KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Marius Bindokas, Aidas Kasperavičius

Požymių modelių kūrimas ir transformacijos

Magistro darbas

Darbo vadovas

Dr. R. Damaševičius

Kaunas, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Marius Bindokas, Aidas Kasperavičius

Požymių modelių kūrimas ir transformacijos

Magistro darbas

Recenzentas

Prof. R. Butleris

2012-05-

Darbo vadovas

Dr. R. Damaševičius

2012-05-

Atliko

IFM-0/4 gr. stud.

Marius Bindokas

2012-05-25

IFM-0/2 gr. stud.

Aidas Kasperavičius

2012-05-25

Kaunas, 2012

Turinys

LENTELIŲ TURINYS

PAVEIKSLŲ TURINYS

SANTRAUKA

Šiame darbe sprendžiamas vis aktualesnis, pakartotinis programų panaudojamumo klausimas. Tačiau dažniausiai jau sukurtą programą pakartotinai panaudoti arba būna labai sunku, arba išvis neįmanoma. Todėl programinę įrangą, kurią norima arba galvojama pakartotinai panaudoti atskiriems egzemplioriams kurti, patogiausia būtų kurti naudojant programų šeimos idėją.

Kad sukurti programų šeimą reikia sukurti tos srities, kurioje bus naudojama programų šeima, metamodelį. Tam tikslui yra keletas kalbų ir keletas įrankių. Šiame darbe pasirinkome požymių modelių tyrimui naudoti jau sukurtą įrankį, taip pat papildyti jį transformavimu, tarp skirtingais būdais aprašytų požymių modelių, bei šias transformacijas ištirti.

Darbo tikslas – ištirti programų variantiškumo modelių, specifikuotų požymių diagramomis, transformacijas į įvairius srities modelius.

SUMMARY

In this work we deal with programs reusability question, which becomes more and more relevant as time passes. It is very difficult to reuse program, or sometimes even impossible. Therefore, the software wanted to be reused or planned to create instances is best placed to develop using the idea of product line.

Metamodel for specific scope needs to be created, to create a product line. For this purpose, there are several languages and few tools. In this paper we choose to use an existing feature modeling tool and investigate feature modeling. We also implement the transformations of feature models to this tool and investigate them.

Main goal of this work is to investigate program variability models and transformations between them.

1 ĮVADAS

Organizuojant didelio masto programinės įrangos gamybą, kai turima daug užsakovų, kurių reikalavimai galutiniam produktui skiriasi, dažnai naudojama produktų linijų metodologija [1]. Produktų variantams modeliuoti kuriami požymių (užsakovui svarbių sistemos charakteristikų) modeliai, kurie aprašomi požymių diagramomis. Apsirašius kuriamos programos srities metamodelį transformavimo pagalba galima greičiau sukurti produktą, negu atliekant visus projektavimo darbus nuo pradžių. Pagrindinis transformacijų tikslas yra kuo našiau ir efektyviau sukurti naują sistemą automatizuojant sistemos projektavimo uždavinius, taip pat pasiekti didesnį suderinamumą su kitais įrankiais ir turėti galimybę panaudoti kitais formatais aprašytus požymių modelius.

1.1 Darbo tikslas ir uždaviniai

Darbo tikslas – ištirti programų variantiškumo modelių, specifikuotų požymių diagramomis, transformacijas į įvairius srities modelius.

Uždaviniai:

1. Pasiūlyti požymių modelių metamodelį ir pateikti jo formalų aprašą.
2. Išanalizuoti ir realizuoti požymių modelių įvairaus lygmens transformacijas:
    a. Požymių modelis → požymių modelis (homogeninė transformacija);
    b. Požymių modelis → kitas srities modelis (heterogeninė transformacija: kito srities modelio, pvz., UML klasių diagramos, konfigūravimas).
3. Ištirti realizuotų požymių modelių transformacijų savybes.
4. Atlikti realizuotų požymių modelių transformacijų tyrimą su realia sistema.

1.2 Temos tikslingumas ir aktualumas

Labai dažnai susiduriama su poreikiu sukurti kelias skirtingas programinės įrangos versijas, užsakomas skirtingų užsakovų su skirtingais reikalavimais, turinčias skirtingą funkcionalumą, bet kartu ir besidalijančias tam tikra aibe bazinio funkcionalumo. Iš pirmo žvilgsnio tai gali pasirodyti gana trivialus uždavinys, tačiau praktika rodo, kad programinės įrangos sistemoms plečiantis ir kylant poreikiui išleisti vis daugiau skirtingų sistemos variantų, ši užduotis tampa gana kebli. Automatizuojant šios problemos sprendimo procesą galima sutaupyti nemažai laiko ir lėšų.

2 TERMINŲ IR SANTRUMPŲ ŽODYNAS

- ATL (*ATLAS Trasformation Language*) – modelių transformavimo kalba.
- BNP (*Backus-Naur Form*) – metakalba formalios kalbos sintaksei apibrėžti.
- BPMN (*Business Process Model Notation*) – verslo procesų modelių notacija.
- CASE (*Computer-Aided Software Engineering*) – automatizuotas kompiuterinis programinės įrangos projektavimas.
- EMF (*Eclipse Modeling Framework*) – Eclipse platformos modeliavimo karkasas.
- EPL (*Eclipse Public License*) – Eclipse programos kūrėjų sukurta viešam naudojimui skirta licencija. Pagal šią licenciją kuriamos atviro kodo programos.
- FD2 – sukurto įrankio požymių modelio saugojimo formatas paremtas XML.
- FDL (*Feature Description Language*) – požymių aprašymo kalba.
- FODA (*Feature-Orented Domain Analysis*) – požymiais paremta srities analizė.
- IDE (*Integrated Development Environment*) – integruota kūrimo aplinka – programa, turinti daug priemonių, palengvinančių ir pagreitinančių programinės įrangos kūrimo procesą.
- JRE (*Java Runtime Environment*) – programavimo kalba *Java* parašytų programų vykdymo aplinka.
- LKC (*Linux Kernel Configurator*) – Linux OS branduolio konfigūravimo priemonė, turinti savo specifinę konfigūravimo kalbą.
- Metakalba – kalba, skirta kitoms kalboms aprašyti.
- Metamodelis – modelis, aprašantis kitą modelį.
- OCL (*Object Constraint Language*) – deklaratyvi ribojimų kalba, skirta aprašyti taisykles taikomas UML modeliams.
- OWL (*Web Ontology Language*) – OWL yra XML grįsta kalba, kuri remiasi RDF metaduomenų atvaizdavimo formatu.
- Požymis (angl. *feature*) – tam tikra programinės įrangos vieneto skiriamoji charakteristika.
- Požymių diagrama (angl. *feature diagram*) – grafinių modeliavimo kalba, naudojama programinės įrangos produktų linijos reikalavimams aprašyti.
- Požymių modelis (angl. *feature model*) – glaustas programinės įrangos produktų linijos visų produktų požymių aprašas.
- Produktų linija (angl. *software product line* arba *SPL*) – susijusių programinės įrangos produktų, susidedančių iš bendrų komponentų, klasių, ar kitų savybių, aibė.
- PROLOG – bendro naudojimo loginė programavimo kalba.

- RDF (*Resource Description Framework*) – meta duomenų apdorojimo pagrindas, skirtas suderinti informacijos srautams tarp programų, kurios internete keičiasi informacija.
- SXFM (*Simple XML Feature Model*) – XML grįstas požymių atvaizdavimo formatas.
- Tefkat – modelių transformavimo kalba ir taip pat besivadinantis modelių transformavimo variklis.
- TVL – Boucher ir kt. pasiūlyta kalba požymių modeliams aprašyti.
- UML (*Unified Modeling Language*) – unifikuota modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.
- XMI (*XML Metadata Interchange*) – standartas metaduomenų keitimuisi tarp programų, panaudojant XML kalbą.
- XML (*eXtensible Markup Language*) – W3C kompanijos rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

3 ĮŽANGA Į POŽYMIŲ MODELIAVIMO SRITĮ

3.1 Programinės įrangos produktų linija

Programinės įrangos produktų linija (angl. *software product line* arba *SPL*) – tai susijusių programinės įrangos produktų, susidedančių iš bendrų komponentų, klasių, ar kitų savybių, aibė [1]. Programinės įrangos produktai iš vienos produktų linijos pasižymi bendru funkcionalumu, tačiau taip pat turi ir savitų, unikalių savybių.

Programinės įrangos produktų linija pagrįstas kūrimas orientuojasi į produktų linijos atskirų produktų besiskiriančius reikalavimus, funkcionalumą.

3.2 Variantiškumo valdymas

Variantiškumo valdymas (angl. *variability management*) – tai esminė programinės įrangos produktų linija pagrįsto kūrimo dalis išskirianti šį programinės įrangos kūrimo būdą iš kitų [2]. Variantiškumo valdymo esmė – problemų, susijusių su efektyviu skirtingų produktų linijos produktų vystymu.

3.3 Požymis

Požymis (angl. *feature*) yra tam tikra programinės įrangos vieneto skiriamoji charakteristika [3]. Požymis pradžioje būna apibrėžiamas reikalavimais ir programinėje įrangoje atsispindi tam tikru funkcionalumu ar kokiu kitu nefunkciniais reikalavimai nustatytu požymiu. Būtent šiais požymiais dalijasi ir/arba išsiskiria atskiri programinės įrangos produktų linijos produktai [4].

3.4 Požymių modelis

Požymių modelis (angl. *feature model*) – tai glaustas programinės įrangos produktų linijos visų produktų požymių aprašas. Požymių modelis paprastai įsivaizduojamas kaip požymių medis [5]. Požymių modeliai vizualiai atvaizduojami požymių diagramomis. Požymių modeliai yra naudojami visu programinės įrangos produktų linijos kūrimo proceso metu.

3.5 Požymių diagramos

Požymių diagramos (angl. *feature diagrams*) – tai grafinių modeliavimo kalbų šeima, naudojama programinės įrangos produktų linijos reikalavimams aprašyti [6]. Požymių diagramas pirmą kartą pristatė K. Kang 1990 metais kaip FODA (angl. *Feature Oriented Doamin Analysis*)

metodo dalį [7]. Nuo tada buvo sukurta įvairių praplėtimų ir modifikacijų FODA požymių diagramoms, tam kad būtų galima išvengti dviprasmybių, pagerinti tikslumą ir išraiškingumą.

Požymių diagrama yra vizualus požymių modelio aprašymas. Požymių diagramos, kuriomis aprašomi požymių modeliai neturi vieningo standarto. Yra nemažai įvairių notacijų modifikacijų, tačiau pagrindinės, bazinės savybės išlieka tos pačios.

Požymių diagrama yra medžio tipo arba tiesinis, aciklinis grafas, kurį sudaro rinkinys viršūnių, tiesinių briaunų ir briaunų kombinacijų. Šakninis elementas apibūdina aukščiausią požymio lygį (pvz. sistema, sritis). Vidutinės viršūnės apibūdina sudėtinius požymius, paskutinės – atominius požymius, kurie nėra skaidomi į mažesnius požymius duotoje srityje. Briaunos naudojamos palaipsniui suskaidyti sudėtinius požymius į labiau detalizuotus požymius. Grafo briaunos nurodo ryšius arba priklausomybes tarp požymių [8].

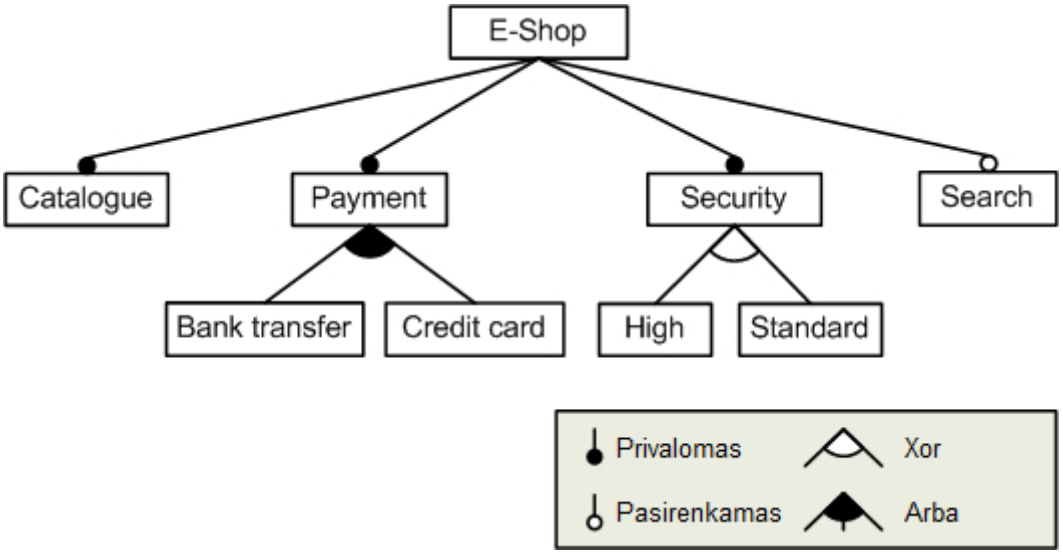
Pagrindiniai ryšiai tarp tėvo ir vaiko požymių gali būti tokie [9]:

- Privalomus – vaiko požymis yra privalomas tėvo požymiui;
- Pasirenkamus – vaiko požymis yra pasirenkamas tėvo požymiui – jo gali ir nebūti;
- „Arba“ tipo – bent vienas vaiko požymis turi būti pasirinktas;
- „Xor“ tipo – lygiai vienas vaiko požymis turi būti pasirinktas.

Papildomai, be šių tėvo-vaiko požymių ryšių gali būti ryšiai tarp bet kurių kitų dviejų požymių, kurie nurodo kad:

- A reikalauja B – požymis A reikalauja, kad būtų pasirinktas ir požymis B;
- A prieštarauja B – požymiai A ir B negali būti abu viename produkte.

Pavyzdinė požymių diagrama pateikiama 1 paveiksle.



1 pav. Pavyzdinė požymių diagrama

3.6 Tekstinės požymių modelių specifikavimo kalbos

Grafinių požymių modelių aprašymo kalbų trūkumus ir tekstinių kalbų pranašumus suformulavo Boucher ir kt. [10]. Grafinis aprašas lengviau suprantamas žmonėms be techninių žinių, tačiau dirbant su didelėmis požymių diagramomis išryškėja šios notacijos trūkumai: požymių modelio medis užima didelę erdvę fiziškai, ryšiai tarp požymių persipina tarpusavyje, todėl darosi nepatogu skaityti ir kurti tokias diagramas. Dėl šių priežasčių išaugo tekstinių požymių modelių specifikavimo kalbų poreikis.

Tekstinė požymių modelių specifikavimo kalba – tai kalba, kuria galima aprašyti požymių modelį grynu tekstu. Toks požymių modelio aprašymo būdas leidžia glausčiau aprašyti požymių modelius, taip pat suteikia galimybę įvairiems įrankiams keistis tarpusavyje požymių modelių informacija, galiausiai suteikia galimybę naudoti įprastines, grynajam tekstui pritaikytas priemones, tokias kaip teksto redaktorius ar versijų kontrolės sistemas.

Šiuo metu yra daug neatitikimų tarp įvairių modeliavimo programų naudojamų saugojimo formatų. Sistemos negali apsikeisti duomenimis, nėra būdo, kaip perkelti meta modelius iš vienos sistemos į kitą. Tekstinis diagramų saugojimo formatas yra vienintelis būdas, leidžiantis perduoti meta duomenis iš vienos sistemos į kitą. Todėl kuriama požymių modeliavimo programinė įranga būtinai turi specifikuoti požymių diagramas tekstiniu formatu.

3.7 Modelių transformacija

Transformacija yra fundamentalus informatikos ir programinės įrangos inžinerijos dalykas. Bet koks skaičiavimas tam tikra prasme yra duomenų transformacija. Modelių transformacijos yra glaudžiai susijusios su programų transformavimu. Iš tiesų nėra aiškios ribos tarp šių dviejų transformacijų būdų ir iš dalies jie persidengia. Tačiau programų transformacijos paprastai būna paremtos matematinėmis koncepcijomis, o modelių transformacijos remiasi objektiškai orientuotais modeliais [11].

Grafų transformacijos gali būti naudojamos modelių transformacijoms išreikšti, todėl daug tam skirtų programinių įrankių naudoja šią techniką kaip vidinį savo transformavimo variklio mechanizmą. Ypač vizualių modelių transformacijos gali būti natūraliai suformuluotos pagal grafų transformacijas, kadangi grafai puikiai tinka aprašyti modelių struktūroms [12].

4 LITERATŪROS APŽVALGA

4.1 Požymių modeliavimo įrankiai

Yra sukurta nemažai požymių modeliavimo įrankių, tačiau nėra vieningo požymių modelių standarto, todėl visi šie įrankiai gana skirtingi, turintys savų privalumų ir trūkumų. Šiame skyriuje, 1 lentelėje pateikiama populiariausių požymių modeliavimo įrankių apžvalga.

1 lentelė. Požymių modeliavimo įrankiai

Požymių modeliavimo įrankis	**Privalumai**	**Trūkumai**
Feature IDE [13]	• Nemokamas (atviro kodo) • XMI duomenų failas • diagramos klaidų tikrinimas • programinio kodo kūrimas	• Naudojama tik 1 grafine notacija • Neįlieta į kitas programas • Nesusieta su UML

Xfeature [14]	• Nemokamas (atviro kodo) • XMI duomenų failas • diagramos klaidų tikrinimas • programinio kodo kūrimas • Galimos skirtingos diagramų sintaksės • Eclipse programos priedas	• Nesusieta su UML
Ecore / Feature Modelling Plug-in [15]	• Nemokamas (atviro kodo) • XMI duomenų failas • diagramos klaidų tikrinimas • programinio kodo kūrimas • Eclipse programos priedas	• Diagramos vaizduojamos kaip išskleidžiamas medis. • Nesusieta su UML
Requiline [16]	• Nemokamas (atviro kodo) • XMI duomenų failas • diagramos klaidų tikrinimas	• nenumatytas programos kodo kūrimas • Neįlieta į kitas programas • Nesusieta su UML
FAMA [17]	• Nemokamas (atviro kodo) • XMI duomenų failas • diagramos klaidų tikrinimas • programinio kodo kūrimas • Eclipse programos priedas	• Nesusieta su UML
KumbangTools [18]	• Nemokamas (atviro kodo) • XMI duomenų failas • diagramos klaidų tikrinimas • programinio kodo kūrimas • Eclipse programos priedas	• Nesusieta su UML
pure::variants [19]	• XMI duomenų failas • diagramos klaidų tikrinimas • programinio kodo kūrimas • Eclipse programos priedas	• Komercinė licencija • Nesusieta su UML

4.2 Požymių modelių transformavimo priemonės

Įvairioms požymių modelių transformacijoms yra sukurta nemažai priemonių. Keletas jų apžvelgiama šiame skyriuje.

4.2.1 ATLAS transformavimo kalba

ATLAS transformavimo kalba arba tiesiog ATL (angl. *ATLAS Trasformation Language*) – tai modelių transformavimo kalba ir priemonių rinkinys kuriamas ir palaikomas OBEO ir INRIA. ATL išleista pagal Eclipse viešąją licenciją. Šiai kalbai yra sukurtas M2M (angl. *Model-to-Model*) Eclipse platformos komponentas – EMP (angl. *Eclipse Modeling Project*) projekto dalis.

ATL leidžia transformuoti įvairius turimus modelius į daugelį kitų. Apskritai ši kalba nėra pritaikyta būtent požymių modelių transformavimui. Ji yra kur kas bendresnės paskirties, tačiau yra galimybė transformuoti ir požymių modelius. Straipsnyje „Improving the Design of Highly Variant-Rich Business Process Models“ [20] nagrinėjamas ir aprašomas ATL priemonėmis atliekamas požymių modelių transformavimas į verslo procesų modelių notaciją – BPMN (angl. *Business Process Model Notation*).

4.2.2 Tefkat

Tefkat – tai modelių transformavimo kalba ir kartu modelių transformavimo variklis. Ši kalba paremta F-logika. Tefkat modelių transformavimo variklis pritaikytas Eclipse platformai kaip Eclipse Modeling Framework (EMF) įskiepis (angl. plug-in).

Tefkat apibrėžia sąsają tarp pradinių ir galutinių metamodelių rinkinių. Tefkat transformacija susideda iš taisyklių, modelių ir šablonų. Taisyklės susideda iš pradinės sąlygos ir galutinės sąlygos [21].

4.2.3 Požymių modelių transformavimo priemonių apžvalga

Yra gana daug labai įvairių modelių transformavimo kalbų ir priemonių, tačiau daugelis jų yra skirti abstraktesniems modeliams transformuoti. Nė vienas išbandytas įrankis nebuvo pritaikytas būtent požymių modelių transformacijoms.

4.3 Požymių modelių specifikavimo kalbos

4.3.1 FDL

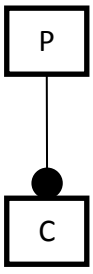
FDL (angl. *Feature Description Language*) yra kalba, kuria siekiama paprastai ir suprantamai aprašyti požymių modelius.

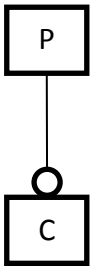
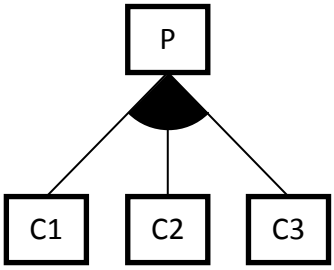
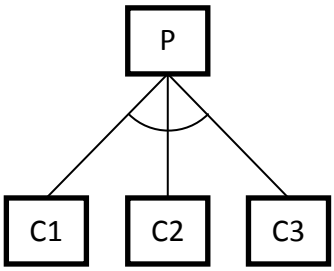
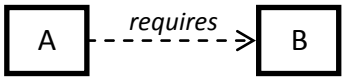
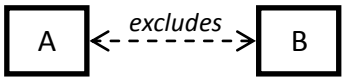
FDL kalba susideda iš tokių elementų [22]:

- Atominis požymis – tai požymis, kuris pats kitų požymių neturi ir priklauso kitam požymiui;
- Sudėtinis požymis – tai vardą turintis požymis, kuris susideda iš kitų požymių;
- Pasirenkamas požymis – tai požymis, kuris gali būti pasirenkamas, išskiriamas simboliu „?“;
- Privalomi požymiai – tai požymių rinkinys aprašytas raktiniu žodžiu *all (požymis1, požymis2…)*;
- Alternatyvūs požymiai – tai vieno požymio pasirinkimas iš aibės, apibrėžtos *one-of (požymis1, požymis2…)*;
- Vienas ir daugiau požymių – tai vieno arba kelių požymių pasirinkimas iš aibės apibrėžtos *more-of (požymis1, požymis2…)*;
- Nustatyta požymio reikšmė – *default*;
- Aprašius visus požymius aprašomi tam tikri apribojimai požymiams:
    - *požymis1 requires požymis2* – nurodo, kad pasirinkus pirmąjį požymį būtina pasirinkti ir antrąjį;
    - *požymis1 excludes požymis2* – nurodo, kad pasirinkus pirmąjį požymį, antrasis požymis nebegali būti pasirinktas.

Lentelėje 2 pateikiamos pagrindinės požymių modelių konstrukcijos aprašytos požymių modelių diagramomis bei FDL kalba.

2 lentelė. Pagrindinės požymių modelių konstrukcijos FDL kalba

Konstrukcija	Požymių diagrama	FDL
Privalomas		`P: all(C)`

Pasirenkamas		`P: all(?C)`
Vienas ir daugiau („*arba*“)		`P: more-of(C1, C2, C3)`
Alternatyva („*case*“)		`P: one-of(C1, C2, C3)`
Reikalavimas pasirinkti („*requires*“)		`A requires B`
Reikalavimas nesirinkti („*excludes*“)		`A excludes B`

4.3.2 OWL

OWL (angl. *Web Ontology Language*) yra vieningas žinių apie tai, kas egzistuoja realiame pasaulyje, pateikimo internete formatas. OWL plačiai naudojama žinių inžinerijoje, kuriant semantinį tinklą. OWL yra XML grįsta kalba, kuri remiasi RDF (angl. *Resource Description Framework*) metaduomenų atvaizdavimo formatu.

Požymių modelių panašumą į OWL kalba aprašomas ontologijas pirmieji pastebėjo Czarnecki ir kt. [23], o Wang ir kt. [24] pasiūlė konvertuoti požymių modelius į OWL, kad būtų galima atlikti jų semantikos verifikavimą naudojant standartinius ontologijų analizės įrankius.

Specifikuojant požymių modelius OWL kalba, kiekvienas požymis specifikuojamas kaip OWL klasė. Privalomi požymiai yra *MandatoryFeature* klasės išvestinės klasės, o pasirenkami požymiai yra *OptionalFeature* klasės išvestinės klasės. Bazinė *MandatoryFeature* ir *OprionalFeature* klasių klasė yra *Feature*. Požymių grupavimas žymimas naudojant <owl:unionOf> konstrukciją. Apribojimas *excludes* specifikuojamas naudojant <owl:complementOf> konstrukciją, o *requires* – naudojant <owl:equivalentClass> konstrukciją. Apribojimais nesusietiems požymiams žymėti naudojama <owl:disjointWith> konstrukcija.

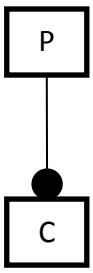
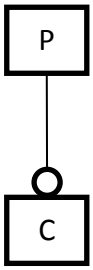
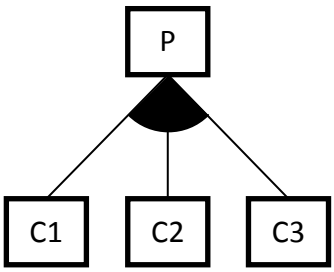
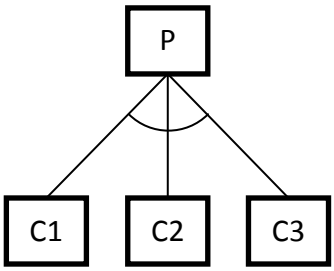
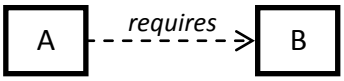
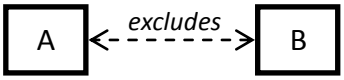
Požymių modelių saugojimo OWL formatu pranašumai yra galimybė atlikti požymių modeliavimą naudojant daugybę žinių modeliavimui sukurtų įrankių (pvz. *Protege*, *NeOn Toolkit*), o taip pat SPARQL užklausų programavimo galimybė naudojant, pvz., Jena paketą.

4.3.3 SXFM

SXFM (angl. *Simple XML Feature Model*) yra XML grįstas požymių atvaizdavimo formatas [25]. SXFM formatą šiuo metu naudoja keli požymių modeliavimo įrankiai, kaip *SPLOT* ir *4WhatReason*. Šios kalbos trūkumas yra tai, kad kalbos autoriai nepateikė formalaus sintaksės aprašymo, o tik tos kalbos failų sintaksinį analizatorių. XML formatu specifikuojami tik požymių modelio metaduomenys. Pats požymių modelis neužrašomas XML formatu, o koduojamas naudojant specialią sintaksę, kurios elementų pavyzdžiai pateikti žemiau esančioje lentelėje 3. Šios sintaksės kodas atitinkamai dedamas tarp XML žymų:

- <*feature_tree*> – jei kodas skirtas aprašyti požymių modelio medžio struktūrą;
- <*constraints*> – jei kodas skirtas aprašyti papildomus apribojimus tarp požymių.

3 lentelė. Pagrindinės požymių modelių konstrukcijos SXFM kalba

Konstrukcija	Požymių diagrama	SXFM
Privalomas		... <feature_tree> :r p (id_p) :m c (id_c) </feature_tree> ...
Pasirenkamas		... <feature_tree> :r p (id_p) :o c (id_c) </feature_tree> ...
Vienas ir daugiau („*arba*“)		... <feature_tree> :r p (id_p) :g [1,*] : c1 (id_c1) : c2 (id_c2) : c3 (id_c3) </feature_tree> ...
Alternatyva („*case*“)		... <feature_tree> :r p (id_p) :g [1,1] : c1 (id_c1) : c2 (id_c2) : c3 (id_c3) </feature_tree> ...
Reikalavimas pasirinkti („*requires*“)		... <constraints> constr1: ~id_a or id_b </constraints> ...
Reikalavimas nesirinkti („*excludes*“)		... <constraints> constr2: ~id_a or ~id_b </constraints> ...

4.3.4 PROLOG

PROLOG loginė programavimo kalba gali būti pritaikyta požymių modeliams aprašyti [26]. P. Paškevičius ir kt. [27] pasitelkia PROLOG gramatikos poaibį, parinktą pagal požymių aprašymo kalbos (FDL) pavyzdį, taip pat pateikia 2 paveiksle esančią naudojamos gramatikos aprašymą BNF (angl. *Backus-Naur Form*) forma.

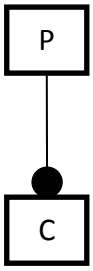
```
<feature_model> ::= { <feature_definition> }
<feature_definition> ::= <rule> | <fact>
<fact> ::= <feature>.
<rule> ::= <feature> :- <rule_body>.
<rule_body> ::= <selector> {, <constraint>}
<selector> ::= <and_selector> | <or_selector> | <oneof_selector>
<and_selector> ::= all ( <alt_feature> { , <alt_feature> } ).
<or_selector> ::= more_of ( <feature> { ; <feature> } ).
<oneof_selector> ::= one_of ( <feature> { , <feature>} ).
<constraint> ::= <requires> | <excludes>
<requires> ::= requires ( <feature> , <feature> ).
<excludes> ::= excludes ( <feature> , <feature> ).
<alt_feature> ::= alt ( <feature> ) | <feature>
<feature> ::= <lower_case> {<char>}
```

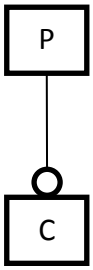
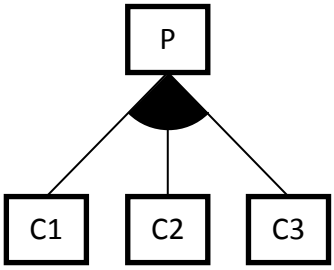
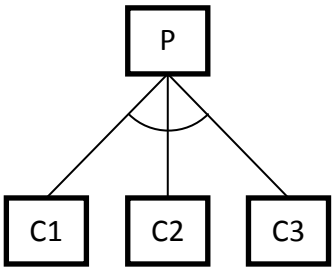
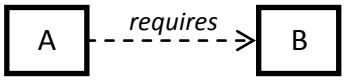
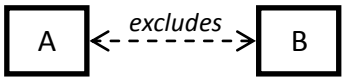
2 pav. PROLOG požymių modelių gramatikos BNF aprašymas

PROLOG formato esminis pranašumas, lyginant su kitomis požymių modelių aprašymo kalbomis, yra tas, kad požymių modelis yra ir vykdomoji specifikacija, kurią galima įvykdyti, patikrinti jo teisingumą, formuluoti užklausas, atlikti formalią loginę analizę.

Lentelėje 4 pateikiamos pagrindinės požymių modelių konstrukcijos aprašytos požymių modelių diagramomis bei PROLOG kalba.

4 lentelė. Pagrindinės požymių modelių konstrukcijos PROLOG kalba

Konstrukcija	Požymių diagrama	PROLOG
Privalomas		`p :- all(c)`

Pasirenkamas		`p :- all(alt(c))`
Vienas ir daugiau („*arba*“)		`p :- more_of(c1, c2, c3)`
Alternatyva („*case*“)		`p :- one_of(c1, c2, c3)`
Reikalavimas pasirinkti („*requires*“)		`requires(a, b)`
Reikalavimas nesirinkti („*excludes*“)		`excludes(a, b)`

4.3.5 TVL

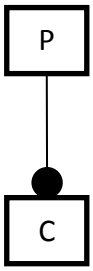
TVL kalbą pasiūlė Boucher ir kt. [10]. TVL kalbos sintaksė yra panaši į C. Ši kalba yra žymiai sudėtingesnė už kitas nagrinėtas kalbas, turi daug papildomų, tikroms programavimo kalboms būdingų požymių, pvz., aritmetinių išraiškų skaičiavimą. Kai kurie sintaksės elementai pasiskolinti iš FDL kalbos. Pagrindinis šios kalbos trūkumas (nors kartu ir privalumas) yra jos

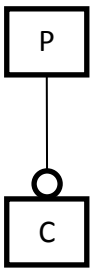
sudėtingumas. Tai yra beveik programavimo kalba, turinti daugelį programavimo kalboms būdingų bruožų.

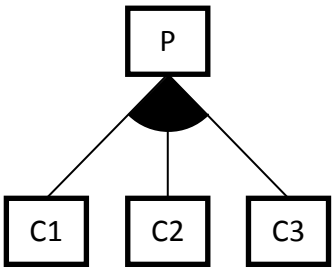
Lentelėje 5 pateikiamos pagrindinės požymių modelių konstrukcijos aprašytos požymių modelių diagramomis bei TVL kalba.

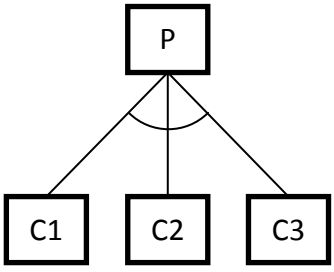
5 lentelė. Pagrindinės požymių modelių konstrukcijos TVL kalba

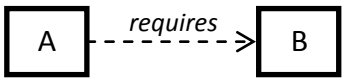
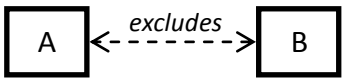
Konstrukcija	**Požymių diagrama**	**TVL**

Privalomas		`root P {`
		`  group allOf {`
		`  C`
		`  }`
		`}`

Pasirenkamas		`root P {`
		`  group allOf {`
		`  opt C`
		`  }`
		`}`

Vienas ir daugiau („*arba*“)		`root P {`
		`  group someOf {`
		`  C1, C2, C3`
		`  }`
		`}`

Alternatyva („*case*“)		`root P {`
		`  group oneOf {`
		`  C1, C2, C3`
		`  }`
		`}`

Reikalavimas pasirinkti („*requires*“)		*Gali būti aprašoma įvairiai, naudojant dvejetaines išraiškas.*
Reikalavimas nesirinkti („*excludes*“)		*Gali būti aprašoma įvairiai, naudojant dvejetaines išraiškas.*

4.3.6 LKC

LKC (angl. *Linux Kernel Configurator*) sudarytas iš LKC kalbos sintaksės analizatoriaus (angl. *parser*) ir priklausomybių tikrintuvo (angl. *dependency checker*) ir leidžia pasirinkti tam tikrą konfigūraciją pagal konfigūracijos duomenų bazę. Konfigūracijos duomenų bazė – tai medžio struktūros konfigūracijos pasirinkimų įrašų rinkinys. Kiekvienas įrašas turi savo priklausomybes, kurios nustato jo pasirenkamumą.

