modeliu.
- Analizės metu buvo palyginti XML schemos bei OP duomenų modeliai. Pateiktos ir apibrėžtos transformavimo taisyklės. Sudarytas OP hipergrafo padengimo metodas, kurio metu gautas padengiantis medis panaudojamas vykdant OP į XML transformavimo taisyklės. Šios taisyklės:
  - yra paprastos, nes leidžia semantinius konstruktus vienareikšmiškai atvaizduoti XML semantiniais konstruktais
  - realizuoja atributų agregatinius ryšius, būdingus reliacinėms schemoms
  - realizuoja objektų apibendrinimo ryšius, kurie nespecifikuojami reliacinėmis schemomis
- Sukurtas grafinio OP modelio redaktoriaus projektas. Apžvelgtos realizacijos galimybės pasirinktose platformose, jų specifika, pasirinkta realizavimo platforma (Eclipse/Java). Nustatyti reikalavimai redaktoriaus realizacijai.
- Sudaryta bei aprašyta OP modelio dokumento realizacija Java kalboje. Pateikta klasių diagrama, Java klasių atitikimas konkretiems OP notacijos dariniams.
- Apžvelgta Eclipse atviro kodo platforma, kurios bazėje buvo realizuotas grafinis redaktorius. Išnagrinėtas ir panaudotas MVC programavimo šablonas, nurodyta jo panaudojimo specifika GEF aplinkoje. Apibrėžta realizuoto redaktoriaus architektūra.
- Sudaryta trumpa iliustruota redaktoriaus vartojimo instrukcija, nurodant specifines panaudojimo savybes, OP grafo padengimo medžiu procesą.

- Sudaryti eksperimentiniai OP modelio dokumentai, kurie panaudoti testuojant redaktoriaus realizaciją, OP modelio transformavimo į XML schemą algoritmą. Trumpai išnagrinėti testuojamų OP modelio darinių transformavimo atvejai, nurodant jų specifiką.
- Darbe neaprėptos visų įmanomų OP modelio darinių realizacijos ir nepilnas transformavimo į XML metodas. Dėl ribotos darbo apimties redaktoriuje nėra realizuota galimybė apibrėžti atributo reikšmę kaip kito objekto subtipą, kas išskiria OP modelį iš EER, ORM be to gali būti atvaizduojama XML schemos apraše.
- Puoselėjant darbą reikia:
    - grafinio redaktoriaus galimybes papildyti atributo reikšmės pavaizdavimu nauju objektų tipu;
    - sudaryti metodiką XML medžių šakniniams objektams identifikuoti, panaudojant verslo procesams modeliuoti naudojamas komunikacines kilpas
- OP modelio transformavimo į XML schemą principai buvo apibrėžti straipsnyje ir paskelbti 10-oje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinės Technologijos 2005“.

NAUDOTA LITERATŪRA

[1] KONTRIMAS Vilius. Federacinės duomenų bazės. Konferencijos "Informacinės Technologijos'2004" pranešimų medžiaga. Kaunas: Technologija, 2004. 465–469 psl.

[2] PARADAUSKAS Bronius, NEMURAITĖ Lina. Duomenų bazės ir semantiniai modeliai. Monografija. Kaunas, Technologija, 2002. 39–50, 159–172 psl.

[3] W3C. Extensible Markup Language (XML) 1.0 (Third Edition). Interneto prieiga: http://www.w3.org/TR/2004/REC-xml-20040204/, 2004.

[4] ISO/IEC. JTC1/SC34 (Standardization in the field of document structures, languages and related facilities for the description and processing of compound and hypermedia documents). Interneto prieiga: http://y12web2.y12.doe.gov/sgml/sc34/sc34oldhome.htm, 2005.

[5] W3C. HTML 4.0 Specification. Interneto prieiga: http://www.w3.org/TR/WD-html40/, 1998.

[6] W3C. XML Schema Specifications and Development. Interneto prieiga: http://www.w3.org/XML/Schema#dev, 2004.

[7] DING Chen, YUEGUO Chen, WEIWEI Cheng. A Survey Study on XML Schema Design, 2002. Interneto prieiga: http://www.comp.nus.edu.sg/~jaga/reports,

[8] PAPA John. XML Features in SQL Server 2000. Interneto prieiga: http://msdn.microsoft.com/library/default.asp?url=/msdnmag/issues/0800/sql2000/toc.asp, 2005.

[9] TALPIN Terry. ORM In Detail. Interneto prieiga: http://www.orm.net/index.html, 2005.

