KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Milda Balandytė

Valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimo metodika

Magistro darbas

Darbo vadovė
doc. L. Nemuraitė

Kaunas, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas
_____ doc. dr. R. Butleris
(parašas)
2004 05

Valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimo metodika

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
_____ dr. J. Mikelionienė
(parašas)
2004 05

Vadovas
_____ doc. dr. L. Nemuraitė
(parašas)
2004 05

Recenzentas
_____ doc. dr. V. Pilkauskas
(parašas)
2004 05

Atliko
_____ IFM-7/2 gr. stud. M. Balandytė
(parašas)
2004 05

Kaunas, 2004

TURINYS

1 Santrauka

This work presents development of Human Resource Management System based on analysis of modern development trends as well as fundamentals of key functions and data structures of human resource management systems in organizations. Introduced general-purpose model meets the requirements of human resource management systems and fits for the organizations of any size and structure. The system, mentioned above, was designed using ICONIX, Martin-Odell, RUP, XP, UMM methodologies, tested using black and white box techniques and implemented by means of Lotus Notes/Domino SDK[1]. Created software was installed for the trial period at the joint-stock company 'PTI Technologijos'.

The conceptual part of the theses represents component interface specification technique capturing control flow rules. It describes a clear process for moving from business requirements to system specification and identifying system behavior rules conditioning interfaces of the system. Proposed model facilitates dealing with the change and substitutability of business rules, what can be achieved only if the system is properly specified. Interface specification technique was used in practice to design the interface between human resource management system and accountancy system.

[1] SDK – Software Development Kit

2 Įvadas

Atsižvelgiant į nuolat augantį poreikį valdyti žmogiškuosius išteklius organizacijose, todėl buvo suformuotas inžinerinis magistro studijų tikslas – sukurti organizacijos žmoniškųjų išteklių valdymo sistemą, tinkančią mažoms ir didelėms įmonėms bei valstybinėms institucijoms, apimančią visas funkcijas, susijusias su žmoniškųjų išteklių valdymu, ir pasirinkus Lietuvos įmonę pagal jos poreikius realizuoti būtiniausią sistemos dalį.

Kuriant organizacijos žmoniškųjų išteklių valdymo sistemą buvo siekiama:

- sukurti lengvai naudojamą sistemą, pagrįstą internetine sąsaja;
- realizuoti platesnį nei šiuo metu egzistuojančių sistemų funkcijų spektrą;
- užtikrinti su personalu susijusių dokumentų tvarkymą;
- internetinės savitarnos priemonėmis padidinti kiekvieno personalo nario informuotumą, pateikiant duomenis apie jį patį ir organizacijos informacinius išteklius pagal darbuotojo teises;
- siekti ekonominio efektyvumo, automatizuojant organizacijos personalo duomenų valdymo, įdarbinimo, mokymo funkcijas – tokiu būdu sutaupant nemažai laiko ir pinigų sąnaudų;
- užtikrinti tinkamų darbuotojų priskyrimą pareigybėms ir atskiroms užduotims, remiantis kompetencijų modeliais;
- užtikrinti darbuotojų kompetencijos didinimą, mokymą, darbo išteklių planavimą;
- stebėti ir vertinti pareigybes, darbuotojus bei jų darbo sąlygas;
- užtikrinti duomenų perdavimo buhalterinės apskaitos sistemai sąsają;
- sistemą pritaikyti Lietuvos sąlygoms, kad būtų galima pritaikyti daugelyje įmonių.

Kuriant sistemą buvo siekiama įgyvendinti programinių priemonių lankstumo ir pakartotinio panaudojimo principus.

Įdiegus organizacijos žmogiškųjų išteklių valdymo sistemos dalį, atsirado nauji funkciniai poreikiai – sujungti sistemą su buhalterinės apskaitos sistema. Organizacijos žmogiškųjų išteklių valdymo sistemos ir buhalterinės apskaitos sistemos sąsajos modeliavimo metu išryškėjo valdymo taisyklės, veikiančios komponentų tarpusavio priklausomybes. Panašūs būdingi sąryšiai pastebėti ir kitais atvejais. Norint užtikrinti kuriamos programinės įrangos lankstumą, ieškota sprendimo būdų, kaip galima būtų adaptuoti nuolat besikeičiančias veiklos taisykles, minimaliai koreguojant programinių komponentų kodą. Pastebėta, kad dauguma sąsajas veikiančių valdymo taisyklių yra susiję su sąsajas sudarančių operacijų vykdymo sąlygomis. Todėl nuspręsta šias valdymo taisykles išskirti bei jų pagrindu sudaryti atskirus

valdymo objektus, atsakingus už sąsajų valdymo taisyklių realizavimą - tokiu būdu užtikrinant lankstų sąsajos valdymo mechanizmų pakeičiamumą. Tačiau bet kokiai programinio vieneto realizacijai būtina tiksli specifikacija. Kadangi sistema buvo projektuojama taikant objektinio ir komponentinio modeliavimo principus, iškilo klausimas, kokiu metodu vadovautis sudarant komponentų sąsajos, ribojamos valdymo taisyklėmis, specifikaciją.

Suformuluotas mokslinis darbo tikslas – išanalizuoti komponentinių sąsajų specifikavimo metodus bei suformuluoti specifikavimo metodiką, kurioje būtų įvertinamos sąsajas ribojančios valdymo taisyklės.

Komponentų sąsajų specifikavimo metodų analizės metu buvo išnagrinėti sistemų elgsenos aspektai, įvykių ir sekų modeliavimo galimybės, taikant Martin-Odell metodą, bei atliktas Catalysis ir Dalniels komponentų sąsajų specifikavimo metodų tyrimas. Analizės metu buvo pastebėta, jog sąsajos operacijų prieš-sąlygose bendrai aprašomi visi abribojai, tačiau valdymo taisyklės neatskiriamos nuo sąlygų, atspindinčių sąsajos informacinių objektų būsenas.

Darbo metu sudaryta valdymo taisyklėmis ribojamų komponentų sąsajų specifikacijos metamodelis bei specifikavimo metodika, kurios pagrindu pasirinktas Daniels metodas. Sudaryta metodika galėtų papildyti komponentų sąsajų specifikavimo metodus tais atvejais, kai specifikuojamos sąsajos operacijų vykdymas priklauso nuo tiriamoje aplinkoje galiojančių valdymo taisyklių.

Valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimo metodika buvo pritaikyta sudarant žmogiškųjų išteklių valdymo sistemos ir buhalterinės apskaitos sistemos sąsajos specifikaciją – gauta specifikacija pateikta eksperimentinės dalies aprašyme.

Magistro studijų metu buvo suprojektuota ir sukurta žmogiškųjų išteklių valdymo sistema, tinkanti įvairių tipų įmonėms bei apimanti naujoviškas žmogiškųjų išteklių valdymo funkcijas. Išnagrinėjus Lietuvoje ir pasaulyje egzistuojančias sistemas bei atlikus inžinerinių sprendimų analizę, sistema suprojektuota taikant iteracinio ir evoliucinio kūrimo, ICONIX, Martin-Odell, RUP, XP, UMM metodų principus. Sistema realizuota, naudojant Lotus Notes/Domino programinę įrangą, bei įdiegta užsakovų įmonėje.

Studijų metu dalyvauta sekančiose konferencijose:

- sukurtos sistemos pagrindu suformuluotos šiuolaikinių žmogiškųjų išteklių valdymo sistemų kūrimo tendencijos, kurios, kartu su bendraautoriais, pristatytos konferencijoje „Informacinės technologijos 2003“. Publikacija [12] pateikta prieduose.
- sudarytas integruotos žmogiškųjų išteklių valdymo sistemos modelis kartu su bendraautoriais pristatytas konferencijoje „Integruotos projektavimo sistemos“.

3 Žmogiškųjų išteklių valdymo sistemų analizė

Magistrinio projekto vykdymo metu pirmiausia buvo siekiama apibrėžti organizacijos žmogiškųjų išteklių valdymo sistemos universalų modelį, tinkantį mažoms ir didelėms įmonėms bei valstybinėms institucijoms, apimantį visas funkcijas, susijusias su žmogiškųjų išteklių valdymu. Tuo tikslu buvo atliktas egzistuojančių žmogiškųjų išteklių valdymo sistemų tyrimas [13]. Taip pat buvo nagrinėjami projektavimo ir darbų sekų valdymo metodai bei programų kūrimo priemonės, labiausiai atitinkančios kuriamo produkto specifiką.

3.1 Esamų programinių produktų analizė

Sudarant organizacijos žmogiškųjų išteklių valdymo sistemos universalų modelį, buvo nagrinėjami tiek Lietuvoje, tiek užsienyje sukurti programiniai paketai, susiję su žmogiškųjų išteklių valdymu, tačiau besiskiriantys sistemą sudarančių modulių ir funkcijų įvairove. Atliekant žmogiškųjų išteklių valdymo sistemų palyginimą, buvo taikomas vieningas šablonas, charakterizuojantis programinę įrangą sekančiais parametrais: *pavadinimas, sistemos tipas, sistemos moduliai ir funkcijos, realizuojančios organizacijos žmogiškųjų išteklių valdymą.*

Analizės metu išnagrinėtų programinių produktų sąrašas pateiktas 1 lentelėje.

1 lentelė. Žmogiškųjų išteklių valdymo sistemos

Pasaulyje egzistuojančios sistemos	Lietuvoje egzistuojančios sistemos
Navision Financials	Scala
Hansa Financials	Balansas 2000
Best Abra HR	Konto
Best Imperative HRMS	Apskaita
Ulti Pro HR	Akf apskaita
Dynamics	Personalas
Sage Enterprise Suite	Cs asmuo
eEnterprise	Ci avs 2001
Navision Axapta	Debetas
Oracle Small Business Suite	Alga 2000
Mysap HR	Du000
Peoplesoft HR	Ontime
Oracle HR	Personalas
	Alga
	Pragma
	Skaita 2000
	Atlygis
	Stekas-apskaita

Išnagrinėjus užsienyje bei Lietuvoje egzistuojančias sistemas buvo gautos sekančios išvados:

- didelės sistemos, tokios kaip Oracle HR, PeopleSoft HR, mySAP, pasižymi išvystytomis žmogiškųjų išteklių valdymo funkcijomis, tačiau yra labai brangios;
- dauguma užsieninių sistemų nepritaikytos Lietuvos specifikai;

- daugelis nagrinėtų sistemų netenkina funkcinių reikalavimų, keliamų personalo valdymui – išskirti galima tik “DU000” ir “ALGA2000” sistemas.
- dauguma lietuviškų sistemų pritaikytos tik vienai darbo vietai – nepalaiko grupinio darbo sąsajos.

Analizės metu išryškėjo papildomų funkcijų spektras:

- platus darbuotojų asmeninių duomenų valdymas;
- organizacinis valdymas;
- naujų darbuotojų paieškos ir atrankos organizavimas;
- personalo ugdymas ir mokymas;
- darbuotojų vertinimas;
- darbo laiko apskaita, apimanti atostogų valdymo mechanizmą;
- sveikatos priežiūra;
- statistinių duomenų ir įvairaus pobūdžio ataskaitų formavimas.

3.2 Projektavimo metodų analizė

Inžinerinių sprendimų analizės metu buvo suformuluotas tikslas - pasirinkti projektavimo būdą ar metodą, kuris labiausiai atitiktų kuriamo produkto specifiką.

Analizės metu buvo nagrinėjamos sekantys sistemų modeliavimo metodai, nusakantys skirtingus projektavimo aspektus:

- *objektinis projektavimas*: Martin-Odell, OMT, Moses;
- *komponentinis projektavimas*: RUP, Catalysis;
- *greitas kūrimas*: XP, ICONIX.

Atlikus sistemų modeliavimo priemonių analizę, nuspręsta projektuoti sistemą, taikant keletą metodologijų:

- iteracinio ir evoliucinio kūrimo principus;
- minimalų diagramų kiekį pagal ICONIX metodo patirtį;
- įvykių sekų modeliavimas, pagrįstas Martin-Odell metodu;
- sistemos dokumentavimas pagal RUP ir Kruchten metodų reikalavimus, analizuojančius sistemą *loginiu, procesų, fiziniu, kūrimo, elgsenos* pjūviais;
- užduočių testavimo procedūrų užtikrinimas, vadovaujantis XP metodu.
- grafinis sistemos atvaizdavimas *Rational Rose 2002* projektavimo paketu, pagrįstu UML notacija.

3.3 Darbų sekų valdymo metodų analizė

Nagrinėjant su personalo valdymu susijusius kompiuterizuojamus procesus, tokius kaip atostogų valdymas, personalo ugdymas ir mokymas, darbuotojų vertinimas, naujų darbuotojų paieška ir atranka, buvo nustatyta, kad kuriama sistema turi užtikrinti:

- efektyvų darbo paskirstymą turimiems resursams – darbų skaidymas į užduotis ir jų priskyrimas reikiamą kvalifikaciją turintiems veiklos dalyviams;
- dokumentų srautų apdorojimą;
- dokumentų tvirtinimo ciklus;
- užduočių, darbų ir procesų būsenų identifikavimą;
- greitą ir patogų duomenų apdorojimą;
- heterogenines kūrimo priemones;
- duomenų konfidencialumą.

Šiuos klausimus nagrinėja darbų sekų valdymo sritis, apimanti darbų sekų sudarymo standartus, modeliavimo ir analizės metodus, egzistuojančius programinius sprendimus. Todėl buvo suformuotas tikslas - panaudojant geriausią praktiką, aprašyti organizacijoje vykstančius žmogiškųjų išteklių valdymo procesus tinkamais darbų sekų modeliais.

Darbų sekų valdymo metodų analizės metu buvo išnagrinėti WfMC[2] ir OMG[3] darbų sekų standartai, apibrėžtos pagrindinės darbų sekas charakterizuojančios sąvokos, pateikta darbų sekų sistemų klasifikacija bei jų sudarymo metodų (*Action Workflow, Enterprise Modelling, UMM*[4]) tyrimas.

Išanalizavus darbų sekų valdymo aspektus, nustatyta, jog:

- kuriama žmogiškųjų išteklių valdymo sistema sunkiai nusakoma vienu kuriuo nors DSVS kriterijumi, nes pasižymi įvairiomis savybėmis: *autonominė, administracinė, kolektyvinio darbo, orientuota į žmogų, transakcinė, orientuotos į veiklas, jų tarpusavio ryšius ir būsenas, komunikacinė;*
- tikslinga sudaryti nagrinėjamos dalykinės srities darbų sekų šablonus – tipinius modelius, tiksliai aprašančius organizacijoje vykstančius su personalo apskaita susijusius veiklos procesus, remiantis veiklos transakcijas aprašančia UMM metodologija – bus gaunamas greitesnis sistemos veikimas ir paprastesnis naudojimas.

[2] WfMC – Workflow Management Coalition
[3] OMG - Object Management Group
[4] UMM - UN/CEFACT's Modeling Methodology

3.4 Informacinių technologijų analizė

Informacinių technologijų analizės tikslas – išanalizuoti programines priemones, leidžiančias įgyvendinti nagrinėjamai sistemai keliamus reikalavimus veikimo aplinkai.

Analizės metu buvo nagrinėjami grupinio darbo aplinką palaikantys programiniai paketai:

- *Teamware Office;*
- *Novell Groupwise;*
- *Microsoft Exchange;*
- *Lotus Notes/Domino.*

Išanalizavus programines priemones, leidžiančias įgyvendinti kuriamai sistemai keliamus reikalavimus, sistemos realizacijai pasirinktas IBM firmos Lotus Notes/Domino paketas, paremtas kliento/serverio architektūra. Galima išskirti sekančius Lotus Notes/Domino paketo privalumus:

- *grupinio darbo sąsaja* - sistemos vartotojai gali komunikuoti organizacijos viduje ir už jos ribų;
- *užtikrinami keli slaptumo lygiai* – kiekvienam sistemos dokumentui nustatoma tam tikra prieiga (galimybė dokumentą skaityti ir redaguoti);
- *internetinė sąsaja* – suteikiama galimybė vartotojui naudotis sistema per Interneto naršyklę;
- *integravimas su reliacinėmis DB*;
- *pašto standartų palaikymas*;
- *veiklos koordinavimas* – resursų, darbų sekų valdymas.

4 Komponentų sąsajų specifikavimo metodų analizė

Analizės tikslas – išanalizuoti komponentinių sąsajų specifikavimo metodus bei nustatyti, kuris metodas labiausiai tinka, specifikuojant valdymo taisyklėmis ribojamų komponentų sąsajas.

Pagal UML[5], *sąsaja* - tai elemento statinę bei dinaminę elgseną charakterizuojantis operacijų rinkinys [5], nusakantis sistemos komponentų elgseną. Todėl tikslinga išsiaiškinti sistemų elgsenos aspektus, valdymo elementų identifikavimo principus bei apžvelgti egzistuojančius sąsajų specifikavimo metodus.

4.1 Sistemų elgsenos aspektai

Esminis objektinės ideologijos principas – sistemos elgsenos ir su ja susijusių duomenų nagrinėjimas sistemos objektų kontekste [3].

Sistemos elgseną nusako aibė *žingsnių*, UML notacijoje vadinamų *būsenomis*, *veiksmais* ir *subveiklomis*. Identifikuojant sistemos elgsenos modelio žingsnius bei jų pobūdį, C. Bock [3], išskiria trijų tipų sistemų elgsenos aspektus (2 lentelė):

- *valdymo aspektai*, kuriuos galima atspindėti veiklos diagramose,
- *būsenų kaita*, atvaizduojama būsenų perėjimų diagramose,
- *objektų srautai*, atitinkantys operacijų įėjimus/išėjimus.

Be to, dar išskiriamos sąveikos, kurios vaizduojamos sekų/bendradarbiavimo diagramomis.

