KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Birutė Dūdonytė

MERODE metodikos taikymas informacinėms sistemoms kurti

Magistro darbas

Darbo vadovė
doc. Rita Butkienė

KAUNAS
2005

TURINYS

1 ĮVADAS

Modeliavimo metodikos yra skirtos modeliuoti informacinės sistemos projektui. Suprojektavus informacinę sistemą, lengviau nusakyti jos realizavimui reikalingus resursus, programavime galima išvengti dviprasmiškumų. Labiausiai paplitusi ir modeliavimo standartu patapusi yra UML modeliavimo kalba. Tačiau yra ir kitų modeliavimo metodikų, kurios siūlo savo privalumus. Metodikas galima naudoti kiekvieną atskirai, o galima ir kombinuoti, iš kiekvienos pasirenkant tai, kas yra tinkamiausia kuriamam projektui.

Šio darbo tikslas – ištirti MERODE modeliavimo metodiką, siūlančią griežtą modelių patikrą ir detalesnius iš modelių gaunamus klasių aprašus. Šiuo belgų mokslininkų pasiūlytu metodu sumodeliuojamas šeimos klinikos informacinės sistemos modelis bei jis išbandomas praktiškai programuojant.

- Antrame skyriuje pasirenkamos analizės priemonės, analizuojama kuo skirsis šeimos klinikos veikla ją kompiuterizavus nuo dabartinės situacijos, šeimos klinikos poreikiai, analizuojamos egzistuojančios panašios sistemos. Pasirenkama duomenų valdymo sistema ir realizavimo priemonės. Pasirenkamos projektavimo metodai ir priemonės.
- Trečiame skyriuje pateikiama projekto dalis: techninė užduotis ir detalūs kuriamos sistemos modeliai, specifikacijos, aprašymai suprojektuoti UML ir MERODE metodikomis.
- Ketvirtame skyriuje aprašytas eksperimentinis tyrimas. Palyginimamas programos išeitinis tekstas, gautas MERODE metodika ir Rational Rose įrankiu.

2 ANALIZĖS DALIS

2.1 Tyrimo sritis, objektas, problema, darbo aktualumas

Magistriniame darbe tiriamos MERODE modeliavimo metodikos pritaikymo galimybės kuriant šeimos klinikos informacinę sistemą. Naudojant MERODE modeliuojamas sistemos projekto modelis, realizacijos modelis ir pradiniai programos klasių aprašai.

Tyrimo sritis: MERODE metodikos praktinis taikymas.

Tyrimo objektas: MERODE metodika.

Sprendžiamos problemos: MERODE metodikos praktinis pritaikymas: šeimos klinikos informacinės sistemos modeliavimas, naudojant MERODE metodiką, projekto praktinis įgyvendinimas.

2.2 Analizės metodų ir priemonių parinkimas

2.2.1 Metodų pasirinkimas

2.2.1.1 MERODE motyvacija ir apžvalga

2.2.1.1.1 Sprendžiamos problemos

Ilgą laiką programinės įrangos modeliavimo metodai buvo orientuoti arba vien į objektus, arba vien į įvykius. Tačiau tikrovėje procesai ir duomenys tarpusavyje tampriai susiję objekto sąvokoje. Be to, dauguma metodų sistemos specifikacijai naudoja natūralios kalbos ir pusiau formalaus grafinio žymėjimo mišinį. Tokie žymėjimai gali būti naudojami neformalioms analizėms, bet griežta patikra ir atestavimas yra beveik neįmanomi. Dar viena problema kyla dėl to, kad programinės įrangos realizavimas įtakoja daugelį modeliavimo technologijų. Orientacija į objektus pirmiausia atsirado programavime. Tie patys principai buvo pritaikyti modeliavime, idėjos neperžiūrėjus iš esmės. Analizė turėtų tik aprašyti dalykinę sritį, bet neįtakoti galimų problemos sprendimo variantų. Objektiškai orientuotos sistemos paprastai aprašomos kaip tarpusavyje bendraujančių objektų rinkinys. Augant projekto apimčiai, objektų modeliai yra linkę išsiplėsti iki sudėtingų tarpusavyje bendraujančių tinklų. Nepaisant daugelio objektiškai orientuotos sistemos teikiamų pranašumų, sukurtą didelę sistemą sunku palaikyti.

MERODE buvo sukurtas siekiant išspręsti šias problemas: savybių, garantuojančių specifikacijos kokybę, trūkumą; objektiškai orientuotų analizės priemonių polinkį į realizaciją; aukšto lygio agregavimo priemonių trūkumą taikant objektiškai orientuotą

metodiką [7]. Tai įtakojo pagrindines MERODE ypatybes: modeliu pagrįstą sistemos kūrimo metodiką, abstraktesnes objektų sąveikos sąvokas, formalų visų priemonių aprašymą.