[10] WIEBICKE Ralf. XML Query Languages for Repositiories Based on XML Documents. Interneto prieiga: http://rw7.de/ralf/gbeleg99/intro.html, 2003.

[11] Wikipedia. XML. Interneto prieiga: http://en.wikipedia.org/wiki/XML, 2005.

[12] W3C. XML Schema Part 2: Datatypes Second Edition. Interneto prieiga: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html, 2004.

[13] STAKEN Kimbiro. Introduction to Native XML Databases. Interneto prieiga: http://www.xml.com/pub/a/2001/10/31/nativexmldb.html, 2001.

[14] KONALOVAS Tomas, PARADAUSKAS Bronius. Nehomogeninių duomenų bazių integracija naudojant XML formatus. Aštuntosios magistrantų ir doktorantų konferencijos pranešimų medžiaga, Kaunas: Technologija, 2003, p. 45–48.

[15] W3C. XML Path Language (XPath), W3C Recommendation. Interneto prieiga: http://www.w3.org/TR/xpath, 2005.

[16] GOLUMBIC Martin Charles. Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York. 1980. Psl. 1–18.

[17] W3C. XML Schema Part 1: Structures Second Edition. Interneto Prieiga: http://www.w3.org/TR/xmlschema-1/, 2005.

[18] Sun Microsystems. Java 2 Platform, Standard Edition (J2SE). Interneto prieiga: http://java.sun.com/j2se/index.jsp, 2005.

[19] No Magic, Inc. MagicDraw UML. Interneto prieiga: http://www.magicdraw.com/, 2005.

[20] Eclipse. About Eclipse. Interneto prieiga: http://www.eclipse.org/org/, 2005.

[21] Eclipse. Graphical Editing Framework (GEF). Interneto prieiga: http://www.eclipse.org/gef/, 2005.

[22] Eclipse. XML Schema Infoset Model (XSD). Interneto prieiga: http://www.eclipse.org/xsd/, 2005.

[23] Object Technology Internation, Inc. Eclipse Platform Technical Overview, 2003. Interneto prieiga: http://www.eclipse.org/whitepapers/eclipse-overview.pdf, 2005.

[24] W3C. Document Object Model *(DOM)*. Interneto prieiga: http://www.w3.org/DOM/, 2005.

[25] THEODORATOS Dimitri. Semantic Integration and Quering of Heterogeneous Data Sources Using a Hypergraph Data Model. *Visual Languages and Computing*, 6(1), 3–36 psl, 1995.

TERMINŲ IR SANTRUMPŲ ŽODYNAS

CASE – Kompiuterizuotas sistemų kūrimas (*angl.* Computer Aided System Engineering)

XML dokumentas – XML žymių kalbos taisyklėmis apibrėžtas dokumentas

XML schema – XML dokumentą apibrėžianti schema

JAVA – aukšto lygio kompiliuojama kalba, sukurta Sun Microsystems kompanijoje

PĮ – programinė įranga

DB – duomenų bazė (*angl.* Database*)*

OS – operacinė sistema (*angl.* Operating System)

CVS – lygiagrečių versijų sistema (*angl.* Concurrent Versioning System)

IS – informacinė sistema

IDE – integruota PĮ kūrimo aplinka (*angl.* Integrated Development Environment)

EER – išplėsta esybių ryšių diagrama (*angl.* Extended Entity Relationship Diagram)

API – programinė taikomosios programos sąsaja (*angl.* Application Programming Interface)

XSD – XML schemos modelio API (*angl.* XML Schema Infoset Model)

GUI – grafinė vartotojo sąsaja (*angl.* Graphical User Interface)

EMF – struktūrizuotų duomenų modelių API (*angl.* Eclipse Modelling Framework)

SWT – standartinis Eclipse Java GUI priemonių rinkinys (*angl.* Standard Widget Set)

DOM – dokumento objektų modelis – struktūrizuoto dokumento programinio redagavimo API (*angl.* Document Object Model)

PRIEDAI

1. PRIEDAS. Pavyzdžių bei eksperimentų XML ir XML schemos dokumentai

1. Konceptualaus XML medžio, pateikto 1.7 pav. XML dokumentas

```
<?xml version="1.0" encoding="utf-8"?>
<prekiu_sarasas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="pavyzdys.xsd">
      <preke nr="1" virtuali="true">
            <kodas>GK32</kodas>
            <pavadinimas>GK Televizorius 32</pavadinimas>
            <gamintojas>GK Kompanija</gamintojas>
            <rekomenduojama_kaina>500.00</rekomenduojama_kaina>
      </preke>
      <preke nr="2" virtuali="false">
            <kodas>SJ12</kodas>
            <pavadinimas>SJ Lygintuvas 12</pavadinimas>
            <gamintojas>SJ Kompanija</gamintojas>
            <rekomenduojama_kaina>50.00</rekomenduojama_kaina>
            <papildoma_informacija>
                  Rekomenduojama tik vidurio Lietuvos regionui.
            </papildoma_informacija>
      </preke>
</prekiu_sarasas>
```