Konfigūracijos failas yra tekstinis failas, parašytas specifine LKC kalba. Ši kalba nėra skirta būtent požymių modeliams aprašyti, tačiau J. Sincero ir W. Schröder-Preikschat parodo, kad LKC galima naudoti ir požymių modeliavimui [28]. Pagrindiniai LKC kalbos raktiniai žodžiai, kurie naudojami požymių modelių aprašymui:

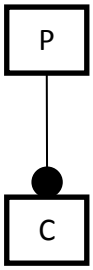
- *config* – naudojamas požymio apibrėžimui;
- *boolean* – dvejetainio tipo kintamasis, nusakantis, ar požymis pasirinktas;
- *depends on* – priklausomybės ryšys, nusakantis, kad požymis priklauso nuo kito tam tikro požymio.
- *select* – atvirkštinės priklausomybės ryšys, nusakantis, kad pasirinkus požymį reikia pasirinkti ir kitą požymį;
- *menu* – naudojamas požymių apgrupavimui, kai leidžiama pasirinkti vieną ir daugiau požymių („*arba*“ grupė);
- *choice* – naudojamas požymių apgrupavimui, kai leidžiama pasirinkti tik vieną požymį (alternatyvinė/„*case*“ grupė);

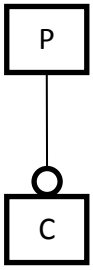
- *requires* – reikalavimo ryšys, nusakantis, kad požymis reikalauja, jog būtų pasirinktas tam tikras kitas požymis (arba atvirkščiai – būtų nepasirinktas, jei nurodytas „!“ simbolis prieš požymio pavadinimą). Naudojamas požymių ryšiams „*requires*“ bei „*excludes*“ aprašyti.

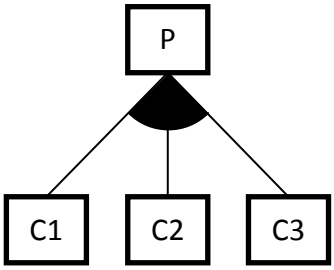
Lentelėje 6 pateikiamos pagrindinės požymių modelių konstrukcijos aprašytos požymių modelių diagramomis bei LKC kalba.

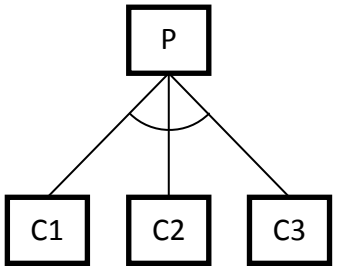
6 lentelė. Pagrindinės požymių modelių konstrukcijos LKC kalba

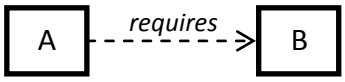
Konstrukcija	**Požymių diagrama**	**LKC**

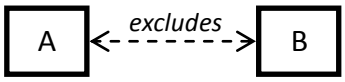
Privalomas		`config P`
		`  boolean "P"`
		`  select C`
		`config C`
		`  boolean "C"`

Pasirenkamas		`config P`
		`  boolean "P"`
		`config C`
		`  depends on "P"`
		`  boolean "C"`

Vienas ir daugiau („*arba*“)		`menu "P"`
		`  config P`
		`  boolean "P"`
		`  config C1`
		`  boolean "C1"`
		`  select P`
		`  config C2`
		`  boolean "C2"`
		`  select P`
		`  config C3`
		`  boolean "C3"`
		`  select P`
		`endmenu`

Alternatyva („*case*“)		`choice`
		`  prompt "P"`
		`  config C1`
		`  boolean "C1"`
		`  config C2`
		`  boolean "C2"`
		`  config C3`
		`  boolean "C3"`
		`endchoice`

Reikalavimas pasirinkti („*requires*“)		`config A`
		`  boolean "A"`
		`  requires B`
		`config B`
		`  boolean "B"`

Reikalavimas nesirinkti („*excludes*“)		`config A`
		`  boolean "A"`
		`  requires !B`
		`config B`
		`  boolean "B"`
		`  requires !A`

4.3.7 Požymių modelių specifikavimo kalbų apibendrinimas

Apžvelgtų požymių modelių specifikavimo kalbų pagrindiniai privalumai ir trūkumai pateikti 7 lentelėje.

7 lentelė. Požymių modelių specifikavimo kalbų palyginimas

Kalba	**Privalumai**	**Trūkumai**
FDL	Paprasta ir aiški struktūra, todėl nesunkiai perprantama. Puikiai tinka rankiniam redagavimui. Populiariausia moksliniuose straipsniuose.	Mažas kiekis įrankių, naudojančių šią kalbą.
OWL	Yra nemažai įrankių, pritaikytų šiai kalbai.	Neskirta būtent požymių modeliams aprašyti. Mažai paplitęs požymių modelių aprašymas šia kalba.
SXFM	Yra nemažai įrankių, pritaikytų šiai kalbai.	Gana sudėtinga XML struktūra, todėl nelabai tinkama rankiniam redagavimui.

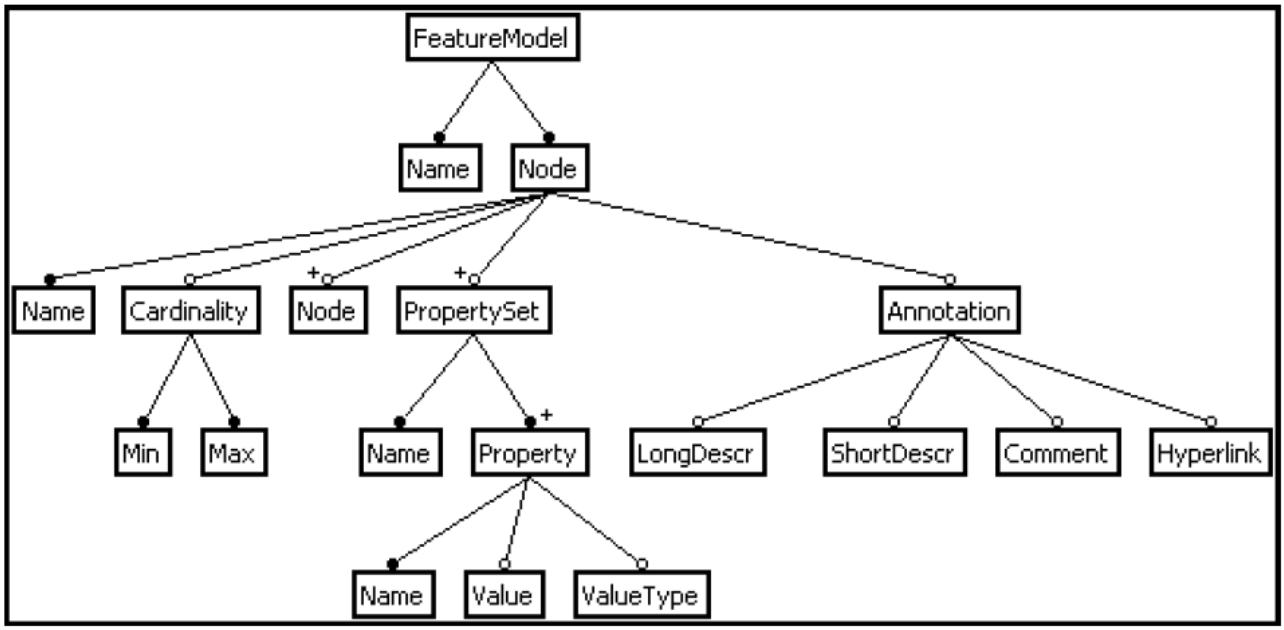
PROLOG	Šia kalba aprašytas požymių modelis yra ir vykdomoji specifikacija, kurią galima įvykdyti, patikrinti jo teisingumą, formuluoti užklausas, atlikti formalią loginę analizę PROLOG priemonėmis.	Neskirta būtent požymių modeliams aprašyti. Mažai paplitęs požymių modelių aprašymas šia kalba.
TVL	Artimesnė programavimo kalbai, todėl leidžia sudėtingesnes konstrukcijas ir išraiškas, o ne vien požymių modeliams įprastus aprašymo būdus.	Įprastam požymių modelių aprašymui per daug sudėtinga.
LKC	Yra nemažai įrankių, pritaikytų šiai kalbai. Realiai naudojama Linux branduolio konfigūravimui.	Neskirta būtent požymių modeliams aprašyti.

5 POŽYMIŲ IR JŲ TRANSFORMACIJŲ METAMODELIS

5.1 Požymių metamodelis

5.1.1 O. Rohlik ir kt. pasiūlytas požymių metamodelis

Norėdami suprasti kaip keičiant įvairius požymius ir ryšius tarp jų, galima gauti naują produktą priklausantį produktų šeimai, turime suprasti iš ko susideda pats požymių modelis ir kaip jį galima transformuoti į naują produktą. Yra keletas siūlomų požymių metamodelių: 1) O. Rohlik ir kt. [29] siūlo metamodelį (3 pav.), kuris konceptualiai nusako tam tikrą modelio struktūrą, kurios pagalba galima patikrinti ar požymių modelis yra validus. Validus požymių modelis bus tuomet, kuomet tai bus požymių meta-modelio atskiras atvejis (angl. *Instance*).

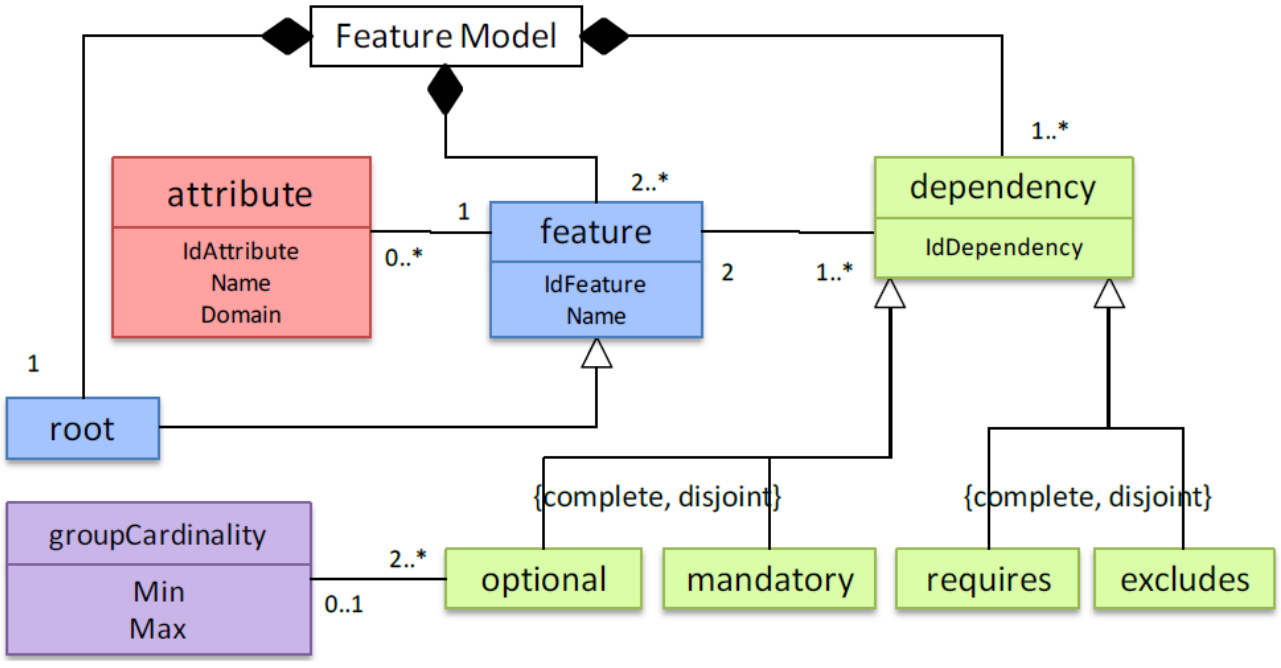


3 pav. O. Rohlik ir kt. siūlomas požymių metamodelis

5.1.2 R. Mazo ir kt. pasiūlytas požymių metamodelis

R. Mazo ir kt. [30] siūlomas požymių metamodelis pavaizduotas (4 pav.). Jie taip pat teigia, jog: „sudarytas požymių metamodelis yra labai svarbus aspektas iš metamodelio sudarytų modelių teisingumui ir nuoseklumui patikrinti“ .

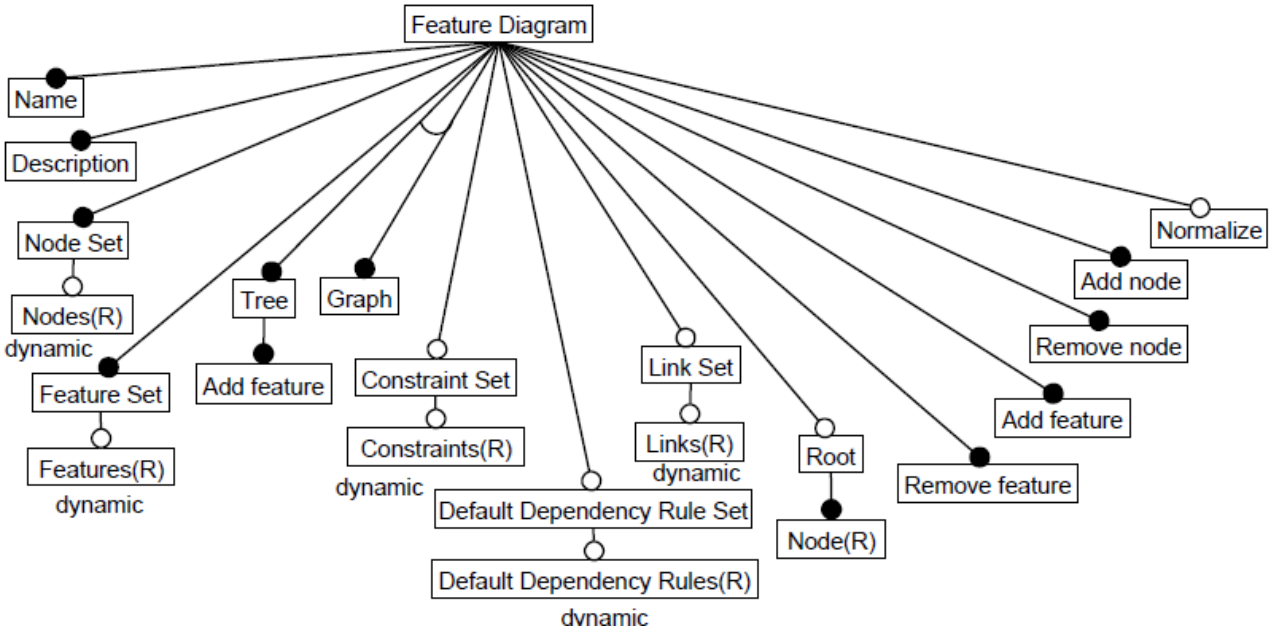
Jie kaip ir O. Rohlik ir kt. taip pat teigia, kad metamodelio pagalba galima patikrinti naujai sudaryto modelio validumą – tai turi būti atskiras metamodelio atvejis (angl. *Instance*).



4 pav. R. Mazo ir kt. siūlomas požymių metamodelis

5.1.3 V. Vranic ir kt. pasiūlytas požymių metamodelis

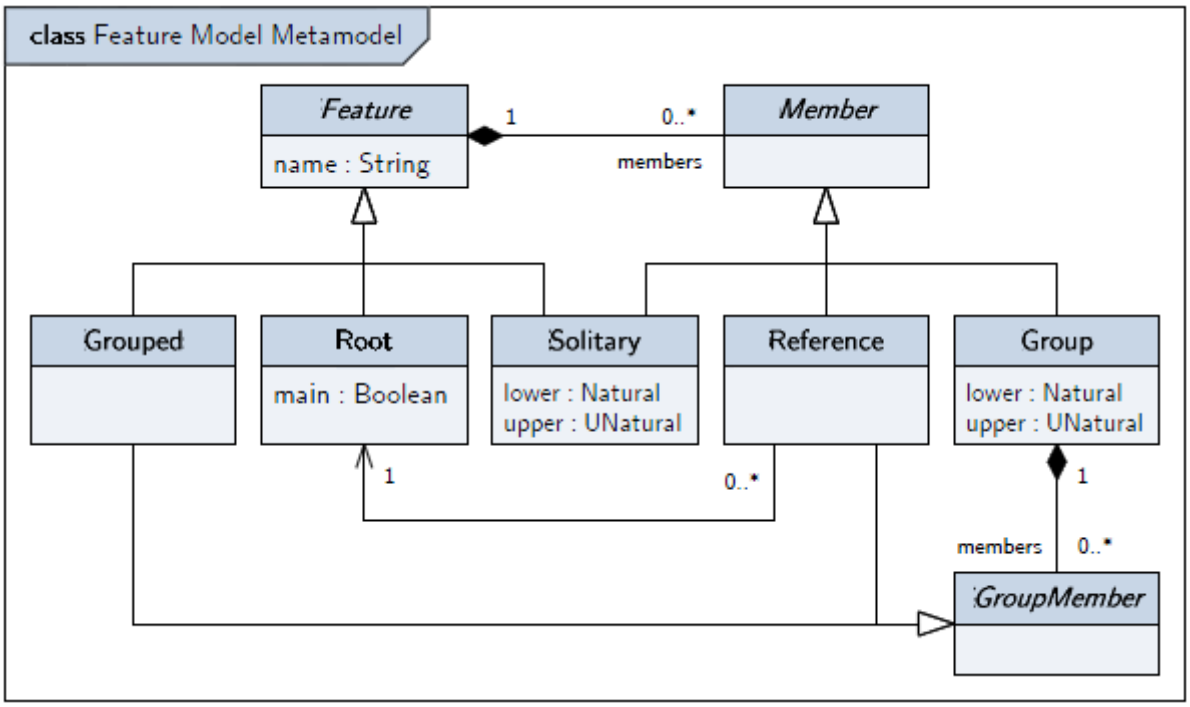
V. Vranic teigia, jog požymių metamodelio (5 pav.) pagalba galime nustatyti naujai sudarytų modelių bendrumus ir srities kintamumus. Taip pat V. Vranic pažymi, jog požymių metamodelis gali būti naudojamas naujo produkto pagrindimui ir jo pagalba galima kurti, bei tobulinti jau esamus požymių modelių kūrimo įrankius [31].



5 pav. V. Vranic siūlomas požymių diagramų metamodelis

5.1.4 P. O. Rossel ir kt. pasiūlytas požymių metamodelis

P. O. Rossel ir kt. [32] teigia, jog metamodelis (6 pav.) yra naudojamas kaip priemonė išreikšti konkrečią srities modeliavimo kalbą ir jo pagalba apibrėžiami naudojami artifaktai. Taip pat P. O. Rossel ir kt. teigia, jog naudojantis šiomis priemonėmis galima pasiekti automatinį taikomųjų programų kūrimą tam tikroje srityje.



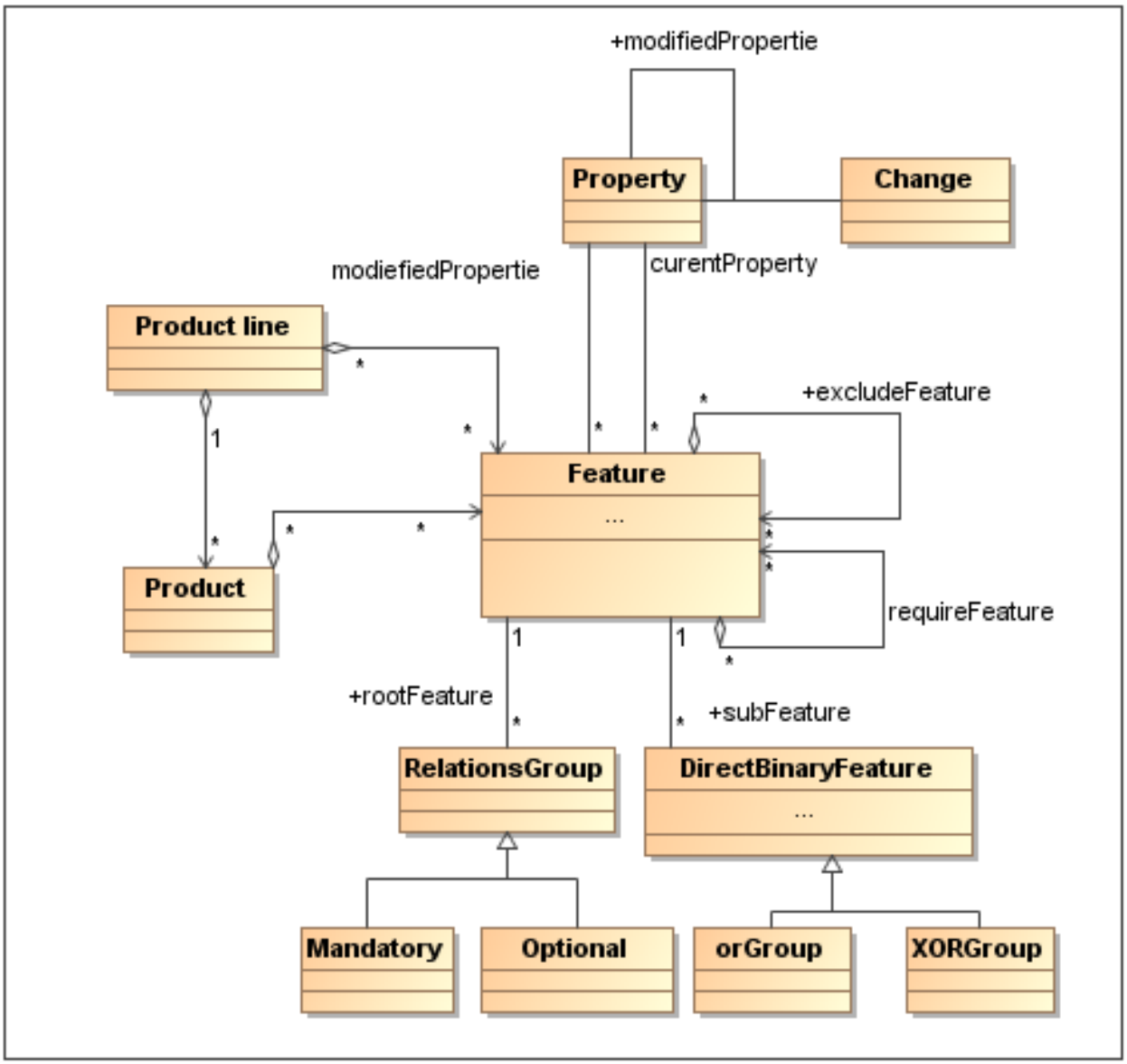
6 pav. P. O. Rossel ir kt. siūlomas požymių metamodelis

5.1.5 Siūlomas požymių metamodelis

Išnagrinėję siūlomus požymių metamodelius pastebėjome jog pagrindinis požymių modelių elementas yra požymis. M Fowler ir kt. [33] požymius ir jų metamodelius siūlo panaudoti reikalavimo aprašymo kalboms kurti. Mes atsižvelgę į kitų autorių komentarus sukūrėme atskirą požymių metamodelį, kurį atvaizdavome UML klasių diagrama. Iš metamodelio diagramos matyti, kad pagrindinis požymių diagramos elementas yra požymis (angl. *Feature*). Kiekvienas požymis turi ryšius su kitais požymiais. Ryšiai gali būti kelių tipų: 1) Būtinas (angl. *Mandatory*) – tai ryšio tipas kurį naudojant požymio pasirinkimas, sudarant naują atskiro atvejo diagramą (angl. *Instance*), yra būtinas; 2) Pasirenkamas (angl. *Optional*) – tai ryšio tipas kurį naudojant požymio pasirinkimas, sudarant naują atskiro atvejo diagramą (angl. *Instance*), požymis gali būti pasirenkamas arba ne; 3) Išskiriantis (angl. *Exclude*) – jeigu du požymiai yra jungiami išskiriančiuoju ryšiu, tuomet tik vienas požymis, iš dviejų požymių kurie yra sujungti išskiriančiuoju ryšiu, gali būti pasirenkamas; 4) Reikalaujantis (angl. *Require*) – jeigu du požymiai yra jungiami reikalaujančiuoju ryšiu, tuomet abu požymiai, iš dviejų požymių kurie yra sujungti reikalaujančiuoju ryšiu, turi būti pasirenkami;

Iš požymių grupės galime gauti tam tikrą produktą (angl. *Product*), o iš keleto produktų galime gauti produktų šeimą (angl. *Product line*). Kiekvienas požymių modelis yra sudarytas bent iš dviejų požymių. Kiekvienas požymis turi savo savybes (angl. *property*). Požymių savybės gali būti bet kokia aibė, pvz.: požymio pavadinimas, požymio ryšiai su kitais požymiais.

Tokį metamodelio apibrėžimą pasirinkome dėl to, kad jis yra minimalistinis ir apibrėžia visus galimus požymių modelių variantus. Jame atspindima, jog pagrindinis požymių modelių – produktų ir produktų šeimų elementas yra požymis. Požymio savybės gali būti lengvai modifikuotos (pakeistos), kadangi požymis yra tai kas matoma iš išorės, jos gali kisti pvz.: pavadinimas. taip pat apibrėžiami visi ryšiai – pradedant reikalaujančiu/išskiriančiu ryšiais, baigiant tiesioginiais dvejetainiais ryšiais.



7 pav. Požymių metamodelis

5.2 Požymių modelių transformacijos metamodelis

5.2.1 Modelio transformacijos apibrėžimas

Požymių modelių transformavimas yra sąlyginai nauja sritis, nes nėra nustatytų konkrečių modelių tipų transformavimo standartų.

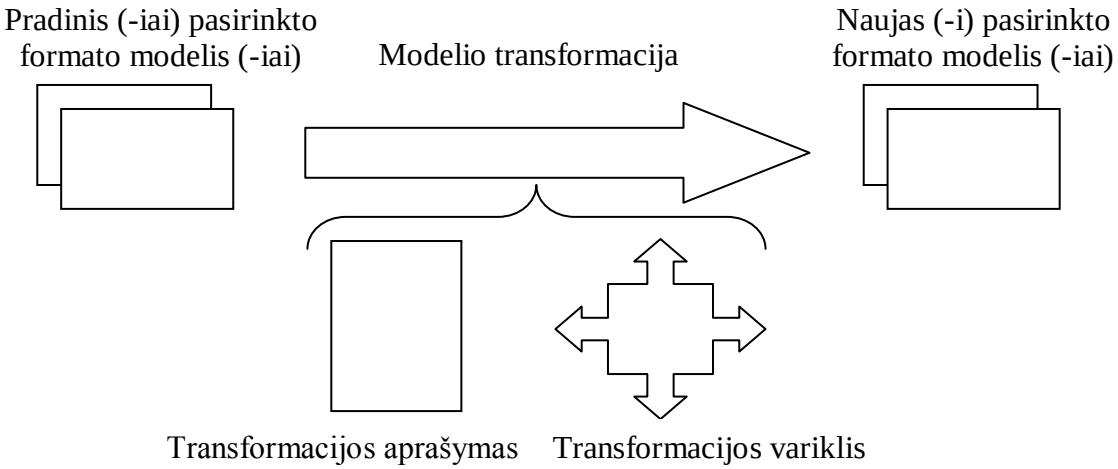
Programų inžinerijos standartizavimo asociacija *Object Management Group* (OMG), modeliais paremtos architektūros (angl. *model-driven architecture (MDA)*) kontekste transformaciją apibrėžia taip: „transformacija – tai procesas kuomet tos pačios sistemos modelis yra konvertuojamas į kitą modelį“ [34].

Kleppe ir kt. [35]. modelių transformaciją apibrėžia taip: „automatinis naujo modelio generavimas iš pasirinkto pradinio modelio, naudojant transformacijos aprašymą“.

Mens ir kt. [36] išplėtė Kleppe ir kitų apibrėžimą teigdami jog pradinių ir gautų modelių gali būti keletas ir modelių transformaciją apibrėžė sekančiai: „automatinis vieno arba kelių naujų modelių transformavimas iš vieno ar kelių pasirinktų pradinių modelių, naudojant transformacijos aprašymą“.

Mes siūlome transformaciją apibrėžti taip: „tai automatinis vieno arba kelių naujų modelių transformavimo procesas iš vieno ar kelių pasirinktų pradinių modelių, naudojant transformacijos aprašymą ir pasirenkant pradinį, bei galutinį modelio formatą“.

8 paveiksle yra modelio(-ių) transformacijos iliustracija.



8 pav. Požymių Modelio transformacijos elementai

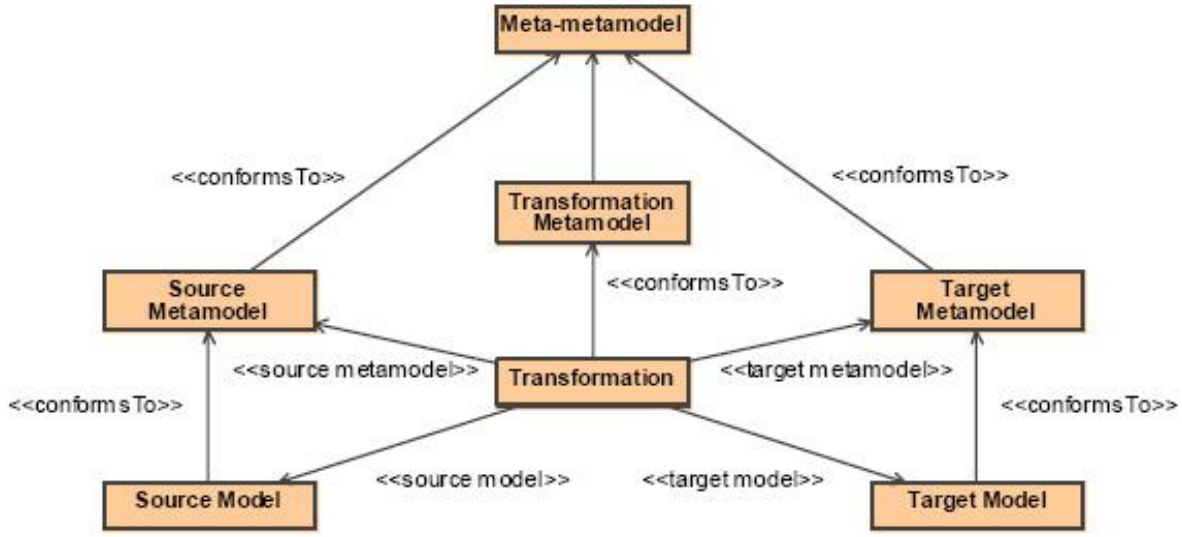
Taigi modelio modifikacija tai procesas kuomet pradinis ir gautas modeliai yra to paties metamodelio atskiri atvejai (angl. *Instance*). Tuomet galima tarti, jog toks procesas yra modelio būsenos pakeitimas – egzistuojantys elementai gali būti pakeisti, nauji gali būti pridėti ar netgi dalis elementų gali būti pašalinti.

Transformacijas galima būti išskirti į dvi pagrindines grupes: 1) modelis – kodas transformacija; 2) modelis – modelis transformacija

Modelis – kodas transformacija, tai transformacija kuomet modelis yra konvertuojamas į tekstą. Dažniausiai tai programos tekstas neapribotas objektiškai orientuotų kalbų atžvilgiu, bet išreikštas tekstu.

Modelis – modelis transformacija, tai transformacija kuomet modelis yra konvertuojamas į kitą modelį. Kaip pavyzdį galima pateikti FDL kalba aprašytą modelį ir FD2 įrankio pagalba nubraižytą požymių modelį. Taip pat transformacija galima modelį transformuojat iš vieno abstrakcijos lygio į kitą.

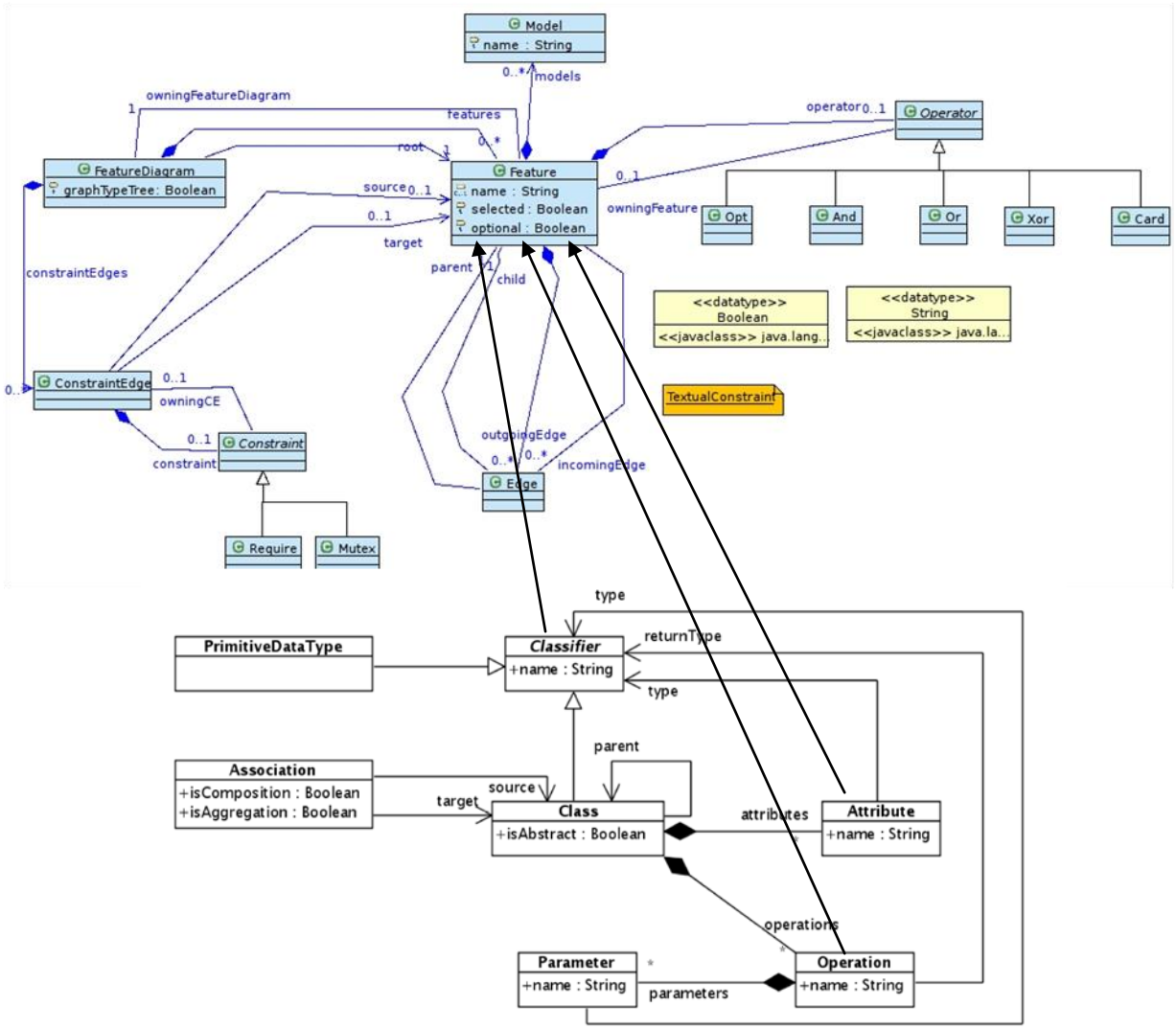
A. Kraus [37] modelis – modelis tipo transformacijos susieja pradinio metamodelio (nusakant jo tipą) ir galutinio metamodelio (nusakant jo tipą) atskirus atvejus. 9 paveikslas iliustruoja ryšius tarp modelių, metamodelių ir transformacijų. Tai pat A. Kraus teigia jog svarbus aspektas yra tai, kad pati transformacija savaime jau yra modelis, pvz.: transformacijos metamodelio atskiras atvejis. Tai leidžia atlikti aukštesnio abstrakcijos lygmens transformacijas, pvz.: transformacijos kurios generuoja transformacijas. Taigi iš to išplaukia, jog visi metamodeliai yra meta-meta modelio atskiras atvejis. Taip pat transformacijos gali būti įvairiakryptės, bet dažniausiai vykdomos vienos krypties transformacijos.