2 lentelė. Sistemos elgsenos aspektai

Sistemų elgsenos tipas	Aprašymas
Valdymo srautų modeliai	Sistemos elgseną nusakantys žingsniai iškviečiami įvykdžius tam tikrą aibę žingsnių, tačiau neatsižvelgiant į žingsnių parametrus. Žingsnių įėjimo parametrai apibrėžiami pereinant nuo vieno žingsnio prie kito, tačiau jie nėra būtini, iškviečiant konkretų žingsnį. Šis metodas dažniausiai taikomas modeliuojant įterptinių ir verslo sistemų veiklą, kai norima akcentuoti sistemos elgsenos žingsnių sekas, veikiamas išorinių įvykių.
Duomenų srautų modeliai	Sistemos elgseną nusakančių žingsnių iškvietimas priklauso nuo suformuotų įėjimo parametrų. Šiuo atveju žingsnių įėjimo parametrai – kitų žingsnių suformuoti rezultatai. Todėl duomenų srautų modeliuose akcentuojama ne žingsnių vykdymo seka, o žingsnių įėjimo parametrų suformavimo momentai, nulemiantys žingsnių iškvietimą.
Būsenų mašinų modeliai	Sistemos elgseną nusakantys žingsniai iškviečiami atsižvelgiant į sistemos aplinkoje vykstančius įvykius. Žingsnių vykdymo seką nulemia išoriniai įvykiai. Įėjimo parametrai suformuojami žingsnių vykdymo eigoje. Dažniausiai šie modeliai naudojami modeliuojant įterptines bei realaus laiko sistemas, kurių veikimas modeliuojamas kaip atsakymas į galimus išorinius poveikius.

[5] Unified Modelling Language – vieninga modeliavimo kalba, pripažinta „de facto“ standartu, kuriant taikomųjų programų sistemas

Pateikti elgsenos modeliai charakterizuoja vykdomus žingsnius keliais aspektais (3 lentelė):

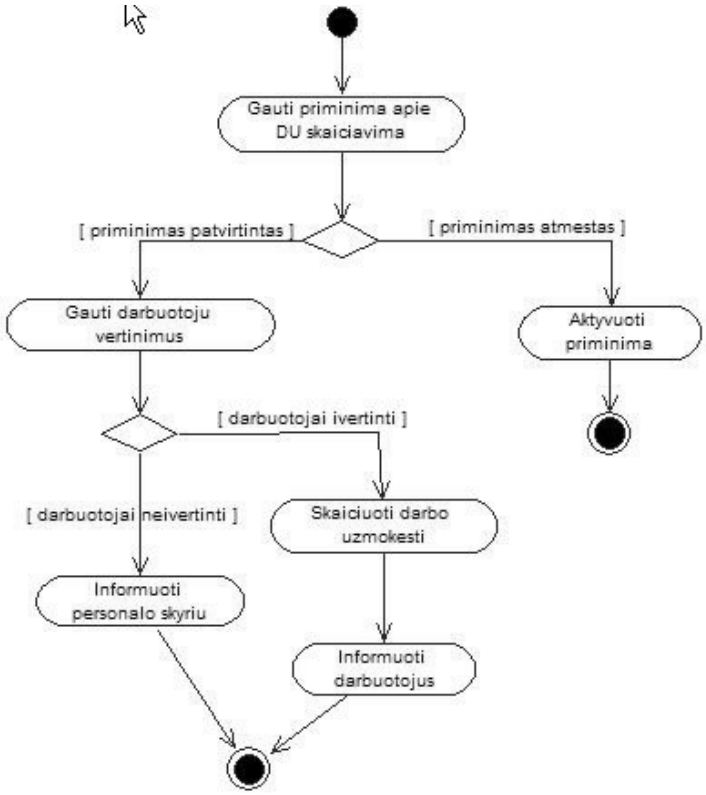
- *išorinėmis* arba *vidinėmis* sąlygomis, nulemiančiomis žingsnio iškvietimą;
- *įėjimo parametrų nustatymo momentais*.

3 lentelė. Elgsenos metodų charakteristikos

	Valdymo modeliai	*Duomenų srautų modeliai*	*Būsenų modeliai*
Žingsnio iškvietimo sąlygos	vidinės	vidinės	išorinės
Įėjimo parametrų apibrėžimas	iškviečiant žingsnį	prieš iškviečiant žingsnį	iškviečiant žingsnį

4.1.1 Objektinis elgsenos modeliavimas

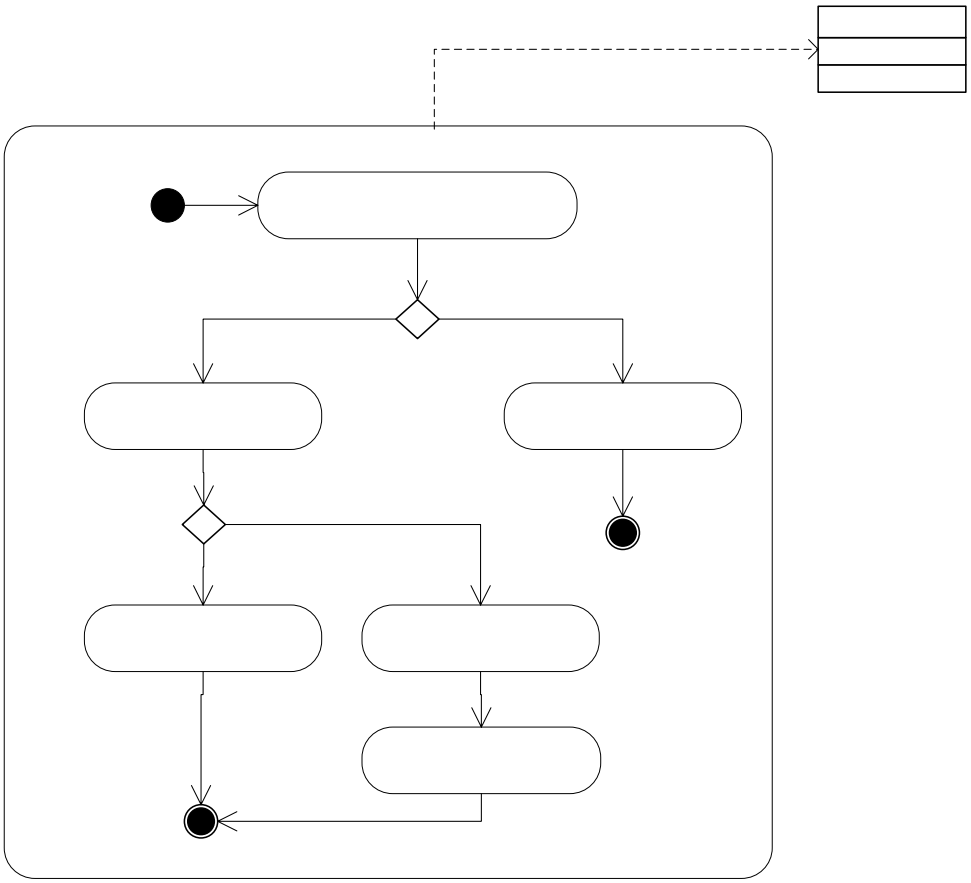
Apibūdinant sistemų veikimą, objektinėje teorijoje išskiriamos *operacijos* ir jas vykdančių *metodų* sąvokos. Aprašant elgsenos modelius objektiniame kontekste, sistemų elgsenos žingsnių funkcijos siejamos su operacijomis, o patys elgsenos modeliai – su metodais. Modeliuojant sistemos elgseną objektinėje aplinkoje funkcijų iškvietimas pakeičiamas pranešimų siuntimu – taip atskiriant funkcijų iškvietimo ir jų vykdymo sąvokas ir pereinant prie objektinio elgsenos modeliavimo teorijos, nekeičiant funkcijų kodo. Tokiu būdu taikant polimorfizmo principą elgsenos modeliams, pastarieji (1 pav.) lengvai gali būti atvaizduoti objektinėje aplinkoje (2 pav.).



1 pav. Neobjektinis valdymo modelis

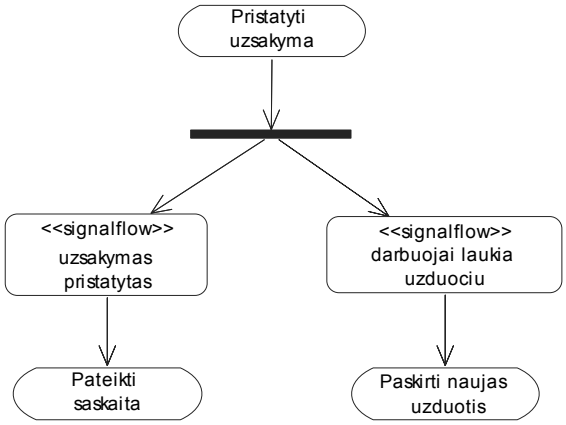
Valdymo srautų modelis, pateiktas 1 pav., iliustruoja bendrą sistemos veikimą, tačiau šiame modelyje nematyti sąsajos tarp sistemos objektų, atsakančių už sistemos elgsenos žingsnius. Pagal objektinės teorijos principą, sistemos elgsena modeliuojama siejant sistemos objektus su apdorojamais duomenimis. Šiuo atveju, sistemos elgsenos žingsniai priskiriami konkretiems sistemos objektams, gaunantiems atitinkamą pranešimą. Naudojant OOIE ([2]) notaciją, pranešimus apdorojantys objektai žymimi laužtiniuose skliaustuose bei identifikuojamas ryšys, jungiantis valdymo modelį su jį realizuojančia sistemos klase. Tokiu būdu priskiriant sistemos elgseną ir ją sudarančius žingsnius valdantiems sistemos objektams, gaunamas objektinis valdymo modelis, pateiktas 2 pav.

Pabrėžiant objektinio elgsenos modeliavimo principus, reikia paminėti, kad valdymo modelį (2 pav.) sudarančios operacijos vykdomos nepriklausomai viena nuo kitos – tokiu būdu užtikrinamas operacijų pakartotinio panaudojimo principas įvairiuose metoduose (elgsenos modeliuose), nulemiančiais operacijų iškvietimo seką, koordinuojant sistemos būsenas.



2 pav. Objektinis valdymo modelis

Objektiniai elgsenų modeliai taip pat gaunami siejant elgsenos žingsnių nulemtus sistemos objektų pasikeitimus [3]. Grafinis elgsenos žingsnių priežasčių ir poveikių modeliavimo fragmentas pateiktas 3 pav.



3 pav. Elgsenos žingsnių priežasčių ir poveikių modeliavimas

Verta pastebėti, kad šis metodas, sudarant duomenų ir būsenų srautų modelius retai taikomas, nes šiuose modeliuose žingsnių poveikiai tik netiesiogiai daro įtaką objektų struktūrai, veikiančiai sekančius sistemos elgsenos žingsnius.

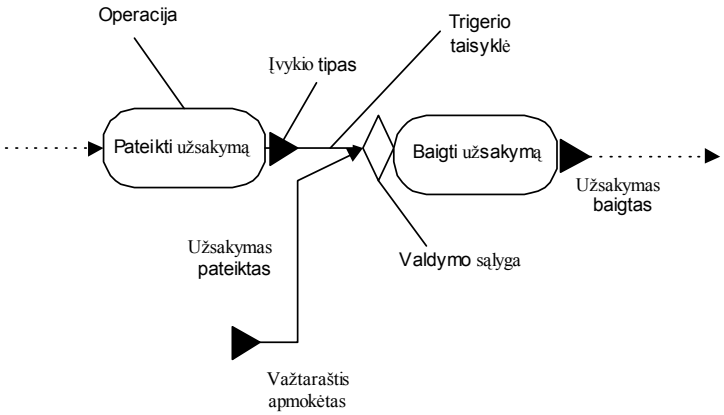
Taip pat egzistuoja keletas objektinio elgsenos modeliavimo metodų, būdingų tik tam tikroms modelių grupėms. Objektiniuose duomenų srautų modeliuose reikalaujama, kad tarp elgsenos žingsnių perduodami duomenys būtų tam tikros klasės objektai, esantys apibrėžtoje būsenoje. Tokiu būdu suformuojamas perduodamų objektų srautas. Objektiniuose būsenų mašinų modeliuose reikalaujama susieti elgsenos modelį su konkretaus tipo klase bei vykdomais žingsniais, atvaizduojant atitinkamą objekto būseną ar objektą sudarančių duomenų konfigūraciją.

4.1.2 Įvykių ir sekų modeliavimas Martin-Odell metodu

Veiksmų modeliavimo požiūriu vienas iš labiausiai išvystytų objektinio projektavimo metodų yra *Martin-Odel* (M-O) metodas, kurio pagrindu sudarytos UML veiklos (*activity*) diagramos. M-O modelis pagrįstas įvykių diagramomis. Jos išreiškiamos naudojant 4 pagrindinius procesus identifikuojančius elementus:

- operacijas,
- įvykius,
- trigerius,
- valdymo sąlygas.

Operacijos yra procesai, vykdantys būsenų pasikeitimus. Įvykių tipai apibrėžia būsenų pasikeitimus, iššaukiančias kitas operacijas per trigerių taisykles. Valdymo sąlygos užtikrina tam tikrų būsenų egzistavimą, prieš iškviečiant operaciją. Šios sąvokos apibendrinamos, naudojant grafinę notaciją, vadinamą įvykių diagramomis (pav. 3.)



4 pav. Įvykių grafinis atvaizdavimas

Kiekvieną kartą iškvietus operaciją "pateikti užsakymą", tikimasi rezultato — įvykio "užsakymas pateiktas". Įvykus šiam įvykiui, trigeris iškviečia operaciją "baigti užsakymą". Tačiau prieš operacijos iškvietimą yra tikrinama valdymo sąlyga, garantuojanti užsakymo pateikimo ir sąskaitos apmokėjimo faktą. Jei operacijai "baigti užsakymą" nėra aprašyta valdymo sąlyga, operacija įvykdoma tuoj pat po bet kurio įvykio atlikimo.

Įvykio tipas "apmokėtas važtaraštis" neturi jokios susijusios operacijos, todėl būdas, kuriuo apmokamas važtaraštis, yra už specifikacijos ribų. Įvykio tipas pateikiamas, siekiant parodyti proceso vykdymo stimulą.

Iš esmės, yra dvi būsenų pokyčių rūšys:

- *pridėti (add)* - įveda naują objektą arba ryšį;
- *panaikinti (remove)* - pašalina objektą arba ryšį.

Objektai gali būti kuriami, baigiami, grupuojami, išgrupuojami, pergrupuojami, jungiami, išskiriami, suvienijami, atskiriami. Kiekvienas įvykis gali būti klasifikuojamas kaip vienas iš šių būsenų pokyčių rūšių – kiekvienas įveda pridedamus ir pašalinamus objektus.

Įvykiai yra procesų, iššaukiančių būsenų pokyčius, pasekmė. Įvykiai skirstomis į 3 grupes:

- *vidiniai* – vidinės operacijos rezultatas;
- *išoriniai* - išorinės operacijos rezultatas;
- *laiko* - laiko operacijų rezultatas;

Vidiniai bei išoriniai įvykiai identifikuojami tiriant dalykinę sritį. Tuo metu, kai įvykiai atlieka būsenų pokyčius, operacijos apdoroja elementus, kurie atlieka tuos pokyčius. Kiekvienai operacijai reikalingas objektas. Objektinio modeliavimo kontekste operacijos suprantamos kaip transakcijos. *Transakcijos* - procesai arba procesų sekos, veikiančios kaip elementai, keičiantys objektų būsenas. Operacijos specifikacija vadinama *metodu***.** Objektinio programavimo kalbose metodai specifikuojami kodo eilutėmis. Įvykių pasirodymas iššaukia reakcijas arba

atsakomuosius procesus. Ryšys tarp įvykio ir atsakomojo proceso vadinamas *trigeriu*. Trigeriai nusakomi taisyklėmis, nusakančiomis 3 elementus:

- *įvykio tipą* (priežastį);
- *operaciją* (veiksmą);
- *atvaizdavimą* – vienos operacijos rezultatai perkeliami į tuos objektus, kurie reikalauja jų kaip įvedimo kintamųjų.

Taip pat egzistuoja *valdymo sąlygos* - tai apribojimai, kuriuos reikia patikrinti prieš vykdant operaciją. Operacijos prieš sąlygos gali būti laikomos operacijos dalimi, nes jos apribojimai turi būti patenkinti, iššaukiant operaciją.

Be įvykių diagramų, M-O apibrėžė objektų srautų diagramas, kuriose vaizduojami operacijų įėjimo/išėjimo objektai. Ši idėja buvo pritaikyta UML veiklos diagramose, modeliuojančiose objektų srautus.

4.1.3 Dinaminiai objektų sąryšiai

Šaltinio [8] teigimu, dauguma objektinio modeliavimo metodų atskiria sistemos struktūrinių ir dinaminių savybių nagrinėjimą. Sistemos komponentai atvaizduojami objektų modeliuose, o sistemos elgsena pateikiama veiklos ir duomenų srautų modeliuose, dalinai jungiančiuose objektų charakteristikas ir objektų tarpusavio sąryšius. Todėl autoriai [8] pateikia metodą, sudarantį galimybes specifikuoti sistemas kaip aibę abstrakčių objektų, jungiančių statinius ir dinaminius sistemos elgsenos aspektus. Objektų specifikacijos apima objektų struktūros ir elgsenos aprašymą. Objektų egzempliorių vidinės būsenos stebimos ir keičiamos naudojant objektų sąsajas. Struktūrines objektų savybes nusako objektų atributai, o objektų elgsena valdoma įvykiais, kurie apibrėžiami kaip objektų būsenas keičiančios operacijos ar metodai. Sistemos objektų struktūra nusakoma agregavimo ryšiais bei specializavimo hierarchijomis. Sistemos dinamika aprašoma objektų egzempliorių sąveikomis. Objektų komunikavimas modeliuojamas, iškviečiant arba identifikuojant susijusių objektų įvykius. Specifikuojant modulines sistemas, apibrėžiama sistemą sudarančių komponentų architektūra, identifikuojant tarpusavyje komunikuojančių skirtingų posistemių objektus. Semantine prasme objektai traktuojami kaip matomi procesai, komunikuojantys sinchroninių sąveikų pagalba. Procesai interpretuojami kaip įvykių sekų aibė, charakterizuojanti objektų gyvavimo ciklą. Procesais aprašoma objekto egzempliorių dinamika.