2. Išskirtinio *isa* ryšio tipu susietų esybių XML schemos, pateiktos 2.5 pav. tekstinis dokumentas

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="A" type="A" />
      <xsd:element name="B" type="B" />
      <xsd:element name="C" type="C" />
      <xsd:complexType name="A">
            <xsd:sequence>
                  <xsd:element name="a" type="xsd:string" />
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="B">
            <xsd:complexContent>
                  <xsd:extension base="A">
                        <xsd:sequence>
                              <xsd:element name="b" type="xsd:string" />
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="C">
            <xsd:complexContent>
                  <xsd:extension base="A">
                        <xsd:sequence>
                              <xsd:element name="c" type="xsd:string" />
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:schema>
```

3. Tranzityviu *isa* ryšiu susietų esybių XML schemos, pateiktos 2.7 pav. tekstinis dokumentas

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="A" type="A" />
      <xsd:element name="B" type="B" />
      <xsd:element name="C" type="C" />
      <xsd:element name="D" type="D" />
      <xsd:element name="E" type="E" />
      <xsd:complexType name="A">
```

```
            <xsd:sequence>
                  <xsd:element name="a" type="xsd:string" />
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="B">
            <xsd:complexContent>
                  <xsd:extension base="A"></xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="C">
            <xsd:complexContent>
                  <xsd:extension base="A">
                        <xsd:sequence>
                              <xsd:element name="c" type="xsd:string" />
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="D">
            <xsd:complexContent>
                  <xsd:extension base="C"></xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="E">
            <xsd:complexContent>
                  <xsd:extension base="C">
                        <xsd:sequence>
                              <xsd:element name="e" type="xsd:string" />
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:schema>
```

4. OP objektų vidinių ryšių priklausomybės XML schemos, pateiktos 2.9 pav. tekstinis dokumentas

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="A" type="a"/>
      <xsd:element name="B" type="b"/>
      <xsd:complexType name="a">
            <xsd:sequence>
                  <xsd:element name="A_" type="a_"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="b">
            <xsd:sequence>
                  <xsd:element name="B_" type="b_"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="a_">
            <xsd:sequence>
                  <xsd:element name="A1" type="xsd:string"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="b_">
            <xsd:complexContent>
                  <xsd:extension base="a_">
                        <xsd:sequence>
                              <xsd:element name="B2" type="xsd:string"/>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:schema>
```

5. XML schemos pavaizduotos 4.19 pav. tekstinis dokumentas

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
      <xsd:complexType name="studentas">
            <xsd:complexContent>
                  <xsd:extension base="universiteto_narys">
                        <xsd:sequence>
                              <xsd:element name="stud_nr" type="xsd:string"></xsd:element>
                        </xsd:sequence>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
      <xsd:element name="studentas" type="studentas"></xsd:element>
      <xsd:complexType name="universiteto_narys">
            <xsd:sequence>
                  <xsd:element name="vardas" type="xsd:string"></xsd:element>
                  <xsd:element name="pavarde" type="xsd:string"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

6. XML schemos pavaizduotos 4.21 pav. tekstinis dokumentas

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="paskaita">
            <xsd:sequence>
                  <xsd:element name="ID" type="paskaita_ID">
                        </xsd:element>
                  <xsd:element name="destytojas" type="destytojas"></xsd:element>
                  <xsd:element name="disciplina" type="disciplina"></xsd:element>
                  <xsd:element name="grupe" type="grupe"
                        maxOccurs="unbounded" minOccurs="1"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="paskaita_ID">
            <xsd:sequence>
                  <xsd:element name="auditorija" type="xsd:string"></xsd:element>
                  <xsd:element name="laikas" type="xsd:dateTime"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="destytojas">
            <xsd:sequence>
                  <xsd:element name="vardas" type="xsd:string"></xsd:element>
                  <xsd:element name="pavarde" type="xsd:string"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="disciplina">
            <xsd:sequence>
                  <xsd:element name="ID" type="disciplina_ID"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="disciplina_ID">
            <xsd:sequence>
                  <xsd:element name="pavadinimas" type="xsd:string"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="grupe">
            <xsd:sequence>
                  <xsd:element name="paskaitos" type="xsd:string"
                        maxOccurs="unbounded" minOccurs="1"></xsd:element>
                  <xsd:element name="kuratorius" type="destytojas"
                        maxOccurs="1" minOccurs="0"></xsd:element>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="paskaita" type="paskaita"></xsd:element>
</xsd:schema>
```