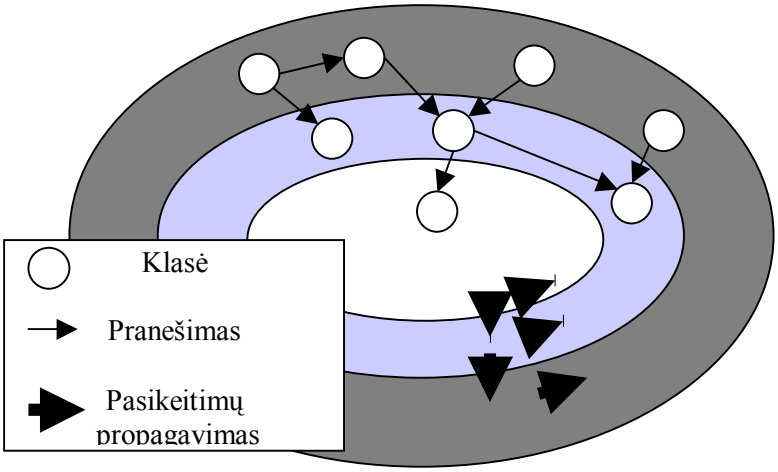
2.2.1.1.2 Modeliu grįstas sistemos kūrimas

Modeliu grįstas projektavimas vadovaujasi idėja, kad informacinė sistema sudėtinga tiek, kiek sudėtinga projektuojama dalykinė sritis [7]. Informacinių sistemų projektuotojai, teikiantys pirmenybę realaus pasaulio modelio, o ne norimos sistemos funkcionalumo projektavimui, geriau susitvarko su projektuojamos sistemos sudėtingumu.

Analizės modelis – tai abstrakcija to, ką sistema turi daryti, o ne to, kaip ji tai turi daryti. Objekto modelis, sukurtas analizės fazėje apibrėžia objekto klases, jų bruožus, metodus ir tarpusavio sąveiką. Bendrai paėmus, objektinės metodikos nesuskirsto specifikuotų objektų klasių į kategorijas. Tačiau šis suskirstymas yra reikalingas, jei norime valdyti ir palaikyti specifikaciją (ir realizaciją).

Norint sistemiškai suskirstyti objektų klases, reikia atsižvelgti į tam tikras specifikacijos savybes bei jos prigimtį. Pavyzdžiui, interfeiso reikalavimai keičiasi daug dažniau nei veiklos modelis. Geriau objektų klases sugrupuoti į sluoksnius, atsižvelgiant į jų kategoriją, nei turėti vientisą objektų rinkinį. Vidinio sluoksnio objektai stabiliausi, o išoriniai – labiau linkę keistis. Klasės gali naudotis tik arčiau vidaus, bet ne arčiau išorės nuo jų esančių klasių paslaugomis. Tai užtikrintų, kad išoriniuose sluoksniuose vykdomi pakeitimai nepaveiktų arba mažai paveiktų vidines klases. Į vidinį sluoksnį įdėjus stabilias klases, jos gali išlikti nepakitusios keičiant klases, esančias išoriniam sluoksnyje (Pav. 1).

Norint gauti tokį sluoksniuotą objektų rinkinį, specifikacija taip pat turėtų būti sluoksniuota.



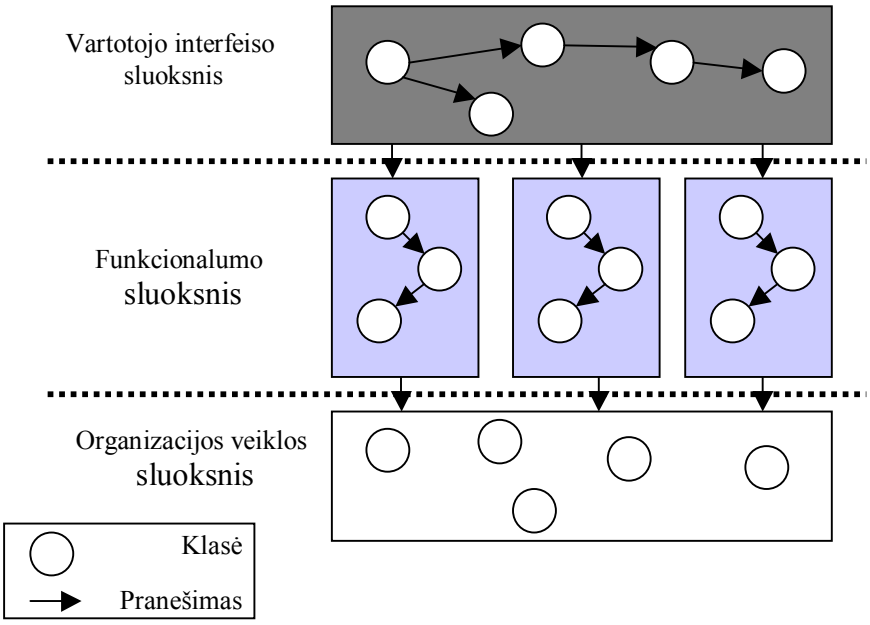
Pav. 1. Objektai sluoksniuose.