4.2 Komponentų sąsajų specifikavimo metodai

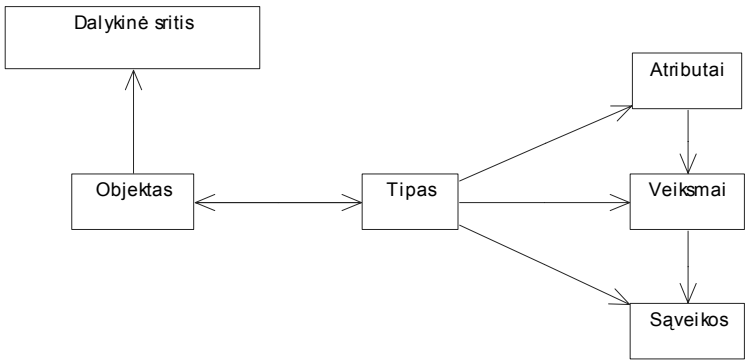
Šiame skyriuje pateikiama Catalysis ir Daniels modeliavimo metodų, apimančių komponentų sąsajų specifikavimo metodikas, apžvalga.

4.2.1 Catalysis metodas

Catalysis – tai sudėtingas ir detalus komponentinio kūrimo metodas, pagrįstas UML standartu. Catalysis metodas apibrėžia sistemingą ir lankstų sistemų kūrimo procesą, pasižymintį sekančiomis savybėmis:

- sudaro galimybes vartotojams ir programinės įrangos kūrėjams dalintis aiškiu ir tiksliu žodynu;
- nagrinėja sistemos veiklą bei sistemą sudarančius komponentus viename kontekste;
- tikslus komponentų sąsajų specifikavimas;
- komponentų architektūros sudarymas;
- sistemų kūrimui naudojamos objektinės ir komponentinės technologijos;
- palikuoninių komponentų integravimas, kuriant naujas sistemas;
- veiklos ir aparatūrinės įrangos patikimos komponentų architektūros sudarymą;
- šabloninis dalykinės srities modelių, komponentų, sąsajų, kodo ir procesų pakartotinis panaudojimas;
- veiklos procesų atsekamumo patobulinimas.

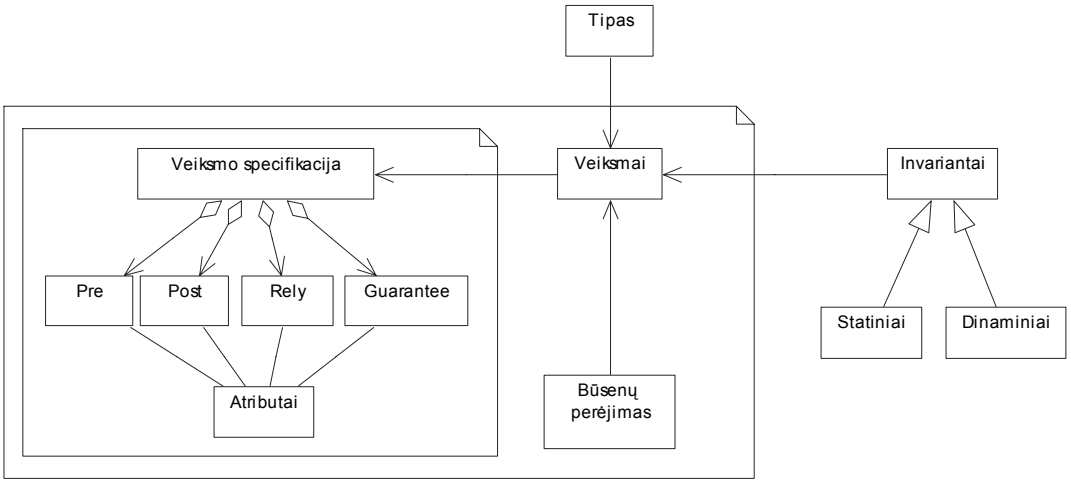
Komponentų specifikacijos apibrėžiamos tipų modeliais, kuriuos sudaro aibė atributų, veiksmų ir sąveikų, nusakančių sistemos elgsenos specifikaciją (5 pav.).



5 pav. Konceptualus modeliuojamų objektų modelis

Projektuojant komponentų elgseną, identifikuojami visi operacijų dalyviai, veikiantys operacijų įėjimo parametrus bei nulemiantys operacijų rezultatus. Todėl įvedama operacijos abstrakcijos sąvoka – *veiksmas* (6 pav.). Kiekvienas veiksmas siejamas su keliais dalyviais –

vienas iš jų yra veiksmo iniciatorius. *Prieš*[6] ir *po*[7] sąlygų specifikacijos nusako veiksme sąveikaujančių dalyvių būsenų pasikeitimus.



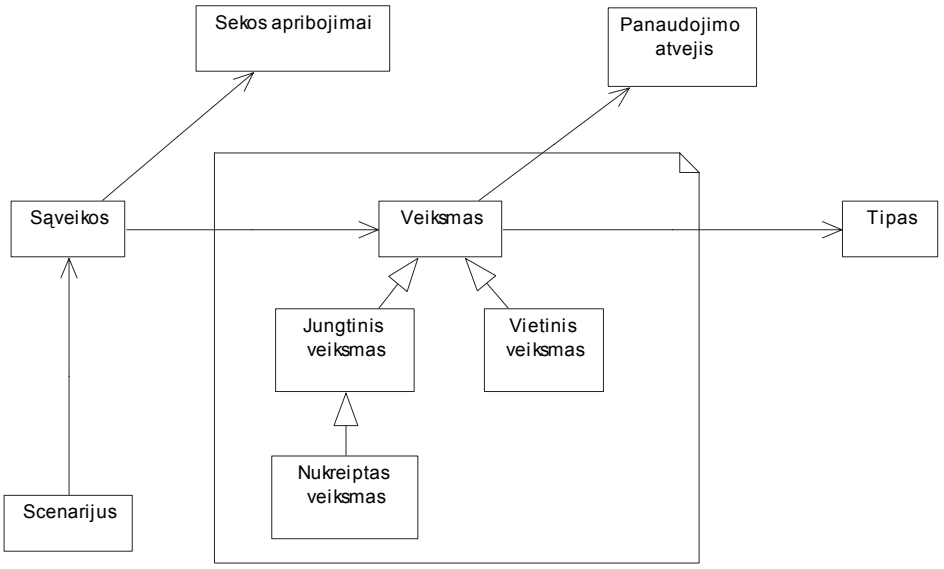
6 pav. Būsenų pasikeitimų modeliavimas

Objektų bendradarbiavimo sąvoka nusako objektų tipų, vaidinančių tam tikras roles, veiksmų aibę. Veiksmai aprašomi tipų modeliais, kuriais dalinasi visi konkrečios sąveikos dalyviai. Veiksmai skirstomi į sekančias grupes:

- *jungtiniai* – veiksmų atsakomybė nesiejama su sąveikoje dalyvaujančiomis objektų tipų rolėmis, veiksmą nusako keletas dalyvių;
- *lokalizuoti* – veiksmų atsakomybė priskiriama konkrečioms sąveikoje dalyvaujančiomis objektų tipų rolėms, veiksme dalyvauja tik „gavėjas“;
- *išoriniai* arba *vidiniai* objektų tipų sąveikos atžvilgiu.

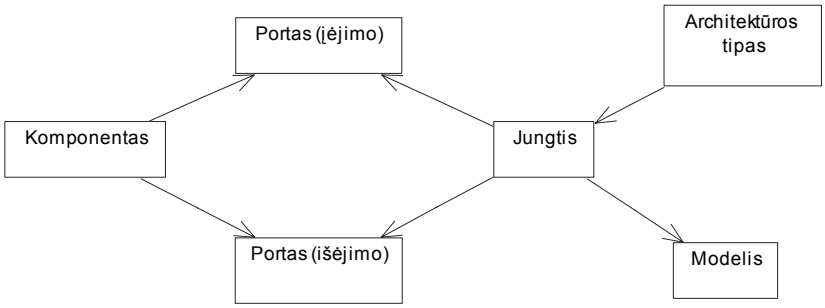
Sekančiame sistemos elgsenos specifikavimo žingsnyje realizuojamas bendradarbiavimo modelių apjungimas. Tokiu būdu, kiekvienas objektų tipų bendradarbiavimo atvejis apibrėžia konkrečios objektų aibės jungtinius veiksmus (7 pav.).

[6] Sąlygos, prie kurių garantuojama operacijų po sąlyga
[7] Operacijos poveikis operacijų parametrų ir informacinio modelio atžvilgiu



7 pav. Sąveikų modeliavimas

Modeliuojant komponentų architektūrą, komponentai jungiami, naudojant komponentų sąveikos elgseną charakterizuojančias jungtis (8 pav.), kurių pobūdis nusakomas įvairiais stereotipais: <<Event>>, <<Property>>, <<Trandfer>>, <<Transaction>>.



8 pav. Komponentų ir jų sąsajų modeliavimas

4.2.2 Daniels metodas

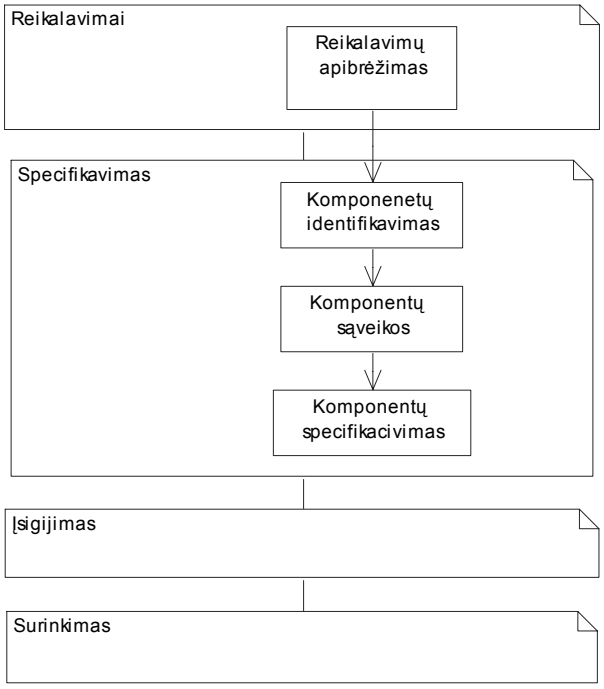
Pastaruoju metu UML tapo sistemų objektinio modeliavimo standartu, schematiškai aprašančiu skirtingus sistemos aspektus. Statiniai sistemos modelių apribojimai užrašomi OCL[8] tekstinėmis išraiškomis, papildančiomis UML diagramas. UML komponentų sąvoka apima tik sistemos veikimo metu žemiausiame lygmenyje egzistuojančius vienetus [10], tačiau neidentifikuoja komponentų konceptualaus ar specifikacijos lygmens ypatybių.

Autoriai, J. Cheesman ir J. Daniels, pateikia UML modeliavimo galimybių taikymo metodą, projektuojant komponentines sistemas. Taikant *Daniels* metodą, sistemos komponentai nagrinėjami konceptualiame lygmenyje bei specifikuojami naudojant standartinę UML semantiką. Tokiu būdu sudaromos komponentų specifikacijos, charakterizuojamos komponentų

[8] OCL – Object Constraint Language

sąsajos, apibrėžiami komponentų objektai bei jų realizacija. Daniels metode taip pat taikomi kontraktais pagrįsto komponentinio modeliavimo principai. Kontraktais aprašomi komponentų siūlomos bei reikalaujamos sąsajos. Charakterizuojant komponentų kontraktus, naudojamos OCL loginės išraiškos bei UML grafinė bendradarbiavimo diagramų notacija. OCL išraiškomis apibrėžiamos komponentų sąsajų operacijų *prieš* ir *po* sąlygos, o bendradarbiavimo diagramomis atvaizduojamos komponentų sąveikos.

Daniels pateiktas sistemų komponentinio projektavimo metodas išsamiai aprašo komponentinių sąsajų projektavimo procesą, pradedant veiklos reikalavimų nustatymu ir baigiant sistemos architektūros sudarymu (9 pav.), bei atskiria komponentų specifikavimo ir jų realizavimo sąvokas.



9 pav. Komponentų specifikavimo procesas

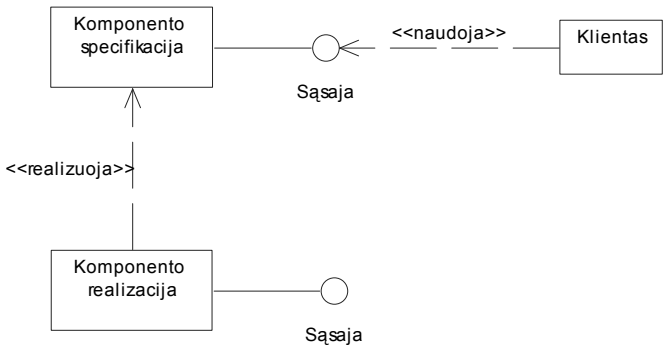
Komponentai – tai programinės įrangos vienetai, teikiantys paslaugas kitiems komponentams, bei reikalaujantys tam tikrų paslaugų iš kitų komponentų. Komponentų siūlomos/reikalaujamos priklausomybės aprašomos kontraktais.

Skiriami dviejų tipų kontraktai, kurių grafinis atvaizdavimas pateiktas 10 pav.

Naudojimo – komponento objekto ir kliento[9], naudojančio tą objektą, sąsajos kontraktas - sistemos vykdymo kontraktas;

Realizavimo – kontraktas tarp komponento specifikacijos ir jo realizuotojo – sistemos projektavimo kontraktas.

[9] Programinė įranga, "iškviečianti" ar naudojanti komponento sąsajos operacijas



10 pav. Kontraktų tipai

Naudojimo kontraktai – tai komponentų sąsajų specifikacijos, kurias sudaro sąsajos operacijų sąrašas, apimantis operacijų aprašymus ir signatūras, bei sąsajų informaciniai modeliai, identifikuojantys klientų ir objektų sąsajose dalyvaujančius veiklos tipus bei sąsajas veikiančius apribojimus. Operacijų specifikacijos apima įėjimo, išėjimo parametrus, jų tarpusavio ryšius, operacijų vykdymo sąlygas bei *prieš* ir *po* sąlygų aprašymą. Operacijų grupavimas į sąsajas motyvuojamas praktinio operacijų panaudojimo tarpusavio priklausomybėmis, aprašančiomis sistemos elgseną.

Realizavimo kontraktai – sistemos komponentų specifikacijos, detalizuojančios komponentų realizavimo ir diegimo ribas. Komponentų specifikacijos apima komponentų funkcionalumo aprašymą bei komponentų sąveikos priklausomybes. Komponentų sąveikos pateikiamos bendradarbiavimo diagramose arba aprašomos deklaratyviais apribojimais, apimančiais komponentų sąveikas charakterizuojančius informacinius modelius. Sąsajų ir komponentų specifikacijų skirtumai pateikiami 4 lentelėje.

4 lentelė. Komponentų ir jų sąsajų specifikacijų sandaros palyginimas

Sąsajos specifikacija	Komponento specifikacija
Operacijų sąrašas	Siūlomų sąsajų sąrašas
Informaciniai modeliai	Sąsajų informacinių modelių tarpusavio priklausomybės
Kontraktas su klientu	Kontraktas su realizuotoju
Operacijų poveikis informacinių modelių kontekste	Komponento realizavimo ir veikimo visuma
Vietinio poveikio aprašas	Operacijų realizavimo aprašas naudojamų sąsajų atžvilgiu

4.3 Analizės išvados

Šiame skyriuje buvo norima išsiaiškinti sistemų elgsenos aspektus, komponentų sąsajas ribojančių valdymo elementų identifikavimo principus bei pasirinkti metodą, labiausiai tinkantį valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimui. Tuo tikslu buvo analizuojami dinaminiai objektų sąryšiai, objektinio sistemų elgsenos modeliavimo būdai, įvykių ir sekų projektavimo galimybės, taikant Martin-Odell metodą. Taip pat atlikta Catalysis ir Dalniels komponentų sąsajų specifikavimo metodų analizė. Todėl galima teigti, jog:

- komponenetų sąsajas ribojančios valdymo taisykles pakankamai tiksliai galima identifikuoti naudojant Martin-Odell metodą, kurio pagrindu sudarytos UML veiklos diagramos;
- komponentų sąsajų specifikavimo metodai bendrai nagrinėja sąsajas veikiančių taisyklių semantiką, tačiau atskirai nedetalizuoja sąsają ribojančių valdymo elementų;
- sąsajas ribojančios valdymo taisyklės aprašomos operacijų specifikacijose – tokiu būdu sumažinamas programinių priemonių lankstumas bei daugkartinio komponentų panaudojimo galimybės – pasikeitus valdymo pobūdžio veiklos taisyklėms, tektų koreguoti visą operacijos specifikaciją, o po to – ir realizaciją;

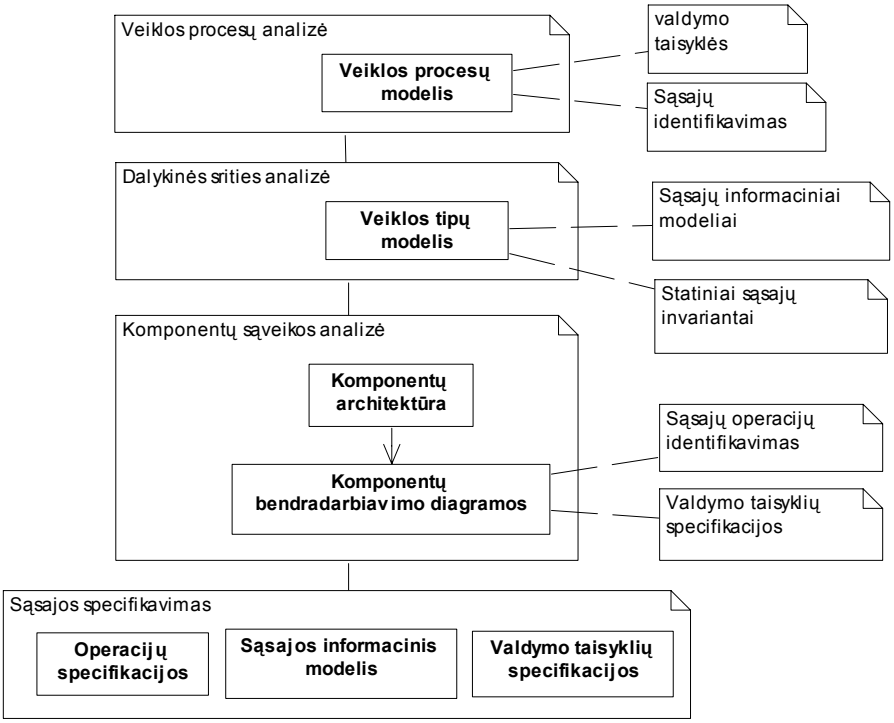
Todėl atskirai aprašius komponentų sąsajas ribojančias valdymo taisykles, būtų galima projektuoti atskirus komponentus, realizuojančius atskirus valdymo mechanizmus.