9 pav. Kraus pasiūlytas metamodelis

5.2.2 Požymių diagramos konvertavimas į UML

FD2 įrankio viena iš funkcijų yra FD2 formato požymių diagramos konvertavimas į UML klasių diagramą. Šie formatai yra panašūs ir naudojant aprašytas taisykles UML diagramą galima pakeisti į požymių diagramą. Šiame darbe rėmėmės V. Štuikio ir R. Damaševičiaus [38] siūlomu požymių diagramų transformacijos į UML diagramas metamodeliu (10 pav.).



10 pav. Požymių diagramų – UML klasių diagramų susietumo metamodelis

Apibrėžti požymių diagramų – UML klasių diagramų transformaciją galima taip: pasirenkamas požymių metamodelio elementas ir nurodomas vienas ar keletas ekvivalentų UML metamodelyje. Tai reiškia, kad reikia pasirinkti vieną iš galimų dizaino mechanizmų, kad anotuoti galimą požymių modelių ir UML klasių diagramų transformaciją [38].

Transformacijos idėja būtų tokia:

a) Transformuoti kiekvieną požymį į klasę;
b) Transformuoti būtinus požymius į agregacijos/kompozicijos UML ryšius;
c) Transformuoti pasirenkamus ryšius į UML 0..1 asociacijas;
d) Transformuoti alternatyviuosius ryšius į UML generalizaciją;

e) Galiausiai transformuoti reikalaujantį (angl. *require*) ir išskiriančius (angl. *exclusive*) ryšius į OCL išraiškas (angl. *expressions*).

6 POŽYMIŲ MODELIAVIMO IR TRANSFORMAVIMO ĮRANKIO PROJEKTAVIMAS

6.1 Sistemos paskirtis

Šiuo metu masiškai kuriant programinę įrangą įvairiems uždaviniams spręsti, pakartotinis esamų programų panaudojimas tapo labai populiarus. Tačiau pakartotinis anksčiau sukurtos programos panaudojimas kitam uždaviniui spręsti dažniausiai tampa neįmanomas. Tuomet ir yra pasitelkiama į pagalbą įvairios jau esamų programų sprendimų transformacijos. Kad suprasti kaip atlikti transformaciją reikia suprasti daug įvairių dalykų apie programos veikimą, specifines savybes, ryšį tarp įvairių elementų, naudojamus atributus ir t.t. Dažniausiai, kad tai padaryti reikia nustatyti kuriai programų šeimai ji priklauso. Programų šeima – tai programų rinkinys su labai panašiomis savybėmis.

Kitas svarbus aspektas yra programų produktų linijos. Programų produktų linijos inžinerija skatina sisteminį pakartotinį panaudojimą visame programos kūrimo procese. Produktų linijos tai rinkinys programinės įrangos produktų, kurie dalinasi panašiomis savybėmis, tenkina specifinius reikalavimus ir naudoja tam tikras pagrindinės sistemos dalis.

Požymių variantiškumas, programų šeimos, produktų linijos aktualūs beveik kiekvienam programinės įrangos paketui. Labai dažnai susiduriama su poreikiu sukurti kelias skirtingas programinės įrangos versijas, turinčias skirtingą funkcionalumą, bet kartu ir besidalijančias tam tikra aibe bazinio funkcionalumo. Iš pirmo žvilgsnio tai atrodo gana trivialus uždavinys, tačiau praktika rodo, kad programinės įrangos sistemoms plečiantis ir kylant poreikiui išleisti vis daugiau skirtingų sistemos versijų, ši užduotis tampa gana kebli.

Automatizuojant šį procesą būtų galima sutaupyti nemažai laiko ir lėšų, todėl požymių variantiškumo modeliavimo ir šių modelių transformavimo įrankis, padedantis spręsti šią problemą yra išties aktualus ir galėtų būti realiai panaudojamas. Dėl šių priežasčių buvo nuspręsta sukurti šių užduočių sprendimą palengvinantį įrankį, kuriuo būtų galima modeliuoti požymių diagramas, jas transformuoti į kitus modelius arba kuriamos programos kodo „griaučius“.

6.2 Sistemos tikslai

Sistemos pagrindinis tikslas palengvinti programinės įrangos produktų linijos požymių modeliavimą požymių diagramomis ir ypač šių modelių transformavimą. Sukurtas įrankis padėtų efektyviau plėtoti kuriamos programinės įrangos šeimą, taigi leistų taupyti laiką ir lėšas. Šio įrankio potencialūs naudotojai – IT įmonės, kuriančios ir palaikančios įvairią programinę įrangą bei turinčios poreikį ją atitinkamai pritaikyti įvairiems užsakovams ar dėl kitų priežasčių naudojančios produktų linijos metodologiją.

6.3 Vartotojų tipai

Požymių modeliavimo ir transformavimo įrankiu naudosis trys vartotojų grupės, išvardintos 8, 9, 10 lentelėse.

8 lentelė. Vartotojų tipai: architektas

Vartotojo kategorija	**Architektas**
Vartotojo sprendžiami uždaviniai	Sistemų architektūros kūrimas pagal požymių modelius.
Patirtis dalykinėje srityje	Architektūros specialistas. Aukšto lygio programuotojas
Patirtis informacinėse technologijose	IT Specialistas
Papildomos vartotojo charakteristikos	Gali reikti pačiam pasimokyti naudotis įrankiu ir tinkamai interpretuoti gautus rezultatus
Vartotojo prioritetai	Svarbus vartotojas

9 lentelė. Vartotojų tipai: analitikas

Vartotojo kategorija	**Analitikas**
Vartotojo sprendžiami uždaviniai	Atliekami požymių modelių (diagramų) kūrimo ir redagavimo darbai
Patirtis dalykinėje srityje	Vidutinio arba aukšto lygio analitikas
Patirtis informacinėse technologijose	IT Specialistas
Papildomos vartotojo charakteristikos	Gali reikti pačiam pasimokyti naudotis įrankiu ir tinkamai interpretuoti gautus rezultatus.
Vartotojo prioritetai	Svarbus vartotojas

10 lentelė. Vartotojų tipai: užsakovas

Vartotojo kategorija	**Užsakovas**
Vartotojo sprendžiami uždaviniai	Peržiūrimi požymių modeliai (diagramos)
Patirtis dalykinėje srityje	Vidutinio arba aukšto lygio testuotojas
Patirtis informacinėse technologijose	Patirtis IT nebūtina norint naudotis įrankiu
Papildomos vartotojo charakteristikos	Gali reikti pačiam pasimokyti naudotis įrankiu ir tinkamai interpretuoti gautus rezultatus.
Vartotojo prioritetai	Svarbus vartotojas

6.4 Projekto apribojimai

6.4.1 Apribojimai sprendimui

Kuriama ne atskira programa, bet jau egzistuojančios platformos Eclipse IDE priedas, kuris parašytas Java programavimo kalba. Dėl šios priežasties kuriamas įrankis bus rašomas taip pat Java kalba, todėl vartotojo kompiuteryje turės būti įdiegta JRE (angl. *Java Runtime Environment*).

6.4.2 Diegimo aplinka

Įrankio diegimas bus įmanomas visur, kur bus galima įdiegti *Eclipse* IDE platformą. *Eclipse* platformą galima įdiegti ir naudoti Windows, Linux ir Mac OS X operacinėse sistemose su JRE.

6.4.3 Bendradarbiaujančios sistemos

Su kuriamu įrankiu glaudžiai bendradarbiaus *Eclipse* IDE platforma, nes įrankis bus jos priedas, naudos bazinį šios platformos teikiamą funkcionalumą.

6.4.4 Svarbūs faktai ir prielaidos

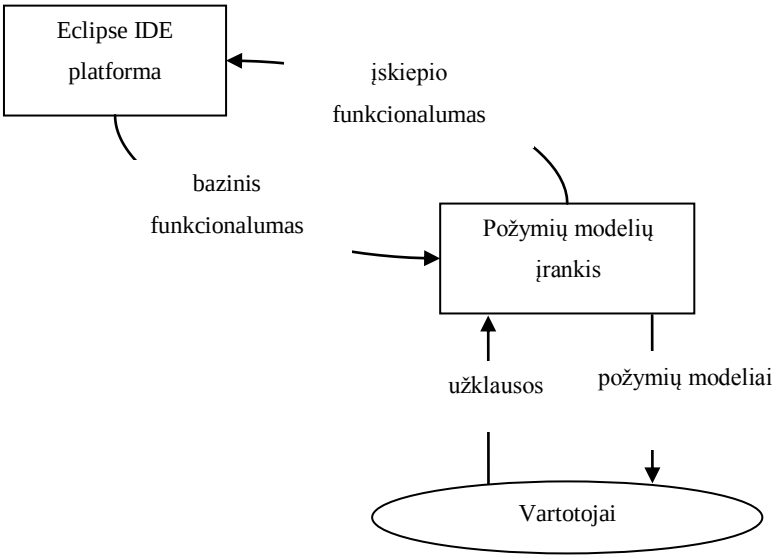
1. Duomenų saugojimas reliacinėje duomenų bazėje nebus naudojamas. Visi duomenys saugomi failuose.
2. *Eclipse IDE* platforma suteiks bazinį funkcionalumą, kuris bus naudojamas, norint užtikrinti nuoseklumą ir darnumą.

6.5 Funkciniai reikalavimai

6.5.1 Veiklos sritis

6.5.1.1 Veiklos kontekstas

Kuriamo įrankio kontekstas pavaizduotas 11 paveikslėlyje.

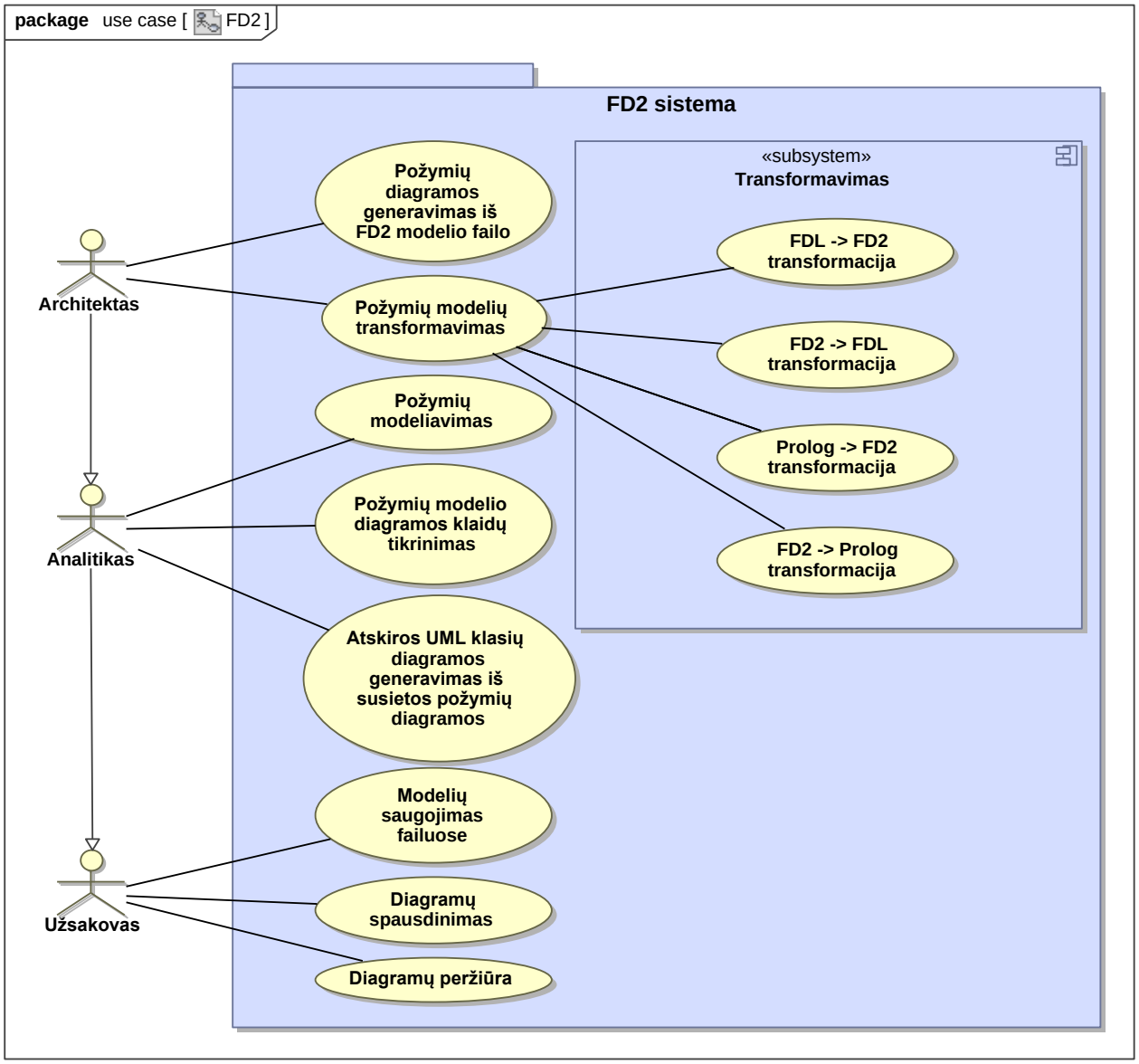


11 pav. Konteksto diagrama.

6.5.2 Sistemos sudėtis

6.5.2.1 Sistemos ribos

Požymių modeliavimo ir transformavimo įrankio panaudojimo atvejų diagrama pateikiama 12 paveikslėlyje.



12 pav. Panaudojimo atvejų diagrama

6.5.2.2 Panaudojimo atvejų sąrašas

Įrankio panaudojimo atvejai pateikiami 11-18 lentelėse.

11 lentelė. „Diagramų peržiūra“ PA.

1. Panaudojimo atvejis	**Diagramų peržiūra**
Aktoriai	Užsakovas, analitikas, architektas
Aprašas	Peržiūrėti požymių modelių diagramas
Prieš sąlygos	Turimas išsaugotas požymių modelis
Sužadinimo sąlygos	Paspaudžiamas požymių modelio atidarymo mygtukas
Po sąlygos	Požymių modelio diagrama rodoma

12 lentelė. „Diagramų spausdinimas“ PA.

2. Panaudojimo atvejis	**Diagramų spausdinimas**
Aktoriai	Užsakovas, analitikas, architektas
Aprašas	Spausdinti požymių modelių diagramas
Prieš sąlygos	Atidarytas požymių modelis
Sužadinimo sąlygos	Paspaudžiamas spausdinimo mygtukas
Po sąlygos	Atspausdinta diagrama

13 lentelė. „Modelių saugojimas failuose“ PA.

3. Panaudojimo atvejis	**Modelių saugojimas failuose**
Aktoriai	Užsakovas, analitikas, architektas
Aprašas	Galimybė išsaugoti požymių modelius failuose
Prieš sąlygos	Atidarytas požymių modelis
Sužadinimo sąlygos	Paspaudžiamas saugojimo mygtukas
Po sąlygos	Požymių modelis išsaugotas faile

14 lentelė. „Modelių saugojimas failuose“ PA.

4. Panaudojimo atvejis	**Atskiros UML klasių diagramos generavimas iš susietos UML klasių diagramos**
Aktoriai	Analitikas, architektas
Aprašas	Galimybė automatiškai sugeneruoti atskiro atvejo UML klasių diagramą iš susietos požymių ir UML klasių diagramos
Prieš sąlygos	Atidarytas požymių modelis
Sužadinimo sąlygos	Paspaudžiamas UML klasių diagramos sugeneravimo mygtukas
Po sąlygos	Sugeneruotas atskiro atvejo UML klasių modelis

15 lentelė. „Diagramos klaidų tikrinimas“ PA.

5. Panaudojimo atvejis	**Požymių modelio diagramos klaidų tikrinimas**
Aktoriai	Analitikas, architektas
Aprašas	Galimybė patikrinti, ar požymių diagrama neturi klaidų
Prieš sąlygos	Atidaryta diagrama.
Sužadinimo sąlygos	Spaudžiamas meniu punktas „Tikrinti diagramos klaidas“
Po sąlygos	Pateikiamos diagramos klaidos

16 lentelė. „Požymių modeliavimas“ PA.

6. Panaudojimo atvejis	**Požymių modeliavimas**
Aktoriai	Analitikas, architektas
Aprašas	Galimybė kurti požymių modelius, kuriant/redaguojant/šalinant požymių diagramų elementus.
Prieš sąlygos	Atidaryta požymių diagrama arba sukurta tuščia požymių diagrama
Sužadinimo sąlygos	Paspaudžiamas atitinkamas mygtukas
Po sąlygos	Sukurtas/redaguotas požymių modelis

17 lentelė. „Požymių modelių transformavimas“ PA.

7. Panaudojimo atvejis	**Požymių modelių transformavimas**
Aktoriai	Architektas
Aprašas	Galimybė transformuoti požymių modelius iš vieno aprašo tipo į kitą.
Prieš sąlygos	Atidarytas požymių modelis
Sužadinimo sąlygos	Pasirinktas atitinkamas transformavimo meniu punktas
Po sąlygos	Transformuotas požymių modelis

18 lentelė. „Požymių diagramos generavimas iš FD2 tipo modelio failo“ PA.

8. Panaudojimo atvejis	**Požymių diagramos generavimas iš FD2 tipo modelio failo**
Aktoriai	Architektas
Aprašas	Galimybė generuoti požymių modelius požymių diagramų elementus.
Prieš sąlygos	Atidaryta požymių diagrama arba sukurta tuščia požymių diagrama
Sužadinimo sąlygos	Paspaudžiamas diagramos generavimo mygtukas
Po sąlygos	Sugeneruotas požymių modelis

6.5.3 Funkciniai reikalavimai ir reikalavimai duomenims

6.5.3.1 Funkciniai reikalavimai

Apsibrėžti šie funkciniai reikalavimai:

1) Turi būti galimybė detaliai peržiūrėti požymių diagramų savybes.
2) Modeliuojant požymių diagramas turi būti galimybė kurti požymius.
3) Modeliuojant požymių diagramas turi būti galimybė trinti nereikalingus požymius.
4) Modeliuojant požymių diagramas turi būti galimybė redaguoti požymius.

5) Modeliuojant požymių diagramas klaidos turi būti automatiškai aptinkamos ir parodomos vartotojui.
6) Vartotojas turi galėti pasirinkti įvairius spausdinimo parametrus.
7) Vartotojas turi galėti peržiūrėti spausdinimo rezultatą prieš spausdinant.
8) Vartotojas turi galėti transformuoti požymių modelį iš FDL formato į FD2 formatą.
9) Vartotojas turi galėti transformuoti požymių modelį iš FD2 formato į FDL formatą.
10) Vartotojas turi galėti transformuoti požymių modelį iš FD2 formato į PROLOG formatą.
11) Vartotojas turi galėti transformuoti požymių modelį iš PROLOG formato į FD2 formatą.
12) Vartotojas turi galėti transformuoti požymių modelį iš PROLOG formato į FDL formatą.
13) Vartotojas turi galėti transformuoti požymių modelį iš FDL formato į PROLOG formatą.
14) Turi būti galimybė išsaugoti kuriamą požymių modelį FD2 tipo faile .
15) Turi būti galimybė atidaryti ir redaguoti FD2 tipo failus.
16) Turi būti galimybė keisti apžvalgos langą.
17) Turi būti galimybė keisti peržiūrimų diagramų mastelį.

Reikalavimas duomenims – reliacinė duomenų bazė nebus naudojama ir jokios sudėtingos duomenų struktūros nebus. Tačiau verta paminėti, kad požymių modelių duomenys, diagramos bus saugomi tam tikros struktūros failuose.

6.6 Nefunkciniai reikalavimai

Apsibrėžti šie nefunkciniai reikalavimai:

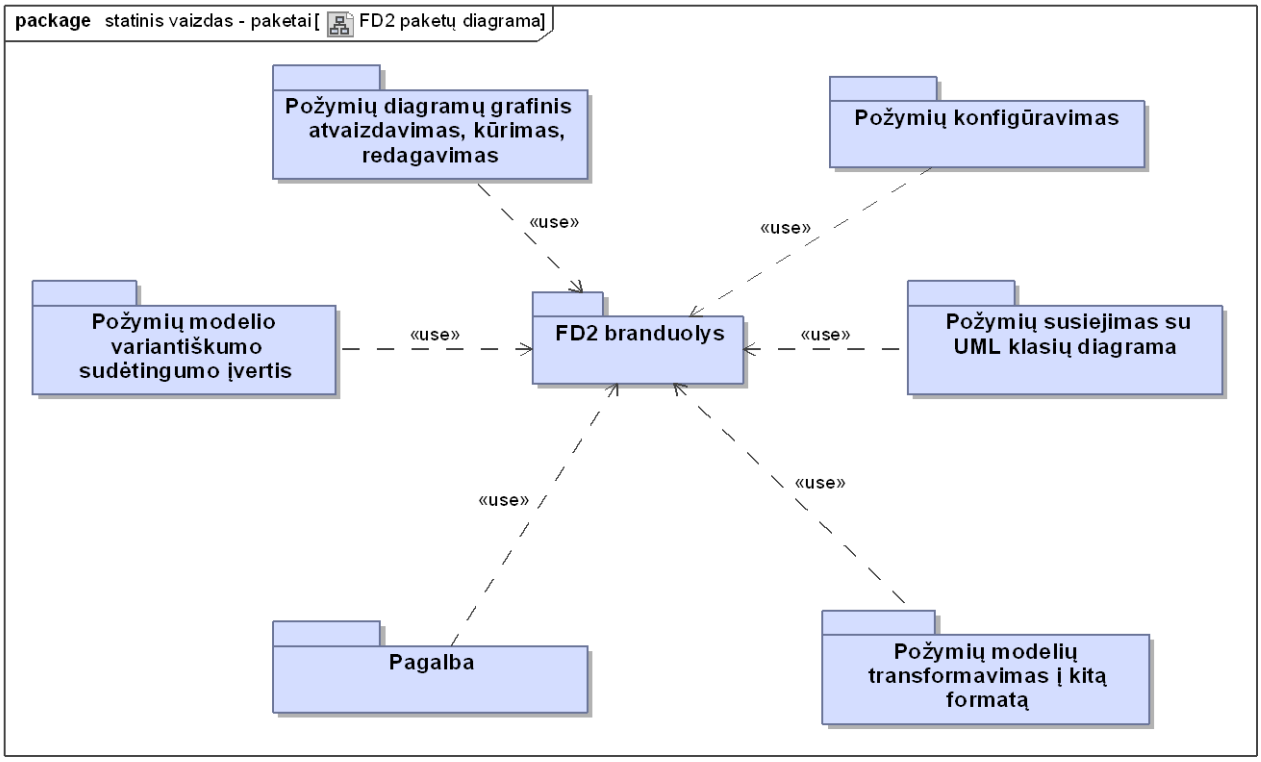
1) Turi būti užtikrintas kuo sklandesnis vartotojo darbas.
2) Įrankio vartotojas neturi būti apkrautas sudėtingu įrankio panaudojimu.
3) Turi būti apsauga dėl vartotojų nepatyrimo galimai netinkamai suvestiems duomenis aptikti.
4) Resursai turėtų būti atlaisvinti, kai tik įmanoma, kad ilgą laiką naudojant įrankį, kompiuteris nesulėtėtų,
5) Aukštas patikimumas reikalingas norint užtikrinti produktyvų vartotojo darbą.
6) Įrankis turi būti palaikomas kuo lengviau, jo galimybės praplečiamos nesudėtingai

6.7 Statinis vaizdas

Įrankis susideda iš 7 paketų (13 pav.):

1) FD2 branduolys – pagrindinis paketas atsakingas už sistemos veikimą, kurį naudoja visi kiti paketai;
2) Požymių diagramų grafinio atvaizdavimo, kūrimo ir redagavimo paketas;

3) Požymių konfigūravimo paketas;
4) Požymių susiejimo su UML klasių diagrama paketas;
5) Požymių modelių transformavimo į kitus formatus paketas;
6) Požymių modelio variantiškumo sudėtingumo įvertinimo paketas;
7) Įrankio pagalbos paketas.



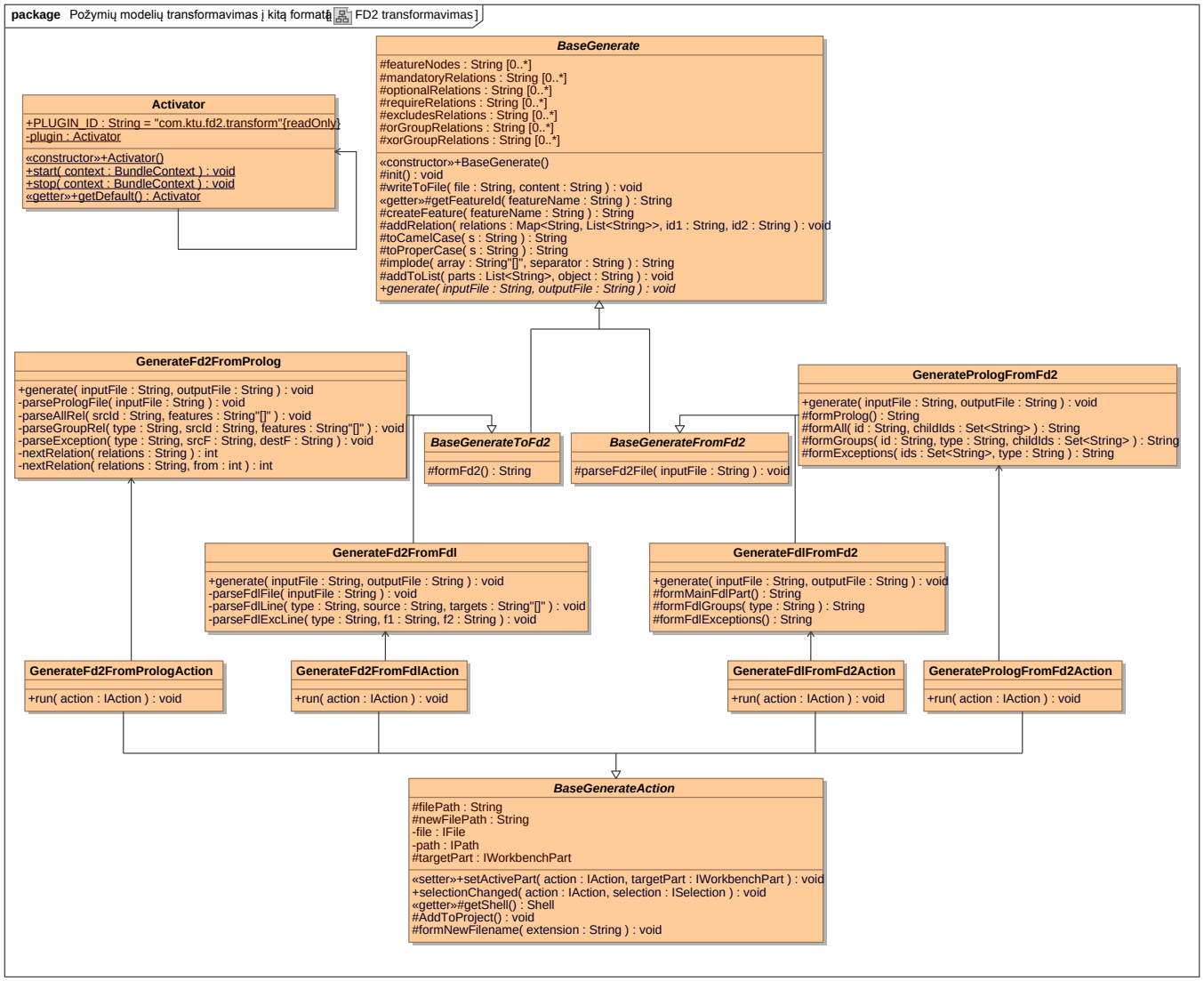
13 pav. Statinis vaizdas

6.8 Klasių diagrama

Požymių modeliavimo ir transformavimo įrankis susideda iš daugybės klasių. Atvaizdavome, mūsų pasiūlyto ir integruoto į įrankį transformavimo paketo klases. Transformavimo paketas susideda iš šių klasių (14 pav.):

1) *Activator* – klasė realizuojanti sąsają tarp transformavimo modulio ir *Eclipse* karkaso;
2) *BaseGenerate* – bazinė abstrakti klasė visoms transformavimo logikos klasėms;
3) *BaseGenerateToFD2* – bazinė abstrakti klasė visoms transformavimo į FD2 formatą logikos klasėms;
4) *BaseGenerateFromFD2* – bazinė abstrakti klasė visoms transformavimo iš FD2 formato logikos klasėms;
5) *GenerateFD2FromProlog* – transformavimo iš PROLOG į FD2 logikos klasė;

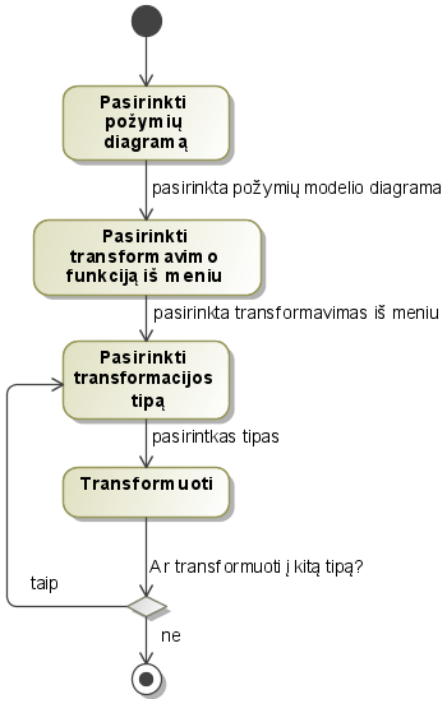
6) *GenerateFD2FromFDL* – transformavimo iš FDL į FD2 logikos klasė;
7) *GenerateFD2FromProlog* – transformavimo iš FD2 į PROLOG logikos klasė;
8) *GenerateFDLFromFD2* – transformavimo iš FD2 į FDL logikos klasė;
9) *BaseGenerateAction* – bazinė abstrakti transformavimo atlikimo klasė;
10) *GenerateFD2FromPrologAction* – transformavimo iš PROLOG į FD2 atlikimo klasė;
11) *GenerateFD2FromFDLAction* – transformavimo iš FDL į FD2 atlikimo klasė;
12) *GenerateFD2FromPrologAction* – transformavimo iš FD2 į PROLOG atlikimo klasė;
13) *GenerateFDLFromFD2Action* – transformavimo iš FD2 į FDL atlikimo klasė.



14 pav. Transformacijų modulio klasių diagrama

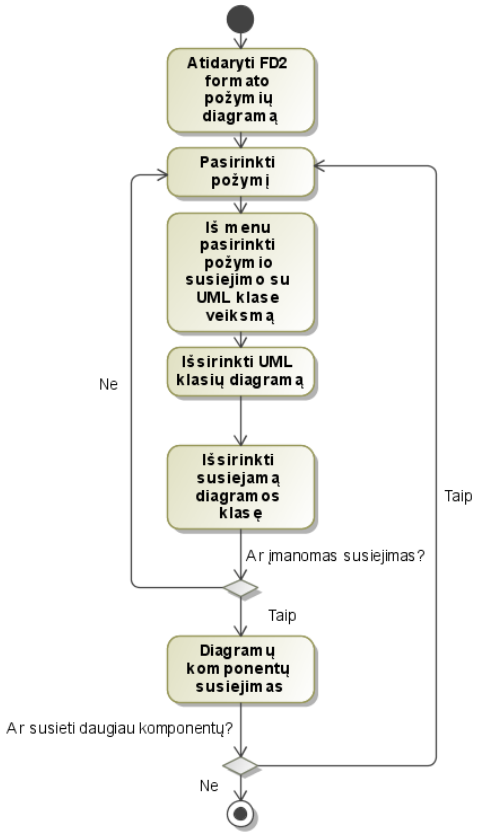
6.9 Veiklos diagramos

Transformavimo veiklos diagrama pateikta 15 paveiksle atvaizduoja požymio modelio transformavimą iš vieno formato į kitą. Transformavimas vyksta tokia seka: Pasirenkama požymių diagrama → Iš menu pasirenkama pasirenkama transformavimo funkcija → Pasirenkamas transformacijos tipas → Vykdoma transformacija → Vartotojui pageidaujant transformuojama į kitą tipą, priešingu atveju transformacija užbaigiama.



15 pav. Transformavimo veiklos diagrama

Požymių diagramos susiejimas su UML diagrama atvaizduotas 16 paveiklse. Susiejimas vyksta tokia seka: Atidaroma FD2 formato požymių diagrama → Pasirenkamas požymis → Iš kontekstinio menu pasirenkamas požymio susiejimo su UML klase veiksmas → Išsirenkama UML klasių diagrama → Išsirenkama susiejama diagramos klase → Jeigu susiejimas įmanomas požymis ir klasė susiejami, priešingu atveju grįžtama į požymio pasirinkimo stadiją → Jeigu vartotojas pageidauja susieti daugiau požymių gali vykdyti susiejimą toliau, jeigu ne – susiejimas baigiamas ir galima vykdyti atskiro atvejo diagramos generavimą.