2.2.1.1.3 Specifikacijos sluoksniavimas

MERODE metodu kuriamos sistemos turi natūralią sluoksnių struktūrą grupuojant specifikacijas pagal objekto aspektus [7]. Kai kurios specifikacijos kyla iš esminių verslo reikalavimų. Kitos specifikacijos paprastai susijusios su esama informacine sistema ir kyla iš daugiau administracinio tipo funkcijų, tokių kaip duomenų įvedimas, ataskaitų generavimas ir pan. Pirmojo tipo specifikacijos sudaro organizacijos veiklos sluoksnį, apie jį esantis funkcionalumo sluoksnis sudarytas iš antro tipo specifikacijos. Jis teikia duomenų įvedimo ir išvedimo paslaugas.

Dauguma dabartinių programinės įrangos projektavimo metodų neatskiria organizacijos veiklos nuo funkcionalumo. Jie paprastai veiklos objektui priskiria tiek veiklos atributus ir paprogrames, tiek įvedimo ir išvedimo procedūras. MERODE toks grupavimas negalimas specifikavimo stadijoje. Jis gali atsirasti tik realizavimo stadijoje.

Ant funkcionalumo sluoksnio dedamas vartotojo sąsajos sluoksnis suklijuoja individualias funkcionalumo sluoksnio funkcijas tarpusavyje ir leidžia vartotojui sukelti, įsiterpti ir pratęsti funkcijų veikimą. Gauname labai lanksčią sluoksniuotą programos architektūrą (Pav. 2). Pastebėkite Pav. 2, kad organizacijos veiklos sluoksnio objektai tarpusavyje nebendrauja siųsdami žinutes. Tai parodo MERODE ypatingą požiūrį į objektų sąveiką: organizacijos objektai sąveikauja tarpusavyje tik drauge įtraukti į veiklos įvykius.



Pav. 2. Informacinės sistemos architektūra pagal MERODE.

Šios architektūros nereikėtų maišyti su trijų sluoksnių architektūra paskirstytoms sistemoms. Trijų sluoksnių architektūroje atskiriama vartotojo sąsaja, logika ir duomenys. MERODE architektūroje funkcionalumo sluoksnyje yra tiek programos logika, tiek duomenys ir organizacijos veiklos sluoksnis turi tiek organizacijos objektų duomenis, tiek veiklos logiką.

2.2.1.1.4 Modeliavimas taikant MERODE metodiką

MERODE metodikoje išskiriami du pagrindiniai sistemos kūrimo etapai: specifikavimas ir realizavimas, o specifikavimo etape išskiriamos tokios trys fazės:

1. Organizacijos veiklos modeliavimas,
2. Funkcionalumo modeliavimas,
3. Interfeiso modeliavimas.

MERODE organizacijos veiklos modeliui sudaryti naudojamos trijų tipų priemonės: objektų tipų gyvavimo priklausomybių grafas, objektų-įvykių lentelė ir objektų elgsenos schemos. Gyvavimo priklausomybės grafo dėka užtikrinamas suderinamumas tarp dalinių organizacijos veiklą aprašančių modelių ir tokiu būdu galima pasiekti geresnę specifikacijos kokybę nei atskirai tikrinant sintaksinį atitikimą tarp dalinių schemų. MERODE metodikoje specifikacijos kokybę nusako specifikacijos vidinė darna (skirtingi modeliai modeliuoja skirtingus sistemos aspektus, bet turi persidengiančios semantikos) ir korektiškumas. Griežta ir nedviprasmiškai apibrėžta MERODE modeliuose naudojamų sąvokų sintaksė ir semantika (šiam tikslui panaudota procesų algebra) sudaro sąlygas formaliai patikrinti specifikacijos vidinį suderinamumą bei korektiškumą.