Tuo tikslu nuspręsta apibrėžti komponentų sąsajų, ribojamų valdymo taisyklėmis, specifikavimo metodiką, kurios pagrindu pasirinktas Daniels metodas, tiksliai apibrėžiantis sąsajų apribojimus, tačiau neišskiriantis valdymo taisyklių.

5 Valdymo taisyklėmis ribojamų sąsajų specifikavimo metodika

Pakankamai tiksliai komponentų sąsajas apibrėžia Daniels metodo ideologija, tačiau neidentifikuoja sąsajų ribojančių valdymo mechanizmų. Praplėtus Daniels metode sąsajų specifikacijas valdymo taisyklių formaliais aprašymais, galima sudaryti valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimo metodiką.

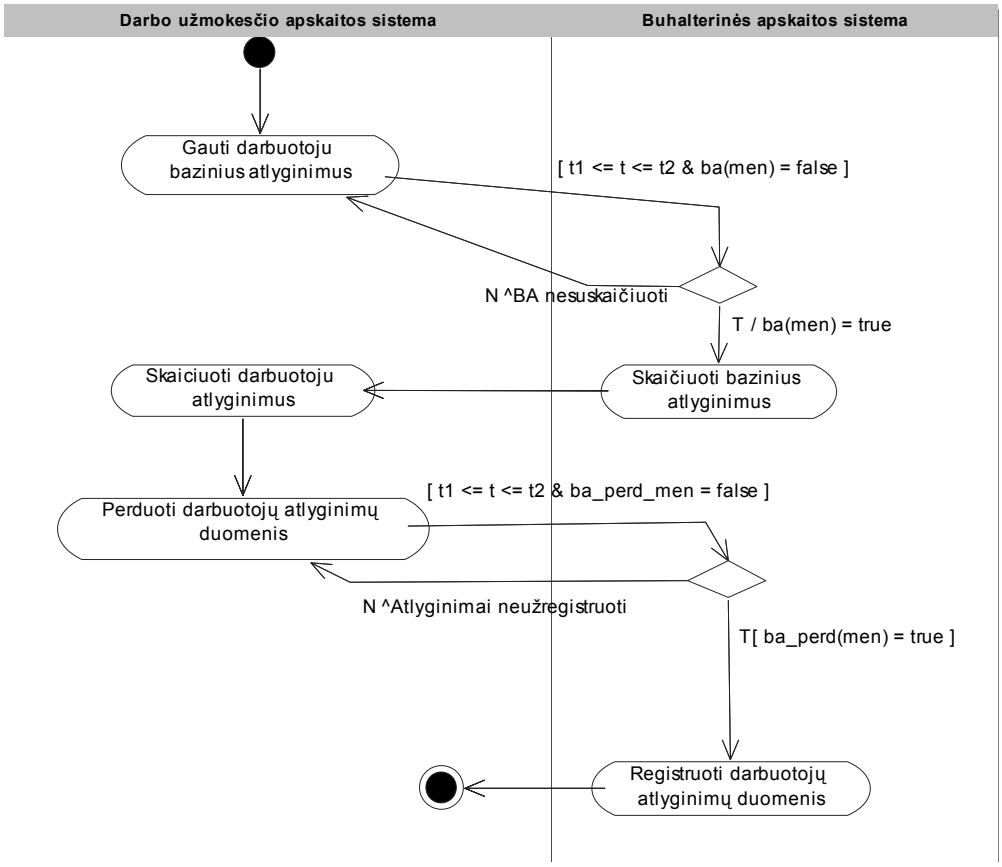
Siekiant gauti vienareikšmišką bei išsamią sąsajų specifikaciją, nepriklausomą nuo sąsajos realizavimo aplinkos, siūlomas metodas aprašomas naudojant UML notaciją bei OCL logines išraiškas. UML, kaip standartizuota objektinės analizės ir modeliavimo priemonė [5], leidžia specifikuoti sistemų komponentus nepriklausomai nuo jų realizavimo aplinkos. Papildant UML diagramas OCL deklaratyviomis loginėmis išraiškomis, pašalinamas natūralios kalbos dviprasmiškumas bei gaunama vienareikšmiška aprašymų interpretacija. 11 pav. pateikiamas komponentų sąsajos, ribojamos valdymo taisyklėmis, specifikacijos sudarymo procesas.



11 pav. Komponentų sąsajos, ribojamos valdymo taisyklėmis, specifikacijos sudarymo procesas

5.1 Veiklos proceso modelio sudarymas

Sistemų komponentų sąsajos identifikuojamos, nagrinėjant sistemų elgseną nusakančius veiklos procesus. Sistemos veiklą nusakantys veiklos procesai aprašomi UML veiksmų diagramomis, identifikuojančiomis valdymo ir duomenų srautus (12 pav.).



12 pav. Veiklos procesų modelis

12 pav. pateiktame veiklos procesų modelyje identifikuojamos 2 valdymo taisyklės, sąlygojančios *Buhalterinės apskaitos sistemos* ir *Darbo užmokesčio sistemos* sąsają:

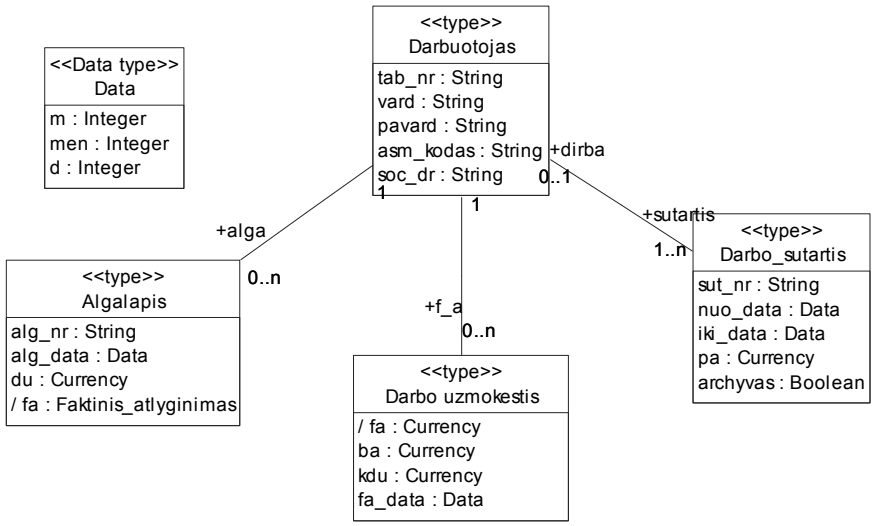
- Bazinių atlyginimų skaičiavimas turi būti atliekamas kartą per mėnesį – nustatytame mėnesio dienų intervale. Bazinis atlyginimas skaičiuojamas, nustatant n ankstesnių atlyginimų vidurkį;
- Darbuotojų atlyginimų skaičiavimas turi būti atliekamas kartą per mėnesį – nustatytame mėnesio dienų intervale.

Naudojant UML veiklos diagramas, charakterizuojamos komponentų sąsajos bei jas ribojančios valdymo taisyklės.

5.2 Veiklos tipų modelio sudarymas

Sekančiame sąsajos apibrėžimo žingsnyje sudaromas veiklos tipų modelis (13 pav.). Šis modelis sudaromas naudojant UML klasių diagramas, tačiau turi kiek kitokią semantiką.

Veiklos tipų modelis – tai vienas iš sąsajų apibrėžiančios specifikacijos modelių, todėl jame modeliuojama ne programinės klasės ar reliacinių duomenų bazių lentelės, bet dalykinės srities informaciniai tipai, tiesiogiai susiję su projektuojamomis sąsajomis. Modelyje dalykinės srities tipai žymimi <<*type*>> stereotipu. Pažymėtina, kad dalykinės srities tipai neturi operacijų. Informaciniai tipai charakterizuojami atributais, susietais su atitinkamais duomenų tipais. Naudojant UML klases, veiklos tipų modelyje taip pat gali būti atvaizduoti ir struktūriniai duomenų tipai (konkrečiu atveju, „Data“). Ryšiai tarp dalykinės srities objektų tipų aprašomi kaip ir klasių diagramose, nurodant asociacijų kardinalumus bei rolių pavadinimus.



13 pav. Veiklos tipų modelis

Dalykinės srities tipų modelį charakterizuojančios veiklos taisyklės išreiškiamos apribojimais, veikiančiais veiklos tipus ar jų tarpusavio asociacijas. Šie veiklos tipų modelio apribojimai aprašomi OCL išraiškomis ir pateikiami kaip modelio priedas.

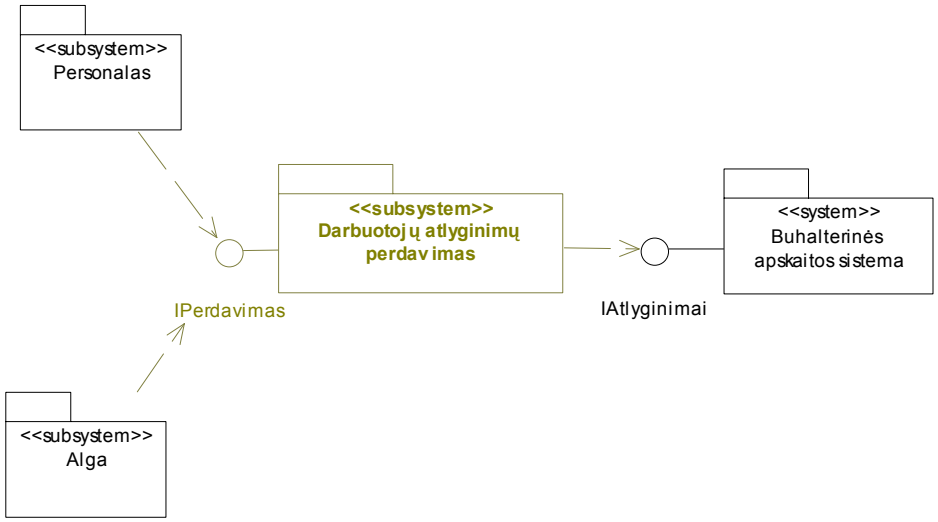
```
--tabelio numeris turi buti unikalus
context Darbuotojas inv:
Darbuotojas.allInstances()->forAll( d1, d2 | d1
<> d2 implies d1.tab_nr <> d2.tab_nr )

--sutarties numeris turi buti unikalus
context Darbuotojas inv:
self.sutartis->forAll( ds1, ds2 | ds1 <> ds2
implies ds1.sut_nr <> ds2.sut_nr )
```

14 pav. Veiklos tipų modelio apribojimai

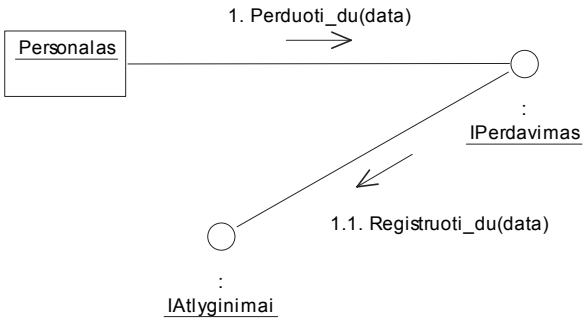
5.3 Sąveikos modelio sudarymas

Vadovaujantis kontraktų metodo teorija, sąsajos apibrėžiamos komponentų specifikacijose, identifikuojant siūlomų ir naudojamų sąsajų aibes. Tokiu būdu sudaroma pradinė loginė komponentų architektūra (15 pav.).



15 pav. Loginė komponentų architektūra

Komponentų specifikacijose apibrėžiamos sąsajos bei jų operacijų realizavimui būdingi apribojimai, nusakantys įvairias veiklos taisykles. Šie apribojimai identifikuojami modeliuojant sistemos komponentų objektų sąveikas, naudojant UML bendradarbiavimo diagramų notaciją (16 pav.).



16 pav. Bendradarbiavimo diagrama

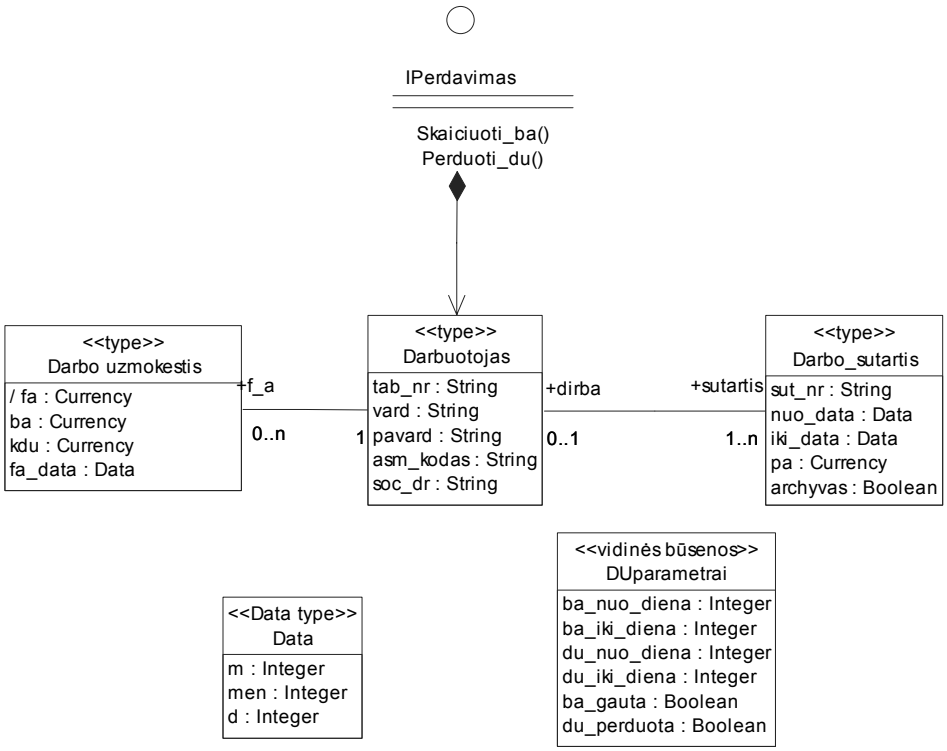
Bendru atveju, komponentų sąveikų diagramose akcentuojama konkrečių komponentų objektai bei jų sąveika su kitais komponentais arba išplėstinė aibė tarpusavyje sąveikaujančių komponentų objektų. Tačiau pranešimų siuntimas galimas tik tarp sistemos specifikacijose apibrėžtų komponentų objektų, nes pranešimų siuntimas tolygus komponentų funkcijų iškvietimui.

5.4 Sąsajų apibrėžimas

Sąsajos apibrėžiamos sudarant sąsajų specifikacijas, kurias sudaro:

- sąsajos pavadinimas;
- sąsajos informacinis modelis;
- sąsajos operacijų specifikacijos;
- papildomi informacinio modelio ar operacijų savybių invariantai.

Specifikuojant sąsajas, siekiama sudaryti maksimaliai nepriklausomus sąsajų aprašymus. Tuo tikslu kiekviena sąsaja susiejama su atitinkama dalykinės srities tipų modelio dalimi, nusakančia nagrinėjamą sąsają, kurios informacinis modelis pateikiamas sąsajų specifikacijos diagramose (17 pav.).



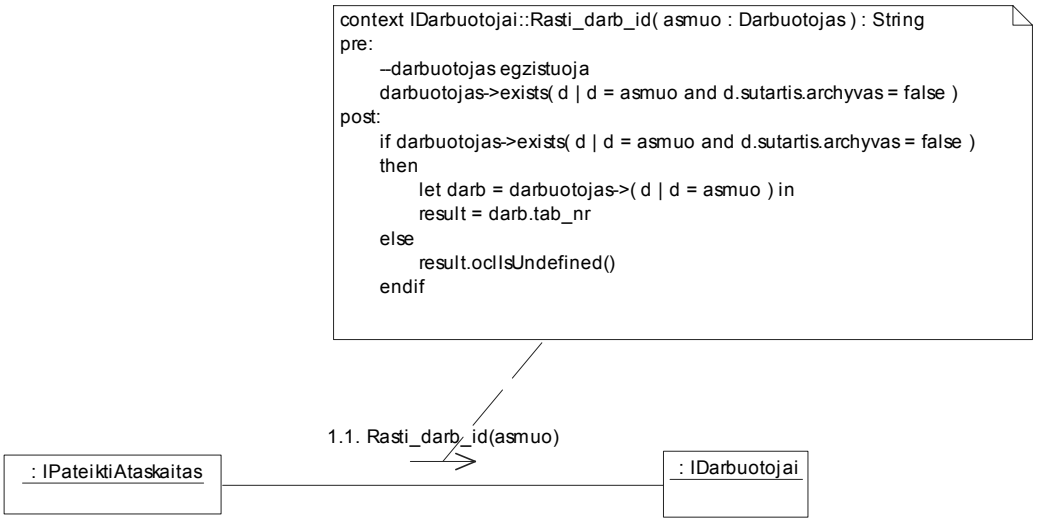
17 pav. Sąsajos "IPerdavimas" informacinis modelis

5.5 Operacijų apibrėžimas

Operacijų specifikacijas sudaro jų *signatūros* bei *prieš* ir *po* sąlygų rinkinių aprašymai. Operacijų specifikacijos aprašomos OCL loginėmis išraiškomis bei susiejamos su komponentų objektų sąveikas bendradarbiavimo diagramose vaizduojančiomis operacijomis (18 pav.).

Operacijos signatūra – tai operacijos antraštė, apibrėžianti operacijos įėjimo parametrus. Operacijų parametrai gali būti įvairių tipų: nuoroda į komponento objektą, paprastas ar struktūrinis duomenų tipas bei mišrus duomenų tipų rinkinys.