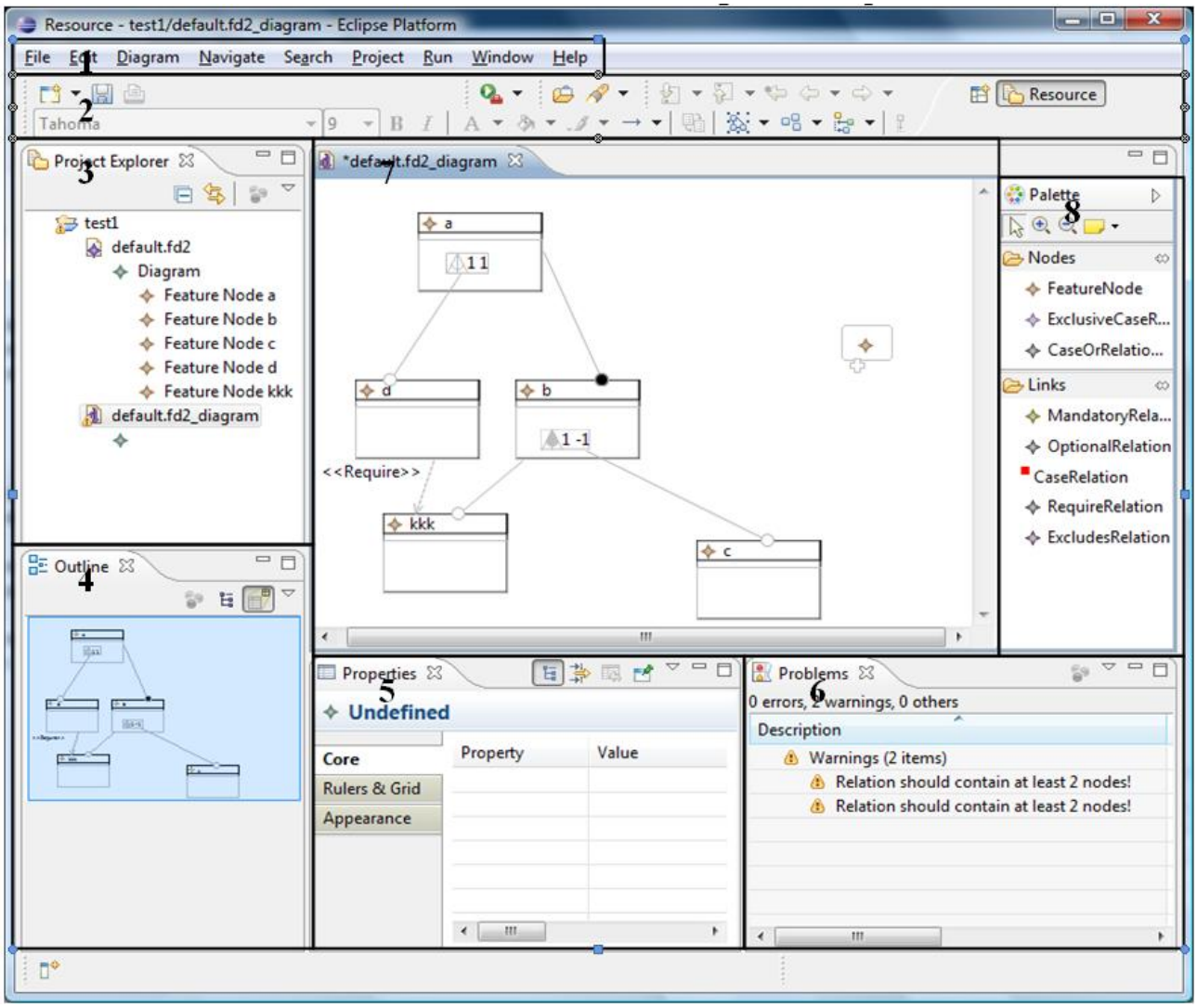


16 pav. Požymių diagramų susiejimas su UML klasių diagramom

7 POŽYMIŲ MODELIAVIMO IR TRANSFORMAVIMO ĮRANKIO APRAŠYMAS

7.1 Vartotojo sąsaja

Standartinis požymių modeliavimo ir transformavimo įrankio lango vaizdas ir vartotojo sąsaja pavaizduota 17 paveiksle ir aprašytą 19 lentelėje.



17 pav. Įrankio vartotojo sąsaja

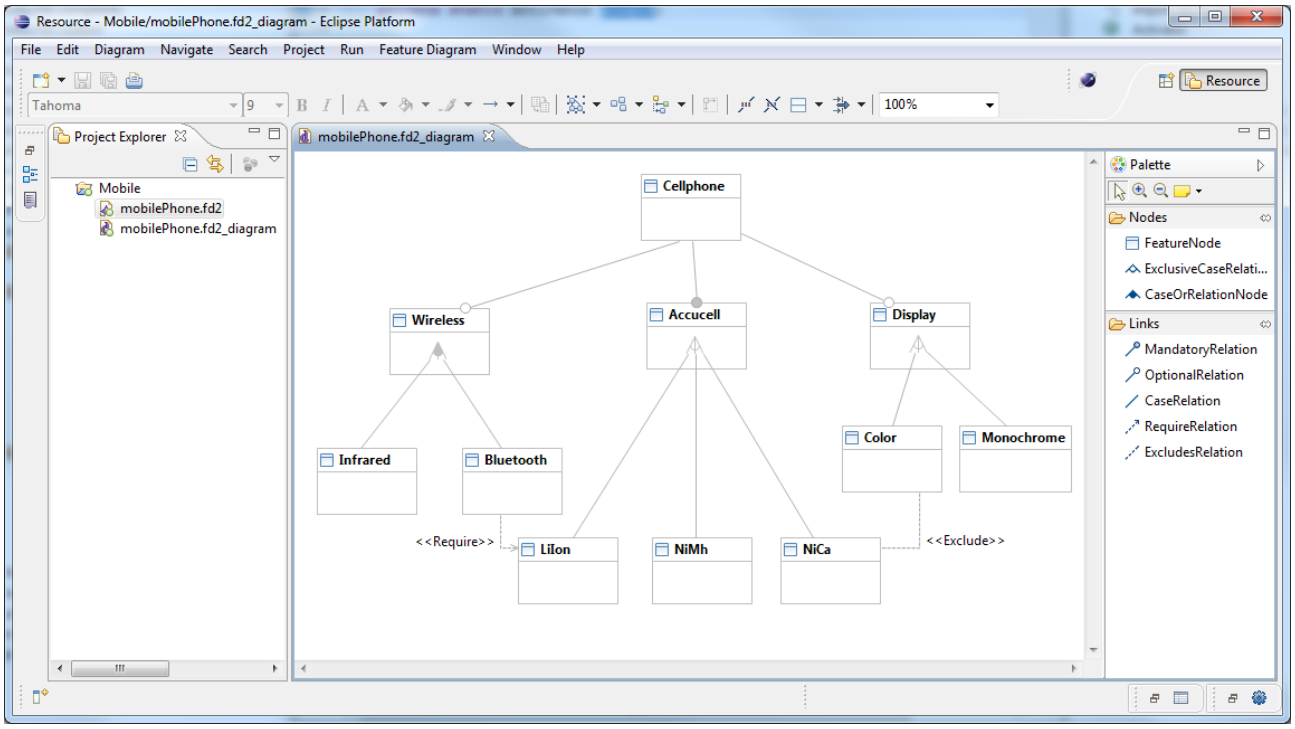
Įrankio vartotojo sąsaja suskirstyta į 8 dalis: menu, įrankių juosta, projekto naršyklė, apybraiža, savybės, klaidos, diagramų langas ir įrankių juosta. Kiekviena dalis suteikia paprastesnę prieigą prie modeliavime naudojamų įrankių ir tai padeda sutaupyti laiko. Taip pat reikėtų pabrėžti jog patys vartotojai gali susikonfigūruoti aplinką taip, kad jiems būtų patogiau dirbti

19 lentelė. Vartotojo sąsajos elementai

Vartotojo sąsajos elementas	**Aprašymas**
1. Menu	Menu pasirinkimai
2. Įrankių juosta	Pagrindinių įrankių juosta
3. Projekto naršyklė	Galimi modeliai, diagrama, ryšiai ir t.t.
4. Apybraiža	Sumažintas požymių diagramos vaizdas pavaizduotas diagramos lange
5. Savybės	Požymių modelio savybių sąrašas
6. Klaidos	Sąrašas klaidų aptiktų požymių modelyje
7. Diagramų langas	Langas požymių kūrimui, redagavimui ir atvaizdavimui
8. Įrankių dėžė	Įrankiai naudojami požymių diagramai kurti

7.2 Pavyzdinis požymių modelio transformavimo darbo scenarijus

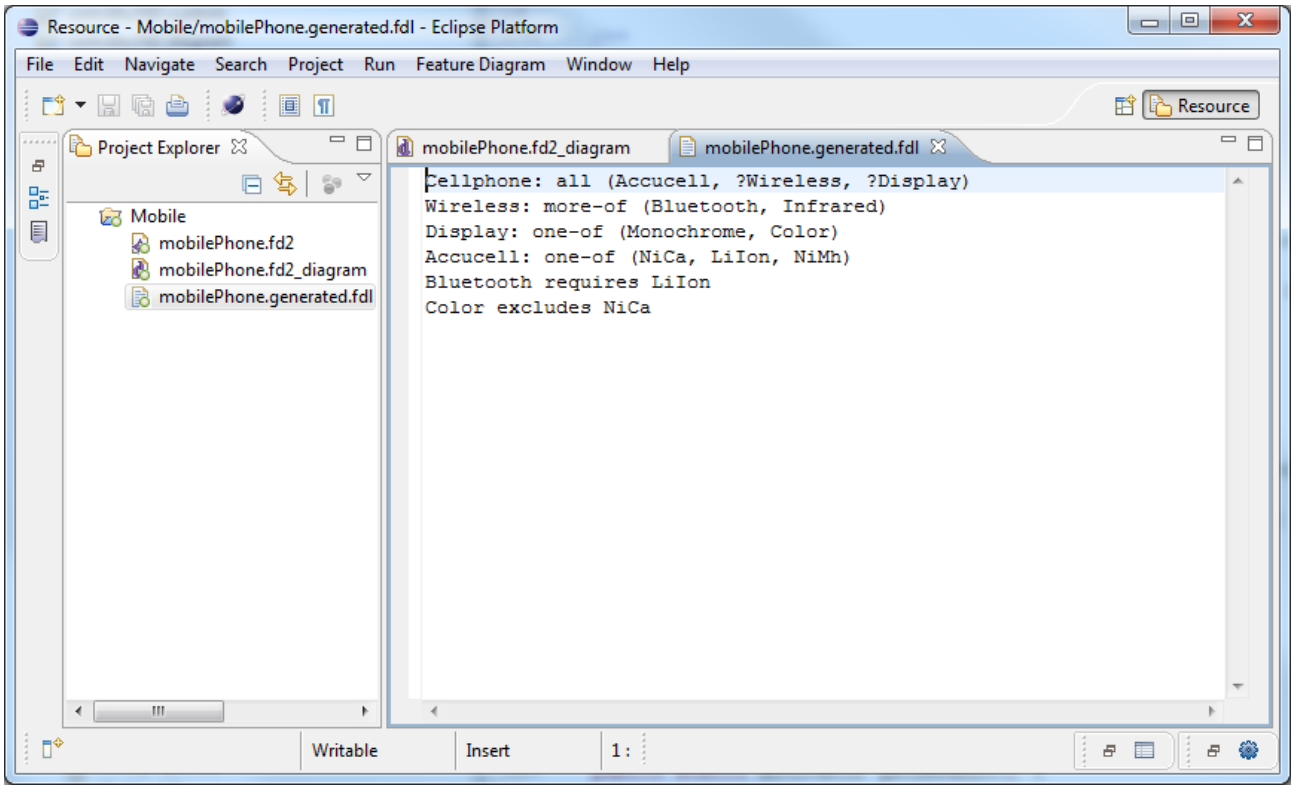
Šiame skyriuje bus aprašomas pavyzdinis požymių modelio transformavimo scenarijus. Pavyzdžiui, sukuriame mobiliojo telefono požymių modelį naudodami įrankį. Matome vaizdą, pateiktą 18 paveiksle: požymių diagrama, ir du failai (požymių modelio medžio ir požymių diagramos) projekto naršyklėje kairėje.



18 pav. Įrankiu sukurtas požymių modelis

Norėdami atlikti transformaciją, pavyzdžiui į FDL kalba aprašytą požymių modelį, pasirenkame atitinkamą punktą iš požymių modelio failo kontekstinio meniu. Gautas rezultatas

pateikiamas 19 paveiksle. Matome požymių modelį FDL kalba, taip pat lango kairėje esančioje projekto naršyklėje atsiradusį naują FDL failą.



19 pav. Įrankiu transformuotas požymių modelis

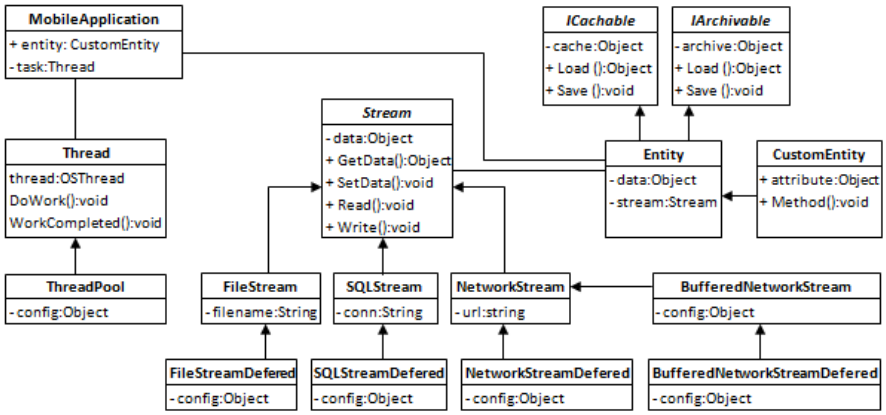
Analogiškai pasirinkus kitą transformavimo meniu punktą, galima atlikti kitų tipų požymių modelių transformacijas.

7.3 Pavyzdinis darbo scenarijus su požymių modelių ir UML klasių diagramų integracija

7.3.1 Pradinės klasių diagramos ir požymių modelio kūrimas

Kaip pavyzdį nagrinėsime produktų šeimos kūrimą naudojant mobiliojo telefono aplikacijos kūrimo galimybes atsižvelgiant į efektyvumą, veikimo greitį, reprezentatyvumą, kūrimo išlaidas. Vartotojai mobiliąją aplikaciją gali naudoti skirtingose aplinkose, ko pasekoje reikalingos skirtingos produkto konfigūracijos t.y. skirtingi požymių rinkiniai. Dirbdamas su mobiliąja aplikacija vartotojas gali pageidauti skirtingų failų laikymų būdų (failų sistema, SQL duomenų bazė, tinklo talpyklos), multi-taskinimo technikos (vienos gijos, multi-gijų) ir duomenų kešavimo arba suspaudimo laipsnis. Visos šios žemo lygio funkcijos turi poveikį aukšto lygio funkcijoms,

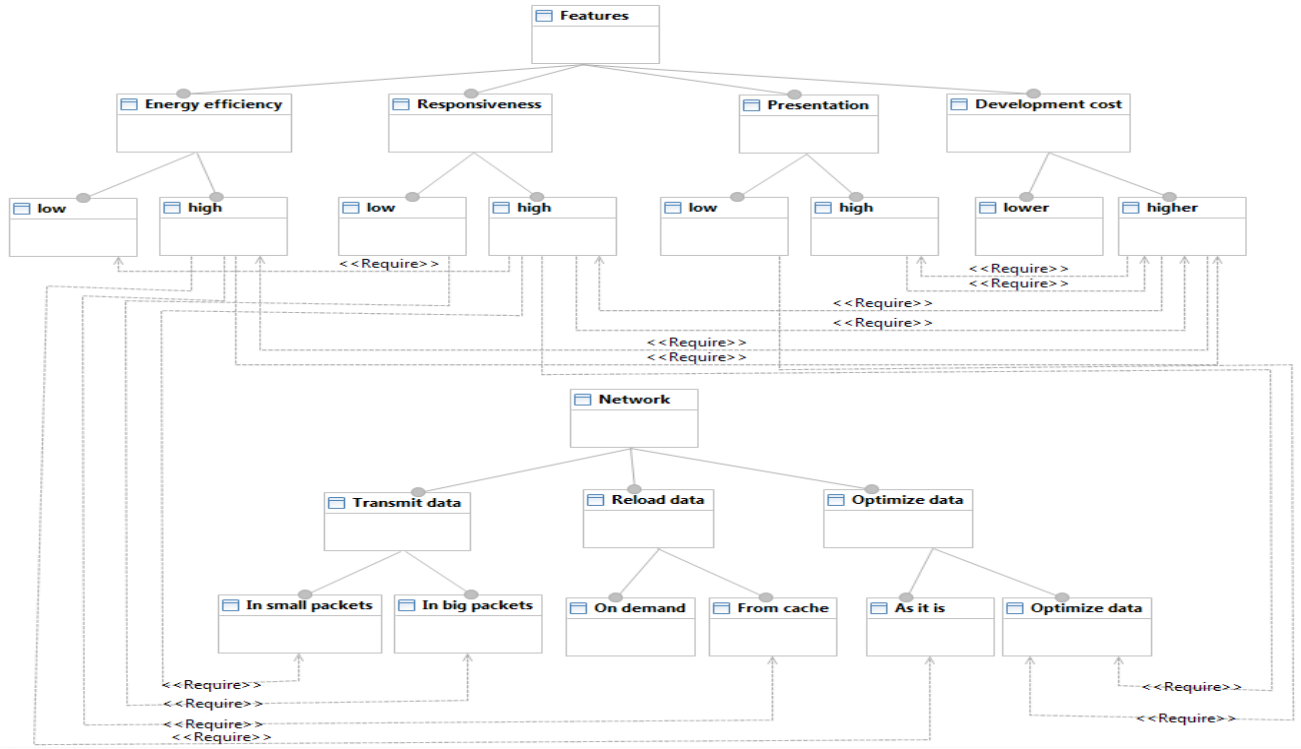
tokioms kaip: energijos išnaudojimo efektyvumas, atsako greitis, reprezentatyvumas ir kūrimo išlaidos.



20 pav. Mobilios aplikacijos UML klasių modelis

20 paveiksle pavaizduotas mobilios aplikacijos UML klasių modelis. Čia, Entity – ryšys apima informaciją apie mobilios aplikacijos objektą, Thread – gija ir ThreadPool – multi gijos apima vidinių procesų logiką veiksmus atliekant vienoje arba multi gijose, sql stream – sql srautas apima duomenų išlaikymą ir ICachable – kešavimas kartu su IArchivable – archyvavimas apibūdina keršavimo ir archyvavimo logikas. CustomEntity – pasirinkta esybė yra naudojama kaip mobilios aplikacijos duomenimis paremtas objektas, kuris susideda iš pasirinktos verslo logikos ir suteikia rašymo, skaitymo būdus, optimizavimo, bei suspaudimo būdus ir vienos, bei multi-gijų veikimo galimybes mobiliojoje aplikacijoje.

21 paveiksle yra pavaizduotas mobilios aplikacijos požymių modelis. Mobilios aplikacijos požymių specifikavimas padalintas į dvi dalis: vartotojo perspektyva (context – kontekstas) ir architekto perspektyva (product – produktas). Žvelgiant iš vartotojo perspektyvos mobili aplikacija reikalauja aukšto lygio požymių specifikacijos, tokių kaip: energijos naudojimo efektyvumas, atsako greitis, reprezentatyvumas ir kūrimo išlaidos. Kuomet į sistemą žvelgiama iš kūrėjo perspektyvos mobilioji aplikacija reikalauja žemesnio lygio požymių, tokių kaip tinklo duomenų siuntimas (dideliais arba mažais paketais), duomenų kešavimas, duomenų suspaudimo būdas.

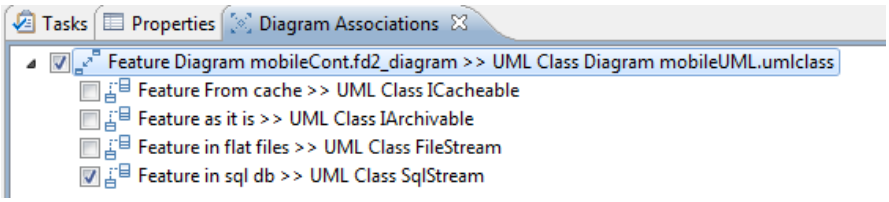


21 pav. Mobilios aplikacijos požymių modelis

7.3.2 Požymių ir klasių susiejimas bei konfigūracija

Požymių ir klasių susiejimas, tai procesas kurio metu požymis yra susiejamas su UML klasių diagrama. Susieti požymius ir klases galima naudojantis kontekstiniu FD2 meniu. Pavyzdinis susiejimo procesas:

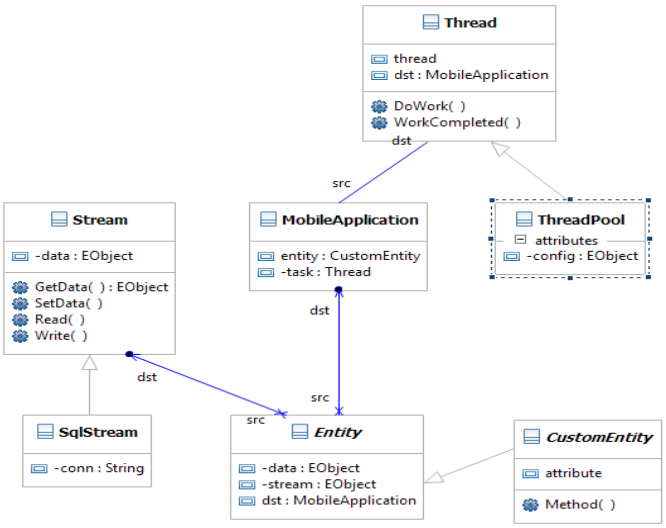
1. Pasirenkame požymį ir spaudžiame dešinįjį pelės mygtuką ant jo,
2. Iš kontekstinio meniu pasirenkame „Assign to UML Class“, tuomet atveriama klasių diagrama,
3. Pasirenkame klasę iš UML klasės diagramos ir spaudžiame dešinįjį pelės klavišą,
4. Iš kontekstinio meniu pasirenkame „Assign to Feature“, naujas įrašas atsiranda „Diagram Associations“ (22 pav.) susiejimo lange.



22 pav. Požymių konfigūravimo procesas

7.3.3 Atskiro atvejo UML klasių modelio sukūrimas

Kuomet atliekamas visas susiejimo procesas, galima atlikti naujos sistemos konfigūraciją. Tai paprastas procesas: tereikia sužymėti kuriuos susietus elementus naudosime naujoje diagramoje. Žiūrėti pavyzdį 22 paveiksle. Kuomet pasirenkama visi elementai iš kurių bus sudaroma nauja UML klasių diagrama, reikia paspausti generavimo mygtuką. Naujoji diagrama susidės tik iš pasirinktų elementų. (žiūrėti paveikslas 23, ir palyginti su paveikslas 20).



23 pav. Atskiro atvejo UML klasių modelis

8 POŽYMIŲ MODELIAVIMO IR TRANSFORMAVIMO ĮRANKIO KOKYBĖS ĮVERTINIMAS

8.1 Įvadas

Šiame skyriuje pateikiama informacija susijusi su požymių modeliavimo ir transformavimo įrankio kokybės įvertinimu, testavimu. Aprašomas įrankio kodo tyrimas naudojant kokybės bei kiekybės metrikas, pačios metrikos; taip pat įrankio testavimo sąlygos, procedūros, metodologijos.

8.2 Įrankio kodo tyrimas naudojant kokybės ir kiekybės metrikas

8.2.1 Kodo tyrimo naudojant kokybės ir kiekybės metrikas tikslai

Požymių modeliavimo ir transformavimo įrankio kodo tyrimo naudojant kokybės ir kiekybės metrikas esminiai tikslai:

- įvertinti įrankio kodo sudėtingumą, suprantamumą ir palaikomumą;
- nustatyti įrankio kodo silpniausias, taisytinas vietas.

8.2.2 Kodo kokybės ir kiekybės metrikų aprašymas

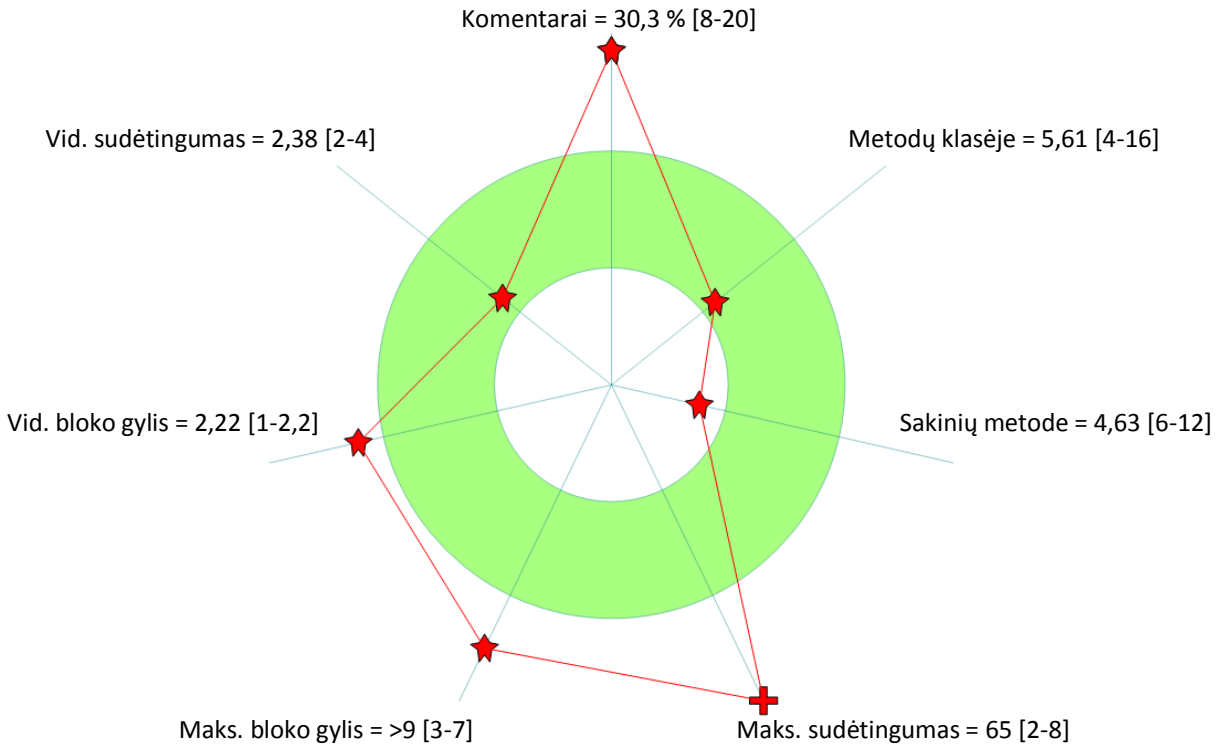
Požymių modeliavimo ir transformavimo įrankio *Java* kodas buvo tiriamas ir įvertintas naudojant kodo metrikas, tokias kaip:

- failų skaičius – bendras įrankio kodo failų skaičius;
- eilučių skaičius – bendras įrankio kodo eilučių skaičius;
- sakinių skaičius – mažiausių atskirųjų (angl. *standalone*) imperatyvios programavimo kalbos elementų skaičius;
- metodų iškvietimo sakinių skaičius – sakinių, kuriuose iškviečiami metodai, skaičius;
- išsišakojimo sakinių procentinė dalis – sakinių, kuriuose algoritmo seka šakojasi, procentinė dalis iš visų sakinių;
- komentaro eilučių procentinė dalis – eilučių su komentaru procentinė dalis iš visų eilučių;
- klasių ir interfeisų skaičius – programos kodo objektinių sudedamųjų dalių – klasių ir interfeisų skaičius;
- vidutinis metodų skaičius klasėje – metodų ir klasių kiekių santykis;
- vidutinis sakinių skaičius metode – sakinių ir metodų kiekių santykis;
- vidutinis bloko gylis (angl. *average nested block depth*) – kodo blokų gylių vidurkis;
- maksimalus bloko gylis (angl. *max nested block depth*) – maksimalus kodo blokų gylis;
- vidutinis sudėtingumas – visų įrankio metodų sudėtingumų (pagal S. McConnell [39]) vidurkis;
- maksimalus sudėtingumas – sudėtingiausio įrankio metodo sudėtingumas (pagal S. McConnell [39]).

8.2.3 Įrankio kodo tyrimo naudojant kokybės ir kiekybės metrikas rezultatai

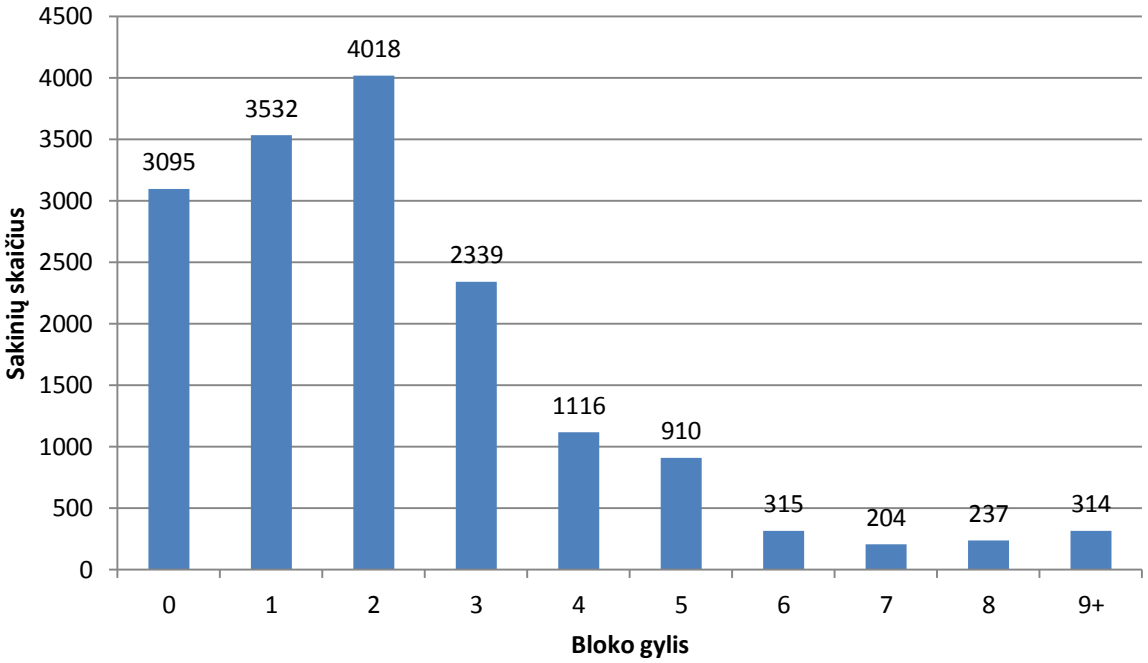
Pagrindinės įrankio kodo metrikos Kiviat diagramos pavidalu pateiktos 24 paveiksle, pagal kurį matome, kad vidutinio metodų skaičiaus klasėje ir vidutinio sakinių skaičiaus metode metrikos yra labai žemos, vadinasi, kodas yra itin gerai struktūrizuotas. Vidutinio sudėtingumo bei vidutinio bloko gylio metrikos yra gana žemos, kas parado, jog didžioji kodo dalis nėra sudėtinga ar paini. Vis dėl to maksimalus bloko gylis ir maksimalus sudėtingumas yra aukšti – tai reiškia, kad įrankio kodas turi kritinių vietų, kurias reikėtų optimizuoti, pertvarkyti (angl. *refactor*). Galiausiai

matome, kad komentarų eilučių procentinė dalis yra labai aukšta, kas parodo, kad kodas yra išsamiai aprašytas. Ši metrika iš dalies susijusi su nedideliu vidutiniu sakinių skaičiumi metoduose. Tai parodo didesnį metodų skaičių, kur kiekvienas jų aprašomas komentaru.



24 pav. Įrankio kodo metrikų Kiviat diagrama

Detaliau ištirta įrankio kodo sakinių skaičiaus priklausomybė nuo bloko gylio. Kaip matome paveiksle 25, nuo pradinio nulinio iki antrojo bloko gylio sakinių skaičius auga. Antrojo gylio blokuose sakinių skaičius yra didžiausias ir tai yra gerai. Gilesniuose blokuose sakinių skaičius ženkliai mažėja, bet nedidelis kiekis sakinių išlieka ir septinto bei didesnio gylio blokuose, ko reikėtų vengti, nes tai apsunkina kodo suprantamumą ir palaikomumą.



25 pav. Įrankio kodo sakinių skaičius pagal blokų gylius

Visas išmatuotų ir nagrinėtų įrankio kodo kiekybės bei kokybės metrikų sąrašas pateikiamas 20 lentelėje.

20 lentelė. Įrankio kodo kiekybės bei kokybės metrikos

Parametras	**Reikšmė**
Failų skaičius	213
Eilučių skaičius	36238
Sakinių skaičius	14135
Metodų iškvietimo sakinių skaičius	8829
Išsišakojimo sakinių procentinė dalis	15,6 %
Komentaro eilučių procentinė dalis	30,3 %
Klasių ir interfeisų skaičius	345
Vid. metodų skaičius klasėje	5,61
Vid. sakinių skaičius metode	4,63
Vid. bloko gylis	2,22
Maks. bloko gylis	>9
Vid. sudėtingumas	2,38
Maks. sudėtingumas	65

8.2.4 Kodo tyrimo naudojant kokybės ir kiekybės metrikas išvados

Apibendrinant kodo tyrimo naudojant kokybės ir kiekybės metrikas rezultatus, galima pasakyti, kad kodas gerai struktūrizuotas, vidutinis apskaičiuotas kodo sudėtingumas nėra didelis, vidutinis kodo bloko gylis taip pat pakankamai žemas, komentarų eilučių procentinė dalis aukšta. Visa tai rodo gana aukštą kodo kokybę. Nustatyta pagrindinė silpnoji vieta – maksimalus sudėtingumas kritinėse vietose labai aukštas. Šias kodo vietas, tokias kaip požymių diagramų atvaizdavimo ir redagavimo modulį, reikėtų perrašyti, struktūrizuoti.

8.3 Įrankio testavimas

8.3.1 Testavimo tikslai ir objektai

Kuriant programinę įrangą vienas iš kūrimo tikslų yra pateikti programinę įrangą su kuo mažesniu defektų kiekiu. Šiame skyriuje aprašomas įrankio testavimo procesas, užtikrinantis kiek galima mažesnį defektų kiekį. Svarbiausias testavimo rezultatas – teisingas įrankio logikos vykdymas. Mažiau reikšmingos, tačiau taip pat labai svarbios sritys yra vartotojo sąsajos patogumas bei veikimo sparta.

8.3.2 Testavimo apimtis ir tipai

Įrankio testavimą galima suskirstyti į šias dalis:

- vienetų testavimas,
- integracinis testavimas,
- grafinės sąsajos testavimas,
- transformacijų testavimas,
- priėmimo testavimas.