2. PRIEDAS. Straipsnis. Objektų savybių modelio transformavimas į XML schemą

OBJEKTŲ SAVYBIŲ MODELIO TRANSFORMAVIMAS Į XML SCHEMĄ

Saulius Menkevičius, Bronius Paradauskas
Kauno technologijos universitetas
Informacijos sistemų katedra, Studentų 50-308

Pasiūlyta integruotų informacijos sistemų XML srautų schemas projektuoti panaudojant paskirstytų duomenų konceptualiąją *objektų savybių* schemą, kuri suformuojama informacijos sistemos funkcinių reikalavimų analizės fazėje ir kurioje be dinaminių savybių apibrėžiamos ir duomenų struktūrinės savybės. *Objektų savybių* modelio notacijoje apibrėžiami taikomosios srities objektų išoriniai apibendrinimo ir vidiniai kompoziciniai ryšiai, objektų privalomi ir neprivalomi, sudėtiniai ir paprastieji atributai. Straipsnyje pateikiamos puoselėjamo metodo taisyklės objektų savybių modelio dariniams transformuoti į išplėstinės žymių kalbos XML schemų dokumentus. Šioms taisyklėms keliami reikalavimai, kad sugeneruotose XML schemose būtų išsaugojama kuo daugiau originalaus objektų savybių modelio semantikos ir kad dokumentus būtų galima formuoti tiražuojamų reliacinių arba objektinių DBVS aplinkose.

1. Įvadas

OMG (Object Management Group) [1] specifikavo *modeliu pagrįstą architektūrą* (MDA – *angl.* model driven architecture), kurios tikslas yra integruoti įvairius žinomus IS analizės ir projektavimo metodus. Pasinaudojant MDA metodu analitikai puoselėja įvairiuose IS kūrimo etapuose skirtingus modelius, integruoja ir transformuoja juos, siekdami informaciją apie duomenų savybes, sukauptą vartotojų funkcinių reikalavimų analizės fazėje išsaugoti ir panaudoti duomenų bazių vientisumui palaikyti ir programiniams produktams validuoti. IS kūrime naudojami modeliai skirstomi į dvi klases :

- nuo skaičiavimų nepriklausančius (CIM – *angl.* computation independent model) ir nuo platformos nepriklausančius (PIM – *angl.* platform independent model) modelius;
- konkrečios platformos modelius (PSM – *angl.* platform specific model).

CIM [2] skirti verslo taisyklėms ir vartotojų reikalavimams nurodyti. Jie neapriboja pateikiamos informacijos išraiškos būdus ir nespecifikuoja skaičiavimo algoritmus. CIM aprašomi verslo procesai, aktorių sąveikos informacinis turinys ir duomenų pateikimo ir navigacijos reikalavimai, tačiau sąsajos techninės detalės nenurodomos. PIM [2] nepateikiama informacija apie konkrečią platformą ar technologiją (J2EE, CORBA, .Net). Tai formalus konceptualus modelis, atspindintis verslo procesų intencionalinį turinį. MDA pateikia sisteminį būdą kaip atvaizduoti PIM į vieną ar daugiau PSM. PSM [2] apima informaciją apie konkrečią technologiją ir susideda iš tai platformai būdingų elementų. PSM gali būti panaudotas generuoti ir testuoti programų kodą, žinučių formatus, bendradarbiavimo protokolus, fizines duomenų struktūras, DBVS konfigūracijos parametrus.