Operacijų *prieš* ir *po* sąlygomis nusakomos statinės sąsajų savybės. *Prieš* ir *po* sąlygos – specifikacijų artefaktai, leidžiantys detaliai apibrėžti operacijas, tačiau į operacijų kodą neįtraukiami. *Prieš-sąlyga* – tai loginė išraiška, kuri teoriškai įvertinama prieš operacijos vykdymą, tačiau jokiu būdu nesiejama su operacijos iškvietimu. Prieš-sąlygos patenkinimas garantuoja po-sąlygos rezultato gavimą, priešingu atveju – komponento elgsena neapibrėžiama. *Po-sąlyga* - tai loginė išraiška, kuri teoriškai įvertinama įvykdžius operaciją. Ši sąlyga atspindi sistemos būsenų pasikeitimus, įvykdžius operaciją.



18 pav. Patikslinta komponentų objektų bendradarbiavimo diagrama

5.6 Sąsajos valdymo taisyklių apibrėžimas

Komponentų sąsajas ribojančios valdymo taisyklės, išskirtos pirmame specifikacijos sudarymo etape, apibrėžiamos OCL loginėmis išraiškomis, sudarančiomis atskirus sąsajų invariantus. Skirtingai negu Daniels metode, sąsajas ribojančios valdymo taisyklės išskiriamos iš sąsajos operacijų *prieš* sąlygų bei pateikiamos kaip sąsajų specifikacijų sudedamoji dalis. Dažnai sąsajas ribojančios valdymo taisyklės priklauso nuo vidinių sistemos būsenų kintamųjų. Šie sistemos būsenų parametrai charakterizuojami dalykinės srities tipų modelyje atskirų klasių pavidalu, pažymint jas <<*vidinės_būsenos*>> stereotipu. Sąsajas ribojantys parametrai įtraukiami į atitinkamų sąsajų specifikacijas, t.y. į sąsajų informacinius modelius. 19 pav. pateiktas sąsajos valdymo taisyklės – periodiškai skaičiuoti darbuotojų atlyginimų vidurkį apibrėžtame datų intervale – OCL aprašymas bei taisyklę charakterizuojantys sistemos būsenų parametrai.

```
context IPateiktiAtlyginimus inv:
--operacija turi buti iskviesta laiku
    if IPateiktiAtlyginimus.reception.hasMatchingSignature( Skaiciuoti_ba( men_sk,
data ) ) and ( data.d >= DUparametrai.ba_nuo_diena and data.d <=
DUparametrai.ba_iki_diena ) and
    then
        if not DUparametrai.ba_gauta
        then
            let pran : oclMessage = IAtlyginimai^Gauti_fa_vid( men_sk, data ) in
            if pran.hasReturned() and pran.result()
            then
                DUparametrai.ba_gauta = true
            else
                DUparametrai.ba_gauta = false
            endif
        endif
    else
        DUparametrai.ba_gauta = false
    endif
```

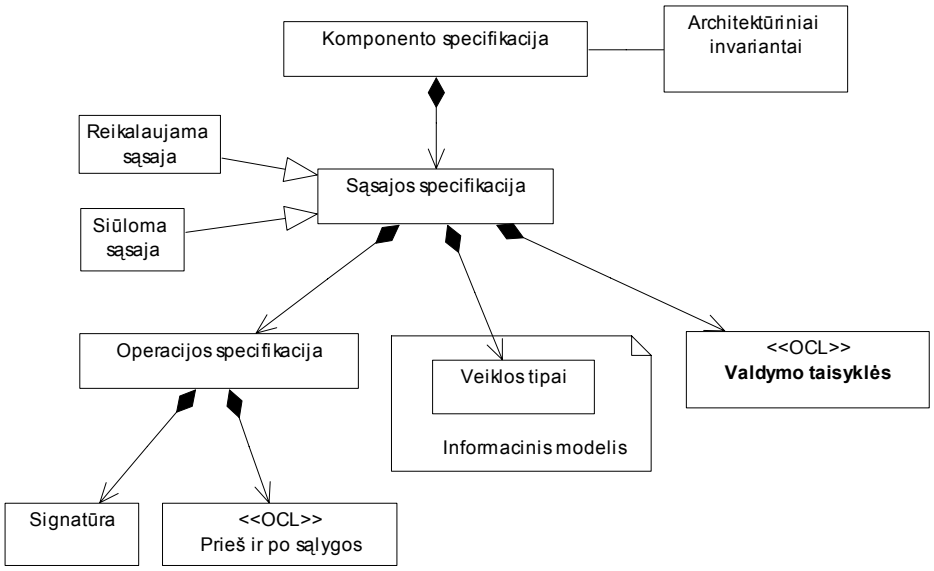
<<vidinės būsenos>>
DUparametrai

ba_nuo_diena : Integer
ba_iki_diena : Integer
du_nuo_diena : Integer
du_iki_diena : Integer
ba_gauta : Boolean
du_perduota : Boolean

19 pav. Sąsają ribojančios valdymo taisyklės aprašas

5.7 Valdymo taisyklėmis ribojamos sąsajos specifikacijos metamodelis

Pasiūlyta valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimo metodika aprašo 20 pav. pateiktą metamodelį, išskiriantį komponento, sąsajos bei operacijų sąvokas.



20 pav. Valdymo taisyklėmis ribojamos sąsajos specifikacijos metamodelis

Iš 20 pav. matyti, kad komponentų tarpusavio priklausomybes nulemia architektūriniai invariantai. Komponentai nusakomi komponentų specifikacijomis, sudarytomis iš sąsajų specifikacijų, vienareikšmiškai apibrėžiančių tiek komponentų siūlomas, tiek reikalaujamas sąsajas. Komponentų sąsajų specifikacijos sąvoka apima sąsajų operacijų specifikacijas, sąsajų informacinių modelius bei sąsajas ribojančias valdymo taisykles. Operacijų specifikacijos nusako operacijų signatūras bei operacijų *prieš* ir *po* sąlygų rinkinius. Sąsajų informaciniuose modeliuose identifikuojami nagrinėjamą dalykinę sritį nusakantys bei tarpusavyje susiję veiklos tipai.

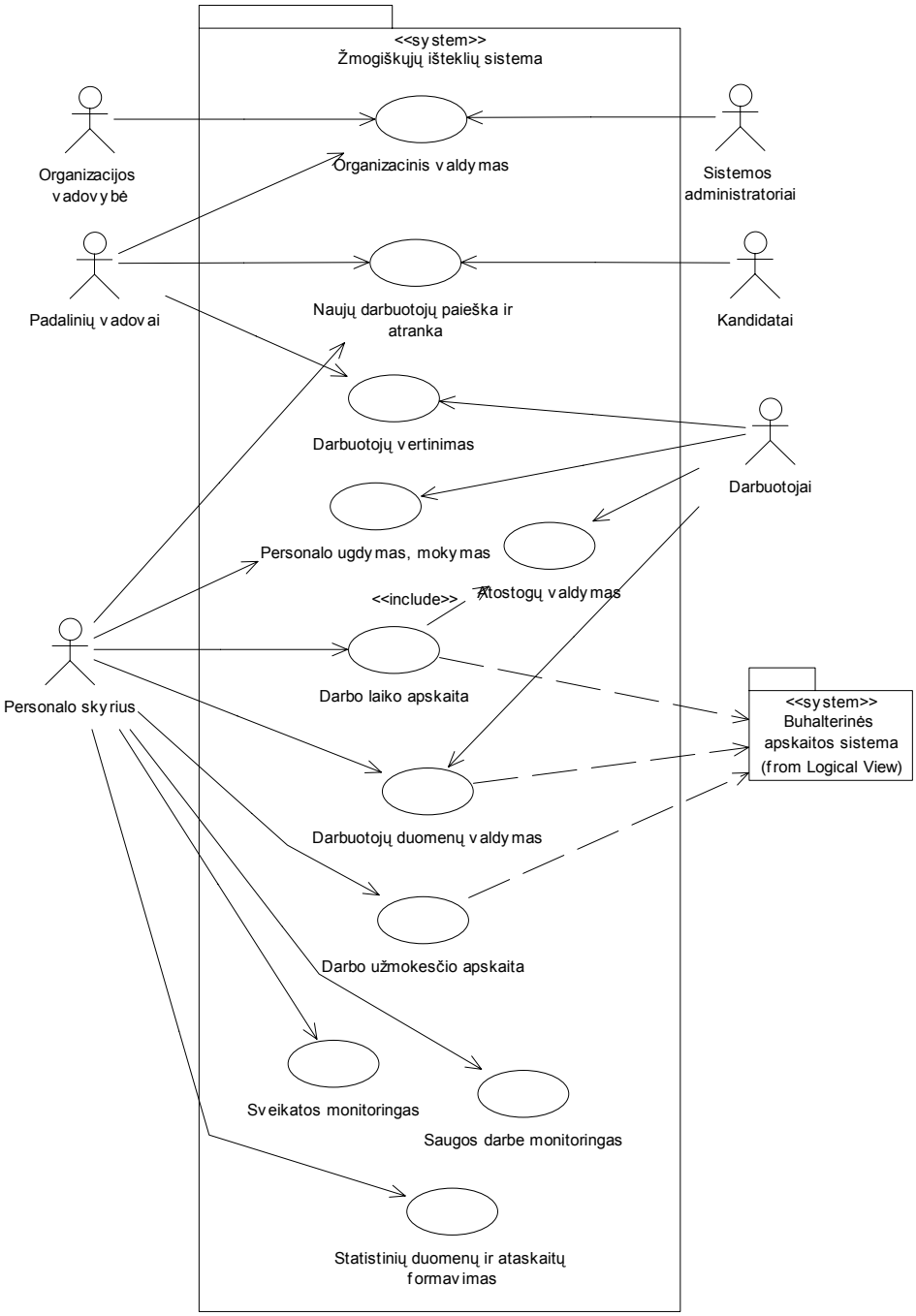
Kaip matyti iš komponentų sąsajų specifikacijos metamodelio, sąsajas ribojančios valdymo taisyklės apibrėžiamos nepriklausomai nuo operacijų *prieš* ir *po* sąlygų bei pateikiamos kaip sąsajų specifikacijų sudedamoji dalis. Tokiu būdu sudaromos galimybės lanksčiai pritaikyti kintančias veiklos taisykles, nekeičiant operacijų kodo.

6 Organizacijos žmogiškųjų išteklių valdymo sistema

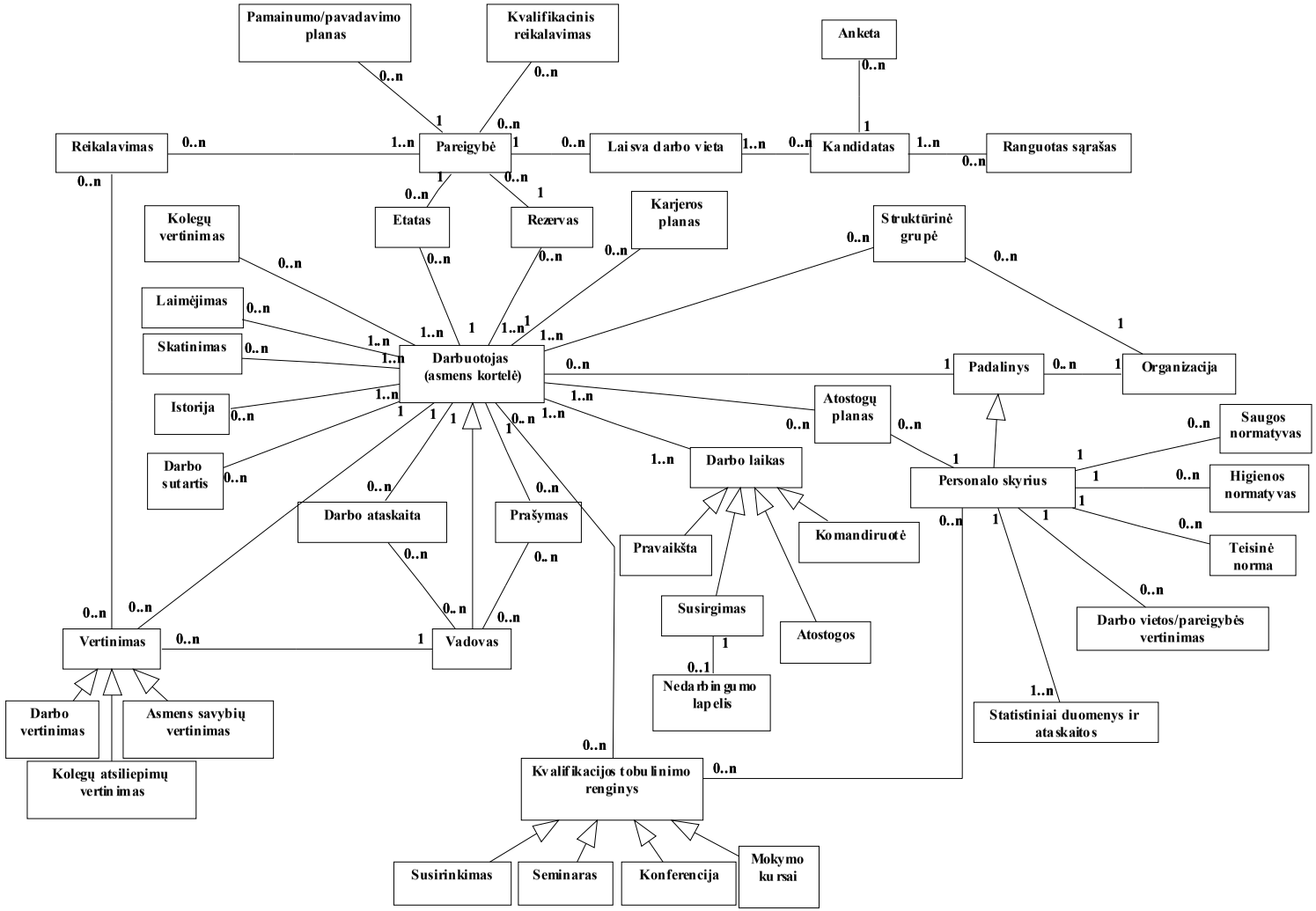
Kuriant organizacijos žmogiškųjų išteklių valdymo sistemo sistemą, buvo siekiama:

- sukurti lengvai naudojamą sistemą, paremtą internetine sąsaja;
- realizuoti platesnį nei šiuo metu egzistuojančių sistemų funkcijų spektrą;
- užtikrinti su personalu susijusių dokumentų tvarkymą;
- suteikti kiekvienam darbuotojui komunikavimo galimybes;
- padidinti kiekvieno personalo nario informuotumą, suteikiant duomenis apie jį patį ir organizacijos informacinius išteklius pagal darbuotojo teises;
- siekti ekonominio efektyvumo, automatizuojant organizacijos valdymo, įdarbinimo, mokymo funkcijas, tokiu būdu sutaupant nemažai laiko bei piniginių sąnaudų;
- užtikrinti tinkamų darbuotojų priskyrimą pareigybėms ir atskiroms užduotims, remiantis kompetencijų modeliais;
- užtikrinti darbuotojų kompetencijos didinimą, mokymą, darbo išteklių planavimą;
- kaupti istoriją apie kiekvieną darbuotoją;
- stebėti ir vertinti pareigybes, darbuotojus bei jų darbo sąlygas;
- sistemą pritaikyti Lietuvos sąlygoms, kad būtų galima panaudoti daugelyje įmonių.

Taikant UML panaudojimo atvejų modelį, 21 pav. pavaizduotos organizacijos žmogiškųjų išteklių sistemos ribos. Panaudojimo atvejų notacija žymi atskirą susijusių funkcijų poaibį.



21 pav. Sistemos ribos



22 pav. Dalykinės srities duomenų modelis

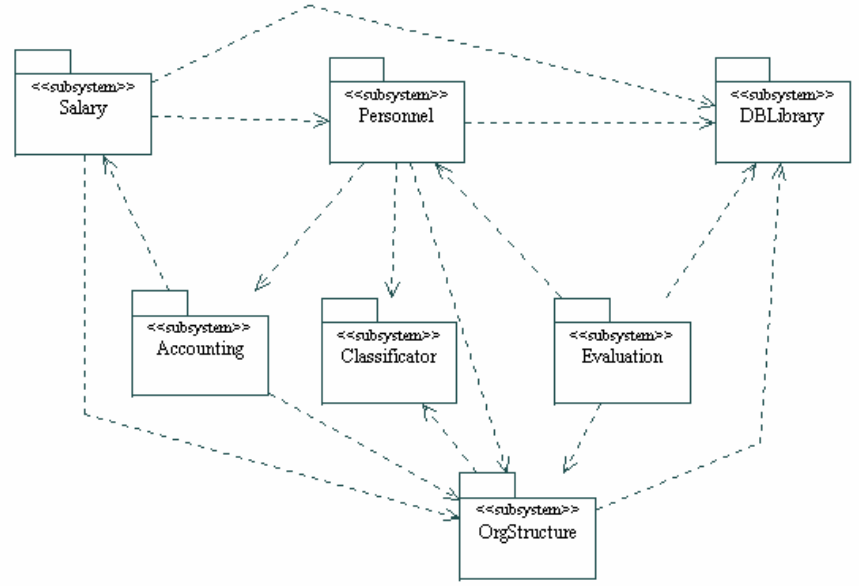
Sistemos dalykinės srities duomenų modelis, vaizduojantis sistemoje cirkuliuojančius duomenų vienetus, pateiktas 22 pav.

Vadovaujantis sudarytu organizacijos žmogiškųjų išteklių valdymo modeliu, pagal UAB "PTI Technologijos" poreikius suprojektuota, realizuota ir įdiegta "Organizacijos žmogiškųjų išteklių valdymo sistemos" dalis, užtikrinanti patogų informacijos apie personalą valdymą bei duomenų perdavimo sąsają tarp sistemos ir buhalterinės apskaitos programos, nustatanti darbuotojų darbo užmokestį, naudojant specializuotą vertinimo sistemą bei formuojanti statistinius ir suminius rezultatus. Realizuojant pasirinktą žmogiškųjų išteklių valdymo sistemos dalį, buvo vadovaujamasi inžinerinių sprendimų pasirinkimo analizės išvadomis [14] - [16].