Vienetų testavimas atsakingas už atskirų, pavienių metodų, algoritmų testavimą, kai tikrinamas jo veikimo teisingumas. Integracinio testavimo metu buvo imama viena klasė ir sujungiama su daugiau besisiejančių klasių. Tokiu būdu buvo galima sužinoti apie įrankio visumos veikimą.

Grafinės sąsajos testavimo metu buvo tikrinama, ar sąsajos elementai veikia korektiškai įvairiose aplinkose. Kadangi sistema pritaikyta dirbti įvairiose OS (Windows, Linux, Mac), todėl buvo tikrinama kaip atrodo ir funkcionuoja grafinė sąsaja šiose OS. Taip pat buvo atsižvelgiama į sąsajos patogumą vartotojui.

Transformacijų testavimas būtent šiam projektui specifinis testavimo būdas, kurio metu buvo tikrinamas transformacijų algoritmų teisingumas. Visos įrankio atliekamos transformacijos buvo realizuotos dvikryptės, t.y. jei galimas transformavimas iš notacijos A į notaciją B, turi būti galimas transformavimas ir atgal – iš notacijos B į notaciją A. Taigi testuojant transformacijas buvo imami šabloniniai požymių modeliai tam tikroje pradinėje notacijoje, atliekamas transformavimas į kitą notaciją ir gautam rezultatui vėl atliekamas transformavimas į pradinę notaciją. Galutinis produktas lyginamas su pradiniu šabloniniu modeliu ir tikrinama ar jie ekvivalentiški. Jei abu požymių modeliai ekvivalentiški, vadinasi transformacijos atliktos korektiškai; kitu atveju – kažkuriame transformacijų žingsnyje yra klaidų.

Priėmimo testavimo metu buvo atliekamas patikrinimas pagal „juodos dėžės“ principą užbaigtam produktui. Sistema buvo nuodugniai ištestuota pagal reikalavimų specifikaciją ir patikrinta, ar sistema atitinka vartotojo poreikius. Radus neatitikimą tarp sistemos ir vartotojo poreikių, buvo patikrinama reikalavimų specifikacijoje. Radus sistemos ir specifikacijos neatitikimą buvo registruojama klaida.

8.3.3 Reikalavimai testavimui

Norint atlikti programinės įrangos testavimą reikalinga:

- Pilna programinės įrangos specifikacija išreikšta panaudojimo atvejų diagramomis ir panaudojimo scenarijais.
- Pilna ir testavimui parengta programinė įranga.
- Nustatyta procedūra, ką daryti su aptiktomis klaidomis.
- Klaidos ataskaitos šablonas.

8.3.4 Testavimo prioritetai

Atliekant sistemos testavimą pagrindinis dėmesys bus kreipiamas į šiuos faktorius. Jie pateikti svarbumo mažėjimo tvarka.

- Funkcionalumas – ar reikalingos programinės įrangos funkcijos yra ir atliekam joms priskirtas roles taip kaip reikia (atitinka specifikaciją)
- Saugumas – ar vartotojas gali prieiti ir dirbti tik su jam skirtais duomenimis.
- Panaudojamumas – kaip vartotoja sąsaja yra patogi vartotojui ir draugiška.

- Našumas – ar programinė įranga dirba laikantis specifikacijoje nurodytais našumo reikalavimais.

8.4 Formalios techninės peržiūros

Java programavimo kalba – objektinė programavimo kalba, todėl programoje naudojamos duomenų struktūros taip pat naudojamos objektiškai – projekte naudojamas objektinis programavimas. Programuojant naudojami modeliavimo šablonai (angl. „design patterns“), kurie padaro programinį kodą lengviau suprantamą, palaikomą ir perpanaudojamą.

8.5 Reikalavimų išpildymas

Reikalavimų išpildymo patikrinimo rezultatai pateikiami 21 lentelėje.

21 lentelė. Reikalavimų išpildymas

Reikalavimas	**Išpildymas**
Transformavimas iš FD2 į FDL	Išpildyta
Transformavimas iš FDL į FD2	Išpildyta
Transformavimas iš FD2 į PROLOG	Išpildyta
Transformavimas iš PROLOG į FD2	Išpildyta
Transformavimas iš FDL į PROLOG	Išpildyta (naudojant dviejų žingsnių transformavimą: FDL → FD2 → PROLOG)
Transformavimas iš PROLOG į FDL	Išpildyta (naudojant dviejų žingsnių transformavimą: PROLOG → FD2 → FDL)
Požymių modelio klaidų tikrinimas	Sąlyginai – klaidų patikrinimas realizuotas, bet kai kurios sudėtingai aptinkamos klaidos praleidžiamos.

8.6 Kokybės įvertinimas pagal kriterijus

Programinės įrangos kokybė įvertinta naudojant 22 lentelėje pateiktus apsibrėžtus kriterijus.

22 lentelė. Kokybės įvertinimo kriterijai

Kriterijus	**Aprašymas**
Saugumas	Programinės įrangos duomenų saugumas
Išplečiamumas	Galimybė praplėsti programinės įrangos funkcijas. Naujų modulių kūrimo galimybė.
Pernešamumas	Galimybė perkelti programinę įrangą ant kitokios techninės įrangos, operacinės sistemos.
Sąsajos galimybės	Ar gali veikti su kitomis sistemomis.
Panaudojamumas	Ar lengva išmokti dirbti su programine įranga.
Patvarumas	Kiek tolerantiška sistema vartotojo klaidoms?
Atsparumas tinklo gedimams	Ar sugeba programinė įranga dirbti nesant tinklui, ar nepraras duomenų jam dingus.

Saugumas

Sistema suprojektuota ir sukurta kaip atsiras sistemos komponentas, todėl dalis saugumo priklauso nuo pasirinkto produkto saugumo sprendimų.

Išplečiamumas

Sistema pagal vartotojo reikalavimus lengvai išplečiama, kadangi ji sukurta kaip atskiras pasirinktos sistemos įskiepis (komponentas), dėl šio privalumo naujų modulių kūrimas yra lengvai įgyvendinamas.

Pernešamumas

Sistema suprogramuota naudojant *Java* programavimo kalbą, taigi ji puikiai veikia *Windows, Mac OS* ir *Unix/Linux* (kurios palaiko grafinę sąsają).

Sąsajos galimybės

Sistema naudoja požymių modelius ir jų diagramas keliais formatais, tarp kurių yra:

- specifinis *FD2* formatas, leidžiantis išsaugoti tiek požymių modelio medžio duomenis, teik grafinės požymių diagramos duomenis;
- plačiai paplitęs *FDL* formatas leidžiantis vartotojui įkelti į sistemą požymių modelį iš kitur;
- *PROLOG* formatas leidžiantis požymių modelį naudoti *PROLOG* aplinkose.

Taip pat sistema veikia kartu su *UML* diagramų braižymo komponentu, išplėsdama jo galimybes požymių diagramų braižymu bei įgalina apjungti diagramas ir atlikti konfigūravimą.

Panaudojamumas

Sistemoje diagramos braižomos naudojant grafinius elementus, juos valdant pelės pagalba. Taip pat sistemoje yra integruota pagalbos funkcija, kurioje pateikiamas sistemos vartotojo vadovas.

Patvarumas

Sistema informuoja vartotoją apie klaidingų ryšių ar elementų naudojimą diagramose, pateikdama detalesnę informaciją apie kiekvieno elemento naudojimo galimybes.

Atsparumas tinklo gedimams

Sistemai veikti interneto ryšys nėra būtinas. Diagramos braižomos lokaliame kompiuteryje, ir turi išsaugojimo į dokumentą galimybę.

8.7 Išvados

Realizuotas požymių modeliavimo ir transformavimo įrankis tinkamas grafiniam požymių modelių kūrimui, atvaizdavimui bei modelių transformavimui į kitas notacijas.

CASE priemonių naudojimas projekte paspartino įrankio sukūrimą, palengvino įvairias programavimo bei testavimo užduotis, taip pat leido pasiekti aukštesnę įrankio kokybę su geresnėmis perpanaudojamumo, suprantamumo, palaikomumo savybėmis. Šias savybes patvirtina atlikta įrankio kodo analizė naudojant kodo kokybės ir kiekybės metrikas.

9 POŽYMIŲ MODELIAVIMO IR TRANSFORMAVIMO ĮRANKIO TYRIMAS

9.1 Tyrimo tikslai

Esminiai požymių modeliavimo ir transformavimo įrankio tyrimo tikslai:

- ištirti visus įrankio atliekamų transformacijų tipus;
- nustatyti ar transformacijos atliekamos korektiškai;
- aptikus nekorektiškų transformacijų, nustatyti, kurie apsibrėžti požymių modelių transformavimo korektiškumo kriterijai tenkinami, o kurie – ne.

9.2 Požymių modelių biblioteka

Transformacijoms tirti buvo naudojami modeliai iš požymių modelių bibliotekos, kurioje yra 18 skirtingų požymių modelių aprašytų *FDL* kalba:

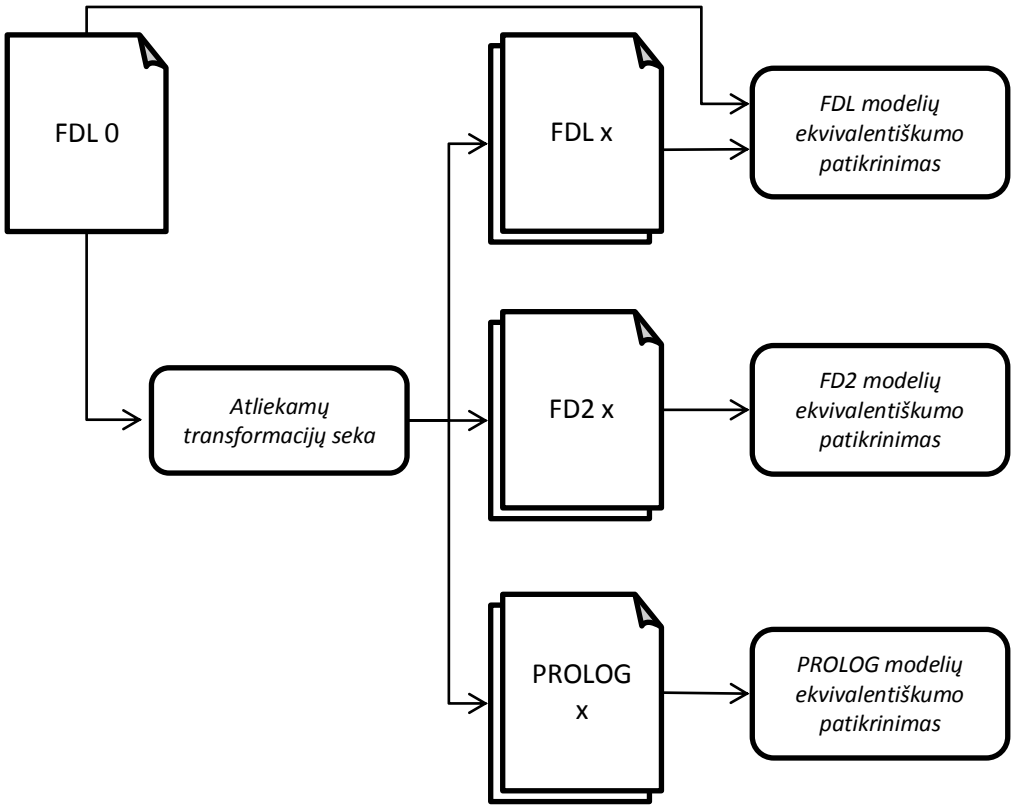
1. *Cellphone-fm* – mobiliojo telefono požymių modelis;
2. *Digital-video-system-fm* – skaitmeninės vaizdo sistemos požymių modelis;
3. *Documentation-generation-fm* – dokumentų generatoriaus požymių modelis;
4. *Eshop-fm* – el. parduotuvės požymių modelis;
5. *Graph-manipulation* – grafų redaktoriaus požymių modelis;
6. *Graph-product-line-fm* – grafais paremtos produktų linijos požymių modelis;

7. *Home-integration-system-fm* – integruotos namų saugumo sistemos požymių modelis;
8. *Insurance-product-fm* – apdaužiamo produkto požymių modelis;
9. *James-fm* – web bendradarbiavimo sistemų karkaso požymių modelis;
10. *Jplug-fm* – Jplug IDE sistemos požymių modelis;
11. *Key-word-in-context-fm* – raktinių žodžių indeksavimo sistemos požymių modelis;
12. *Model-transformation-fm* – modelių transformatoriaus požymių modelis;
13. *Monitor-engine-system-fm* – variklio stebėjimo sistemos požymių modelis;
14. *Telecommunication-system-fm* – telekomunikacinės sistemos požymių modelis;
15. *Text-editor-fm* – tekstų redaktoriaus požymių modelis;
16. *Thread-domain-fm* – daugiagijės sistemos požymių modelis;
17. *Virtual-office-of-future-fm* – virtualaus biuro požymių modelis;
18. *Web-portal-fm* – žiniatinklio portalo požymių modelis.

9.3 Transformacijų tyrimo procesas

Bendras įrankio vykdomų požymių modelių transformacijų tyrimo procesas, pateikiamas paveiksle 26. Šis procesas buvo kartojamas kiekvienam bibliotekos požymių modeliui.

Pradinis *FDL* kalba aprašytas požymių modelis iš požymių modelių bibliotekos žemiau pateiktame paveiksle pažymėtas *FDL 0*. Šis modelis naudojamas kaip įvesties duomenys tam tikrai transformacijų sekai, kurių galutinis produktas – trys skirtingų formatų (*FDL*, *FD2* ir *PROLOG*) požymių modelių aibės, paveiksle atitinkamai pažymėtos kaip *FDL x*, *FD2 x* ir *PROLOG x*. Galiausiai tikrinamas šias aibes sudarančių modelių ekvivalentiškumas tarpusavyje. Apibendrinus ekvivalentiškumo patikrinimo aibėse rezultatus, galima teigti, kurios transformacijos buvo korektiškai, o kurios – ne.



26 pav. Požymių modelių transformacijų tyrimo procesas

Visos atliekamos transformacijos požymių modelius transformuoja į kitą notaciją, tačiau pats požymių modelis turi išlikti semantiškai nepakitęs, ekvivalentiškas pirmtakui. Taigi ir gautų aibių požymių modeliai turi būti ekvivalentiški. Neekvivalentiškų požymių modelių aptikimas aibėje reikštų, kad kažkuri transformacija atlikta nekorektiškai.

9.4 Transformacijų tipai ir jų seka

Transformacijų tyrimui naudojama baigtinė įrankio atliekamų transformacijų tipų aibė, kurią sudaro:

- transformacija iš *FDL* į *FD2*;
- transformacija iš *FD2* į *FDL*;
- transformacija iš *PROLOG* į *FD2*;
- transformacija iš *FD2* į *PROLOG*.

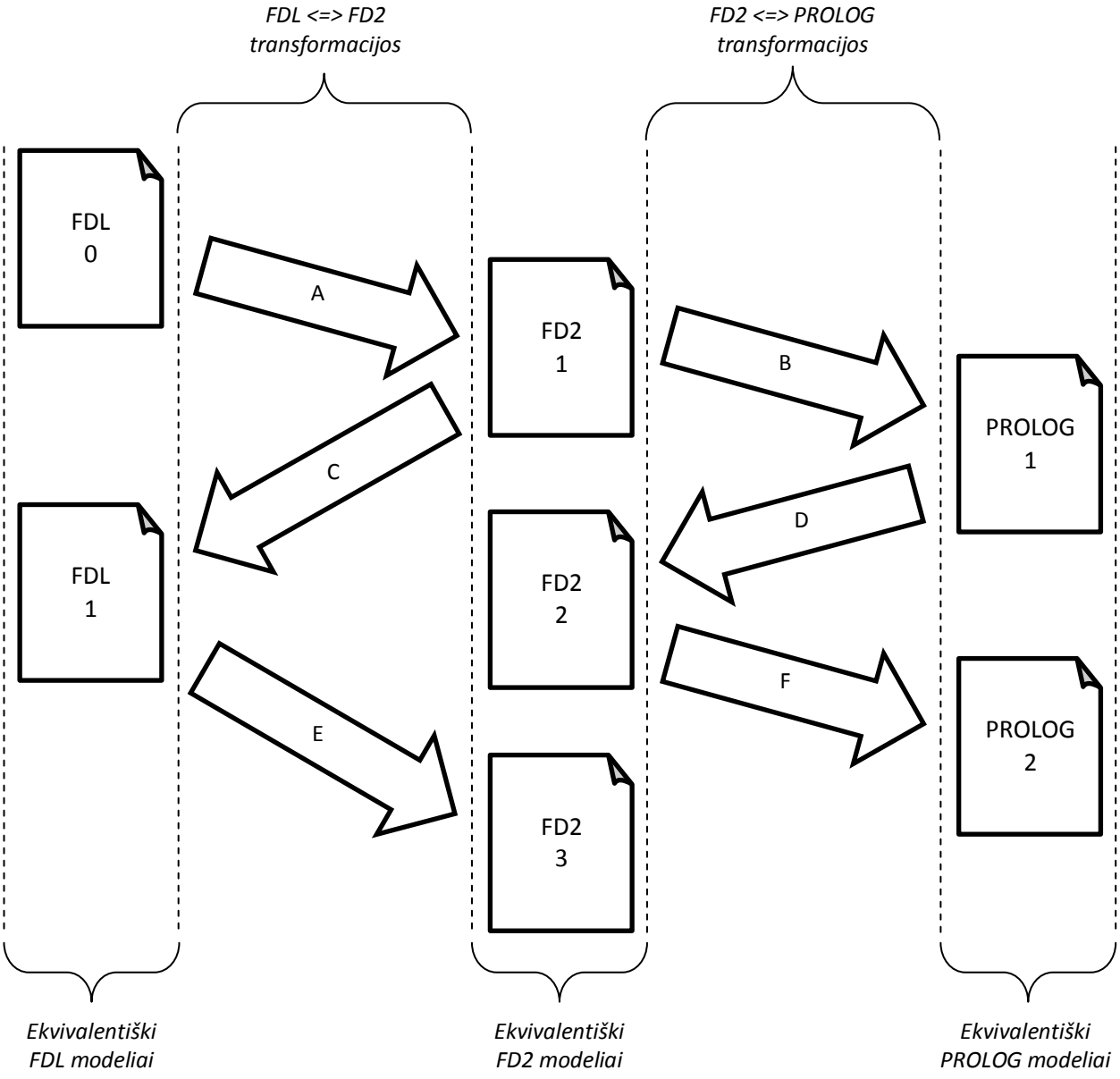
Pasitelkiant aukščiau pateiktą transformacijų tipų aibę, taip pat galimos šios transformacijos:

- transformacija iš *FDL* į *PROLOG* (panaudojant transformacijas: *FDL* → *FD2* → *PROLOG*);

- transformacija iš *PROLOG* į *FDL* (panaudojant transformacijas: *PROLOG* → *FD2* → *FDL*).

Tyrimo metu, kiekvienam atskiram bibliotekos požymių modeliui buvo naudojama tam tikra, 27 paveiksle pateikta transformacijų seka. Šioje sekoje kiekvienas transformacijos tipas panaudojamas bent vieną kartą.

Pagal žemiau pateiktą paveikslą matome, kad sekoje, kiekvienam pradiniam požymių modeliui iš viso atliekamos 6 transformacijos ir taip gaunami 6 nauji požymių modeliai – transformacijų produktai.



27 pav. Požymių modelių transformacijų tyrime naudojamų transformacijų seka

Transformacijų seka, pateikiama 27 paveiksle, pasirinkta todėl, kad ši transformacijų seka turi šias sudedamąsias transformacijų kombinacijas:

1. *FDL* → *FD2* → *FDL* – transformacijos: A ir C;
2. *FD2* → *FDL* → *FD2* – transformacijos: C ir E;
3. *FD2* → *PROLOG* → *FD2* – transformacijos: B ir D;
4. *PROLOG* → *FD2* → *PROLOG* – transformacijos: D ir F;

Šios transformacijų kombinacijos pasirinktos, nes jos apima visų įrankio atliekamų tipų transformacijas kartu su nuosekliai vykdomomis atvirkštinėmis transformacijomis. Taip pradinis ir galutinis požymių modelis gaunamas tos pačios notacijos, kas leidžia lengviau palyginti požymių modelių ekvivalentiškumą.

9.5 Požymių modelių ekvivalentiškumo palyginimo metodai

9.5.1 FD2 požymių modelių ekvivalentiškumo palyginimo metodas

FD2 modelių ekvivalentiškumui patikrinti, visų pirma buvo naudojamos įrankio apskaičiuojamos požymių modelių metrikos, tokios kaip:

- požymių skaičius požymių modelyje;
- privalomumo ryšių skaičius;
- pasirenkamumo ryšių skaičius;
- alternatyvumo ryšių skaičius;
- „vienas ir daugiau“ („*arba*“) ryšių skaičius;
- apribojimų („*requires*“ ir „*excludes*“) skaičius;
- variacijos taškų skaičius – požymių modelio medžio viršūnių, kuriose galimos variacijos, skaičius;
- medžio gylis – požymių modelio medžio gylis.

Jei šių metrikų reikšmės tarp dviejų požymių modelių nesutampa, vadinasi tie požymių modeliai nėra ekvivalentiški. Bet jeigu metrikų reikšmės sutampa, tai dar neįrodo modelių ekvivalentiškumo. Visgi metrikų palyginimas tinkamas preliminariam ekvivalentiškumo patikrinimui. Sekantis patikrinimo žingsnis – rankinis požymių modelių palyginimas naudojant įrankį.

9.5.2 FDL ir PROLOG požymių modelių ekvivalentiškumo palyginimo metodas

FDL ir *PROLOG* kalbomis aprašytų požymių modelių metrikų apskaičiavimas įrankyje nerealizuotas, todėl naudoti metrikų požymių modelių, aprašytų šiomis kalbomis, ekvivalentiškumui tikrinti netikslinga. Šių požymių modelių ekvivalentiškumui tikrinti buvo atliekama tekstinė analizė naudojant *diff* tekstinių failų palyginimo įrankį. Šis įrankis parodo neatitikimus tarp dviejų tekstinių failų, dažniausiai naudojamas įvairių programavimo kalbų skirtingų versijų programinio kodo failams palyginti.

9.6 Požymių modelių ekvivalentiškumo palyginimo rezultatai

9.6.1 FD2 požymių modelių ekvivalentiškumo palyginimo rezultatai

FD2 požymių modelių ekvivalentiškumo patikrinimo rezultatai pateikti lentelėje 23. Tarpusavyje lyginami kiekvieno požymių modelių bibliotekos modelio pradinis *FD2* modelis (23 paveiksle žymimas „*FD2 1*“) ir transformacijų rezultatai (paveiksle žymimi „*FD2 2*“ ir „FD2 3“). Lentelėje pateikiamos visų šių požymių modelių metrikos.

23 lentelė. FD2 požymių modelių ekvivalentiškumo palyginimas

Požymių modelio pavadinimas	Modelio žymėjimas transformacijų sekoje	Variacijos taškai	Medžio gylis	Požymiai	Privalomumo ryšiai	Pasirenkamumo ryšiai	Alterenatyvumo ryšiai	„Vienas ir daugiau“ ryšiai	Apribojimai
Cellphone-fm	FD2 1	4	3	11	2	1	5	2	2
	FD2 2	4	3	11	2	1	5	2	2
	FD2 3	4	3	11	2	1	5	2	2
Digital-video-system-fm	FD2 1	9	4	24	7	18	0	0	3
	FD2 2	9	4	24	7	18	0	0	3
	FD2 3	9	4	24	7	18	0	0	3
Documentation-generation-fm	FD2 1	11	5	44	7	3	9	24	8
	FD2 2	11	5	44	7	3	9	24	8
	FD2 3	11	5	44	7	3	9	24	8
Eshop-fm	FD2 1	x	x	x	x	x	x	x	x
	FD2 2	x	x	x	x	x	x	x	x
	FD2 3	x	x	x	x	x	x	x	x

Graph-manipulation	FD2 1	7	6	30	16	7	6	0	10
	FD2 2	7	6	30	16	7	6	0	10
	FD2 3	7	6	30	16	7	6	0	10
Graph-product-line-fm	FD2 1	5	3	20	4	2	6	7	14
	FD2 2	5	3	20	4	2	6	7	14
	FD2 3	5	3	20	4	2	6	7	14
Home-integration-system-fm	FD2 1	14	7	55	43	10	11	0	4
	FD2 2	14	7	55	43	10	11	0	4
	FD2 3	14	7	55	43	10	11	0	4
Insurance-product-fm	FD2 1	6	4	25	9	1	8	7	**4**
	FD2 2	6	4	25	9	1	8	7	**5**
	FD2 3	6	4	25	9	1	8	7	4
James-fm	FD2 1	4	3	14	4	1	2	6	2
	FD2 2	4	3	14	4	1	2	6	2
	FD2 3	4	3	14	4	1	2	6	2
Jplug-fm	FD2 1	5	3	14	3	6	2	2	2
	FD2 2	5	3	14	3	6	2	2	2
	FD2 3	5	3	14	3	6	2	2	2
Key-word-in-context-fm	FD2 1	8	5	26	9	2	12	2	3
	FD2 2	8	5	26	9	2	12	2	3
	FD2 3	8	5	26	9	2	12	2	3
Model-transformation-fm	FD2 1	32	8	87	18	12	27	31	0
	FD2 2	32	8	87	18	12	27	31	0
	FD2 3	32	8	87	18	12	27	31	0
Monitor-engine-system-fm	FD2 1	3	4	17	10	1	2	3	1
	FD2 2	3	4	17	10	1	2	3	1
	FD2 3	3	4	17	10	1	2	3	1
Telecommunication-system-fm	FD2 1	4	4	12	3	3	5	0	3
	FD2 2	4	4	12	3	3	5	0	3
	FD2 3	4	4	12	3	3	5	0	3
Text-editor-fm	FD2 1	6	4	18	6	8	3	0	0
	FD2 2	6	4	18	6	8	3	0	0
	FD2 3	6	4	18	6	8	3	0	0
Thread-domain-fm	FD2 1	18	12	46	13	15	17	0	0
	FD2 2	18	12	46	13	15	17	0	0
	FD2 3	18	12	46	13	15	17	0	0
Virtual-office-of-future-fm	FD2 1	5	5	12	6	5	2	0	0
	FD2 2	5	5	12	6	5	2	0	0
	FD2 3	5	5	12	6	5	2	0	0
Web-portal-fm	FD2 1	15	5	43	8	17	7	10	6
	FD2 2	15	5	43	8	17	7	10	6
	FD2 3	15	5	43	8	17	7	10	6

Kaip matome pagal rezultatus – 16 iš 18 atskirų požymių modelių skirtingų transformacijų produktų metrikos sutapo. Išsamesnė analizė tik patvirtino, kad šie požymių modeliai buvo transformuoti korektiškai.

Požymių modelio „*Eshop-fm*“ transformacijų produktams apskaičiuoti metrikų naudojant įrankį nepavyko. Taip nutiko dėl itin didelės šio požymių modelio apimties – jo požymių skaičius – net 280. Tokios apimties požymių modeliams įrankis nepritaikytas, todėl šio požymių modelio transformacijų produktų ekvivalentiškumo palyginimas atliktas rankiniu būdu ir parodė, kad ir šis požymių modelis transformuotas korektiškai.

Tiriant požymį „*Insurance-product-fm*“ pastebėta, kad jo transformacijos produktų apribojimų skaičiaus metrika nesutampa. Tai parodo, kad transformacijos produktai-modeliai neekvivalentiški. Kadangi „*FD2 1*“ požymių modelis buvo naudojamas kaip pirminis modelis transformacijose, kurių produktai – „*FD2 2*“ ir „*FD2 3*“ požymių modeliai, galima teigti, jog nekorektiškai atliktos transformacijos iš „*FD2 1*“ į „*FD2 2*“. Pagal transformacijų sekų schemą (27 pav.) matome, kad nekorektiška transformacija *FD2→PROLOG→FD2* (transformacijos B ir D).

Apibendrinant, visi bibliotekos požymių modeliai buvo transformuoti korektiškai naudojant transformacijos seką *FD2→FDL→FD2* ir 17 iš 18 – naudojant transformacijos seką *FD2→PROLOG→FD2*.

9.6.2 FDL ir PROLOG požymių modelių ekvivalentiškumo palyginimo rezultatai

FDL ir *PROLOG* požymių modelių ekvivalentiškumo patikrinimui naudojama tekstinė analizė. Tarpusavyje lyginami kiekvieno požymių modelių bibliotekos modelio pradinis *FDL* / *PROLOG* modelis 27 paveiksle žymimas atitinkamai „*FDL 0*“ arba „*PROLOG 1*“) ir transformacijų rezultatas (27 paveiksle žymimas atitinkamai „*FDL 1*“ arba „*PROLOG 2*“).

Lentelėje 24 pateikiami ekvivalentiškumo patikrinimo rezultatai, kur ženklas „+“ reiškia, kad modeliai ekvivalentiški, o ženklas „–“ reiškia, kad modeliai neekvivalentiški.

24 lentelė. FDL ir PROLOG požymių modelių ekvivalentiškumo palyginimas

Požymių modelio pavadinimas	FDL 0 / FDL 1 požymių modelių palyginimas	PROLOG 1 / PROLOG 2 požymių modelių palyginimas
Cellphone-fm	+	+
Digital-video-system-fm	+*	+
Documentation-generation-fm	+	+
Eshop-fm	+*	+
Graph-manipulation	+	+
Graph-product-line-fm	+	+
Home-integration-system-fm	+*	+
Insurance-product-fm	+	–
James-fm	+	+
Jplug-fm	+	+
Key-word-in-context-fm	+	+
Model-transformation-fm	+	+
Monitor-engine-system-fm	+	+
Telecommunication-system-fm	+	+
Text-editor-fm	+	+
Thread-domain-fm	+	+
Virtual-office-of-future-fm	+	+
Web-portal-fm	+	+

Kaip matome pagal rezultatus, „*Insurance-product-fm*“ *PROLOG* požymių modeliai neekvivalentiški, o tai reiškia, kad šiam požymių modeliui transformacijų seka *PROLOG→FD2→PROLOG* atlikta nekorektiškai.

Taip pat po pirminės tekstinės analizės požymių modelių „*Digital-video-system-fm*“, „*Eshop-fm*“ ir „*Home-integration-system-fm*“ pradinis ir transformacijos rezultato *FDL* modelių tekstinės išraiškos nebuvo ekvivalentiškos, tačiau patys modeliai semantiškai ekvivalentiški (lentelėje pažymėta simboliu „+*“). Taip nutiko todėl, kad pradiniai *FDL* požymių modelių ryšiai buvo aprašyti ne minimalia forma, o modelių po transformacijos ryšiai aprašyti minimalia forma. Lentelėje žemiau pateikiamas *FDL* tekstas, aprašantis pradinį „*Digital-video-system-fm*“ požymių modelį bei tą patį modelį po transformacijos.

25 lentelė. Pradinio ir transformuoto FDL požymių modelio palyginimas

Pradinis požymių modelis (FDL 0)	Požymių modelis po transformacijos (FDL 1)
dvs: all (control, ?networkhw, serverpc, play, ?clientpc, ?handheld) control: all (remote, ?telephone, ?net, ?edit, play) telephone: all (?sms) net: all (?web, ?wap, email) edit: all (?addmusic) **play: all (?slides, ?audio, video)** networkhw: all (?modem, ?ethernet) serverpc: all (?network, ?irdaport) **play: all (?ondemand)** clientpc: all (?network) net requires networkhw handheld requires irdaport addmusic requires audio	dvs: all (control, ?networkhw, serverpc, play, ?clientpc, ?handheld) control: all (remote, ?telephone, ?net, ?edit, play) telephone: all (?sms) net: all (?web, ?wap, email) edit: all (?addmusic) **play: all (?slides, ?audio, video, ?ondemand)** networkhw: all (?modem, ?ethernet) serverpc: all (?network, ?irdaport) clientpc: all (?network) net requires networkhw handheld requires irdaport addmusic requires audio

Apibendrinant rezultatus, galima pasakyti, kad visi bibliotekos požymių modeliai buvo transformuoti korektiškai naudojant transformacijos seką *FDL→FD2→FDL* ir 17 iš 18 – naudojant transformacijos seką *PROLOG→FD2→PROLOG*.

9.7 Detalus požymių modelių transformacijų patikrinimas pagal kriterijus

Tiriant transformacijas buvo atliekamas detalesnis atominių transformacijų tyrimas. Šiam tyrimui taip pat buvo naudojami požymių modeliai iš aprašytos požymių modelių bibliotekos. Buvo tikrinama, ar visų tipų transformacijos tenkina apsibrėžtus kriterijus:

- korektiškai transformuojami požymiai;
- korektiškai transformuojami privalomumo ryšiai;
- korektiškai transformuojami pasirenkamumo ryšiai;
- korektiškai transformuojami alternatyvumo grupės;
- korektiškai transformuojami „vienas ir daugiau“ grupės;
- korektiškai transformuojami apribojimai „*requires*“;
- korektiškai transformuojami apribojimai „*excludes*“.

Šio patikrinimo rezultatai pateikti 26 lentelėje.

26 lentelė. Transformacijų tipų korektiškumas pagal kriterijus

Požymių modelio pavadinimas	Transformacijos tipas	Kriterijai Korektiškai transformuojami:						
		Požymiai	Privalomumo ryšiai	Pasirenkamumo ryšiai	Alterenatyvumo grupės	„Vienas ir daugiau“ grupės	Apribojimai „requires“	Apribojimai „excludes“
Cellphone-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Digital-video-system-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Documentation-generation-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Eshop-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Graph-manipulation	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Graph-product-line-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Home-integration-system-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+

Insurance-product-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	–
	FD2→PROLOG	+	+	+	+	+	+	+
James-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Jplug-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Key-word-in-context-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Model-transformation-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Monitor-engine-system-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Telecommunication-system-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Text-editor-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Thread-domain-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Virtual-office-of-future-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+
Web-portal-fm	FD2→FDL	+	+	+	+	+	+	+
	FDL→FD2	+	+	+	+	+	+	+
	PROLOG→FD2	+	+	+	+	+	+	+
	FD2→PROLOG	+	+	+	+	+	+	+

Kaip matome pagal rezultatus tik vienam požymių modeliui „*Insurance-product-fm*“, transformacijoje *PROLOG→FD2* nustatyta, kad nekorektiškai transformuojami „*excludes*“ apribojimai.

9.8 Tyrimo išvados

Apibendrinant tyrimą, galima teigti, kad didžioji dalis įrankio atliekamų transformacijų vykdomos korektiškai. Vis dėl to vieno iš aštuoniolikos tyrime naudotų požymių modelių transformavime iš *PROLOG* į *FD2* nustatytas nekorektiškas apribojimų „*excludes*“ transformavimas. Visų kitų įrankio atliekamų transformacijų tyrimo metu nekorektiškumų nepastebėta.