Straipsnyje [3] charakterizuota HDBRD (*angl.* - rapid development of hybrid database) metodo esmė. Čia verslo transakcijos vaizduojamos išplėstomis komunikacinių veiksmų kilpomis, kurios leidžia suteikti verslo transakcijoms pragmatinę motyvaciją bei aprašyti su transakcija susijusių duomenų statinius ir dinaminius apribojimus. Tipinėmis veiksmų kilpomis galima aprašyti naujus funkcinius reikalavimus, ištirti semantinį atitikimą tarp naujųjų funkcinių reikalavimų informacinio turinio ir palikuoninės reliacinės duomenų bazės (DB) schemos konceptų. Jei egzistuojančios sistemos duomenų konceptualiosios struktūros leidžia patenkinti tik dalį naujų reikalavimų, puoselėjamos heterogenines (hibridines) DB. Šiuolaikinės duomenų bazės nėra izoliuotos – nuolat keičiasi duomenimis tarp verslo subjektų institucijų ir bet kur, kur naudojamos paskirstytos IS. Kadangi dažnai tos sistemos yra heterogeniškos (t.y. tiesiogiai nesuderinamos, skirtingų gamintojų, veikia įvairiose OS ir pan.), buvo būtina surasti patogų duomenų apsikeitimo formatą, kuris būtų nepriklausomas nuo platformos ar DBVS. Tokiu formatu de-facto tapo XML (*angl.* Extensible Markup Language), patvirtintas W3C konsorciumo [4]. XML dokumentų struktūrai specifikuoti pradžioje buvo naudojamas DTD (*angl.* Document Type Definition), kuriam, kaip tinkamesnis analogas, buvo pasiūlyta XML Schema. XML Schemos patogesnės apdorojimui, be to jos taip pat yra standartiniai XML dokumentai.

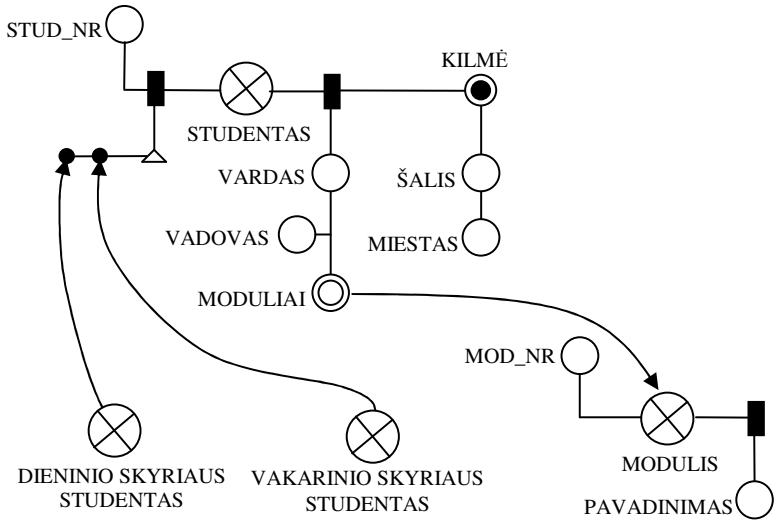
Objektų savybių (OP – *angl.* object property) modelis [5] savo semantinės išraiškos galimybėmis padengia duomenų priklausomybes, kurios yra specifikuojamos EER (*angl.* extended entity relationship) diagramomis, OO (*angl.* object-oriented) modeliais (ORM, UML) ir reliacinėmis schemomis. OP modelio pagrindinis privalumas yra galimybė transformuoti minėtus modelius vieną į kitą ir integruoti federacinių informacijos sistemų nehomogenines duomenų schemas. Šiame straipsnyje pasiūlyta duomenų srautų XML schemas generuoti iš duomenų konceptualiosios OP schemos, kai OP schemoje yra nurodyti "šakniniai" objektų tipai. XML hierarchijų viršūnėms priskiriami komunikacinių kilpų srautų vardai ir sutapatinami su OP modelio "šakniniais" konceptais. Srautų XML hierarchijos medis "auginamas" komunikacinių kilpų (arba naujų IS panaudojimo atvejų) detalizavimo procese [6].

Antrajame skyrelyje charakterizuota OP modelio grafinė notacija Objektų savybių modelį paprasta suvokti grafinėje notacijoje, juolab jis gali būti naudojamas projektuojant pačią informacinę sistemą, kurios duomenų srautus

norima modeliuoti. Trečiajame skyrelyje pateiktos transformavimo taisyklės, o ketvirtajame – tokių transformacijų pavyzdžiai.

2. Konceptualių esybių ir sąryšių notacijos

Objektų savybių modelio notacija privalo būti supaprastinta, kad atitiktų XML Schemos galimybes. Kadangi ji apjungia ir objektinių, ir reliacinių savybių, o XML Schemos aprašo duomenų semantiką, tokie OP notacijos elementai kaip esybės metodų prototipai, transformavus būti išsaugoti į XML Schemą.



1 pav. OP modelio schemos egzemplioriaus pavyzdys.