Sukurta sistema apima šiuos modulius:

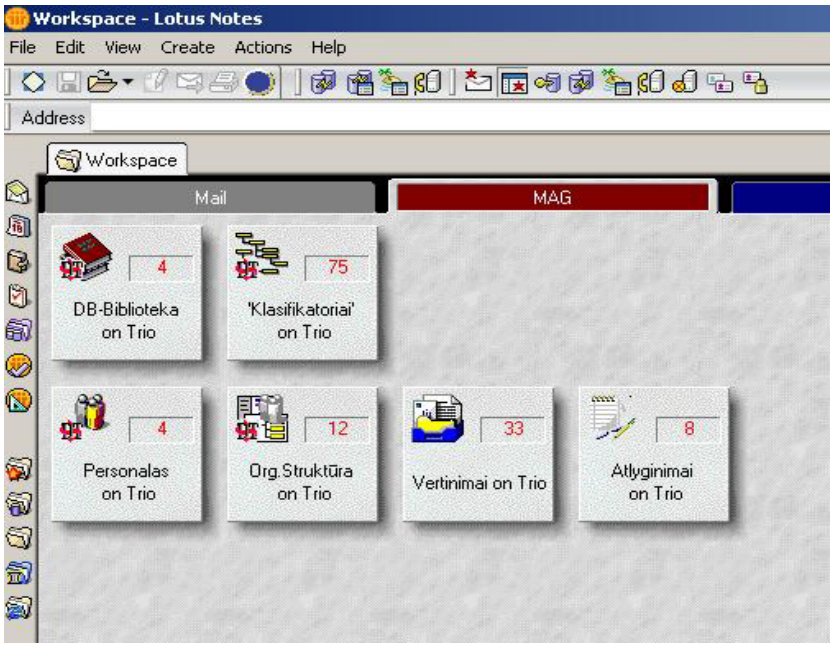
- *DB Biblioteka ("DBLibary")* - registruoja informaciją apie sistemą sudarančias posistemes;
- *Klasifikatoriai ("Classficators")* - registruoja bendrą klasifikuotą sistemoje naudojamą informaciją;
- *Organizacinė struktūra ("OrgStructure")* - aprašo organizacijos padalinių struktūrą bei pareigybių pavaldumo medį;
- *Personalas ("Personnel")* - registruoja ir saugo duomenis apie organizacijos darbuotojus;
- *Atlyginimų skaičiavimo modulis ("Salary")* - pagal gautus vertinimo rezultatus nustato kintamą darbo užmokesčio dalį (KDU), suskaičiuoja *NETO* atlyginimus ir pateikia ataskaitas,
- *Vertinimų modulis ("Evaluation")* - pateikia darbuotojams vertinimo anketas, formuoja vertinimo rezultatus bei ataskaitas.

Loginė sistemos architektūra pavaizduota 23 pav.



23 pav. Loginė sistemos architektūra

24 pav. pavaizdas realizuotos sistemos modulių vaizdas Lotus Notes aplinkoje.



24 pav. Sukurtą sistemą sudarantys moduliai Lotus Notes aplinkoje

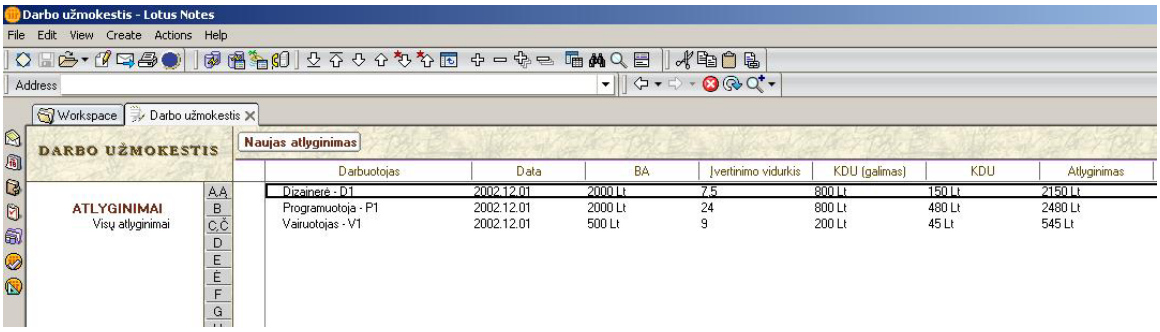
Sistemos kūrimo metu, sudarytas detalus testavimo planas, aprašantis sukurtos programinės įrangos kokybės įvertinimo strategiją. Vadovaujantis sudarytu testavimo planu, realizuota sistema ištestuota taikant šias strategijas:

- *vienetų testavimas* - „baltos“ ir „juodos“ dėžės metodai;
- *integracinis testavimas* – veikimo, funkcinis, saugumo sąsajų;
- *sistemos testavimas.*

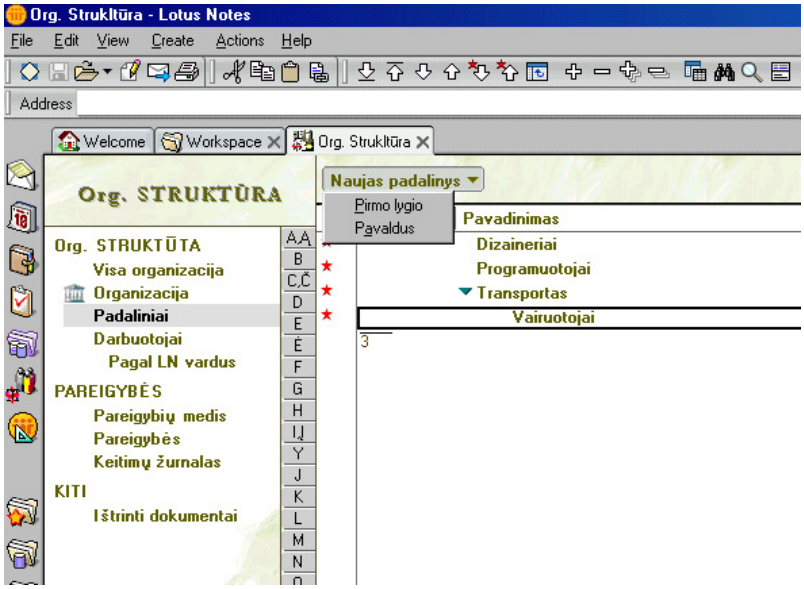
Panaudojant programinės įrangos kokybės užtikrinimo procedūras, sudaryta realizuotos sistemos kokybės įvertinimo ataskaita, apimanti kokybės užtikrinimo procedūrų, projekto etapų ir kokybės užtikrinimo plano etapų sąsajų, techninių, programinių ir žmogiškųjų resursų tyrimą.

Praktikos, kuri buvo atliekama programinės įrangos užsakovo įmonėje UAB "PTI Technologijos", metu buvo sėkmingai įdiegta bandomajam laikotarpiui realizuota organizacijos žmogiškųjų išteklių valdymo sistema. Kartu su sistema pateikta sistemos vartotojų dokumentacija bei sistemos instaliavimo bei administravimo atmintinė.

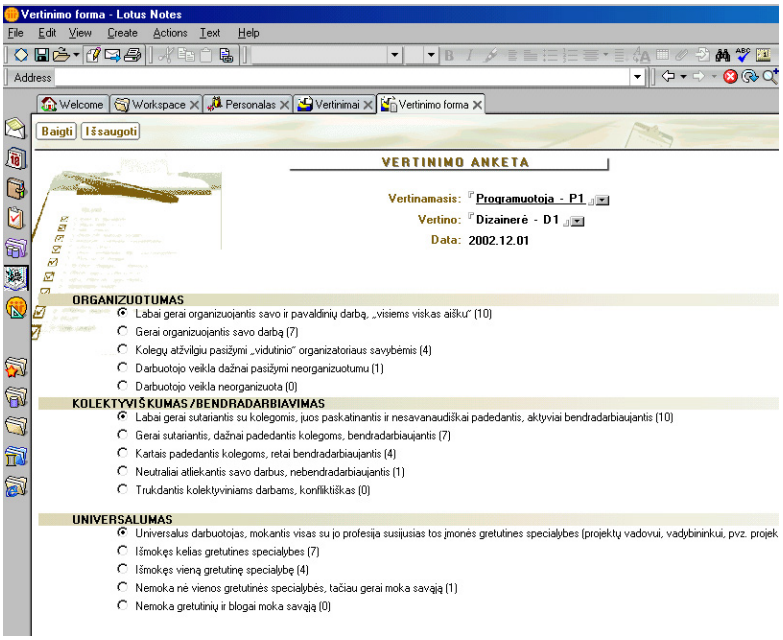
26 pav. – 27 pav. pateikti organizacijų žmogiškųjų išteklių valdymo sistemos veikimo aplinka.



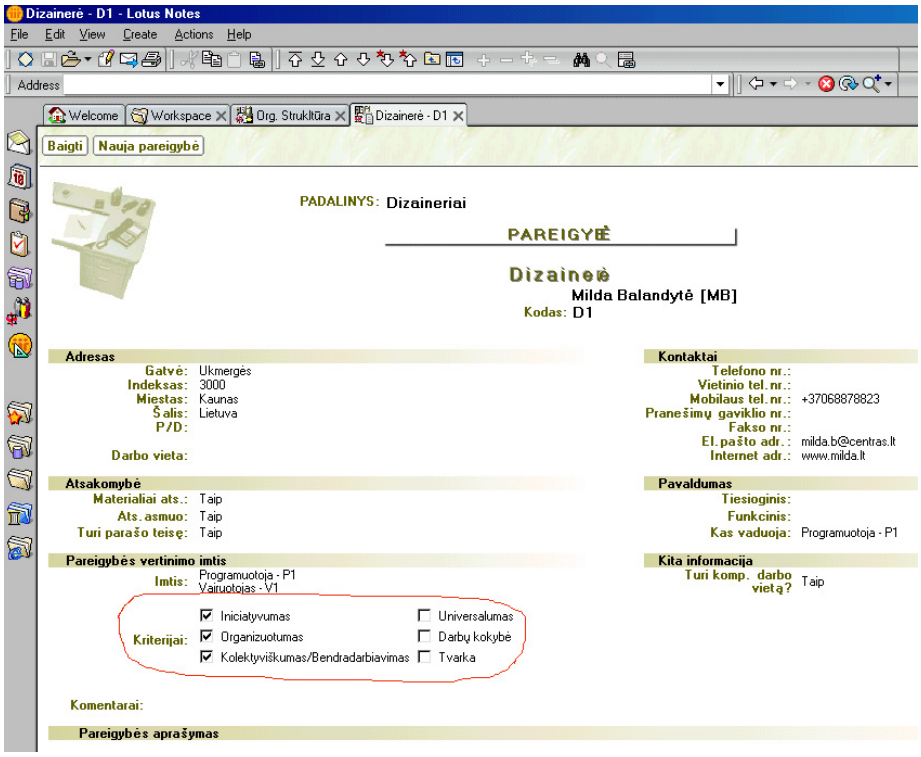
25 pav. Atlyginimų skaičiavimas



26 pav. Organizacijos padalinių struktūra



27 pav. Vertinimo pagal kriterijus forma



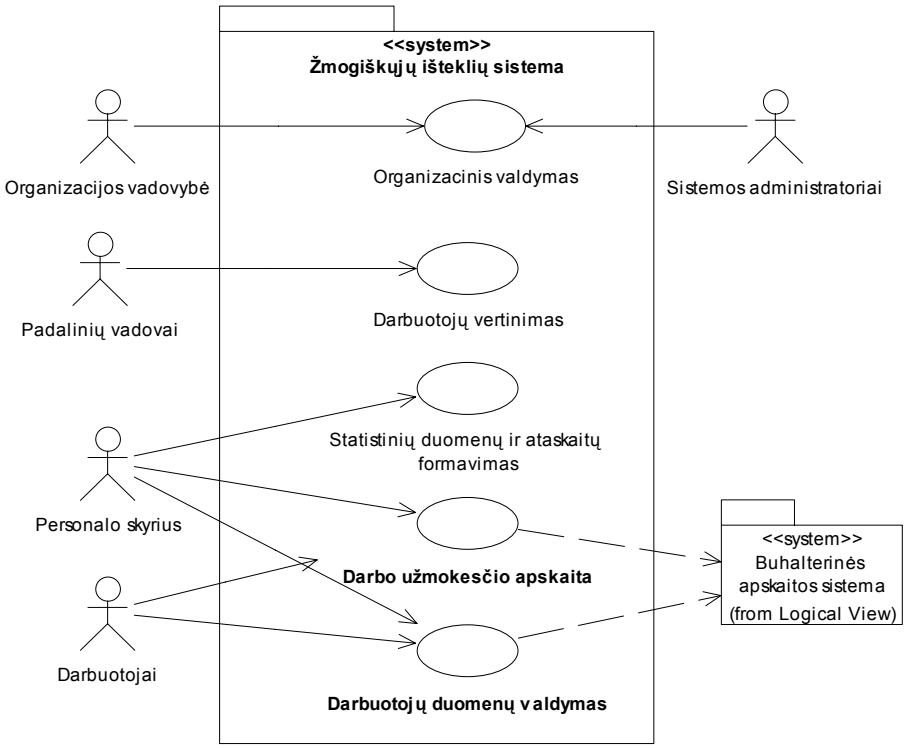
28 pav. Vertinimo kriterijų priskyrimo darbuotojui forma

7 Eksperimentinis sąsajų specifikavimo metodikos taikymas

Vienas iš kuriamai organizacijos žmogiškųjų išteklių valdymo sistemai keliamų tikslų - užtikrinti duomenų perdavimo buhalterinės apskaitos sistemai sąsają. Šios sąsajos modeliavimo metu išryškėjo valdymo taisyklės, veikiančios komponentų tarpusavio priklausomybes. Todėl šiame skyriuje siekiama pritaikyti bei patikrinti pasiūlytos komponentų sąsajos, ribojamos valdymo taisyklėmis, specifikavimo metodikos veiksmingumą.

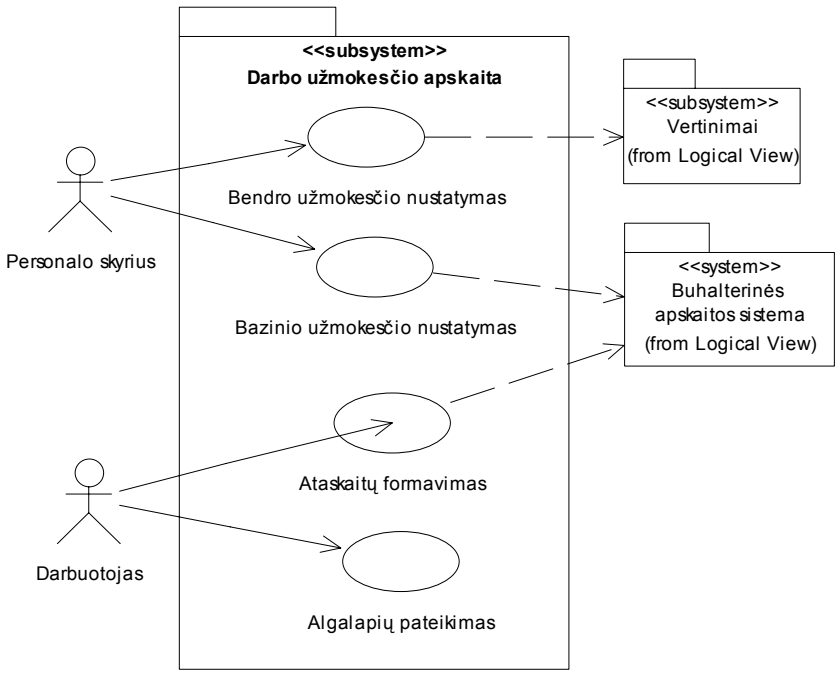
Eksperimentinės dalies tikslas – sudaryti žmogiškųjų išteklių valdymo sistemos ir buhalterinės apskaitos sistemos sąsajos specifikaciją taikant valdymo taisyklėmis ribojamų komponentų sąsajų specifikavimo metodiką.

Nagrinėjamos organizacijos žmogiškųjų išteklių sistemos ribos pateiktos 29 pav.



29 pav. Žmogiškųjų išteklių valdymo sistemos ribos

Iš 29 pav. matyti, kad sąsają su buhalterinės apskaitos sistema turi „Darbo užmokesčio apskaitos“ ir „Darbuotojų duomenų valdymo“ moduliai. Konkrečiu tyrimo atveju pasirinkta Darbo užmokesčio apskaitos modulio ir buhalterinės apskaitos sistemos sąsaja, kurios detalus panaudojimo atvejų vaizdas pateiktas 30 pav.



30 pav. Posistemio "Darbo užmokesčio apskaita" panaudojimo atvejų diagrama

5 lentelė. Posistemio "Darbo užmokesčio apskaita" panaudojimo atvejų aprašas

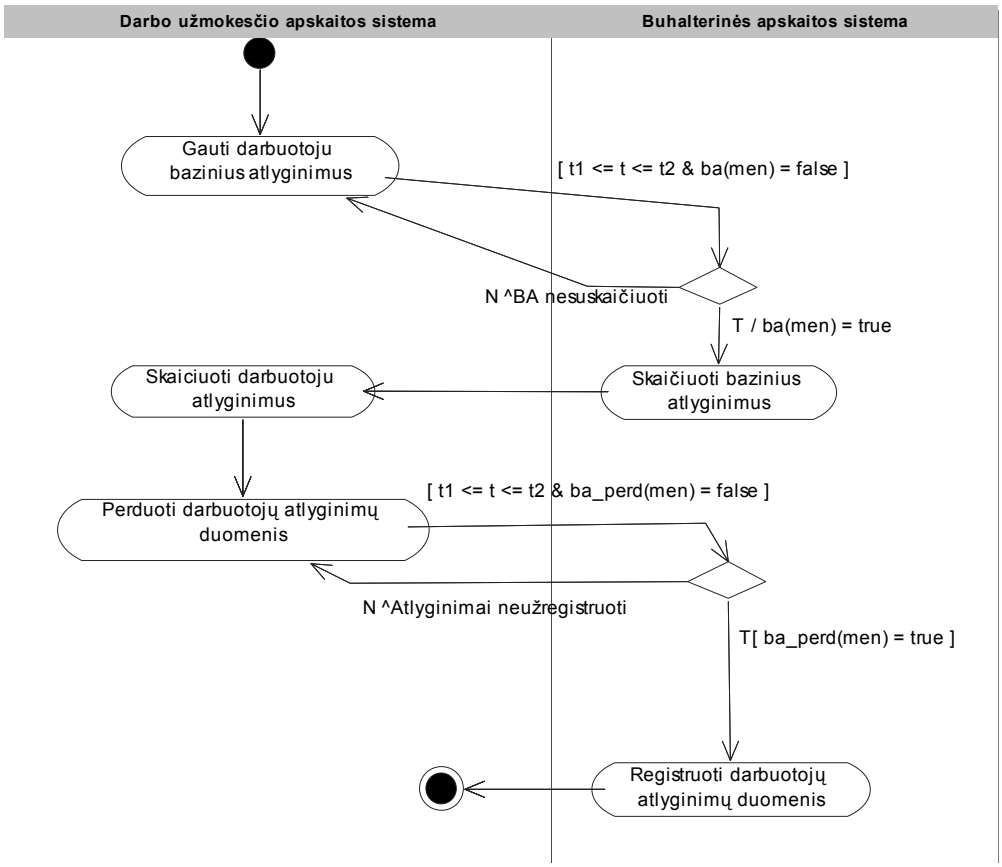
Panaudojimo atvejis	Darbo užmokesčio apskaita
Iniciatorius	Personalo skyrius, Darbuotojai
Tikslas	Nustatyti einamojo mėnesio darbuotojų faktinius atlyginimus
"Sėkmės" scenarijus 1. Personalo skyrius gauna priminimą, kad laikas skaičiuoti einamojo mėnesio darbuotojų kintamas darbo užmokesčio dalis; 2. Personalo skyrius patvirtina priminimą; 3. Sistema nustato darbuotojų bazinius atlyginimus ir kintamas darbo užmokesčio dalis; 4. Sistema nustato darbuotojų faktinį atlyginimą; 5. Sistema perduoda darbuotojų faktinių atlyginimų duomenis buhalterinės apskaitos sistemai; 6. Darbuotojas organizacijos portale pateikia atlyginimų istorijos užklausą; 7. Sistema suformuoja darbuotojo atlyginimų istorijos ataskaitą;	
Scenarijaus praplėtimas 2. Personalo skyrius nepatvirtina priminimo; 1.1 Sistema perveda pranešimą į budėjimo režimą;	

Pagal pateiktą valdymo taisyklėmis ribojamos sąsajos specifikavimo metodiką, pirmiausia sudaromas veiklos procesų modelis, identifikuojantis tiriamos sąsajos valdymo ir duomenų srautus (31 pav.).