Išsamesniam tyrimui vertėtų panaudoti didesnę požymių modelių biblioteką. Bet tokiu atveju taip pat reikėtų pagalvoti ir apie labiau automatizuotą požymių modelių ekvivalentiškumo tyrimą, nes didesnei požymių modelių aibei tirti naudotos metodologijos gali būti neoptimalios.

10 IŠVADOS

- Ištirta požymių modelių struktūra, požymių metamodeliai požymių diagramų transformacijos metamodeliai ir pastebėta jog visų požymių modelių esminis elementas yra požymis, kuris turi ryšius su kitais požymiais. Transformacijos metamodeliai atspindi būdus, kaip modelį iš vieno formato pakeisti kitu. Atsižvelgdami į kitų autorių metamodelius, pasiūlėme savo požymių metamodelį.
- Atlikta požymių diagramų kūrimo įrankių palyginamoji analizė ir pasirinktas jau sukurtas požymių modeliavimo įrankis kaip tinkamiausias mūsų darbui. Kiti nagrinėti įrankiai neatitiko mūsų reikalavimų, tokių kaip: atvirasis kodas, požymių modelių sudėtingumo įvertinimas, tinkama architektūra transformavimui įgyvendinti.
- Atlikta požymių modelių kalbų palyginamoji analizė ir pasirinktos FDL, PROLOG kalbos su kuriomis buvo vykdomi tyrimai.
- Atlikta tobulinamos požymių modeliavimo ir transformavimo įrankio dalies analizė. Pastebėta jog įrankis neturi integruoto transformavimo modulio. Šis modulis buvo suprojektuotas ir integruotas į įrankį.
- Atliktas požymių modeliavimo ir transformavimo įrankio kokybės įvertinimas. Nustatyta pakankamai aukšta įrankio kodo kokybė, kas rodo gerą suprantamumą, palaikomumą.
- Ištirti įrankio atliekamų transformacijų tipai, nustatytas atliekamų transformacijų korektiškumas, pagal apsibrėžtus korektiškumo kriterijus. Ištirta FDL→FD2, FD2→FDL, FD2→PROLOG, PROLOG→FD2 transformacijos su visais turimais FDL bibliotekos modeliais. Nustatyta, kad PROLOG→FD2 transformacija atliekama ne visada korektiškai. Kitose transformacijose nekorektiškumų nepastebėta.

11 LITERATŪROS SĄRAŠAS

[1] P. Borba, L. Teixeira, R. Gheyi. *A Theory of Software Product Line Refinement*. Federal University of Pernambuco, Federal University of Campina Grande, 2010.

[2] K. Schmid, I. John. *Generic Variability Management and Its Application to Product Line Modelling*. Fraunhofer Institute for Experimental Software Engineering, 2003.

[3] IEEE 829 *Standard for Software Test Documentation*, 1998.

[4] P. Höfner, R. Khedri, B. Möller. *Feature Algebra*. Universität Augsburg, McMaster University Canada, 2006.

[5] K. Czarnecki, A. Wąsowski. *Feature Diagrams and Logics: There and Back Again*. University of Waterloo, IT University of Copenhagen, 2007.

[6] P. Y. Schobbens, P. Heymans, J. C. Trigaux. *Feature Diagrams: A Survey and a Formal Semantics*. University of Namur, 2006.

[7] K. Kang, S. Cohen, J. Hess, W. Novak, A. S. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Carnegie Mellon University, 1990.

[8] D. Kreivys. *Programų sistemų variantiškumo modelių, aprašytų požymių diagramomis, tyrimas*. Kauno technologijos universitetas, 2010.

[9] P. Y. Schobbens, P. Heymans, J. C. Trigaux, Y. Bontemps. *Generic semantics of feature diagrams*. University of Namur, 2006.

[10] Q. Boucher, A. Classen, P. Faber, P. Heymans. Introducing TVL, a Text-based Feature Modelling Language. Proc. of the Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10), Linz, Austria, January 27-29, 2010, pp. 159-162.

[11] K. Czarnecki, S. Helsen. *Feature-based Survey of Model Transformation Approaches*. IBM Systems Journal, Vol. 45, No. 3, 2006.

[12] K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovszky, U. Prange, G. Taentzer, D. Varró, S. Varró-Gyapay. *Model Transformation by Graph Transformation: A Comparative Study*. International Workshop on Model Transformations in Practice, 2005.

[13] T. Leich, S. Apel, L. Marnitz, G. Saake. *Tool Support For Feature-Oriented Software Development - FeatureIDE: An Eclipse-Based Approach*. In Proceedings of Oopsla Workshop on Eclipse Technology Exchange, San Diego, USA, 2005.

[14] O. Rohlik, A. Pasetti. *XFeature Modeling Tool*. Automatic Control Laboratory, Eth Zurich, 2005. Prieiga Internete: http://www.pnp-software.com/xfeature/.

[15] M. Stephan, M. Antkiewicz. *Ecore.FMP: A Tool for Editing and Instantiating Class Models as Feature Models*. University of Waterloo, 2008.

[16] T. von der Maßen, H. Lichter. *Requiline: A Requirements Engineering Tool for Software Product Lines*. Springer, 2004.

[17] D. Benavides, S. Segura, P. Trinidad, A. Ruiz-Cortés. *FAMA: Tooling a Framework for the Automated Analysis of Feature Models*. Vamos First International Workshop on Variability Modelling of Software-Intensive Systems, 2007.

[18] V. Myllärniemi, M. Raatikainen, T. Männistö. *KumbangTools*. Software Product Line Conference, Kyoto, Japan, 2007.

[19] D. Beuche. *Modeling and Building Software Product Lines with pure::variants*. Software Product Lines, 12th International Conference, Limerick, Ireland, 2008.

[20] I. Montenero, J. Peña, A. Ruiz-Cortés. *Improving the Design of Highly Variant-Rich Business Process Models*. University of Seville, 2008.

[21] M. Lawley, J. Steel. *Practical Declarative Model Transformation with Tefkat*. Springer, 2006.

[22] A. van Deursen, P. Klint. *Domain-Specific Language Design Requires Feature Descriptions*. CWI, 2002.

[23] K. Czarnecki, C.H.P. Kim, K.T. Kalleberg, *Feature Models Are Views on Ontologies*. Proc. of 10th Int. Software Product Line Conference (SPLC'06), 2006, pp. 41-51.

[24] H.H. Wang, Y.F. Li, J. Sun, H. Zhang, J. Pan. *Verifying Feature Models Using OWL*. Web Semantics 5, 2007, pp. 117-129.

[25] M. Mendonca, M. Branco, D. Cowan, *S.P.L.O.T. – Software Product Lines Online Tools*. Proc. of OOPSLA'09, 2009, 761–762.

[26] A.O. Elfaki, S. Phon-Amnuaisuk, C.K. Ho. *Modeling Variability in Software Product Line Using First Order Logic*. Proc. of 7th ACIS Int. Conf. on Software Engineering Research, Management and Applications (SERA2009), 2009.

[27] P. Paškevičius, G. Ziberkas. *Formalus požymių modeliavimas naudojant programavimo kalbą PROLOG*. Informacinės technologijos 2011 Mag&Dok IT2011, 2011.

[28] J. Sincero, W. Schröder-Preikschat. *The Linux Kernel Configurator as a Feature Modeling Tool*. Friedrich-Alexander University Erlangen-Nuremberg, 2008.

[29] O. Rohlik, A. Pasetti, K. Ekstein, P. Chevalley. *A Meta-Modelling Approach to Feature Modelling*, 2006.

[30] R. Mazo, R. E. Lopez-Herrejon, C. Salinesi, D. Diaz, A. Egyed. *Conformance Checking with Constraint Logic Programming: The Case of Feature Models*. 35th Annual

International Computer Software and Applications Conference (COMPSAC), IEEE Press, Munich-Germany, 2011.

[31] V. Vranic. *Reconciling Feature Modeling: A Feature Modeling Metamodel*. 5th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays 2004), pages 122-137, LNCS 3263, Erfurt, Germany, September 2004. Springer.

[32] P. O. Rossel, D. Perovich, M. C. Bastarrica. *Feature Model to Product Architectures: Applying MDE to Software Product Lines*. European Conference on Software Architecture. WICSA/ECSA, 2009.

[33] M. Fowler. Refactoring: Improving the Design of Existing Code. Addison-Wesley Longman Publishing Co., Inc., 1999.

[34] L. Erlikh. Leveraging Legacy System Dollars for E-Business, IT Professional, vol. 2, no. 3, pp. 17–23, 2000.

[35] A. Kleppe, J. Warmer, W. Bast. *MDA Explained: The Model Driven Architecture – Practice and Promise*. Addison-Wesley, 2003.

[36] T. Mens, K. Czarnecki, P. Van Gorp. *A Taxonomy of Model Transformation*, 2006.

[37] A. Kraus. Model driven software engineering for web applications, PhD dissertation, Ludwig-Maximilians-Universitat Munchen, 2007.

[38] V. Štuikys, R. Damaševičius. *Meta-Programming and Model-Driven Meta-Program Development: Principles, Processes and Techniques*. Springer, Series: Advanced Information and Knowledge Processing, 2012.

[39] S. McConnell. *Code Complete*. Microsoft Press, 1993, p. 395.

12 PRIEDAI

12.1 PAKEITIMŲ POVEIKIO ANALIZĖ POŽYMIŲ MODELIUOSE

Straipsnis pristatytas tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Mag&Dok IT2011“.

Autoriai: Marius Bindokas, Paulius Paškevičius, Robertas Damaševičius.

PAKEITIMŲ POVEIKIO ANALIZĖ POŽYMIŲ MODELIUOSE

Marius Bindokas[1], Paulius Paškevičius[2], Robertas Damaševičius[2]

[1]Kauno Technologijos Universitetas, Informacinių Sistemų Katedra,
[2]Kauno Technologijos Universitetas, Programų Inžinerijos Katedra ,
Studentų 50, LT-51368, Kaunas, Lietuva,
{marius.bindokas, paulius.paskevicius}@gmail.com, robertas.damasevicius@ktu.lt

Santrauka. Kintamumas yra fundamentali programinių sistemų ypatybė. Kiekviena programinė sistema turi evoliucionuoti per visus abstrakcijos lygius (modelis, architektūra, programinis kodas, dokumentacija, t.t.), kad prisitaikytų prie nuolatos besikeičiančių vartotojo poreikių ir aplinkos charakteristikų. Norėdami įvertinti, kiek pasikeitė sistema turime atlikti pokyčių analizę. Šiame straipsnyje mes analizuojame požymių (sistemos išorinių, vartotojui matomų charakteristikų) modelių kintamumą. Kad įvertinti pakeistų požymių validumą, mes siūlome pakeitimo poveikio modelį, kuriame remiantis, požymių priklausomumo matrica, stebimas požymių kaitos sklidimas ir įvertinamas požymių modelio kintamumas naudojantis Jaccard atstumo metriką.

1. Įvadas

Kintamumas yra esminė savybė, įtakojanti programos architektūrą [1]. Siekiant realizuoti sistemą, projektuotojas turi būti pasiruošęs galimiems pakeitimams sistemos gyvavimo metu. Pakeitimus gali įtakoti daugelis veiksnių, kaip pvz., srities reikalavimų pasikeitimas, sistemos fizinės charakteristikos, skirtingi vartotojo-sistemos sąveikos tipai, aparatūrinės dalies pasikeitimas ir t.t. skirtinguose programinės įrangos projektavimo etapuose įskaitant reikalavimų specifikavimą, modelius, architektūrą, dizaino modelius, programinį kodą ir dokumentaciją. Pasiruošimas numatomiems, kaip ir netikėtiems, pakeitimams yra labai svarbus norint sumažinti programinės įrangos gamybos kaštus, padidinti palaikomumą, plėtojamumą ir pratęsti patį sistemos gyvavimo laikotarpį. Be to, *ISO 9126* standarte kaita buvo apibrėžta kaip labai svarbus faktorius programinės įrangos kokybėje ir palaikomume.

Programinės įrangos kaitos numatymas yra viena svarbiausių veiklų palaikant pakeitimus [2]. Norint įvertinti pakeitimų poveikį ir apimtį, reikia atlikti sistemos programinės įrangos poveikio analizės tyrimą. Žinias, gautas iš analizės, galima panaudoti programinės įrangos re-inžinerijos procesui. Visgi, kaitos analizė nėra paprasta užduotis, pakeitimų faktoriai gali sąveikauti ir turėti skirtingus prioritetus, pritaikius programinėje įrangoje. Kuomet svarstomi modeliuojamos sistemos modelio pakeitimai, reikia būtinai identifikuoti keičiamo modelio dalis, kurios bus paveiktos atliekant pakeitimus, bei užtikrinti, jog modelis atlikus pakeitimus ir toliau teisingai atvaizduos sistemą. Taigi, reikia kurti programinės įrangos sistemų kaitos vertinimo, planavimo ir realizavimo metodus.

Pakeitimų poveikio analizė pastaruoju metu dažniausiai taikoma programiniam kodui (žr. pvz., [3]) arba UML modeliams (žr. pvz., [4]). Produktų linijos modeliavimo kontekste, pakeitimų poveikio analizė yra susijusi su abstrakčiu delta modeliavimu [5]. Anquetil *et al.* [6] nagrinėja trasavimą, kaip poveikio analizės produktų linijoje dalinį atvejį. Požymių pakeitimo operacijų katalogai ir taksonomijos pateikiami [7, 8, 9].

Šiame dokumente, mes analizuojame kaip produktų linijos požymių modelis [10] reaguoja į pakeitimą pvz.: atlikus požymių modelių transformacijas. Naudojame požymių modelių pakeitimų poveikio analizę kaip požiūrį į priemonę produktų linijos evoliucijoje. Pagrindinė problema yra pakeitimų sklidimas: jei turimas pagrindinių programos pakeitimų rinkinys – užduotis, yra nustatyti papildomus programos požymius, kurie gali būti paveikti atlikus pagrindinį pakeitimą. Kad identifikuoti šią problemą, mes siūlome metodą, kurio pagalba galima įvertinti pakeitimų mąstą remiantis požymių pakeitimų poveikio modeliu.

2. Požymių modelių evoliucija

Programinės įrangos sistemos variantiškumą galima atvaizduoti požymių modeliais. Požymis – matomas, išorinis sistemos aspektas (pvz.: charakteristikos, paslaugos ir t.t.). Požymių modeliai yra plačiai naudojami programinės įrangos produktų linijos inžinerijoje, paprastai ir aiškiai pavaizduoti visus produkto linijos produktus. Požymių modeliai atvaizduojami grafiškai, naudojant Požymių Diagramas [11] arba tekstu naudojant pvz.: Požymių aprašymo kalbą (Feature Description Language - FDL)[12]. Požymių diagramų būsenų detalių paaiškinimų notacija žr.: [13].

Analizuojant požymių modelių variantiškumą ir evoliuciją, mes perėmėm Fluri ir Gall[14] pasiūlytą programinio teksto pakeitimų taksonomiją. Taksonomija paremta abstrakčios sintaksės medžiu(Angl.: abstract syntax tree (AST)). Kadangi požymių modeliai yra vaizduojami pagal požymių hierarchiją (požymių medžiai), šios taksonomijos dalys gali būti pritaikytos požymių modelių keitimui.

Xue *et al.* [7] pasiūlė panašią taksonomiją priimtą požymių modelių keitimui, kuris susideda iš atominių pakeitimų (pervadinti požymį, pridėti lapinį požymį, pašalinti lapinį požymį ir perkelti požymį) ir sudėtinių požymių(pridėti požymio sub-medį, panaikinti požymių sub-medį, išskaidyti požymį ir sulieti požymius).

Botterweck *et al.* [9] pasiūlė labiau detalų evoliucinių operatorių rinkinį: pridėti [(pasirenkamas|būtinas)] požymis *g* [žemiau *f*], panaikinti požymį *g*, pakeisti g_1 [(būtinas|pasirenkamas)požymis] g_2 požymiu,perkelti *g* požymį į *h*, pervadinti f_1 į f_2, pakeisti požymį *f* į (būtiną|pasirenkamą). Šis būdas taip pat paremtas medžio redagavimo operacijomis (pridėti,pašalinti,perkelti,pakeisti).

Elementarios medžio viršūnės keitimo operacijos yra įdėti, ištrinti, perkelti ir atnaujinti. Įdėti ir ištrinti operacijos yra leidžiamos tik lapinėms viršūnėms (Atskiri požymiai). Perkėlimo operacija perkelia požymį į kitą požymio medžio vietą kartu su visais vaikiniais požymiais. Atnaujinimo operacija pakeičia požymio reikšmę (pvz.: charakteristika ar specialus reikalavimas susiejęs su šiuo požymiu). Požymių medyje pakeitimas išreiškiamas pakeičiant požymio vardą (žymę). Kadangi požymių medis taip pat turi viršūnių ir briaunų dekoracijas, kurios atspindi skirtingų tipų požymių variantiškumus (**būtinas, pasirenkamas**), požymių pasirinkimo taisykles (**visi** požymiai, **vienas iš** požymių, **daugiau iš** požymių) ir požymių apribojimus (**būtinas**, **draudžiantis** arba nei vienas), papildomos medžių viršūnių ir briaunų dekoracijos pakeitimo operacijos gali būti apibrėžtos: atnaujinti požymio statusą, atnaujinti požymio parinkimo taisykles ir atnaujinti požymio taisykles (tik neskaidomiems požymiams)

Žemiau, apibūdiname požymio modelio keitimo operacijas daug nuodugniau:

- $INS(f_x, f_y)$: įvesti požymio viršūnę f_x kaip f_y vaiką.
- $DEL(f_x)$: ištrinti požymio viršūnę f_x iš požymių modelio.
- $MOV(f_x, f_y) \rightarrow DEL(f_x), INS(f_x, f_y)$: perkelti požymio viršūnę f_x kaip vaikinę f_y viršūnę.
- $UPD(f_x, f_y)$: atnaujinti požymio viršūnę f_x kaip f_y.
- $STA(f_x, s)$: pakeisti požymio viršūnės statusą f_x į *s*, kur *s* yra "būtinas" arba "pasirenkamas".
- $SEL(f_x, r)$: pakeisti vaikinio požymio viršūnės pasirinkimo taisyklę f_x į *r*, kur r yra "visi", "vienas iš", arba "vienas ir daugiau".
- $CON(f_x, f_y, c)$: pakeisti apribojimus tarp f_x ir f_y į *c*, kur *c* yra "būtinas", "negalimas" ir "nesantis".

3. Požymių modeliams skirtas pakeitimų poveikio modelis

Pakeitimų poveikio analizės tikslas yra veikiamos sistemos potencialių efektų nustatymas, atsiradusių dėl pasiūlytų pakeitimų [15]. Pakeitimų analizės prielaida yra ta, kad pasiūlytas pakeitimas gali iššaukti nepageidaujamus šalutinius efektus ir/arba kaitos plitimo efektus. Pakeitimo rezultato šalutinis efektas yra sąlyga, kuri veda sistemą prie klaidingos būsenos arba neteisingo funkcionalumo. Kaitos plitimo efektas yra reiškinys, kuomet dėl pakeitimų yra paveikiamos kitos sistemos dalys. Pakeitimų analizės tikslas yra įvertinti kaitos plitimo efektus ir šalutinius efektus atsiradusius dėl pakeitimo. Kuomet yra svarstomas požymių modelio pakeitimas būtina identifikuoti požymių modelio elementus, kurie bus paveikti kaitos plitimo efekto. Pakeitimas taip pat gali vesti prie tokių šalutinių efektų, kaip požymių nesutampamumas, tokiu būdu padarydamas pakeistą požymių modelį neteisingu (pvz., modelis, kuris specifikuoja tuščią produktų rinkinį).

Požymio pakeitimas gali turėti poveikį visam požymių modeliui. Todėl poveikio sklidimas ir pakeisto modelio validumas turi būti gerai apsvarstytas. Suformulavome tokias pakeitimų analizės užduotis: 1) pakeisto požymių modelio validavimas (pakeitimo validumas); 2) dėl keitimo požymių modelyje, paveiktų požymių plitimas; 3) požymio modelio metrinių savybių perskaičiavimas (požymių skaičius, produktų skaičius, t.t.) (kintamumas). Pakeitimų poveikio analizė turi įvertinti šias užduotis. Toliau, pateiksime kiekvienos užduoties detalesnį nagrinėjimą.

1) **Pakeitimo validumas.** Validų požymių keitimą apibrėžiame, kaip duotojo validaus požymio modelio keitimas į validų pakeistą požymių modelį. Tegul *F* yra požymių modelis. Tegul $F' \leftarrow \delta(F)$ yra pakeistas požymių modelis gautas po pakeitimo δ pritaikymo. Tegul *validus* bus predikatas apibrėžtas požymių modelyje *F*, kuris grąžina TRUE jeigu modelis yra validus ir FALSE kitu atveju. Tuomet tegul *validusKeitimas* yra predikatas apibrėžtas *F* ir δ, kuris grąžina TRUE ir pakeistas modelis F' yra *validus*, ir FALSE, kitu atveju: $validusKeitimas(F, \delta) \rightarrow validus(\delta(F))$.

2) **Pakeitimo plitimas.** Modelių kaitos plitimui modeliuoti siūlome komponentų priklausomybės matricą [16]. Apibrėžiame požymių matricą D. Šios matricos elementai reiškia keitimo galimybę tarp dviejų požymių: jeigu požymis f_x yra pakeičiamas ir jeigu $D[f_x, f_y]=1$, tuomet požymis f_x taip pat turi būti pakeistas. Kiekvienai požymių keitimo operacijai sudarome požymių priklausomybės matricą, kaip nurodyta 1 lentelėje.

1 lentelė. Požymių priklausomybės lentelės sudarymas

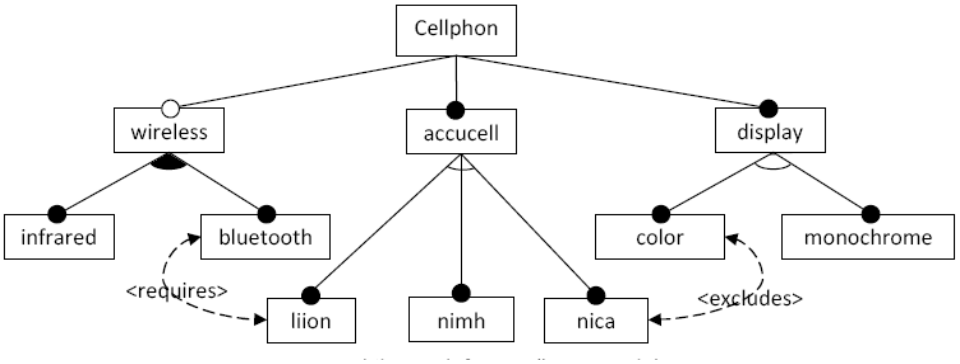
Požymio keitimo operacija	**Požymių priklausomybės matricos elementų reikšmės**
INS (f_x, f_y)	D [f_x, f_y] = 1
DEL (f_x)	D [f_x, *tėvas*(f_x)] = 1
MOV (f_x, f_y)	D [f_x, *vaikas*(f_y)] = 1
UPD (f_x, f_x')	D [f_x, *vaikas*(f_x)] = 1
STA (f_x, s)	D [f_x, *vaikas*(f_x)] = 1, D [f_x, *parent*(f_x)] = 1
SEL (f_x, r)	D [f_x, *vaikaitis*(f_x)] = 1
CON (fx$_x$, f_y, c)	D [f_x, f_y] = 1, D[f_y, f_x] = 1

Kai požymių priklausomybės matrica sudaryta, iš jos gaunama požymių sklidimo matrica tokiu būdu:: $\forall x, y,$ jeigu $D(x,y)=1$, tuomet atliekame D pakeitimą $UPD(y,y)$. Sudaryta keitimų plitimo matrica gali būti naudojama gauti sąrašui požymių kurie buvo paveikti keitimo δ.

3) Kintamumas. Požymių modelio kintamumą apibrėžiame kaip santykį tarp produkto rinkinio egzempliorių, apibrėžtų požymio modelio ir paveiktų požymio keitimo. Kintamumo reikšmė kinta nuo 0: nei vienas produktas nepakeistas, iki 1: visi produktai pakeisti. Tegul F yra požymių modelis. Tegul $F' \leftarrow \delta(F)$ yra pakeistas požymių modelis gautas po pakeitimo δ patvirtinimo. Tegul C bus produktų egzempliorių rinkinys specifikuojamas modelio F ir C' bus produktų egzempliorių rinkinys specifikuojamas modelio F'. Modelio F kintamumas apskaičiuojamas kaip Jaccard'o atstumas tarp F ir F': $ch(F,\delta) = J_\delta(C,C') = 1 - \frac{|C(F) \cap C(\delta(F))|}{|C(F) \cup C(\delta(F))|}$.

4. Pavyzdžio nagrinėjimas

Kaip pavyzdį pateiksime mobilaus telefono požymių modelį [17] (žiūrėti 1 paveikslą). Kiekvienas mobilusis telefonas turi turėti akumuliatorių (*accucell*) ir ekraną (display), ir gali turėti papildomą funkciją – bevielis ryšys (wireless). Bevielis ryšys gali būti realizuotas infraraudonųjų spindulių ryšiu(*infrared*) arba *Bluetooth* ryšiu. *Bluetooth* ryšys reikalauja ličio jonų baterijos (lion (Lithium-Ion)). Ekranas gali būti nespalvotas (*monochrome*) arba spalvotas, o spalvoto ekrano pasirinkimas paneigia galimybę pasirinkti *nica* (Nickel-Cadmium) baterijos tipą. Mobilaus telefono baterijos gali būti: liion, nimh (Nickel-Metal hydride) arba nica tipo. Šis požymių modelis turi 10 požymių ir jo pagalba galima apibūdinti 16 skirtingų validžių produkto konfiguracijų.



1 paveikslas. Mobilaus telefono požymių modelis

Kad įvertinti pakeitimų teisingumą, pakeitimų plitimą ir kintamumą, remiantis pasiūlytu pakeitimų poveikio analizės modeliu, įgyvendinome požymių modelį ir pakeitimus požymių modelyje naudojant loginę programavimo kalbą PROLOG. Kai kurių pakeitimų, padarytų mobilaus telefono požymių modeliui, rezultatai apibendrinti 2 lentelėje.

2 lentelė. Pakeitimų poveikio pavyzdžiai

Požymis	**Požymio pakeitimas**	**Pakeitimo validumas**	**Pakeitimo plitimo paveikti požymiai**	**Skaičius produktų pakeistame modelyje**	**Kintamumas**
cellphon	UPD (cellphon, cellphone)	validus	*none*	16	0
wireless	STA (wireless, mandatory)	validus	infrared, bluetooth, liion	8	0.50
RF	INS (RF, wireless)	validus	*none*	32	0.50
wireless	SEL (wireless, one_of)	validus	infrared, bluetooth, liion	16	0.22
bluetooth	CON (bluetooth, liion, none)	validus	liion	16	0.40
nica	DEL (nica)	validus	color	16	0.67

5. Išvados ir tolimesni darbai

Šiame straipsnyje aptarėme požymių modelių evoliuciją, požymių modelių pakeitimų tipus ir pasiūlėme požymių modeliams skirtą pakeitimų poveikio įvertinimo modelį. Pakeitimų poveikio modelio pagalba galime analizuoti požymių modelyje atliktų pakeitimų efektą, ištirti požymių kaitos plitimą, patikrinti pakeisto požymių modelio validumą ir apskaičiuoti jo kintamumą. Pasiūlėme įvertinti kintamumą požymių modelyje, kaip Jaccard'o atstumą tarp produktų aibių aprašytų originaliame ir pakeistame modeliuose. Išanalizavus, atliktus pakeitimus paprastame požymių modelyje nustatyta, jog net ir paprastas požymio pakeitimas arba ryšio pakeitimas gali turėti žymų (38% vidutiniškai) poveikį produkto konfigūravimo erdvei.

Tolimesni darbai bus orientuoti į pakeitimų poveikio modelio išplėtimą programinio kodo artefaktais. Kuriant žinių bazę, kur kiekvienas požymis yra sujungtas su atitinkamu tą požymį įgyvendinančiu programos kodu, užduotis bus ištirti, kaip pakeitimai požymių modelyje sistemoje, paveiks programinio kodo sistemą.

Literatūra

[1] **Nord R.L., Hofmeister C., Soni D.** Preparing for Change in the Architecture Design of Large Software Systems. *The First Working IFIP Conference of Software Architecture*, San Antonio, Texas, February 1999.

[2] **Kagdi H., Maletic J.I.** Software-Change Prediction: Estimated+Actual. *Proc. of 2nd Int. IEEE Workshop on Software Evolvability, SE '06*, 24-24 Sept. 2006, 38 - 43.

[3] **Delamare R., Munoz F., Baudry B., Le Traon Y.** Vidock: A Tool for Impact Analysis of Aspect Weaving on Test Cases. *Proc. of 22nd IFIP WG 6.1 Int. Conf. on Testing Software and Systems, ICTSS 2010*, Natal, Brazil, November 8-10, 2010. LNCS 6435 Springer 2010, 250-265.

[4] **Briand L.C., Labiche Y., O'Sullivan L., Sowka M.M.** Automated impact analysis of UML models. Journal of Systems and Software, 2006, 79(3), 339-352.

[5] **Clarke D., Helvensteijn M., Schaefer I.** Abstract Delta Modeling. *Proc. of Generative Programming and Component Engineering Conf. (GPCE 2010)*, Eindhoven, October 2010, 13-22.

[6] **Anquetil N., Kulesza U., Mitschk, R., Moreira A., Royer J.-C., Rummler A., Sousa A.** A model-driven traceability framework for software product lines. *Software and System Modeling* 9(4): 427-451, 2010.

[7] **Xue Y., Xing Z., Jarzabek S.** Understanding Feature Evolution in a Family of Product Variants. *Proc. of 17th Working Conference on Reverse Engineering, WCRE 2010*, 13-16 October 2010, Beverly, MA, USA, 109-118.

[8] **Schmid K., Eichelberger H.** A requirements-based taxonomy of software product line evolution. *Electronic Communication of the EASST*, vol. 8, 2007.

[9] **Botterweck G., Pleuss A., Dhungana D., Polzer A., Kowalewski S.** EvoFM: feature-driven planning of product-line evolution. *Proc. of Workshop on Product Line Approaches in Software Engineering (PLEASE '10)*, New York, NY, USA, 24-31, 2010.

[10] **Kang K.C., Cohen S.G., Hess J.A., Novak W.E., Peterson A.S.** Feature-oriented domain analysis (FODA) feasibility study. *Technical Report CMU/SEI-90-TR-021*, SEI, Carnegie Mellon University, November 1990.

[11] **Schobbens P.-Y., Heymans P., Trigaux J.-C.** Feature Diagrams: A Survey and a Formal Semantics. *Proc. of 14th IEEE Int. Conf. on Requirements Engineering (RE 2006)*, 11-15 September 2006, Minneapolis/St.Paul, Minnesota, USA, 136-145.

[12] **van Deursen A., Klint, P.** Domain-Specific Language Design Requires Feature Descriptions. *Journal of Computing and Information Technology*, 2002, 10(1), 1-17.

[13] **Damaševičius, R., Štuikys, V.** Design of Ontology-Based Generative Components Using Enriched Feature Diagrams and Meta-Programming. *Information Technology & Control*, 2008, 37(4), 301-310,.

[14] **Fluri, B., Gall H.** Classifying Change Types for Qualifying Change Couplings. *Proc. of 4th Int. Conference on Program Comprehension (ICPC 2006)*, 14-16 June 2006, Athens, Greece, 35-45.

[15] **Bohner S.A., Arnold R.S.** (Eds.) *Software Change Impact Analysis*. IEEE Computer Society Press, 1996.

[16] **Mao C., Zhang J., Lu Y.** Matrix-based Change Impact Analysis for Component-based Software. *Proc. of 31st Annual Int. Computer Software and Applications Conference (COMPSAC 2007)*, 24-27 July 2007, Beijing, China, (1) 2007: 641-642.

[17] **Maen, von der T., Lichter H.** Determining the variation degree of feature models. *Proc. of 9th Int. Conf. on Software Product Lines, SPLC 2005*, Rennes, France. LNCS vol. 3714, 82-88. Springer, 2005.

12.2 SUDĖTINGŲ KONTEKSTŲ-PRODUKTŲ SĄRYŠIŲ MODELIAVIMAS NAUDOJANT POŽYMIŲ MODELIUS

Straipsnis pristatytas tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Mag&Dok IT2011“.

Autoriai: Aidas Kasperavičius, Robertas Damaševičius.

SUDĖTINGŲ KONTEKSTO-PRODUKTŲ SĄRYŠIŲ MODELIAVIMAS NAUDOJANT POŽYMIŲ MODELIUS

Aidas Kasperavičius, Robertas Damaševičius

Kauno technologijos universitetas, Programų inžinerijos katedra,
Studentų g. 50, LT-51368, Kaunas, Lietuva,
a.kasperavicius@gmail.com, robertas.damasevicius@ktu.lt

Santrauka. Projektuojant nuo konteksto priklausančias sistemas, esminė problema yra kontekstinės informacijos modeliavimas ir valdymas. Kontekstas gali būti pagrindinis faktorius, apsprendžiantis kuriamų produktų variantiškumą. Viena iš problemų kuriant nuo konteksto priklausomą produktų šeimyną yra kontekstinės informacijos atvaizdavimas požymių modelyje. Šiame straipsnyje nagrinėjamas sudėtingų, nuo konteksto priklausomų produktų požymių sąryšių modeliavimas naudojant konteksto ir produktų požymių modelių hierarchijas, formaliai atvaizduojamas meta-grafais. Pateikiamas kultūriškai adaptuojamų tinklalapių kontekstų ir produktų požymių modelis.

1. Įvadas

Nuo konteksto priklausomos sistemos įvertina esamą vartotojų situaciją, kad galėtų pateikti vartotojui pritaikytas paslaugas ir informaciją. Sistemų, priklausomų nuo konteksto, esminė problema yra kontekstinės informacijos modeliavimas, valdymas ir manipuliavimas. Kontekstas yra svarbus sprendžiant programos ir modelio pakartotinio panaudojamumo klausimą, kadangi verslo logika paprastai yra specifikuojama be kontekstų patikslinimo. Atlikus bet kokį konteksto pakeitimą reikia keisti ir programos požymius, o dažni keitimai apsunkina programinės įrangos palaikymą ir priežiūrą. Produkto validavimas taip pat tampa sudėtingesnis, nes gali pasireikšti nenumatytas sistemos funkcionavimas dėl konflikto tarp sistemos ir pasikeitusio konteksto. Siekiant palengvinti konteksto atvaizdavimą, dalijimąsi kontekstine informacija ir semantinę sąveiką, reikia sukurti formalų konteksto modelį, apibrėžiantį konteksto duomenis kompiuteriu apdorojama forma. Konteksto modeliavimas turi būti atliekamas kartu su sistemos modeliavimu, jei įmanoma, panaudojant vieningą notaciją tuose pačiose abstrakcijos lygiuose.