Aukščiau pateiktame paveiksle OP modelio diagramoje galima išskirti šiuos OP modelio elementus:

- apskritimai sukryžiuotomis linijomis – atitinka objektus (esybes); šalia jų nurodytas pavadinimas;
- tušti apskritimai – esybės atributus, su kuriomis jos susietos linijomis (jei apskritimas yra viršuje, kairiau nuo esybės sukryžiuoto apskritimo – tai identifikacinis atributas – pvz. MOD_NR yra MODULIS esybės identifikacinis atributas);
- apskritimai su užpildytu apskritimu viduje – sudėtiniai, jie susieti su žemiau esančiomis jo sudedamosiomis dalimis–atributais tiese;
- apskritimai su neužpildytu apskritimu viduje – daugiareikšmiai atributai;
- ryšiai tarp esybių nurodomi kreivėmis, prie jų taip pat nurodomas kardinalumas;
- potipių aibės nurodomos kairėje apačioje, šalia trikampio; galimos įvairios potipių kombinacijos.

XML Schemos dokumentai aprašo konkretų XML dokumento tipą, nurodo apribojimus šio tipo dokumento duomenims, pateikia žymes bei atributus, naudojamas to tipo dokumentuose, sąryšius tarp šių XML dokumento elementų. Pagrindiniai XML schemos elementai yra:

- *tipai* – aprašo konkrečius duomenų tipus, nurodant jų vidinę elementinę struktūrą; numatytas tipų paveldėjimas naudojantis paveldimo tipo išplėtimu bei apribojimu; šie tipai gali būti paprasti ir sudėtiniai;
- *elementai*, *atributai* – elementai yra naudojami apibrėžti duomenų esybes XML dokumentuose, tarnauja kaip XML dokumento duomenų meta informacija; elementai dokumente išdėstomi hierarchiškai, iš vieno šakninio elemento; atributai gali būti naudojami papildyti jiems priskirto elemento turinį ir privalo būti paprasto tipo (negali turėti kitų elementų ar atributų savo sudėtyje);
- *vardų erdvės* (*angl.* namespace) yra naudojamos apriboti ir įvardinti XML Schemos elementų grupę; naudojant vardų erdves tampa įmanoma į XML Schemos aprašą įjungti kitus du ar daugiau XML Schemos aprašus, kai juose naudojamų tipų ir elementų pavadinimai yra jau vartojami kituose;
- apribojimų ir nuorodų į raktus užtikrinimo mechanizmai: XML Schemoje yra galimybė išlaikyti elementų ir atributų (ar jų grupės) unikalumą ar raktų nuorodų vientisumą tam tikrame XML dokumento medžio lygyje;
- kitos priemonės, leidžiančios apriboti tipų panaudojimą, deklaruoti schemos anotacijas (dokumentaciją), išorinių schemų panaudojimą kitos schemos sudaryme ir kt.

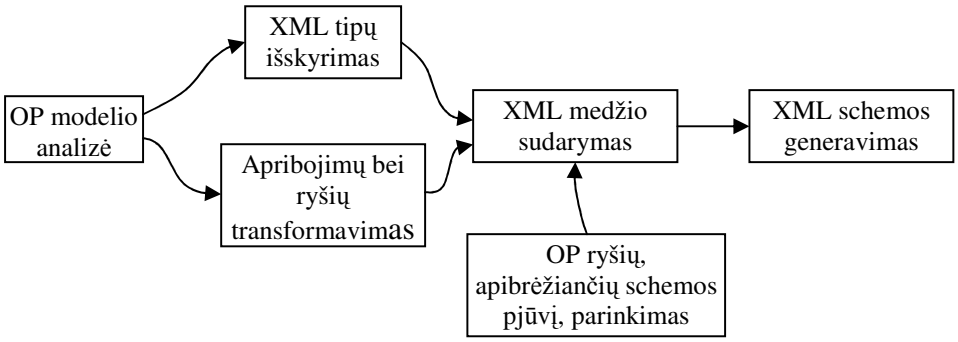
3. OP modelio transformavimas į XML Schemą

XML Schemos generavimui reikalingas metodas kuris apdorotų OP notacijos aprašą ir sugeneruotų tam tikro OP schemos pjūvio schemą. Tam tikroje metodo vietoje yra reikalingas vartotojo patikslinimas, kad nurodyti OP modelio pjūvio apimtį ir konfigūraciją. Metodu aprašomas transformacijos procesas yra suskirstomas į šias dalis:

OP notacijos objektų bei atributų transformavimas į XML elementus. Ši dalis aprašoma taisyklėmis, kurios gali būti vykdomos procedūriškai, be metodo vartotojo įsikišimo. Objektai keičiami XML tipais, objektų atributai keičiami elementais tipuose. Vidiniai sąryšiai transformuojami į sudėtinius tipus, turinčius vidinio sąryšio atributus ir yra įkeliami į juos turintį tipą. Daugiareikšmiai bei nebūtini atributai keičiami įprastai elementais, nurodant jų kardinalumą. Taip pat šalinami objektų elgsenos metodų aprašai – prototipai.