Išskiriamos sekančios nagrinėjamą sąsają veikiančios valdymo taisyklės:

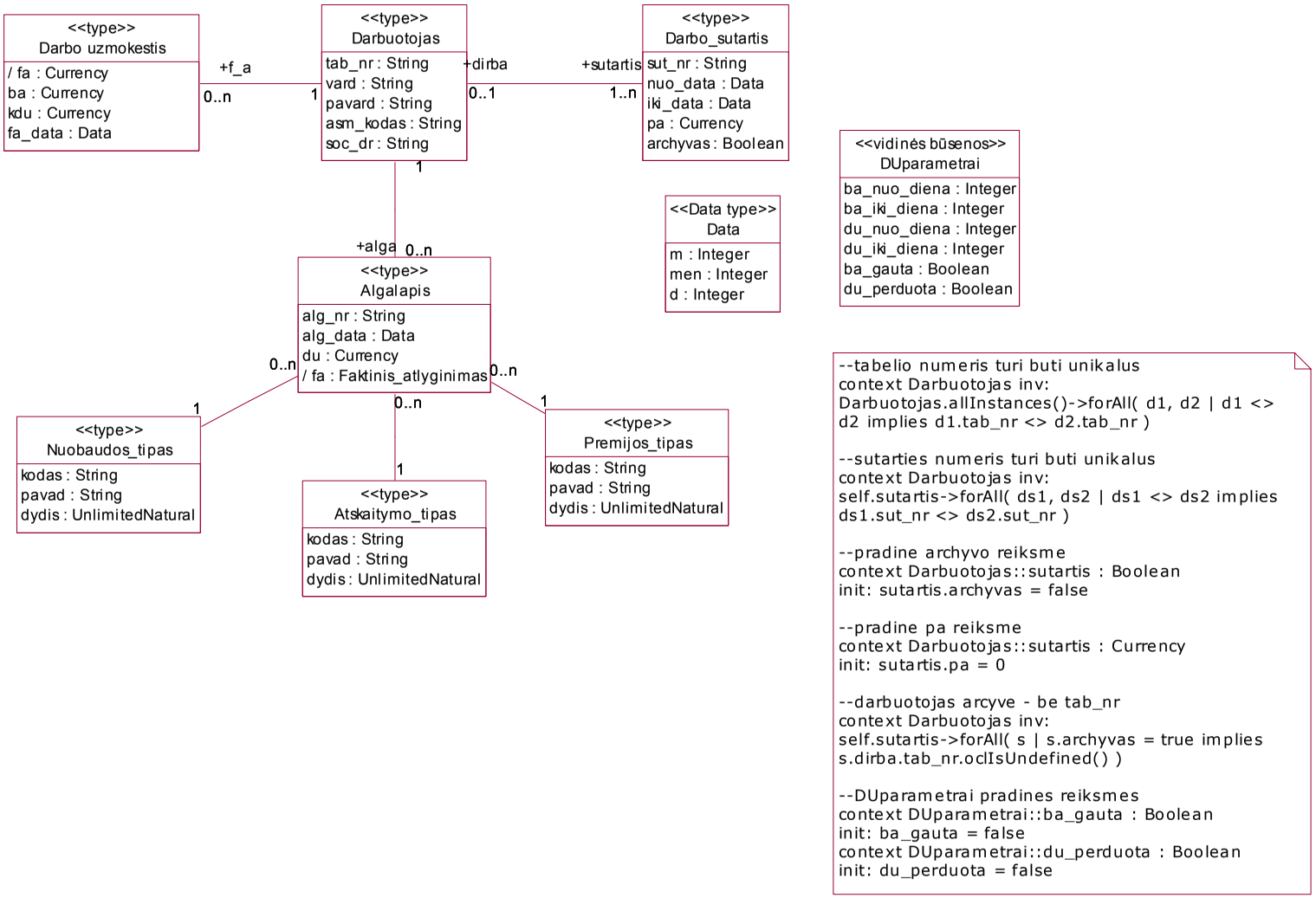
- t1 <= t <= t2 & ba(men) = false
  - bazinių atlyginimų skaičiavimas turi būti atliekamas kartą per mėnesį – nustatytame mėnesio dienų intervale;
  - bazinis atlyginimas skaičiuojamas, nustatant n ankstesnių atlyginimų vidurkį

- t1 <= t <= t2 & ba_perd(men) = false
  - darbuotojų atlyginimų skaičiavimas turi būti atliekamas kartą per mėnesį – nustatytame mėnesio dienų intervale.



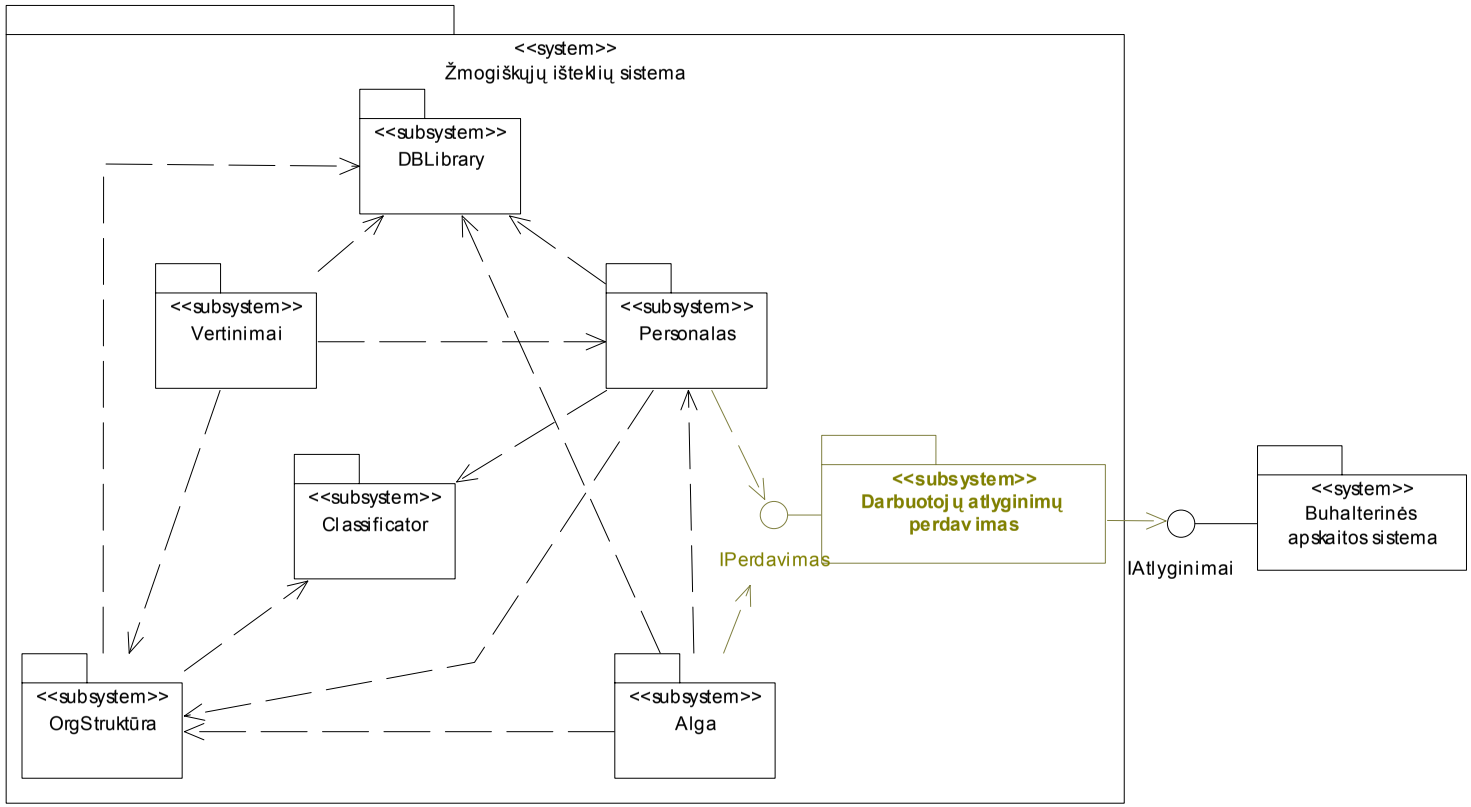
31 pav. Panaudojimo atvejo "Darbo užmokesčio apskaita" veiklos procesų diagrama

Sekančiame sąsajos specifikacijos sudarymo žingsnyje apibrėžiamas veikos tipų modelis (32 pav.), nusakantis dalykinės srities informacinius tipus, tiesiogiai susijusius su projektuojamomis sąsajomis. Veiklos tipus bei jų tarpusavio asociacijas sąlygojančios taisyklės aprašomos OCL loginėmis išraiškomis, kurios pateikiamos veiklos tipų modelio kontekste.



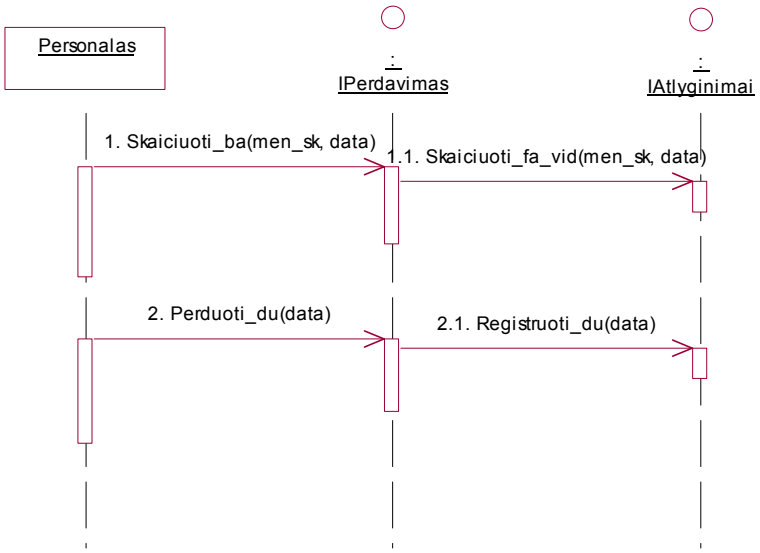
32 pav. Dalykinės srities tipų modelis

Identifikuojant nagrinėjamų komponentų siūlomų ir naudojamų sąsajų aibes, sudaroma pradinė loginė komponentų architektūra (33 pav.).



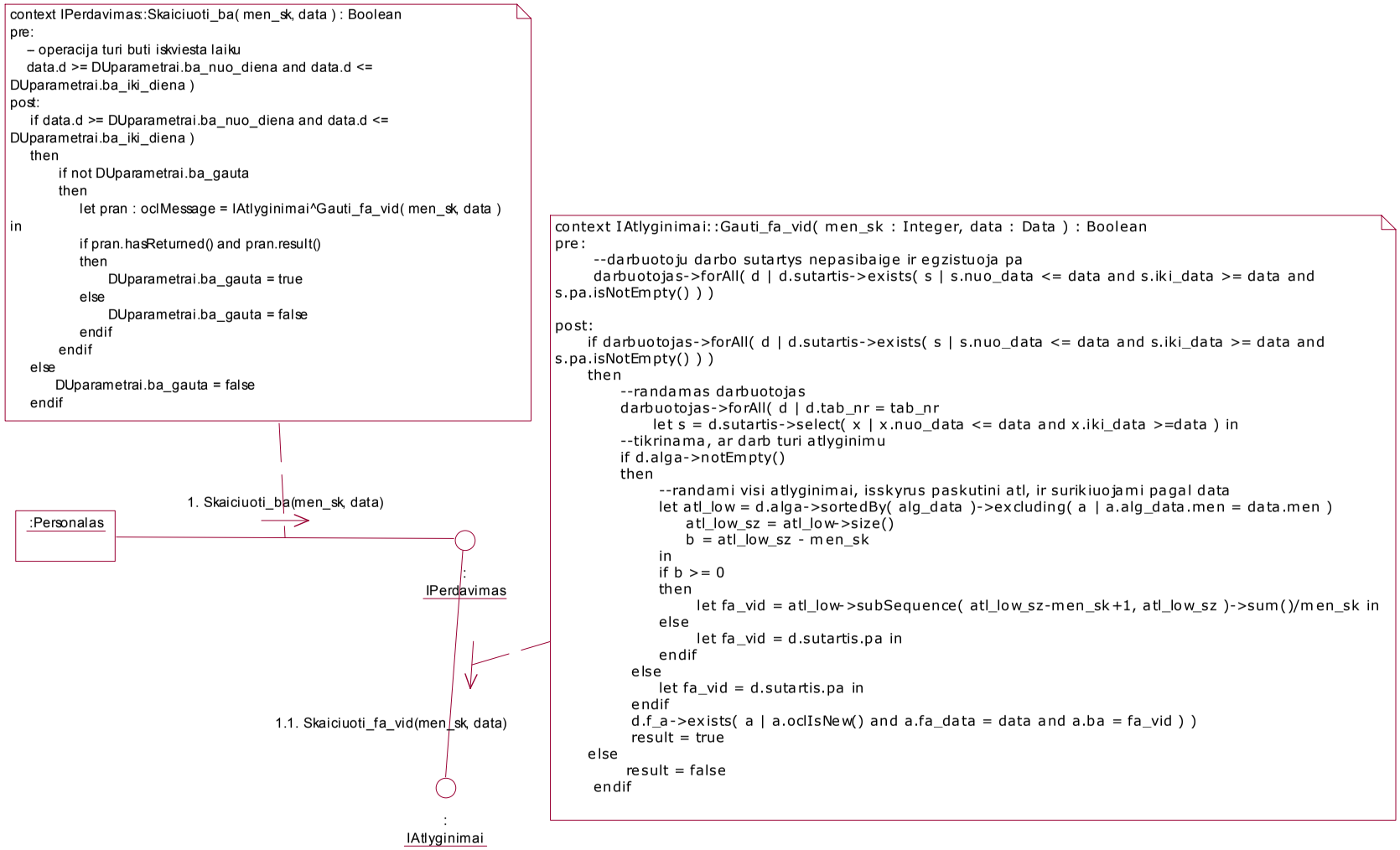
33 pav. Komponentų architektūros specifikacija

Toliau sudaromos komponentų objektų sąveikos, išreikštos UML sekų ir bendradarbiavimo diagramomis, charakterizuojančios komponentų sąsajų operacijas sąlygojančias veiklos taisykles (34 pav.- 36 pav.). Šiame specifikacijos žingsnyje taip pat sudaromos sąsają nusakančių operacijų specifikacijos.