Programinės įrangos produktų šeimynų inžinerijoje [1] sistemos modeliuojamos naudojant požymių (vartotojui matomų sistemos savybių) koncepciją. Požymių variantiškumas ir jų sąryšiai sąlygoja skirtingas produktų konfigūracijas, kurių visuma sudaro produktų šeimyną. Produktų šeimynų inžinerijos tikslas – produktų šeimynų kūrimas ir produktų variantų išvedimo iš šeimynos specifikacijos palengvinimas. Kontekstas gali būti pagrindinis faktorius, apibrėžiantis išvedamus produktus. Viena iš problemų kuriant nuo konteksto priklausomą produktų liniją yra kontekstinės informacijos atvaizdavimas požymių modelyje.

Šiame straipsnyje mes parodome, kad sistemos kontekstas gali būti modeliuojamas naudojant tą pačią abstrakciją (požymių diagramas), kaip ir modeliuojant sistemos požymius. Toliau antrajame skyriuje aprašomi susiję darbai. Trečiajame skyriuje pristatomas konteksto ir produktų šeimynos požymių modeliavimo integravimo karkasas. Ketvirtajame skyriuje pateikiami pavyzdžiai. Galiausiai, penktajame skyriuje apibendrinamos išvados ir aptariami galimi būsimi darbai.

2. Susiję darbai

Konteksto analizės, modeliavimo ir inžinerijos problemos bei temos yra plačiai diskutuojamos programų inžinerijoje. Toliau panagrinėsime, kaip konteksto koncepcijos klausimai sprendžiami programų šeimynų inžinerijoje. Konteksto idėja darosi vis populiaresnė tema produktų šeimynų inžinerijos tyrėjų tarpe. Pavyzdžiui, K. C. Kang ir kt. [2] siūlo metodą požymių modelio skaidymui į kelis sluoksnius, iš kurių vienas yra panaudojimo kontekstas. Buhne ir kt. [3] aprašo reikalavimų variantiškumo tarp kelių produktų šeimynų modeliavimo būdą. Naudojant šį būdą gali būti modeliuojama tik viena konteksto dimensija. Czarnecki ir kt. [4] siūlo naudoti nulinio lygmens požymių modelį konteksto informacijos kaip aukšto lygmens sistemos reikalavimo aprašymui. Wagelaar [5] siūlo atskirti išorinių faktorių sukeltas požymių sąveikas išsamiai modeliuojant išorines priklausomybes. Atskiri konteksto modeliai yra naudojami, tam kad aprašyti susijusius išorinius faktorius, kurie turėtų būti taikomi. Kai šie modeliai apjungiami, galima nustatyti, kurie požymiai gali būti kombinuojami. Požymių modelis ir konteksto priklausomybės yra atskirtos į dvi argumentų erdves. Ubayashi ir kt. [6] sudaro nuo konteksto priklausomas produktų šeimynas iš programų sistemų šeimynų ir kontekstų šeimynų. Sistemų šeimyna atvaizduojama sistemos požymių medžiu, kuris išgaunamas analizuojant produktų šeimynos funkcionalumą ar atributus. Kontekstų šeimyna atvaizduojama konteksto požymių medžiu, kuris išgaunamas analizuojant kontekstų, susietų su produktų šeimyna, požymius.

Ankstesniame mūsų darbe mes nagrinėjome požymių modelių papildymą konteksto informacija [7]. Šiame straipsnyje mes aprašome, kaip sistemos kontekstas gali būti modeliuojamas naudojant tą pačią abstrakciją (požymių diagramas) kaip ir modeliuojant sistemos požymius. Sudaromi du modeliai: 1) kontekstinis požymių modelis leidžia

konteksto modeliuotojui specifikuoti nuo tam tikram kontekstui būdingus požymius, kurie gali būti bendri tarp visų produktų, kurie turi naudoti šį kontekstą. 2) Produkto požymių modelis specifikuoja produkto požymius tam tikrame kontekste. Požymių modelis turi būti atnaujintas kiekvieną kartą kai keičiasi kontekstas. Todėl konteksto požymiai ir produkto požymiai turi būti susieti modelių hierarchijoje, kur aukštesnio lygmens modelis įtakoja žemesnio lygmens modelio požymių pasirinkimą.

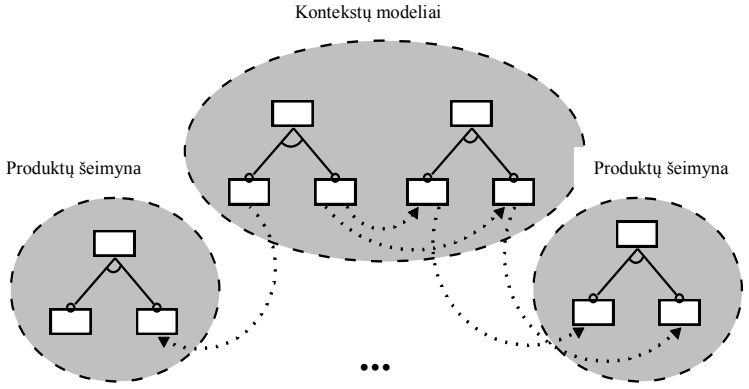
3. Konteksto modeliavimo integravimo į požymių modeliavimą karkasas

Mūsų siūlomam karkasui mes priėmėme šį konteksto apibrėžimą: kontekstas – tai „dalinė srities, kuri susijusi su dalyvio tikslais, būsena“ [8]. Tam kad išvesti nuo konteksto priklausomus produktus, ryšys tarp produktų variantų ir konteksto turi būti aiškiai apibrėžtas ir argumentuotas. Pavyzdžiui, kontekstas gali nulemti, ar požymis yra privalomas, neprivalomas, ar net perteklinis. Laikantis kontekstiškai orientuotų modelių būdo [9], mes siūlome aiškiai atskirti konteksto požymius nuo produktų šeimynos požymių, būdingų taikomai sričiai ir implementavimo technologijoms.

Galimi įvairių tipų konteksto modeliai: socialiniai, kultūriniai, būdingi tam tikrai sričiai (pvz., pedagoginiai), organizaciniai, kokybiniai; priklausantys nuo srities, aplinkos ir reikalavimų. Fricker ir Stoiber [10] pateikia išsamesnę konteksto taksonomiją: techninis kontekstas (produktų struktūra ir dizainas), verslo kontekstas (tikslai, pastangos ir resursai produkto kūrimui), organizacinis kontekstas (produktą kuriančios kompanijos organizacinė struktūra ir funkcionavimas), geografinis kontekstas (atstumas tarp bendradarbių), istorinis kontekstas (produkto ir visos produktų šeimynos evoliucija). Hong [11] pateikia panašią taksonomiją, susidedančią iš šešių fundamentalių kontekstų kategorijų: „kas“ (asmens kontekstas), „kada“ (laiko kontekstas), „kur“ (erdvės/geografinis kontekstas), „ką“ (susiję objektai), „kaip“ (vykstantys procesai) ir „kodėl“ (vartotojo tikslai).

Konteksto/konteksto ir konteksto/produkto ryšiai gali būti kelių rūšių [8]: 1) kontekstas sąlygoja reikalavimų aibę (gali būti modeliuojama naudojant pirmumo logiką). 2) Reikalaujamas kontekstas nustato sąlygas reikalavimų aibės poreikiui (gali būti modeliuojama naudojant loginę implikaciją). 3) Kokybės kontekstas gali įtakoti kiekvieno galimo būdo patenkinti reikalavimus kokybę, pavyzdžiui, kiekvieno modelio varianto kokybę (gali būti modeliuojama naudojant miglotąją logiką).

Apibendrinant galime teigti, kad gali būti kelių lygių hierarchija, sudaryta iš konteksto ir požymio modelių bei apibrėžta meta-grafo struktūra, kur viršūnės atitinka konteksto modelius, lapai atitinka produktų šeimynų modelius, o šakos atitinka funkcinius ryšius tarp (aukštesnio lygio) konteksto požymių ir (žemesnio lygio kontekstų arba) produktų šeimynų požymių. Paprasčiausias pavyzdys pateiktas 1 paveiksle (vienas konteksto modelis yra meta-grafo šaknis, o du produktų šeimynų modeliai yra meta-grafo lapai).



1 pav. Konteksto-produkto modelio meta-medžio pavyzdys

Meta-grafai [12] – tai grafinės struktūros, kuriose briaunos vaizduoja kryptinius ryšius tarp aibių ir elementų. Meta-grafas $S = \langle X, E \rangle$ yra struktūra susidedanti iš dviejų aibių X ir E. Generuojanti meta-grafo aibė yra elementų aibė $X = \{x_1, x_2, ..., x_n\}$, kuri atitinka reikšmių kintamuosius. E yra briaunų, apibrėžtų generuojančiose aibėse, aibė, kur briauna e meta-grafe yra pora $e_k = \{V_k, W_k\}, e_k \in E, V_k \subseteq X, W_k \subset X$.

4. Taikomasis pavyzdys

Kaip pavyzdį pateiksime tinklalapio produktų liniją. Tinklalapio požymiai yra apriboti konteksto požymių, specifikuotų konteksto požymių modelių aibe. Apsvarstysime tokį pavyzdį: tinklalapis turi būti pritaikomas pagal vartotojo kultūros pasirinkimus ir šie pasirinkimai gali būti sužinomi tik iš asmeninės vartotojo informacijos (pastaba: pateiktame pavyzdyje modeliai buvo supaprastinti). Todėl galime apibrėžti du konteksto požymių modelius: asmens konteksto ir kultūrinio konteksto.

Asmens konteksto modelis specifikuotas naudojant FDL kalbą [13]:

```
PersonalContext: all (Gender, Age, Nationality)
Gender: one-of (Male, Female)
Age: one-of (Young, Mature, Old)
Nationality: one-of (European, American, Asian)
```

Kultūrinio konteksto modelis (apibrėžtas pagal kultūrines dimensijas iš [14]):

```
CulturalContext: all (Orientation, PowerDistance, Character, Context, UncertaintyAvoidance)
Orientation: one-of (Individualism, Collectivism)
PowerDistance: one-of (HighPowerDistance, LowPowerDistance)
Character: one-of (Masculinity, Femininity)
Context: one-of (HighContext, LowContext)
UncertaintyAvoidance: one-of (HighAvoidance, LowAvoidance)
```

Sąryšiai tarp asmens ir kultūrinio kontekstų modelių apibėžiami taip:

```
Male requires Masculinity
Female requires Femininity
American requires Individualism
Asian requires Collectivism
Old requires HighAvoidance
```

Tinklalapio požymių modelis gali būti apibrėžtas taip:

```
Webpage: more-of (Title, Layout, Text, Background, Image, Link)
Layout: all(Grid, Template)
Text: all (Font, TextSize, TextColor)
Background: one-of (Texture, Pattern, BackgroundImage)
Pattern: all (Weave, ForegroundColor,BackgroundColor)
Image: more-of (FemalePhoto, MalePhoto, Cars, People, Nature)
BackgroundColor: one-of (BoldColor, PastelColor)
Link: more-of (FAQ, SearchEngine)
TextSize: one_of (LargeText, MediumText, SmallText)
```

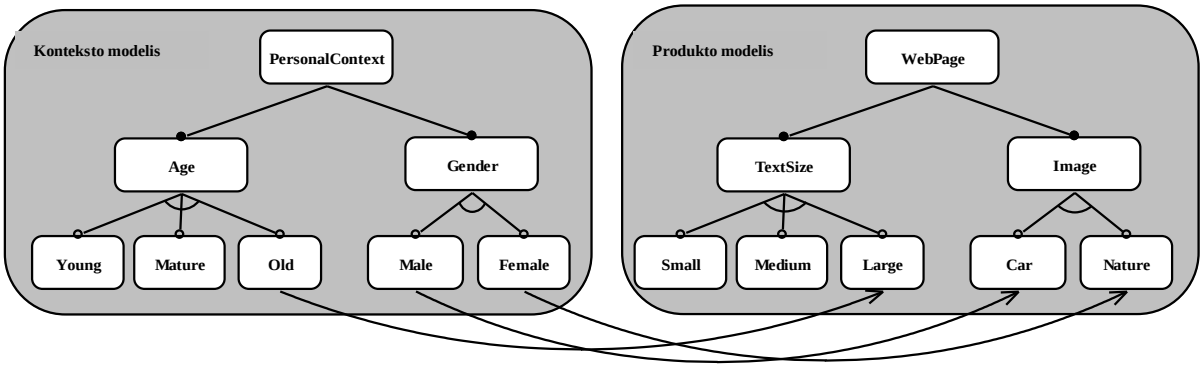
Ryšiai tarp kultūrinio konteksto modelio ir tinklalapio požymių modelio apibrėžti pagal [15]:

```
Collectivism requires People
Masculinity requires FemalePortrait
Masculinity requires Cars
HighContext requires BoldColor
HighAvoidance requires FAQ
```

Ryšiai tarp asmens konteksto modelio ir tinklalapio požymių modelio:

```
Old requires LargeText
```

Supaprastintas konteksto/produkto požymių sąryšio modelio fragmentas (požymių diagramos forma) pateikiamas 2 paveiksle. Modelis gali būti naudojamas kuriant kultūriškai ir socialiai pritaikomus tinklalapius.



2

pav. Konteksto/produkto požymių modelio sąryšiai

5. Išvados ir tolimesni darbai

Mes pasiūlėme kontekstų modelių ir produktų požymių modelių integravimo karkasą, paremtą požymių modeliavimo notacija bei meta-grafų koncepcija. Kontekstų/produktų požymių modeliai gali būti naudojami kuriant nuo konteksto priklausomų, tam tikros srities sistemų produktų šeimynas, pavyzdžiui, socialiai pritaikomus tinklalapius.

Tolimesniuose darbuose numatoma įsigilinti į vykdomųjų kontekstų/produktų požymių modelių kūrimą ir verifikavimą naudojant Prolog loginę programavimo kalbą.

Literatūra

[1] **Pohl K., Bockle G., van der Linden F.** *Software Product Line Engineering.* Springer 2005.

[2] **Kang, K.C., Lee J., Donohoe P.** Feature-Oriented Product Line Engineering. *IEEE Software,* 9(4), 58-65, 2002.

[3] **Buhne S., Lauenroth K., Pohl K.** Modelling requirements Variability across Product Lines. *Proc. of the 13th IEEE Int. Conf. on Requirements Engineering, RE'05*, Paris, France, 41–50.

[4] **Czarnecki K., Helsen S., Eisenecker U.** Staged Configuration through specialization and Multi-Level Configuration of Feature Models. *Software Process Improvement and Practice,* 10(2), 143 – 169, 2005.

[5] **Wagelaar D.** Towards context-aware feature modelling using ontologies. *Proc. of MoDELS 2005 Int. workshop on MDD for Software Product Lines,* 2005.

[6] **Ubayashi N., Nakajima S., Hirayama M.** Context-Dependent Product Line Practice for Constructing Reliable Embedded Systems. *Proc. of 14th Int. Conf. on Software Product Lines, SPLC 2010,* Jeju Island, South Korea, September 13-17, 2010. LNCS 6287, Springer 2010, 1-15.

[7] **[autorių straipsnis]**.

[8] **Ali R.** *Modeling and Reasoning about Contextual Requirements: Goal-based Framework.* PhD thesis, University of Trento, Department of Information Engineering and Computer Science, 2010.

[9] **Viera V., Brézillon P., Salgado A.C., Tedesco P.A.** Context-Oriented Model for Domain-Independent Context Management. *Revue d'intelligence artificielle,* 22, 609-627, 2008.

[10] **Fricker S., Stoiber R.** Relating Product Line Context to Requirements Engineering Processes Using Design Rationale. *Proc. of Software Engineering 2008,* München, Germany. LNI 122 GI 2008, 240-251.

[11] **Hong D., Schmidtke H.R., Woo W.** Linking context modelling and contextual reasoning. *Proc. of 4th Int. Workshop on Modeling and Reasoning in Context (MRC),* 37-48. Roskilde University, 2007.

[12] **Basu A., Blanning R.W., Shtub A.** Metagraphs in hierarchical modeling. *Management Science,* 43: 623-639, 1997.

[13] **van Deursen A., Klint P.** Domain-Specific Language Design Requires Feature Descriptions. *Journal of Computing and Information Technology,* 2002, 10(1), 1-17.

[14] **Hofstede G., Hofstede G.J.** *Cultures and organizations: software of the mind.* New York: McGraw-Hill, 2005.

[15] **Alsarraj, S.** *An approach for Experimenting with Website Designs.* MSc. Thesis. Vrije Universiteit Brussel, 2009.

12.3 BIBLIOTEKOS POŽYMIŲ MODELIŲ APRAŠAI

Pridedami naudotos bibliotekos požymių modelių aprašai FDL kalba.

12.3.1 Cellphone-fm.fdl

```
cellphone: all (?wireless, accucell, display)
wireless: more-of (infrared, bluetooth)
accucell: one-of (liion, nimh, nica)
display: one-of (color, monochrome)
bluetooth requires liion
color excludes nica
```

12.3.2 Digital-video-system-fm.fdl

```
dvs: all (control, ?networkhw, serverpc, play, ?clientpc, ?handheld)
control: all (remote, ?telephone, ?net, ?edit, play)
telephone: all (?sms)
net: all (?web, ?wap, email)
edit: all (?addmusic)
play: all (?slides, ?audio, video)
networkhw: all (?modem, ?ethernet)
serverpc: all (?network, ?irdaport)
play: all (?ondemand)
clientpc: all (?network)
net requires networkhw
handheld requires irdaport
addmusic requires audio
```

12.3.3 Documentation-generation-fm.fdl

```
docgen: all (analysis, presentation, database)
analysis: all (languageanalysis, ?versionmngt, ?subsystems)
languageanalysis: more-of (cobol, jcl, sql, delphi, progress)
cobol: one-of (ibmcobol, microfocuscobol)
presentation: all (localization, interaction, mainpages, ?visualizations)
localization: more-of (english, dutch)
interaction: one-of (static, dynamic)
mainpages: more-of (programpage, copybookpage, statisticspage, indexes, searchpage, subsystempage,
sourcepage, sourcedifference)
programpage: more-of (annotationsection, activationsection, entitiessection, parameterssection)
statisticspage: one-of (withhistory, withouthistory)
visualizations: more-of (performgraph, conditionalperformgraph, jclgraph, subsystemgraph,
overviewgraph)
database: one-of (db2, oracle, mysql)
subsystempage requires subsystems
subsystemgraph requires subsystems
sourcedifference requires versionmngt
performgraph requires cobol
conditionalperformgraph requires cobol
jclgraph requires jcl
static excludes annotationsection
static excludes searchpage
```

12.3.4 Eshop-fm.fdl

```
eShop: all (storefront, businessmanagement)
storefront: all (?homepage, ?registration, catalog, ?wishlist, buypaths, ?customerservice,
?userbehaviourtracking)
homepage: more-of (Staticcontent, Dynamiccontent)
Dynamiccontent: all (Contenttype, Variationsource)
Contenttype: more-of (Welcomemessage, specialoffers)
Variationsource: more-of (Timedependent, Personalized)
registration: all (registrationenforcement, Registrationinformation, ?userbehaviourtrackinginfo)
registrationenforcement: more-of (Registertobrowse, registertobuy, None)
Registrationinformation: all (Logincredentials, ?shippingaddress, ?Billingaddress,
?Creditcardinformation, ?Demographics, ?PersonalInformation, ?preferences, ?Reminders,
?quickcheckoutprofile, ?Customfields)
shippingaddress: all (?Multipleshippingaddresses)
Billingaddress: all (?Multiplebillingaddresses)
Creditcardinformation: all (Cardholdername, Cardnumber, Expirydate, ?Securityinformation)
Demographics: more-of (Age, Income, Education, CustomDemographicfield)
preferences: more-of (Sitelayout, Listsize, Language)
```

```
catalog: all (productinformation, ?categories, ?Multiplecatalogs, ?Searching, ?Browsing,
?Customviews)
productinformation: all (producttype, basicinformation, ?detailedinformation, ?warrantyinformation,
?customerreviews, ?associatedassets, ?productvariants, ?size, ?weight, ?availability,
?customfields)
producttype: more-of (eletronicgoods, physicalgoods, services)
associatedassets: more-of (Documents, Mediafiles)
Mediafiles: more-of (Image, Video, Sound)
Image: more-of (Thumbnail, 2Dimage, 3Dimage, 360degreesimage, Differentperspectives, Gallery)
productvariants: all (?Complexproductconfiguration)
categories: all (categoriescatalog)
categoriescatalog: all (?Categories)
Categories: all (?Multi-level, ?Multipleclassification)
Searching: more-of (Basicsearch, Advancedsearch)
Browsing: all (Productpage, ?categorypage, ?Indexpage)
Indexpage: all (?Sortingfilters)
Sortingfilters: more-of (Price, Quality, Price-Qualityratio, Manufacturername, Customfilter)
Customviews: all (?Seasonalproductviews, ?Personalizedviews)
wishlist: all (?wishlistsavedaftersession, ?emailwishlist, ?Multiplewishlists, ?permissions)
permissions: more-of (Publicaccess, Restrictedaccess, Privateaccess)
buypaths: all (Shoppingcart, Checkout, Orderconfirmation)
Shoppingcart: all (Inventorymanagementpolicy, Cartcontentpage, ?Cartsummarypage,
?Cartsaveaftersession)
Checkout: all (Checkouttype, ?shippingoptions, Taxationoptions, Paymentoptions)
Checkouttype: more-of (registeredcheckout, Guestcheckout)
registeredcheckout: all (?quickcheckout)
quickcheckout: all (?Enableprofileupdateoncheckout)
shippingoptions: all (?Qualityofserviceselection, ?Carrierselection, ?Giftoptions,
?Multipleshipments, Shippingcostcalculation)
Taxationoptions: more-of (Customtaxation, Taxgateways)
Customtaxation: all (Type, Ammountspecification)
Type: more-of (Fixed-ratetaxation, Rule-basedtaxation)
Rule-basedtaxation: all (Taxcodes, Address, ?Resolution)
Address: all (shipping2, ?Billing)
Resolution: more-of (Country, Region, City)
Ammountspecification: more-of (Surcharge, Percentage)
Taxgateways: more-of (CertiTAX, CyberSource, Customtaxgateway)
Paymentoptions: all (Paymenttypes, ?Frauddetection, ?Paymentgateways)
Paymenttypes: more-of (COD, Creditcard, Debitcard, Eletroniccheque, Faxmailorder, Purchaseorder,
Giftcertificate, Phoneorder, Custompaymenttype)
Paymentgateways: more-of (Authorize.Net, CyberSource, LinkPoint, Paradata, SkipJack,
VerisignPayflowPro, Custompaymentgateway)
Orderconfirmation: more-of (Eletronicpage, E-mail, Phone, Mail)
customerservice: more-of (Questionandfeedbackforms, Productreturns, Orderstatusreview,
Shipmentstatustracking)
Questionandfeedbackforms: all (?Questionandfeedbacktracking)
Orderstatusreview: all (Filteringcriteria, ?Requestorderhardcopy)
Filteringcriteria: more-of (Ordernumber, Orderdate, Orderstatus)
Shipmentstatustracking: more-of (Internaltracking, Partnertracking)
userbehaviourtracking: all (Behaviourtracked)
Behaviourtracked: more-of (locallyvisitedpages, externalreferringpages,
behaviourtrackedpreviouspurchases)
businessmanagement: all (Ordermanagement, ?Targeting, ?Affiliates, ?inventorytracking,
?procurement, ?reportingandanalysis, ?Externalsystemsintegration, Administration)
Ordermanagement: all (Fulfillment)
Fulfillment: more-of (physicalgoodsfulfillment, eletronicgoodsfulfillment, servicesfulfillment)
physicalgoodsfulfillment: all (warehousemanagement, shipping)
shipping: more-of (Customshippingmethod, Shippinggateways)
Customshippingmethod: all (Pricing)
Pricing: all (Flatrate, ?Ratefactors)
Ratefactors: more-of (Quantitypurchased, Ordertotal, Weight, Productclassification)
Shippinggateways: more-of (FedEX, UPS, USPS, CanadaPost, Customshippinggateway)
eletronicgoodsfulfillment: all (Filerepository, Licensemanagement)
servicesfulfillment: all (?Appointmentscheduling, ?Resourceplanning)
Targeting: all (Targetingcriteria, Targetingmechanisms, Displayandnotification, ?Campaigns)
Targetingcriteria: more-of (customerpreferences, Personalinformation, Demographics,
targetingcriteriapreviouspurchases, Shoppingcartcontent, wishlistcontent, previouslyvisitedpages,
Dateandtime, Customtargetcriteria)
Targetingmechanisms: more-of (Advertisements, discounts, Sellstrategies)
Advertisements: all (Advertisementtypes, Advertisementsources, ?Advertisementresponsetracking,
?Contextsensitiveads)
Advertisementtypes: more-of (Bannerads, Pop-upads)
Advertisementsources: more-of (Houseadvertisements, Paidadvertisements)
Paidadvertisements: all (Advertisementmanagementinterface)
discounts: all (Discountconditions, Award, Eligibilityrequirements, Graduationby, ?Coupons,
Handlingmultiplediscounts)
Discountconditions: all (Productandquantityscope, Timescope, ?Purchasevaluescope)
Award: more-of (Percentagediscount, Fixeddiscount, Freeitem)
Eligibilityrequirements: all (?Customersegments, ?Shippingaddress)
Graduationby: more-of (Purchasevalue, Quantity)
Sellstrategies: more-of (Productkitting, Up-selling, Cross-selling)
Up-selling: all (Substituteproducts)
Cross-selling: all (Pastcustomersalsobought)
Displayandnotification: more-of (Assignmenttopagetypesfordisplay, Productflagging, E-mails)
E-mails: all (?personalizedemails, ?Responsetracking)
Affiliates: all (Affiliateregistration, Commissiontracking)
inventorytracking: all (?Allowbackorders)
```

```
procurement: all (Stockreplenishment)
Stockreplenishment: all (Manual, ?automatic)
automatic: all (Non-repudiationservice)
reportingandanalysis: all (Reporttypes, Reportformats, Levelofdetail)
Externalsystemsintegration: more-of (fulfillmentsystem, Inventorymanagementsystem,
procurementsystem, Externaldistributorsystem)
Administration: all (Contentmanagement, Storeadministration)
Contentmanagement: all (Productdatabasemanagement, Presentationoptions, Generallayoutmanagement,
?Contentapproval)
Storeadministration: all (Sitesearch, Searchengineregistration, Domainnamesetup)
specialoffers requires discounts
registeredcheckout requires registrationenforcement
registeredcheckout requires registertobuy
customerpreferences requires preferences
quickcheckout requires quickcheckoutprofile
userbehaviourtrackinginfo requires userbehaviourtracking
eletronicgoods requires eletronicgoodsfulfillment
physicalgoods requires physicalgoodsfulfillment
services requires servicesfulfillment
physicalgoods requires size
eletronicgoods requires size
physicalgoods requires weight
availability requires inventorytracking
categorypage requires categories
wishlist requires wishlistsavedaftersession
registration requires wishlistsavedaftersession
emailwishlist requires registration
permissions requires registration
shippingoptions requires shipping
wishlistcontent requires wishlist
previouslyvisitedpages requires locallyvisitedpages
previouslyvisitedpages requires externalreferringpages
```

12.3.5 Graph-manipulation.fdl

```
graph: all (services)
services: all (graphmanipulation, ?doundoredo, ?zoominout)
graphmanipulation: all (delete, select, add, move, ?compose, ?layeradjust)
select: all (selectionmode)
selectionmode: one-of (incrementselection, nonincrementselection)
add: all (entityadd, connectoradd)
move: all (movingmode, ?movingconstraint)
movingmode: one-of (outlinemoving, contentmoving)
movingconstraint: one-of (horizontalconstraint, verticalconstraint)
layeradjust: all (uplayer, downlayer)
doundoredo: all (selectdoundoredo, adddoundoredo, deletedoundoredo, movedoundoredo,
?composedoundoredo, ?layeradjustdoundoredo)
doundoredo requires graphmanipulation
zoominout requires graphmanipulation
compose excludes doundoredo
compose requires composedoundoredo
composedoundoredo requires compose
composedoundoredo requires doundoredo
layeradjust excludes doundoredo
layeradjust requires layeradjustdoundoredo
layeradjustdoundoredo requires doundoredo
layeradjustdoundoredo requires layeradjust
```

12.3.6 Graph-product-line-fm.fdl

```
gpl: all (driver, gtp, ?weight, ?search, algorithms)
driver: all (benchmark)
gtp: one-of (directed, undirected)
weight: one-of (weighted, unweighted)
search: one-of (bfs, dfs)
algorithms: more-of (number, connected, stronglyc, cycle, mstprim, mstkruskal, shortest)
number requires bfs
number requires dfs
connected requires undirected
connected requires weighted
stronglyc requires directed
stronglyc requires dfs
cycle requires dfs
mstkruskal requires undirected
mstkruskal requires unweighted
mstprim requires undirected
mstprim requires unweighted
mstkruskal excludes mstprim
shortest requires directed
shortest requires unweighted
```

12.3.7 Home-integration-system-fm.fdl

```
his: all (services, administration, ?communication, detectiondevices, actiondevices,
monitorcontrol, qualityattributes)
services: all (security, intrusion, fire, ?flood)
intrusion: all (detection, action)
detection: all (motion)
motion: all (monitoringdetecting)
monitoringdetecting: one-of (discretevalue, continuousvalue)
action: all (dooroperation, alarm, ?message)
message: all (?data, voice)
fire: all (detection, action)
detection: all (smoke)
smoke: all (monitoringdetecting)
monitoringdetecting: one-of (discretevalue, continuousvalue)
action: all (water, ?gas)
flood: all (detection, action)
detection: all (moisture)
moisture: all (monitoringdetecting)
monitoringdetecting: one-of (discretevalue, continuousvalue)
action: all (watermain, ?pumping)
administration: all (hmi)
hmi: one-of (standard, advanced)
communication: all (telephone, ?internet)
internet: all (connection)
connection: one-of (tcp, udp)
detectiondevices: all (motionsensor, skokesensor, ?moisturesensor)
actiondevices: all (sprinkler, ?sumppump)
monitorcontrol: all (direct, scheduled, respondingstrategy, eventbased)
scheduled: all (periodic, onetime)
respondingstrategy: all (?sequential, priority)
qualityattributes: all (usability, scalability, reliability)
reliability: all (redundancycontrol)
redundancycontrol: one-of (active)
water requires sprinkler
pumping requires sumppump
flood requires moisturesensor
message requires communication
```

12.3.8 Insurance-product-fm.fdl

```
insuranceproduct: all (insuredobject, coverage, payment, conditions, premium, payee)
insuredobject: one-of (corporation, realty, movableproperty, person)
coverage: more-of (illness, life, unemployment, damage, loss)
payment: all (id2, ?ownrisk)
id2: one-of (service, ammount)
conditions: all (acceptance, exception)
premium: one-of (direct, periodical)
payee: more-of (person, payeecorporation)
corporation excludes illness
person excludes damage
loss requires movableproperty
corporation requires payeecorporation
```

12.3.9 James-fm.fdl

```
james: all (usermngt, ?wsinterface, gui, core, modules)
usermngt: one-of (db, ldap)
gui: more-of (pc, pda)
modules: more-of (calendar, forum, congressmngt, repository)
pda excludes repository
congressmngt requires repository
```

12.3.10 Jplug-fm.fdl

```
jplug: all (interface, ?guibuilder, ?modelcodesynch, ?compiler, ?diagrambuilder)
interface: one-of (sdi, mdi)
guibuilder: all (java, ?qt)
compiler: more-of (javac, mvc)
diagrambuilder: all (uml, ?petrinet)
guibuilder excludes sdi
modelcodesynch requires diagrambuilder
```

12.3.11 Key-word-in-context-fm.fdl

```
kwic: all (inputoutput, circularshift, alphasort, archstyle)
inputoutput: more-of (file, console)
circularshift: all (shiftdata, ?compression)
shiftdata: one-of (implicit, explicit)
compression: all (shiftprocessing, id2, ?noisewords)
id2: one-of (eachline, alllines)
alphasort: all (order, casesensitivity)
order: one-of (descendant, ascendant)
casesensitivity: one-of (sensitive, insensitive)
archstyle: one-of (shareddata, adt, implicitinvocation, pipeandfilter)
compression requires explicit
pipeandfilter requires alllines
pipeandfilter requires explicit
```

12.3.12 Model-transformation-fm.fdl

```
ModelTransformation: all (Transformationrules, ?Ruleapplicationscoping, Source-Targetrelationship,
Ruleapplicationstrategy, Rulescheduling, Ruleorganization, Tracing, Directionality)
Transformationrules: all (LHS/RHS, ?LHS/RHSSyntacticSeparation, ?Bidirectionality,
?Parameterization, ?Intermediatestructures)
LHS/RHS: all (?Variables, id1)
Variables: one-of (Untyped, Syntacticallytyped, Semanticallytyped)
id1: more-of (Patterns, Logic)
Patterns: all (Form, Syntax, Typing)
Form: one-of (Strings, Terms, Graphs)
Syntax: more-of (Abstract, Concrete)
Concrete: more-of (Textual, Graphical)
Typing: one-of (typuntyped, typsyntacticallytyped, typsemanticallytyped)
Logic: more-of (Non-executable, Executable)
Executable: more-of (Imperative, Declarative)
Ruleapplicationscoping: more-of (scopingsource, targetsource)
Source-Targetrelationship: more-of (Newtarget, Existingtarget)
Existingtarget: all (?In-place, Update)
Update: one-of (Destructive, Extensiononly)
Ruleapplicationstrategy: more-of (Deterministic, Non-deterministic, Interactive)
Non-deterministic: more-of (Concurrent, One-point)
Rulescheduling: all (Form, Ruleselection, ?RuleIteration, ?Phasing)
Form: more-of (Implicit, Explicit)
Explicit: one-of (Internal, External)
Ruleselection: more-of (Explicitcondition, Non-determinism, Conflictresolution, Interactive)
RuleIteration: one-of (Recursion, Looping, FixpointIteration)
Ruleorganization: all (?Modulatirymechanisms, ?Reusemechanisms, OrganizationalStructure)
Reusemechanisms: more-of (Inheritance, Logicalcomposition)
OrganizationalStructure: one-of (Source-oriented, Target-oriented, Independent)
Tracing: all (?Dedicatedsupport)
Dedicatedsupport: all (Storagelocation, Control)
Storagelocation: one-of (Model, Separate)
Model: more-of (modelsource, modeltarget)
Control: one-of (Manual, Automatic)
Automatic: one-of (Allrules, Selectedrules)
Directionality: more-of (Unidirectional, Bidirectional)
Bidirectional: one-of (Bidirectionalrules, Complementarypairs)
```

12.3.13 Monitor-engine-system-fm.fdl

```
monitorenginesystem: all (engineperformance, fuelconsumption)
engineperformance: all (temperatures, rpm, exhaustleveltemp)
temperatures: all (?colant, oil, engine, transmission)
fuelconsumption: all (measures, methods)
measures: one-of (lkm, gallonmile)
methods: more-of (distance, typeofdriving, drive)
drive requires rpm
```

12.3.14 Telecommunication-system-fm.fdl

```
tecom: all (?ipvoice, rack, ?messaging)
ipvoice: all (software)
software: one-of (swpack1, swpack2)
rack: all (sizes)
sizes: one-of (large, medium, small)
messaging: all (?upgradeswpack)
swpack1 excludes upgradeswpack
ipvoice excludes messaging
ipvoice requires upgradeswpack
```

12.3.15 Text-editor-fm.fdl

```
editorconfig: all (backup, ?documentclass)
backup: all (?autosave, ?backuponchange, backupextension)
autosave: all (minutes)
backupextension: one-of (filebak, filebakext, fileextbak)
documentclass: all (associatedfileext, commands, ?syntaxhighlighting)
associatedfileext: all (?ext)
commands: all (?removeblanklines, ?spellcheck, ?dosunixconversion)
syntaxhighlighting: all (syntaxdefinitionfile)
```

12.3.16 Thread-domain-fm.fdl

```
thread: all (threadinstance)
threadinstance: one-of (multiple, single)
multiple: all (?coroutine)
coroutine: all (?dispatching, location, size)
dispatching: all (dispachmethod, ?scheduling)
dispachmethod: all (globallife)
scheduling: all (?bundleschedule, ?idlecontrol, ?queryschedule, threadschedule)
bundleschedule: all (?bundlepreemption, bundlestrategy)
bundlestrategy: all (?bundlestrategyreplugging, id1)
id1: one-of (bundlefcfs, bundlelcfs)
idlecontrol: one-of (idleguard, idlepanic, idleuser)
threadschedule: all (?threadpreemption, threadstrategy)
threadpreemption: all (?nonpreemptablethreads, ?schedulerlockable)
threadstrategy: all (id3, ?threadstrategyreplugging)
id3: one-of (priority, threadfcfs, threadlcfs)
priority: all (?schedulesamepriority, prioritybehaviour, prioritycount)
prioritybehaviour: one-of (prioritycooperative, prioritypreemptive)
prioritycount: one-of (priority16, priority32, priority8)
location: one-of (integrated, separated)
size: all (?floatset, ?volatileset, minimalset)
```

12.3.17 Virtual-office-of-future-fm.fdl

```
vof: all (?undefinedonpaper, followme, ?virtualprinter)
followme: all (?undefinedonpaper, userlocalize, logon)
userlocalize: all (userposmethod)
userposmethod: one-of (accesspoint, rfid)
logon: all (?undefinedonpaper)
virtualprinter: all (printerregister, fileupload, ?fileconverter)
```

12.3.18 Web-portal-fm.fdl

```
webportal: all (?addservices, webserver, ?persistence, ?ri, ?performance)
addservices: all (?sitestats, ?sitesearch, ?adserver)
sitestats: all (basic, ?advanced)
sitesearch: all (?images, ?text)
text: all (html, ?dynamic)
adserver: all (reports, ?popups, banners, ?keyword)
banners: all (banimg, ?banflash)
webserver: all (?logging, ?protocol, content)
logging: one-of (db, file)
protocol: more-of (nttp, ftp, https)
content: all (static, ?active)
active: more-of (asp, php, jsp, cgi)
persistence: one-of (xml, database)
ri: more-of (datastorage, datatransfer, userauth)
performance: one-of (ms, sec, min)
keyword requires text
dynamic requires active
db requires database
datatransfer requires https
file requires ftp
https excludes ms
```

12.4 TRANSFORMUOTŲ POŽYMIŲ MODELIŲ APRAŠAI

Pridedami naudotos bibliotekos transformuotų požymių modelių aprašai PROLOG kalba.