OP apribojimų bei ryšių realizavimas. XML Schema leidžia įvesti papildomus unikalumo bei raktų nuorodų apribojimus. Šios galimybės nagrinėjamoje XML Schemos versijoje (1.0) yra nepakankamai išplėtotos todėl nepakankamos, kad užtikrinti visų ryšių perkėlimą iš OP į XML Schemą. Naudojant <unique> elementą yra užtikrinamas objektų raktinių laukų unikalumas konkrečiame XML schemos lygyje. OP tipų apibendrinimas yra perteikiamas naudojant <extension> elementą, kuriame nurodamas paveldimas tipas. Kuomet į XML schemą reikia įterpti paveldimą tipą, ir potipių paveldėjimas yra nepersidengiantis, tipų alternatyva toje vietoje yra realizuojama XML pakaitos elementų grupe (*angl*. substitution group).

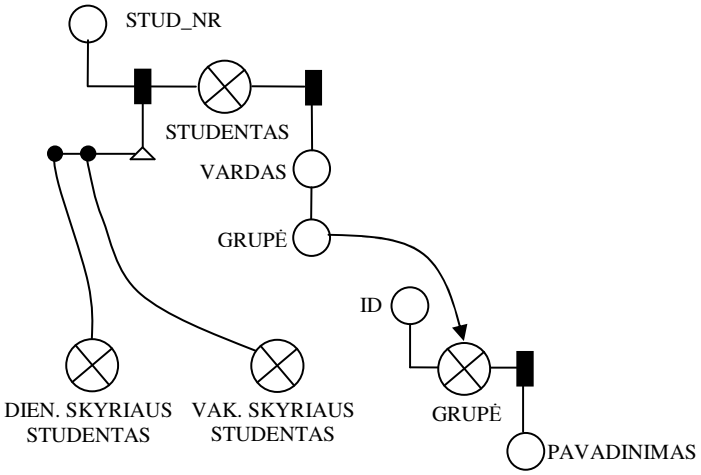
XML medžio sudarymas. Iš OP modelio vartotojas išrenka objektų ryšių medį, kuris aprašo norimą pjūvį iš OP schema realizuotos duomenų bazės. Objektai yra keičiami jų XML elementais su konkrečiais tipais. Kiekvienas ryšio elementas, kuris nurodytas apibrėžiant schemos pjūvį yra nagrinėjamas kaip tėvo–vaiko ryšys XML schemoje, pradedant nuo šakninio. Priklausomai nuo ryšio tipo bei kardinalumo, tėvinio elemento apraše schemoje gali būti nurodomas vaiko elemento pasikartojimas. Tokiu būdu realizavus OP pjūvį gausime įprastą XML dokumento reikalavimus atitinkantį medį.



2 pav. OP modelio transformavimo į XML Schemą procesai

4. Transformavimo pavyzdys

Pateikiame OP modelį, kuriam sukursim atitinkamą XML Schemą, norėdami gauti vienos ar kelių grupių studentų sąrašą.



3 pav. Transformuojamas OP modelis.

Numatydami, kad šakninis elementas generuojamam medžiui bus GRUPĖ, parenkame paprastą medį iš šakninio GRUPĖ tipo ir po jo sekančio lapo – STUDENTAS tipo. Tuomet gauname sekančią XML schemą:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns="http://pavyzdys.org/schema"
       targetNamespace="http://pavyzdys.org/schema"
       elementFormDefault="qualified">

 <xs:element name="ROOT">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="GRUPE" type="GRUPE" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
  <xs:unique name="GRUPE_STUDENTAS_STUD_NR">
   <xs:selector xpath="STUDENTAS"/>
   <xs:field xpath="STUD_NR"/>
  </xs:unique>
 </xs:element>

 <xs:complexType name="STUDENTAS">
  <xs:sequence>
   <xs:element name="STUD_NR" type="xs:integer"/>
   <xs:element name="VARDAS" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:element name="STUDENTAS" type="STUDENTAS" abstract="true"/>