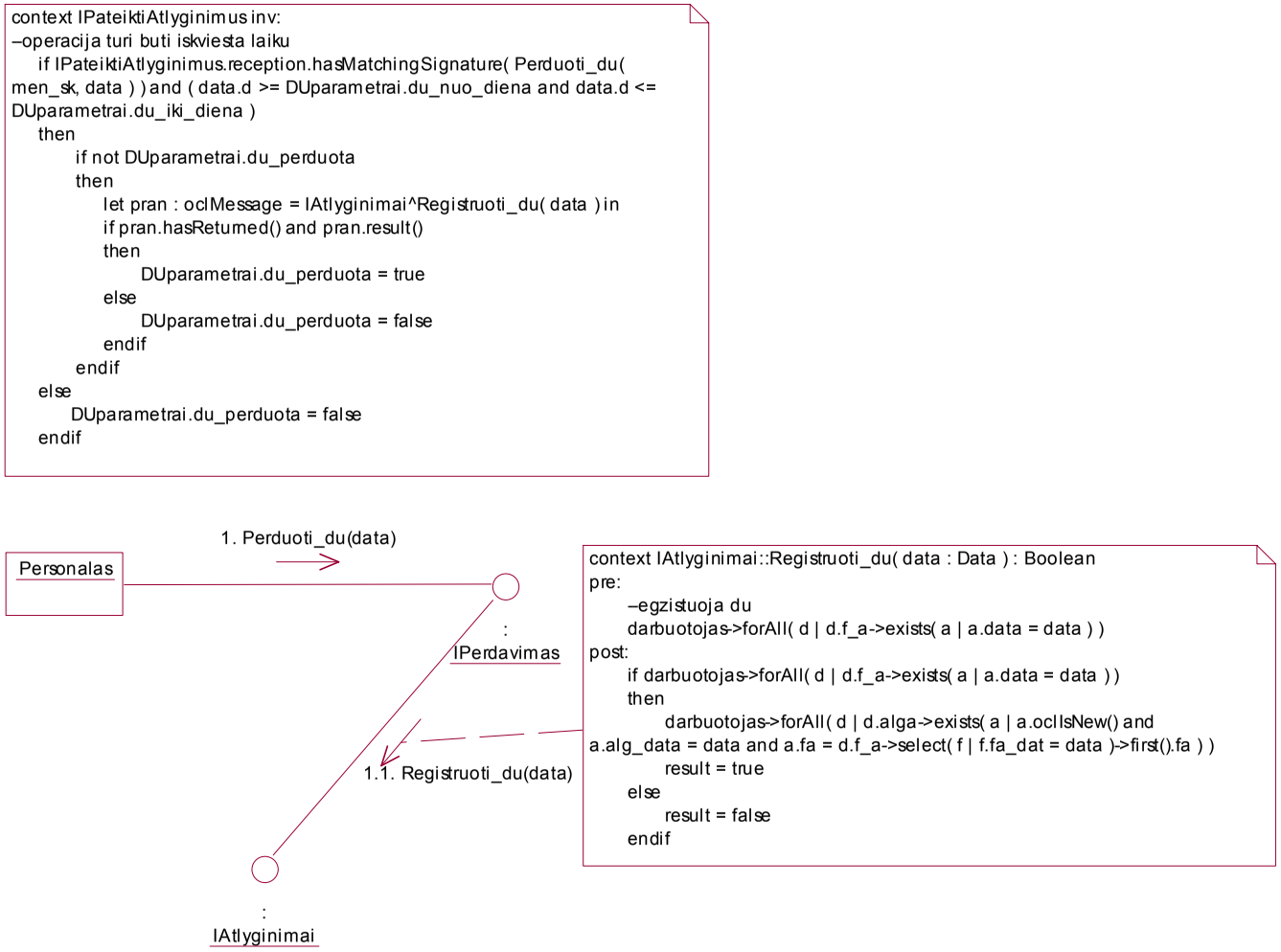


34 pav. Komponentų objektų sąsajos sekų diagrama

Taikant analizės dalyje aprašytus komponentų sąsajų specifikavimo metodus, sąsajas ribojančios valdymo taisyklės užrašomos operacijų *prieš* sąlygose (35 pav.: operacija „*Skaičiuoti_ba(men_sk, data)*“) – taip sumažinamas programinės įrangos lankstumas. 36 pav. sąsają ribojančios valdymo taisyklės atskirtos nuo sąsajos operacijos *prieš* sąlygos bei OCL loginėmis išraiškomis aprašytos kaip atskiras sąsajos invariantas. Tokiu būdu užtikrinamas sąsajos valdymo elementų pakeičiamumas.

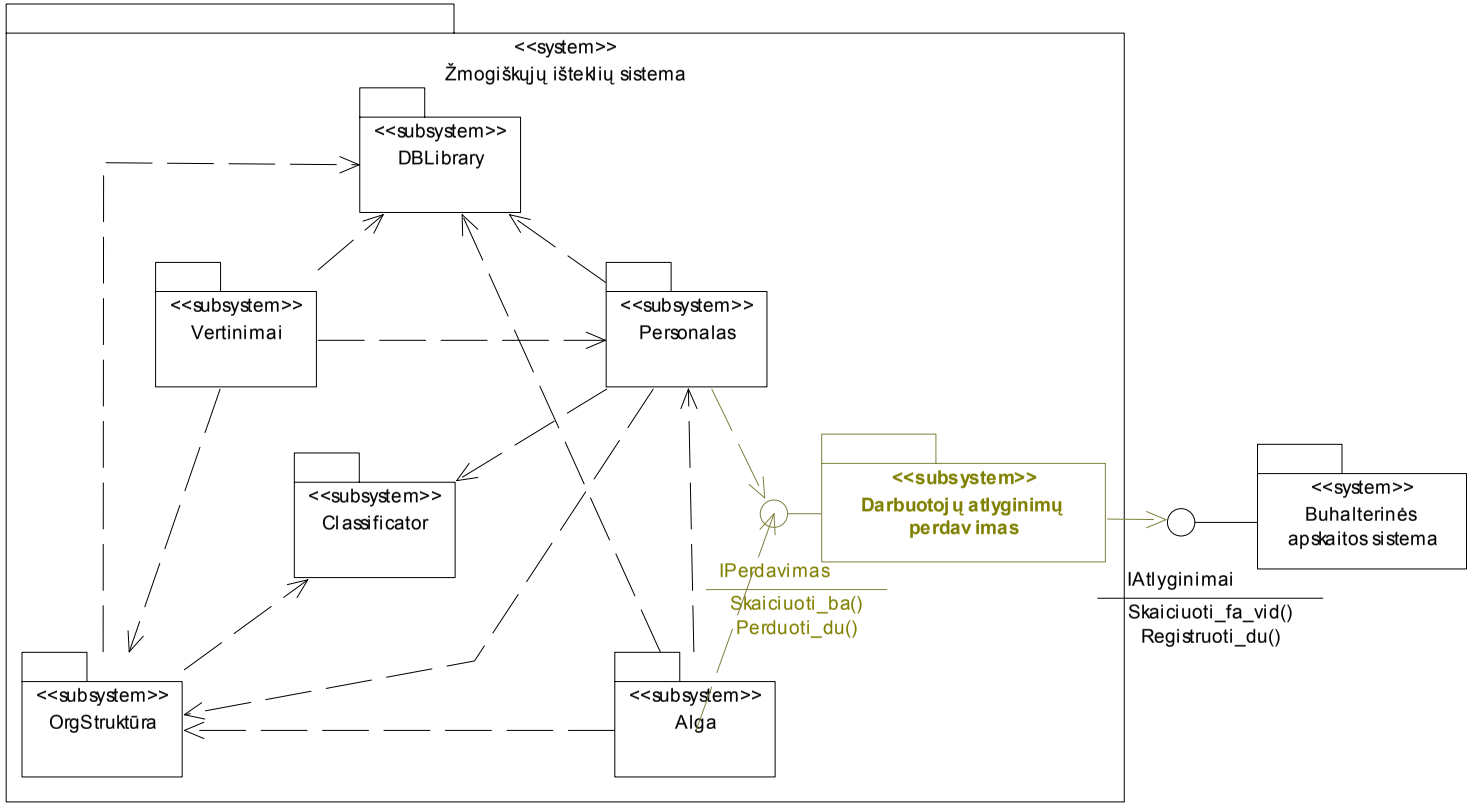


35 pav. Komponentų objektų bendradarbiavimo diagrama operacijai „BA skaičiavimas“ – sąsają ribojanti valdymo taisyklė apibrėžiama operacijos viduje



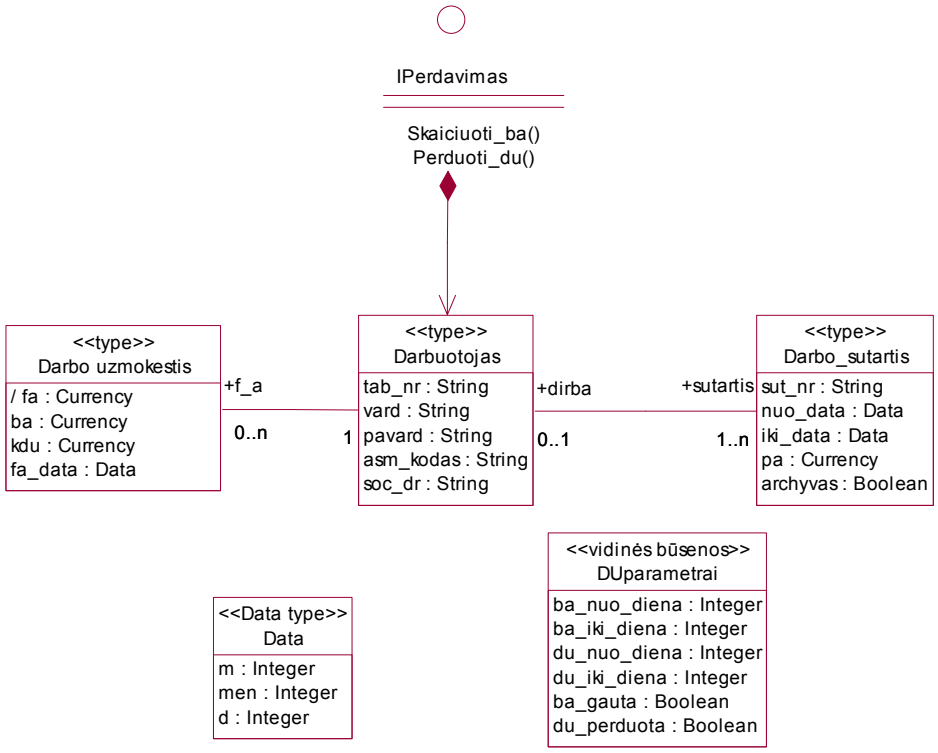
36 pav. Komponentų objektų bendradarbiavimo diagrama operacijai „DU perdavimas“ – valdymo taisyklė, ribojanti sąsają, pateikta kaip atskiras sąsają nusakantis invariantas

Nustačius komponentų sąsajas sudarančias operacijas bei sudarius jų specifikacijas, pateikiama patikslinta loginė projektuojamos sistemos architektūra (37 pav.).

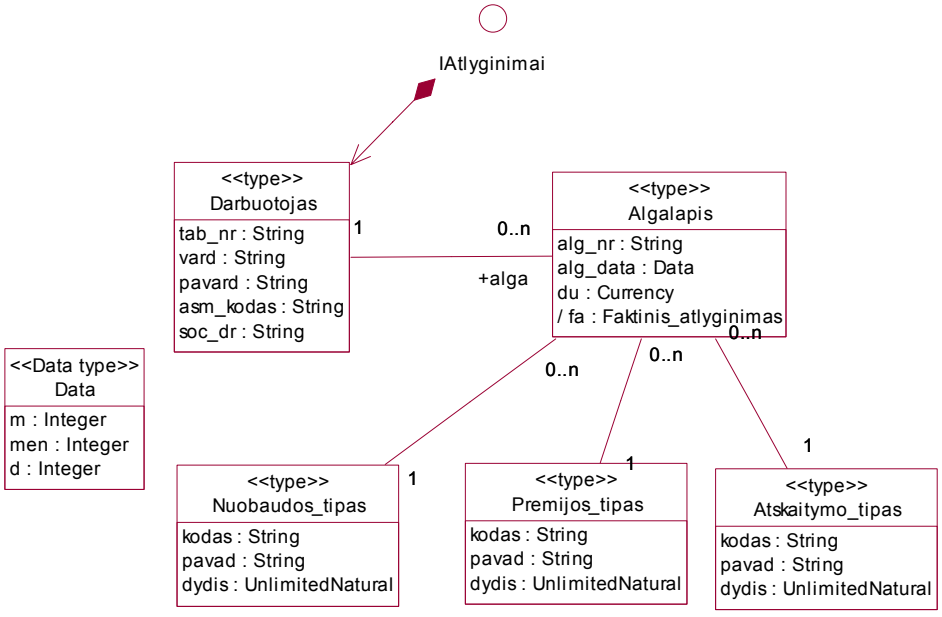


37 pav. Patikslinta sistemos komponentų architektūros diagrama

Sudarant maksimaliai nepriklausomus sąsajų aprašymus, kiekviena sąsaja susiejama su atitinkama dalykinės srities tipų modelio dalimi – tokiu būdu gaunami komponentų sąsajų informaciniai modeliai (38 pav., 39 pav.).



38 pav. Sąsajos "IPerdavimas" informacinis modelis



39 pav. Sąsajos "IAtlyginimai" informacinis modelis

8 Išvados

- Magistro studijų metu buvo suprojektuota ir sukurta žmogiškųjų išteklių valdymo sistema, tinkanti įvairių tipų įmonėms bei apimanti naujoviškas žmogiškųjų išteklių valdymo funkcijas.
- Sistema buvo projektuojama taikant iteracinio ir evoliucinio kūrimo, ICONIX, Martin-Odell, RUP, XP, UMM metodų principus.
- Sistema realizuota, naudojant Lotus Notes/Domino programinę įrangą, ir įdiegta užsakovų įmonėje.
- Įdiegus sistemą, atsirado nauji funkciniai poreikiai – sujungti ją su buhalterinės apskaitos sistema. Sąsajai sukurti reikėjo parengti specifikaciją, todėl buvo atlikta komponentinių sąsajų specifikavimo metodų analizė.
- Išanalizavus sąsajų specifikavimo metodus, pastebėta, jog sąsajos operacijų prieš-sąlygose bendrai aprašomos veiklos taisyklės, neatskiria sąlygų, atspindinčių sąsajas ribojančių valdymo taisyklių.
- Sudarytas sąsajos specifikacijos metamodelis ir specifikavimo metodika, apimanti valdymo taisyklių identifikavimą. Ši metodika buvo pritaikyta žmogiškųjų išteklių ir buhalterinės apskaitos sistemų sąsajai specifikuoti.
- Valdymo taisyklėms aprašyti panaudota OCL 2.0 kalba. Šių taisyklių atskyrimas nuo sąsajos operacijų *prieš* sąlygų leidžia specifikuoti komponentą, turintį didesnes pakartotino naudojimo galimybes - valdymo taisykles pakeisti kitomis, nekeičiant sąsają sudarančių operacijų kodo.
- Sudaryta metodika galėtų papildyti komponentų sąsajų specifikavimo metodus tais atvejais, kai specifikuojamos sąsajos operacijų vykdymas priklauso nuo tiriamoje aplinkoje galiojančių valdymo taisyklių.

9 Literatūra

[1] Cheesman, J.; Daniels, J.; UML Components: A simple process for specifying component-based software. – Addison-Wesley, 2000. – 176 p.

[2] Overview of Object Oriented Information Engineering (OOIE) Method [interaktyvus]. 1995, rugsėjis [žiūrėta: 2004-04-20]. Prieiga per internetą: <http://www.hep.net/chep95/html/papers/p102/html/p102_3.htm>

[3] Bock, C.; Three kinds of behavioural models. Journal of Object-Oriented Programming [interaktyvus], Vol. 12, Nr. 4. 1999, liepa-rugpjūtis [žiūrėta: 2004-04-30]. Prieiga per internetą: <http://www.kabira.com/bock/processcategory.html>

[4] Bock, C.; Unified behaviour models. Journal Of Object-Oriented Programming [interaktyvus] Vol. 12, Nr. 5. 1999, rugpjūtis [žiūrėta: 2004-04-30]. Prieiga per internetą: <http://www.kabira.com/bock/unifiedbehaviormodel.html>

[5] UML 2.0 Superstructure Specification. UML resource page [interaktyvus]. 2004, kovas [žiūrėta: 2004-04-01]. Prieiga per internetą: <http://www.uml.org/>

[6] Conrad, S.; Ramos, J.; Saake, G.; Sernadas, C. Evolving logical specification in informatikon systems. Scientific literature digital library [interaktyvus]. 1998 [žiūrėta: 2004-05-01] Prieiga per internetą: <http://citeseer.ist.psu.edu/46514.html>

[7] Graw, G. Specification of behaviour in component frameworks. Fourth International Workshop on Component-Oriented Programming [interaktyvus]. 1999, birželis. [žiūrėta: 2004-05-01]. Prieiga per internetą: <http://www.abo.fi/~Wolfgang.Weck/WCOP/99/Accepted.html>

[8] Jungclaus, R.; Hartmann, T.; Saake, G. Relationships between dynamic objects. Scientific literature digital library [interaktyvus]. 1998 [žiūrėta: 2004-04-25] Prieiga per internetą: <http://citeseer.ist.psu.edu/jungclaus93relationship.html>

[9] Martin, J.; Odell, J. Object-Oriented Methods: Pragmatic Considerations. – Prentice Hall, 1996 – 560 p.

[10] Juliana Kuester Filipe. A logic-based formalization for component specification. Journal of Object Technology, Vol. 1, No 3. 2002, rugpjūtis. [žiūrėta: 2004-05-15]. Prieiga per internetą: <http://www.jot.fm/issues/issue 2002 08/article13>

[11] Enselme, D.; Florin, F. Legon-Aubry. Design by contracts: Analysis of hidden dependencies in component based applications. Journal of Object Technology, Vol. 3, No 4. 2004, balandis. [žiūrėta: 2004-05-15]. Prieiga per internetą: <http://www.jot.fm/issues/issue_2004_04/article2>

[12] Balandytė, M.; Jašinskaitė, J.; Nemuraitė, L.; Tamulis, G. Šiuolaikinės žmogiškųjų išteklių valdymo sistemų kūrimo tendencijos. Iš: Informacinės technologijos 2003: konferencijos pranešimų medžiaga, Kaunas 2003 sausio 28-29 d.. Kaunas, 2003, p. XIV-47 - XIV-52.

[13] Balandytė, M.; Jašinskaitė, J.; Tamulis, G. Projekto paraiška. 2003, kovas. Prieiga per internetą: <http://www.soften.ktu.lt/~tamugied>

[14] Balandytė, M. Darbų sekų valdymo metodų analizė. 2002, balandis. Prieiga per internetą: <http://www.soften.ktu.lt/~tamugied>

[15] Jašinskaitė, J. Informacinių technologijų analizė. 2002, balandis. Prieiga per internetą: <http://www.soften.ktu.lt/~tamugied>

[16] Tamulis, G. Projektavimo metodų analizė. 2002, balandis. Prieiga per internetą: <http://www.soften.ktu.lt/~tamugied>

10 Priedai

10.1 Publikacija

Balandytė M., Jašinskaitė J., Nemuraitė L., Tamulis G. Šiuolaikinės žmogiškųjų išteklių valdymo sistemų kūrimo tendencijos. Iš: Informacinės technologijos 2003: konferencijos pranešimų medžiaga, Kaunas 2003 sausio 28-29 d.. Kaunas, 2003, p. XIV-47 - XIV-52.

10.2 Pranešimas konferencijoje

Balandytė M., Jašinskaitė J., Nemuraitė L., Tamulis G. Integruotos žmogiškųjų išteklių valdymo sistemos modelis // Integruotos projektavimo sistemos: konferencijos pranešimas, Kaunas 2002 birželio 17 d.