12.4.1 Cellphone-fm.pro

```
cellphone :- all(accucell, display, alt(wireless)).
accucell :- one_of(liion, nimh, nica).
display :- one_of(color, monochrome), excludes(color, nica).
wireless :- more_of(infrared, bluetooth), requires(bluetooth, liion).
nimh.
liion.
nica.
monochrome.
bluetooth.
color.
infrared.
```

12.4.2 Digital-video-system-fm.pro

```
networkhw :- all(alt(modem), alt(ethernet)).
edit :- all(alt(addmusic)), requires(addmusic, audio).
clientpc :- all(alt(network)).
control :- all(remote, play, alt(telephone), alt(net), alt(edit)), requires(net, networkhw).
telephone :- all(alt(sms)).
play :- all(video, alt(slides), alt(audio), alt(ondemand)).
net :- all(email, alt(web), alt(wap)).
serverpc :- all(alt(network), alt(irdaport)).
dvs :- all(control, serverpc, play, alt(networkhw), alt(clientpc), alt(handheld)),
requires(handheld, irdaport).
handheld.
email.
web.
modem.
remote.
sms.
video.
ondemand.
slides.
network.
irdaport.
wap.
addmusic.
audio.
ethernet.
```

12.4.3 Documentation-generation-fm.pro

```
mainpages :- more_of(programpage, copybookpage, statisticspage, indexes, searchpage, subsystempage,
sourcepage, sourcedifference), requires(sourcedifference, versionmngt), requires(subsystempage,
subsystems).
languageanalysis :- more_of(cobol, jcl, sql, delphi, progress).
analysis :- all(languageanalysis, alt(versionmngt), alt(subsystems)).
database :- one_of(db2, oracle, mysql).
docgen :- all(analysis, presentation, database).
statisticspage :- one_of(withhistory, withouthistory).
cobol :- one_of(ibmcobol, microfocuscobol).
presentation :- all(localization, interaction, mainpages, alt(visualizations)).
visualizations :- more_of(performgraph, conditionalperformgraph, jclgraph, subsystemgraph,
overviewgraph), requires(subsystemgraph, subsystems), requires(performgraph, cobol),
requires(conditionalperformgraph, cobol), requires(jclgraph, jcl).
programpage :- more_of(annotationsection, activationsection, entitiessection, parameterssection).
interaction :- one_of(static, dynamic), excludes(static, annotationsection), excludes(static,
searchpage).
localization :- more_of(english, dutch).
microfocuscobol.
static.
copybookpage.
indexes.
jclgraph.
delphi.
versionmngt.
subsystempage.
db2.
oracle.
subsystemgraph.
```

```
annotationsection.
sql.
conditionalperformgraph.
dutch.
sourcepage.
subsystems.
parameterssection.
jcl.
activationsection.
mysql.
progress.
entitiessection.
overviewgraph.
sourcedifference.
dynamic.
ibmcobol.
searchpage.
performgraph.
withhistory.
withouthistory.
english.
```

12.4.4 Eshop-fm.pro

```
homepage :- more_of(staticcontent, dynamiccontent).
advertisementsources :- more_of(houseadvertisements, paidadvertisements).
targetingmechanisms :- more_of(advertisements, discounts, sellstrategies).
producttype :- more_of(eletronicgoods, physicalgoods, services), requires(physicalgoods,
physicalgoodsfulfillment), requires(physicalgoods, size), requires(physicalgoods, weight),
requires(services, servicesfulfillment), requires(eletronicgoods, eletronicgoodsfulfillment),
requires(eletronicgoods, size).
storeadministration :- all(sitesearch, searchengineregistration, domainnamesetup).
creditcardinformation :- all(cardholdername, cardnumber, expirydate, alt(securityinformation)).
ratefactors :- more_of(quantitypurchased, ordertotal, weight, productclassification).
demographics :- more_of(age, income, education, customdemographicfield).
taxationoptions :- more_of(customtaxation, taxgateways).
resolution :- more_of(country, region, city).
indexpage :- all(alt(sortingfilters)).
affiliates :- all(affiliateregistration, commissiontracking).
filteringcriteria :- more_of(ordernumber, orderdate, orderstatus).
customtaxation :- all(type, ammountspecification).
checkouttype :- more_of(registeredcheckout, guestcheckout), requires(registeredcheckout,
registrationenforcement), requires(registeredcheckout, registertobuy).
shipmentstatustracking :- more_of(internaltracking, partnertracking).
shoppingcart :- all(inventorymanagementpolicy, cartcontentpage, alt(cartsummarypage),
alt(cartsaveaftersession)).
categoriescatalog :- all(alt(categories)).
categories :- all(categoriescatalog, alt(multi-level), alt(multipleclassification)).
taxgateways :- more_of(certitax, cybersource, customtaxgateway).
questionandfeedbackforms :- all(alt(questionandfeedbacktracking)).
orderconfirmation :- more_of(eletronicpage, e-mail, phone, mail).
registeredcheckout :- all(alt(quickcheckout)), requires(quickcheckout, quickcheckoutprofile).
permissions :- more_of(publicaccess, restrictedaccess, privateaccess).
fulfillment :- more_of(physicalgoodsfulfillment, eletronicgoodsfulfillment, servicesfulfillment).
behaviourtracked :- more_of(locallyvisitedpages, externalreferringpages,
behaviourtrackedpreviouspurchases).
productinformation :- all(producttype, basicinformation, alt(detailedinformation),
alt(warrantyinformation), alt(customerreviews), alt(associatedassets), alt(productvariants),
alt(size), alt(weight), alt(availability), alt(customfields)), requires(availability,
inventorytracking).
ammountspecification :- more_of(surcharge, percentage).
administration :- all(contentmanagement, storeadministration).
registrationenforcement :- more_of(registertobrowse, registertobuy, none).
pricing :- all(flatrate, alt(ratefactors)).
checkout :- all(checkouttype, taxationoptions, paymentoptions, alt(shippingoptions)),
requires(shippingoptions, shipping).
graduationby :- more_of(purchasevalue, quantity).
eshop :- all(storefront, businessmanagement).
userbehaviourtracking :- all(behaviourtracked).
paymenttypes :- more_of(cod, creditcard, debitcard, eletroniccheque, faxmailorder, purchaseorder,
giftcertificate, phoneorder, custompaymenttype).
up-selling :- all(substituteproducts).
shippingoptions :- all(shippingcostcalculation, alt(qualityofserviceselection),
alt(carrierselection), alt(giftoptions), alt(multipleshipments)).
preferences :- more_of(sitelayout, listsize, language).
targetingcriteria :- more_of(customerpreferences, personalinformation, demographics,
targetingcriteriapreviouspurchases, shoppingcartcontent, wishlistcontent, previouslyvisitedpages,
dateandtime, customtargetcriteria), requires(previouslyvisitedpages, locallyvisitedpages),
requires(previouslyvisitedpages, externalreferringpages), requires(wishlistcontent, wishlist),
requires(customerpreferences, preferences).
paymentgateways :- more_of(authorize.net, cybersource, linkpoint, paradata, skipjack,
verisignpayflowpro, custompaymentgateway).
advertisementtypes :- more_of(bannerads, pop-upads).
paymentoptions :- all(paymenttypes, alt(frauddetection), alt(paymentgateways)).
```

```
buypaths :- all(shoppingcart, checkout, orderconfirmation).
browsing :- all(productpage, alt(categorypage), alt(indexpage)), requires(categorypage,
categories).
registration :- all(registrationenforcement, registrationinformation,
alt(userbehaviourtrackinginfo)), requires(userbehaviourtrackinginfo, userbehaviourtracking).
dynamiccontent :- all(contenttype, variationsource).
variationsource :- more_of(timedependent, personalized).
ordermanagement :- all(fulfillment).
orderstatusreview :- all(filteringcriteria, alt(requestorderhardcopy)).
servicesfulfillment :- all(alt(appointmentscheduling), alt(resourceplanning)).
sortingfilters :- more_of(price, quality, price-qualityratio, manufacturername, customfilter).
contenttype :- more_of(welcomemessage, specialoffers), requires(specialoffers, discounts).
eletronicgoodsfulfillment :- all(filerepository, licensemanagement).
shippinggateways :- more_of(fedex, ups, usps, canadapost, customshippinggateway).
displayandnotification :- more_of(assignmenttopagetypesfordisplay, productflagging, e-mails).
paidadvertisements :- all(advertisementmanagementinterface).
rule-basedtaxation :- all(taxcodes, address, alt(resolution)).
catalog :- all(productinformation, alt(categories), alt(multiplecatalogs), alt(searching),
alt(browsing), alt(customviews)).
businessmanagement :- all(ordermanagement, administration, alt(targeting), alt(affiliates),
alt(inventorytracking), alt(procurement), alt(reportingandanalysis),
alt(externalsystemsintegration)).
billingaddress :- all(alt(multiplebillingaddresses)).
eligibilityrequirements :- all(alt(customersegments), alt(shippingaddress)).
customshippingmethod :- all(pricing).
customerservice :- more_of(questionandfeedbackforms, productreturns, orderstatusreview,
shipmentstatustracking).
searching :- more_of(basicsearch, advancedsearch).
discountconditions :- all(productandquantityscope, timescope, alt(purchasevaluescope)).
image :- more_of(thumbnail, 2dimage, 3dimage, 360degreesimage, differentperspectives, gallery).
targeting :- all(targetingcriteria, targetingmechanisms, displayandnotification, alt(campaigns)).
storefront :- all(catalog, buypaths, alt(homepage), alt(registration), alt(wishlist),
alt(customerservice), alt(userbehaviourtracking)), requires(wishlist, wishlistsavedaftersession),
requires(registration, wishlistsavedaftersession).
externalsystemsintegration :- more_of(fulfillmentsystem, inventorymanagementsystem,
procurementsystem, externaldistributorsystem).
stockreplenishment :- all(manual, alt(automatic)).
e-mails :- all(alt(personalizedemails), alt(responsetracking)).
type :- more_of(fixed-ratetaxation, rule-basedtaxation).
discounts :- all(discountconditions, award, eligibilityrequirements, graduationby,
handlingmultiplediscounts, alt(coupons)).
quickcheckout :- all(alt(enableprofileupdateoncheckout)).
advertisements :- all(advertisementtypes, advertisementsources, alt(advertisementresponsetracking),
alt(contextsensitiveads)).
address :- all(shipping2, alt(billing)).
reportingandanalysis :- all(reporttypes, reportformats, levelofdetail).
cross-selling :- all(pastcustomersalsobought).
automatic :- all(non-repudiationservice).
physicalgoodsfulfillment :- all(warehousemanagement, shipping).
associatedassets :- more_of(documents, mediafiles).
productvariants :- all(alt(complexproductconfiguration)).
shipping :- more_of(customshippingmethod, shippinggateways).
customviews :- all(alt(seasonalproductviews), alt(personalizedviews)).
contentmanagement :- all(productdatabasemanagement, presentationoptions, generallayoutmanagement,
alt(contentapproval)).
procurement :- all(stockreplenishment).
award :- more_of(percentagediscount, fixeddiscount, freeitem).
mediafiles :- more_of(image, video, sound).
registrationinformation :- all(logincredentials, alt(shippingaddress), alt(billingaddress),
alt(creditcardinformation), alt(demographics), alt(personalinformation), alt(preferences),
alt(reminders), alt(quickcheckoutprofile), alt(customfields)).
sellstrategies :- more_of(productkitting, up-selling, cross-selling).
wishlist :- all(alt(wishlistsavedaftersession), alt(emailwishlist), alt(multiplewishlists),
alt(permissions)), requires(permissions, registration), requires(emailwishlist, registration).
shippingaddress :- all(alt(multipleshippingaddresses)).
inventorytracking :- all(alt(allowbackorders)).
multiplebillingaddresses.
fulfillmentsystem.
userbehaviourtrackinginfo.
contentapproval.
previouslyvisitedpages.
education.
procurementsystem.
productkitting.
handlingmultiplediscounts.
authorize.net.
cartsaveaftersession.
multipleclassification.
appointmentscheduling.
manual.
seasonalproductviews.
assignmenttopagetypesfordisplay.
substituteproducts.
qualityofserviceselection.
none.
creditcard.
commissiontracking.
```

region.
productreturns.
video.
welcomemessage.
partnertracking.
quantity.
locallyvisitedpages.
phoneorder.
linkpoint.
eletronicgoods.
sound.
advertisementresponsetracking.
fixed-ratetaxation.
wishlistcontent.
debitcard.
quality.
productflagging.
cartsummarypage.
360degreesimage.
categorypage.
eletroniccheque.
freeitem.
orderstatus.
searchengineregistration.
giftcertificate.
restrictedaccess.
productclassification.
custompaymentgateway.
3dimage.
carrierselection.
ordernumber.
levelofdetail.
staticcontent.
giftoptions.
age.
multiplewishlists.
inventorymanagementsystem.
quickcheckoutprofile.
orderdate.
shipping2.
affiliateregistration.
advertisementmanagementinterface.
basicinformation.
advancedsearch.
enableprofileupdateoncheckout.
warehousemanagement.
productandquantityscope.
publicaccess.
canadapost.
non-repudiationservice.
taxcodes.
city.
campaigns.
reportformats.
cardholdername.
eletronicpage.
presentationoptions.
domainnamesetup.
warrantyinformation.
physicalgoods.
customtargetcriteria.
customerreviews.
externaldistributorsystem.
bannerads.
purchasevalue.
registertobrowse.
reminders.
ups.
services.
personalized.
surcharge.
filerepository.
pastcustomersalsobought.
allowbackorders.
quantitypurchased.
detailedinformation.
phone.
privateaccess.
differentperspectives.
responsetracking.
listsize.
customerpreferences.
productpage.
reporttypes.
2dimage.
contextsensitiveads.
timedependent.

```
price.
requestorderhardcopy.
externalreferringpages.
flatrate.
customdemographicfield.
logincredentials.
personalinformation.
percentage.
fixeddiscount.
purchaseorder.
customshippinggateway.
guestcheckout.
price-qualityratio.
availability.
sitesearch.
certitax.
customersegments.
ordertotal.
cybersource.
paradata.
thumbnail.
purchasevaluescope.
personalizedviews.
fedex.
coupons.
e-mail.
pop-upads.
shippingcostcalculation.
licensemanagement.
multiplecatalogs.
emailwishlist.
inventorymanagementpolicy.
houseadvertisements.
personalizedemails.
mail.
shoppingcartcontent.
internaltracking.
securityinformation.
resourceplanning.
multi-level.
expirydate.
faxmailorder.
productdatabasemanagement.
dateandtime.
verisignpayflowpro.
gallery.
language.
multipleshippingaddresses.
complexproductconfiguration.
sitelayout.
usps.
income.
billing.
skipjack.
customfields.
customtaxgateway.
customfilter.
size.
custompaymenttype.
documents.
behaviourtrackedpreviouspurchases.
timescope.
registertobuy.
wishlistsavedaftersession.
frauddetection.
cartcontentpage.
basicsearch.
targetingcriteriapreviouspurchases.
questionandfeedbacktracking.
country.
specialoffers.
cardnumber.
weight.
cod.
generallayoutmanagement.
percentagediscount.
manufacturername.
multipleshipments.
```

12.4.5 Graph-manipulation.pro

```
doundoredo :- all(selectdoundoredo, adddoundoredo, deletedoundoredo, movedoundoredo,
alt(composedoundoredo), alt(layeradjustdoundoredo)), requires(composedoundoredo, compose),
```

```
requires(composedoundoredo, doundoredo), requires(layeradjustdoundoredo, doundoredo),
requires(layeradjustdoundoredo, layeradjust).
movingconstraint :- one_of(horizontalconstraint, verticalconstraint).
graph :- all(services).
layeradjust :- all(uplayer, downlayer).
graphmanipulation :- all(delete, select, add, move, alt(compose), alt(layeradjust)),
requires(compose, composedoundoredo), requires(layeradjust, layeradjustdoundoredo),
excludes(compose, doundoredo), excludes(layeradjust, doundoredo).
move :- all(movingmode, alt(movingconstraint)).
services :- all(graphmanipulation, alt(doundoredo), alt(zoominout)), requires(zoominout,
graphmanipulation), requires(doundoredo, graphmanipulation).
selectionmode :- one_of(incrementselection, nonincrementselection).
movingmode :- one_of(outlinemoving, contentmoving).
add :- all(entityadd, connectoradd).
select :- all(selectionmode).
selectdoundoredo.
horizontalconstraint.
outlinemoving.
incrementselection.
nonincrementselection.
composedoundoredo.
layeradjustdoundoredo.
downlayer.
adddoundoredo.
zoominout.
deletedoundoredo.
uplayer.
verticalconstraint.
connectoradd.
contentmoving.
movedoundoredo.
entityadd.
compose.
delete.
```

12.4.6 Graph-product-line-fm.pro

```
algorithms :- more_of(number, connected, stronglyc, cycle, mstprim, mstkruskal, shortest),
requires(mstprim, undirected), requires(mstprim, unweighted), requires(shortest, directed),
requires(shortest, unweighted), requires(mstkruskal, undirected), requires(mstkruskal, unweighted),
requires(cycle, dfs), requires(connected, undirected), requires(connected, weighted),
requires(stronglyc, directed), requires(stronglyc, dfs), requires(number, bfs), requires(number,
dfs), excludes(mstkruskal, mstprim).
gtp :- one_of(directed, undirected).
search :- one_of(bfs, dfs).
driver :- all(benchmark).
gpl :- all(driver, gtp, algorithms, alt(weight), alt(search)).
weight :- one_of(weighted, unweighted).
weighted.
mstkruskal.
cycle.
undirected.
stronglyc.
directed.
mstprim.
unweighted.
shortest.
dfs.
bfs.
benchmark.
connected.
number.
```

12.4.7 Home-integration-system-fm.pro

```
redundancycontrol :- one_of(active).
services :- all(security, intrusion, fire, alt(flood)), requires(flood, moisturesensor).
administration :- all(hmi).
monitoringdetecting :- one_of(discretevalue, continuousvalue), one_of(discretevalue,
continuousvalue), one_of(discretevalue, continuousvalue).
action :- all(dooroperation, alarm, water, watermain, alt(message), alt(gas), alt(pumping)),
requires(water, sprinkler), requires(pumping, sumppump), requires(message, communication).
flood :- all(detection, action).
monitorcontrol :- all(direct, scheduled, respondingstrategy, eventbased).
moisture :- all(monitoringdetecting).
his :- all(services, administration, detectiondevices, actiondevices, monitorcontrol,
qualityattributes, alt(communication)).
intrusion :- all(detection, action).
actiondevices :- all(sprinkler, alt(sumppump)).
communication :- all(telephone, alt(internet)).
detection :- all(motion, smoke, moisture).
hmi :- one_of(standard, advanced).
```

```
motion :- all(monitoringdetecting).
internet :- all(connection).
reliability :- all(redundancycontrol).
detectiondevices :- all(motionsensor, skokesensor, alt(moisturesensor)).
fire :- all(detection, action).
connection :- one_of(tcp, udp).
qualityattributes :- all(usability, scalability, reliability).
scheduled :- all(periodic, onetime).
respondingstrategy :- all(priority, alt(sequential)).
message :- all(voice, alt(data)).
smoke :- all(monitoringdetecting).
active.
water.
priority.
eventbased.
pumping.
moisturesensor.
sprinkler.
telephone.
motionsensor.
periodic.
udp.
scalability.
standard.
security.
skokesensor.
tcp.
voice.
sumppump.
dooroperation.
onetime.
direct.
discretevalue.
advanced.
data.
sequential.
continuousvalue.
gas.
watermain.
alarm.
usability.
```

12.4.8 Insurance-product-fm.pro

```
id2 :- one_of(service, ammount).
payee :- more_of(person, payeecorporation), excludes(person, damage).
premium :- one_of(direct, periodical).
insuranceproduct :- all(insuredobject, coverage, payment, conditions, premium, payee).
insuredobject :- one_of(corporation, realty, movableproperty, person), requires(corporation,
payeecorporation), excludes(person, damage), excludes(corporation, illness).
conditions :- all(acceptance, exception).
payment :- all(id2, alt(ownrisk)).
coverage :- more_of(illness, life, unemployment, damage, loss), requires(loss, movableproperty).
loss.
movableproperty.
person.
corporation.
ammount.
unemployment.
damage.
illness.
direct.
acceptance.
realty.
service.
life.
exception.
periodical.
ownrisk.
payeecorporation.
```

12.4.9 James-fm.pro

```
modules :- more_of(calendar, forum, congressmngt, repository), requires(congressmngt, repository).
gui :- more_of(pc, pda), excludes(pda, repository).
usermngt :- one_of(db, ldap).
james :- all(usermngt, gui, core, modules, alt(wsinterface)).
congressmngt.
pda.
forum.
calendar.
ldap.
```

```
wsinterface.
db.
pc.
repository.
core.
```

12.4.10 Jplug-fm.pro

```
interface :- one_of(sdi, mdi).
compiler :- more_of(javac, mvc).
jplug :- all(interface, alt(guibuilder), alt(modelcodesynch), alt(compiler), alt(diagrambuilder)),
requires(modelcodesynch, diagrambuilder), excludes(guibuilder, sdi).
diagrambuilder :- all(uml, alt(petrinet)).
guibuilder :- all(java, alt(qt)).
mdi.
uml.
mvc.
qt.
sdi.
java.
javac.
petrinet.
modelcodesynch.
```

12.4.11 Key-word-in-context-fm.pro

```
casesensitivity :- one_of(sensitive, insensitive).
order :- one_of(descendant, ascendant).
compression :- all(shiftprocessing, id2, alt(noisewords)).
shiftdata :- one_of(implicit, explicit).
circularshift :- all(shiftdata, alt(compression)), requires(compression, explicit).
inputoutput :- more_of(file, console).
alphasort :- all(order, casesensitivity).
archstyle :- one_of(shareddata, adt, implicitinvocation, pipeandfilter), requires(pipeandfilter,
alllines), requires(pipeandfilter, explicit).
kwic :- all(inputoutput, circularshift, alphasort, archstyle).
id2 :- one_of(eachline, alllines).
implicitinvocation.
ascendant.
insensitive.
adt.
file.
noisewords.
console.
pipeandfilter.
eachline.
alllines.
sensitive.
shareddata.
shiftprocessing.
descendant.
implicit.
explicit.
```

12.4.12 Model-transformation-fm.pro

```
automatic :- one_of(allrules, selectedrules).
explicit :- one_of(internal, external).
update :- one_of(destructive, extensiononly).
typing :- one_of(typuntyped, typsyntacticallytyped, typsemanticallytyped).
concrete :- more_of(textual, graphical).
existingtarget :- all(update, alt(in-place)).
transformationrules :- all(lhs/rhs, alt(lhs/rhssyntacticseparation), alt(bidirectionality),
alt(parameterization), alt(intermediatestructures)).
ruleiteration :- one_of(recursion, looping, fixpointiteration).
patterns :- all(form, syntax, typing).
dedicatedsupport :- all(storagelocation, control).
organizationalstructure :- one_of(source-oriented, target-oriented, independent).
tracing :- all(alt(dedicatedsupport)).
syntax :- more_of(abstract, concrete).
ruleorganization :- all(organizationalstructure, alt(modulatirymechanisms), alt(reusemechanisms)).
ruleapplicationscoping :- more_of(scopingsource, targetsource).
control :- one_of(manual, automatic).
modeltransformation :- all(transformationrules, source-targetrelationship, ruleapplicationstrategy,
rulescheduling, ruleorganization, tracing, directionality, alt(ruleapplicationscoping)).
form :- more_of(implicit, explicit), one_of(strings, terms, graphs).
bidirectional :- one_of(bidirectionalrules, complementarypairs).
source-targetrelationship :- more_of(newtarget, existingtarget).
executable :- more_of(imperative, declarative).
```

```
rulescheduling :- all(form, ruleselection, alt(ruleiteration), alt(phasing)).
ruleapplicationstrategy :- more_of(deterministic, non-deterministic, interactive).
logic :- more_of(non-executable, executable).
storagelocation :- one_of(model, separate).
ruleselection :- more_of(explicitcondition, non-determinism, conflictresolution, interactive).
lhs/rhs :- all(id1, alt(variables)).
model :- more_of(modelsource, modeltarget).
id1 :- more_of(patterns, logic).
directionality :- more_of(unidirectional, bidirectional).
non-deterministic :- more_of(concurrent, one-point).
variables :- one_of(untyped, syntacticallytyped, semanticallytyped).
reusemechanisms :- more_of(inheritance, logicalcomposition).
typuntyped.
imperative.
declarative.
phasing.
interactive.
target-oriented.
bidirectionalrules.
independent.
modeltarget.
abstract.
complementarypairs.
external.
semanticallytyped.
non-executable.
extensiononly.
typsemanticallytyped.
newtarget.
looping.
destructive.
inheritance.
manual.
one-point.
syntacticallytyped.
in-place.
conflictresolution.
separate.
textual.
source-oriented.
lhs/rhssyntacticseparation.
strings.
non-determinism.
fixpointiteration.
graphs.
explicitcondition.
selectedrules.
deterministic.
bidirectionality.
graphical.
concurrent.
intermediatestructures.
scopingsource.
allrules.
modulatirymechanisms.
logicalcomposition.
terms.
typsyntacticallytyped.
internal.
targetsource.
untyped.
modelsource.
implicit.
parameterization.
recursion.
unidirectional.
```

12.4.13 Monitor-engine-system-fm.pro

```
engineperformance :- all(temperatures, rpm, exhaustleveltemp).
methods :- more_of(distance, typeofdriving, drive), requires(drive, rpm).
measures :- one_of(lkm, gallonmile).
monitorenginesystem :- all(engineperformance, fuelconsumption).
fuelconsumption :- all(measures, methods).
temperatures :- all(oil, engine, transmission, alt(colant)).
colant.
transmission.
exhaustleveltemp.
typeofdriving.
rpm.
distance.
oil.
gallonmile.
lkm.
```

```
drive.
engine.
```

12.4.14 Telecommunication-system-fm.pro

```
sizes :- one_of(large, medium, small).
rack :- all(sizes).
ipvoice :- all(software).
software :- one_of(swpack1, swpack2), excludes(swpack1, upgradeswpack).
messaging :- all(alt(upgradeswpack)).
tecom :- all(rack, alt(ipvoice), alt(messaging)), requires(ipvoice, upgradeswpack),
excludes(ipvoice, messaging).
medium.
upgradeswpack.
swpack1.
large.
small.
swpack2.
```

12.4.15 Text-editor-fm.pro

```
commands :- all(alt(removeblanklines), alt(spellcheck), alt(dosunixconversion)).
editorconfig :- all(backup, alt(documentclass)).
associatedfileext :- all(alt(ext)).
backupextension :- one_of(filebak, filebakext, fileextbak).
documentclass :- all(associatedfileext, commands, alt(syntaxhighlighting)).
autosave :- all(minutes).
backup :- all(backupextension, alt(autosave), alt(backuponchange)).
syntaxhighlighting :- all(syntaxdefinitionfile).
removeblanklines.
filebakext.
backuponchange.
ext.
fileextbak.
minutes.
dosunixconversion.
syntaxdefinitionfile.
spellcheck.
filebak.
```

12.4.16 Thread-domain-fm.pro

```
location :- one_of(integrated, separated).
priority :- all(prioritybehaviour, prioritycount, alt(schedulesamepriority)).
size :- all(minimalset, alt(floatset), alt(volatileset)).
dispachmethod :- all(globallife).
prioritycount :- one_of(priority16, priority32, priority8).
threadschedule :- all(threadstrategy, alt(threadpreemption)).
bundleschedule :- all(bundlestrategy, alt(bundlepreemption)).
id3 :- one_of(priority, threadfcfs, threadlcfs).
prioritybehaviour :- one_of(prioritycooperative, prioritypreemptive).
idlecontrol :- one_of(idleguard, idlepanic, idleuser).
threadinstance :- one_of(multiple, single).
thread :- all(threadinstance).
threadpreemption :- all(alt(nonpreemptablethreads), alt(schedulerlockable)).
multiple :- all(alt(coroutine)).
scheduling :- all(threadschedule, alt(bundleschedule), alt(idlecontrol), alt(queryschedule)).
bundlestrategy :- all(id1, alt(bundlestrategyreplugging)).
threadstrategy :- all(id3, alt(threadstrategyreplugging)).
dispatching :- all(dispachmethod, alt(scheduling)).
id1 :- one_of(bundlefcfs, bundlelcfs).
coroutine :- all(location, size, alt(dispatching)).
nonpreemptablethreads.
priority8.
bundlelcfs.
queryschedule.
schedulesamepriority.
bundlepreemption.
prioritypreemptive.
priority32.
schedulerlockable.
bundlestrategyreplugging.
volatileset.
floatset.
idleuser.
separated.
bundlefcfs.
threadlcfs.
prioritycooperative.
```

```
threadstrategyreplugging.
priority16.
threadfcfs.
integrated.
minimalset.
idleguard.
idlepanic.
globallife.
single.
```

12.4.17 Virtual-office-of-future-fm.pro

```
userlocalize :- all(userposmethod).
vof :- all(followme, alt(undefinedonpaper), alt(virtualprinter)).
followme :- all(userlocalize, logon, alt(undefinedonpaper)).
userposmethod :- one_of(accesspoint, rfid).
virtualprinter :- all(printerregister, fileupload, alt(fileconverter)).
logon :- all(alt(undefinedonpaper)).
accesspoint.
fileconverter.
fileupload.
printerregister.
undefinedonpaper.
rfid.
```

12.4.18 Web-portal-fm.pro

```
text :- all(html, alt(dynamic)), requires(dynamic, active).
active :- more_of(asp, php, jsp, cgi).
logging :- one_of(db, file), requires(file, ftp), requires(db, database).
sitestats :- all(basic, alt(advanced)).
sitesearch :- all(alt(images), alt(text)).
protocol :- more_of(nttp, ftp, https), excludes(https, ms).
webportal :- all(webserver, alt(addservices), alt(persistence), alt(ri), alt(performance)).
content :- all(static, alt(active)).
banners :- all(banimg, alt(banflash)).
performance :- one_of(ms, sec, min).
adserver :- all(reports, banners, alt(popups), alt(keyword)), requires(keyword, text).
webserver :- all(content, alt(logging), alt(protocol)).
addservices :- all(alt(sitestats), alt(sitesearch), alt(adserver)).
ri :- more_of(datastorage, datatransfer, userauth), requires(datatransfer, https).
persistence :- one_of(xml, database).
reports.
xml.
popups.
datastorage.
keyword.
jsp.
php.
images.
nttp.
dynamic.
basic.
userauth.
file.
datatransfer.
banflash.
database.
ms.
sec.
banimg.
ftp.
asp.
min.
html.
static.
cgi.
https.
advanced.
db.
```