 <xs:complexType name="DIEN_STUDENTAS">
  <xs:complexContent>
   <xs:extension base="STUDENTAS"/>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="DIEN_STUDENTAS" type="DIEN_STUDENTAS" substitutionGroup="STUDENTAS"/>

 <xs:complexType name="VAK_STUDENTAS">
  <xs:complexContent>
   <xs:extension base="STUDENTAS"/>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="VAK_STUDENTAS" type="VAK_STUDENTAS" substitutionGroup="STUDENTAS"/>

 <xs:complexType name="GRUPE">
  <xs:sequence>
   <xs:element name="PAVADINIMAS" type="xs:string"/>
   <xs:element ref="STUDENTAS" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Sugeneruotas OP modelio XML Schemos atitikmuo.

5. Išvados

Išnagrinėjus OP modelį bei XML schemą nustatyta, kad galima atrinkti tokį OP modelio elementų poaibį, kuriuo sudarytas OP modelio schemas galima transformuoti į struktūrinio tipo XML dokumento medžio modelį išlaikant originalios schemos apribojimus, tiesa, ne visus. Nurodytos būtinos OP notacijos elementų transformacijos ir šio proceso eiga. Kadangi OP modelis yra objektinis, nurodantis ir duomenų struktūrą ir elgseną (metodus), o XML – struktūrinis, todėl nėra galimybės į XML Schemą išsaugoti originalaus OP modelio schemos objektų metodų ir kitos papildomos informacijos. Dėl XML Schemos trūkumų ne visi apribojimai iš OP modelio gali būti perkelti į XML Schemą.

Pagal šį formalizuotą metodą buvo atliktos eksperimentinės transformacijos, OP modelio schemą paverčiant į XML schemos atitikmenį ir atitinkamai sugeneruoti XML dokumentus, atitinkančius šią schemą bei nukrypstančius nuo jos. XML dokumento atitikimo jo XML schemai testavimas buvo atliekamas Microsoft .NET platformos XML dokumentų ir schemų programinio apdorojimo bibliotekomis ir priemonėmis.

Naujesnės Microsoft SQL Server versijos bei kitos populiarios DBVS turi priemones duomenų užklausų rezultatus išvesti XML formate. Šie užklausų sugeneruoti dokumentų tipai gali būti apibrėžiami straipsnyje apibrėžto metodo sugeneruota schema. OP modelis, kuris aprašo informacinės sistemos (IS) taikomosios aplinkos objektus ir jų sąryšius, transformuotas į XML schemą skirtingais pjūviais, papildo IS naudojamų duomenų aprašų rinkinį, kurie naudojami deklaruojant informacijos srautus tarp heterogeninių informacijos sistemų.

6. Literatūros sąrašas

[1] Harmon P. Business Processes and the OMG: Overview//*OMG*. 2004. URL: http://www.omg.org/bp-corner/introduction.htm

[2] Hendryx S. **A**. A Home for Business Models in the OMG//*Business Rules Journal*. 2003, Nr. 1. URL: http://www.brcommunity.com/b127.php

[3] Nemuraitė L., Paradauskas B., Salelionis L. Extended Communicative Action Loop for Integration of New Functional Requirements // Information technology and control. ISSN 1392-124X. Kaunas: Technologija, 2002, No. 2(23), p. 18 - 26.

[4] Extensible Markup Language (XML). URL: http://www.w3.org/XML/

[5] Paradauskas B., Nemuraitė L. Duomenų bazės ir semantiniai modeliai. Technologija, Kaunas, 2002, 264 p.

[6] Paradauskas B. Nemuraitė L. From Use Cases to Well Structured Conceptual Schemas // Information Systems Development: Advances in Theory, Practice and Education, edited by O.Vasilecas et al., Springer, 2005 (to be published).

[7] XML Schema. URL: http://www.w3.org/XML/Schema

TRANSFORMATION OF OBJECT PROPERTY MODEL INTO XML SCHEMA

Saulius Menkevičius, Bronius Paradauskas

A proposal to create XML schemas used in integrated information systems using distributed data conceptual *object properties* schema, the instance of which is formed during analysis of the functional requirements for information system and which is capable of defining both dynamic and structural properties for application domain data. The notation of object properties model can be used to define both outer type derivations and internal compositional relations of the objects in the domain, required and optional, complex and simple attributes.

The article provides with rules of the given method, which can be used for transformation of object properties model instances into corresponding XML schema documents. There are additional requirements for these rules so that the resulting XML schema would contain as much semantics of the original object property model as possible, and that the schema could be used in concert with environments of the common objective and relational DBMS.