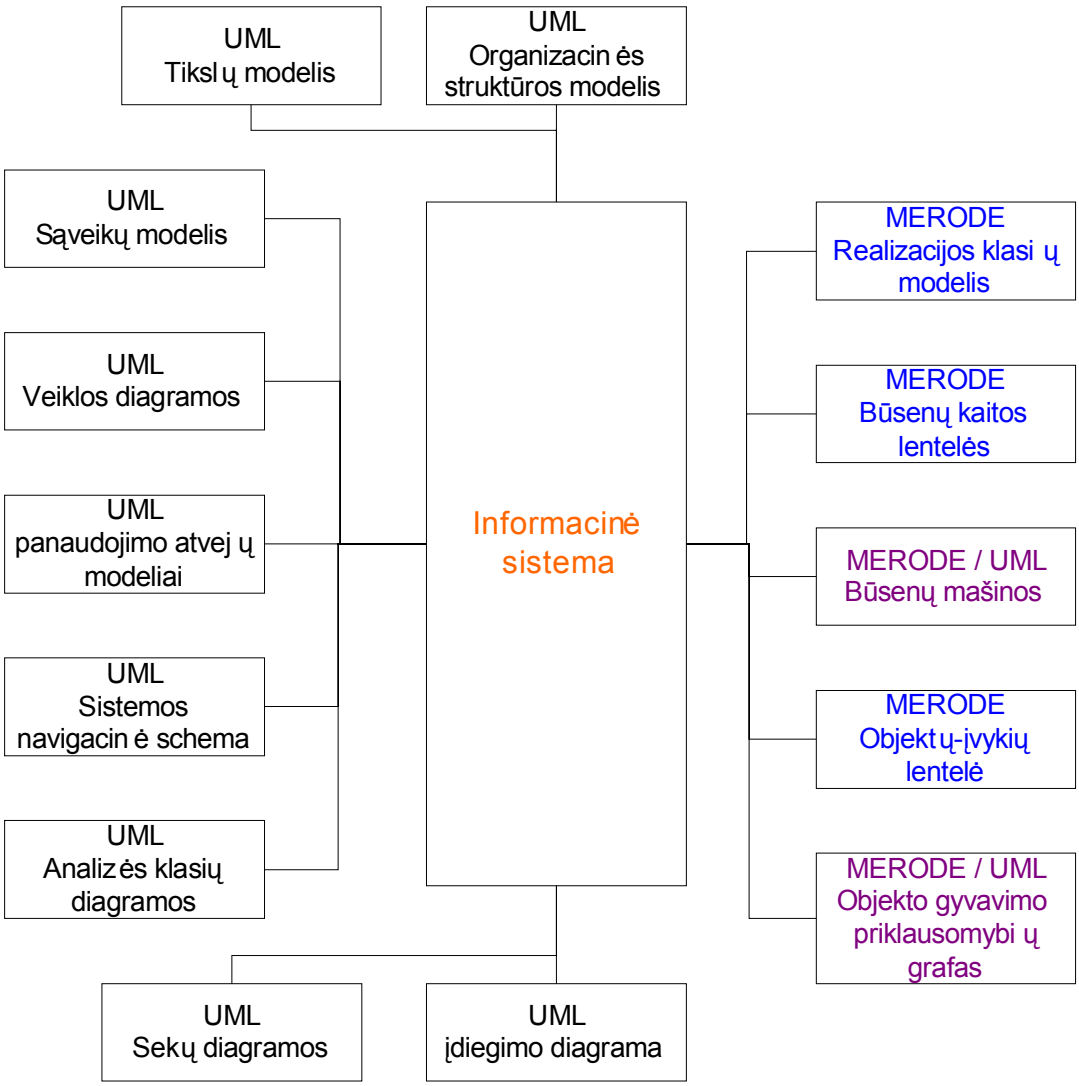
2.2.1.2 UML motyvacija

UML pasirinktas, nes jis:

- Pripažintas standartu – UML pripažintas tarptautiniu standartu ir jau įsisavintas daugelio specialistų. Aiškus – grafiškai pavaizduotas idėjas lengva suprasti;
- Suprantamas – vaizdavimo aiškumas leidžia didesnei auditorijai modelius suprasti;
- Griežtas – modeliavimas, kaip tiksli technologija, užtikrina specifikacijų griežtumą, kas leidžia verslo problemas ir bendrus procesus suprasti vienareikšmiškai.

2.2.1.3 MERODE ir UML suderinimas

UML yra patogus modeliuojant organizacijos veiklą, procesus. UML siūlomi modeliai puikiai tinka prie MERODE metodika sumodeliuotų veiklos modelių. MERODE objekto gyvavimo priklausomybių grafas atitinka ir atstoja UML klasių modelį, o būsenų mašinos atitinka būsenų modelius. Realizacijos modelis taip pat sukurtas pagal MERODE pasiūlytą metodiką. Sistemos projektą papildo objektų-įvykių bei būsenų kaitos lentelės (Pav. 3).



Pav. 3. Informacinę sistemą sudarantys modeliai.

Būsenų mašinas ir objekto gyvavimo priklausomybių grafą galima braižyti naudojant UML notaciją, pagal MERODE metodiką. Objektų gyvavimo priklausomybių grafas atitinka UML klasių diagramą, o būsenų mašinos – būsenų diagramas.

2.2.2 Priemonių pasirinkimas

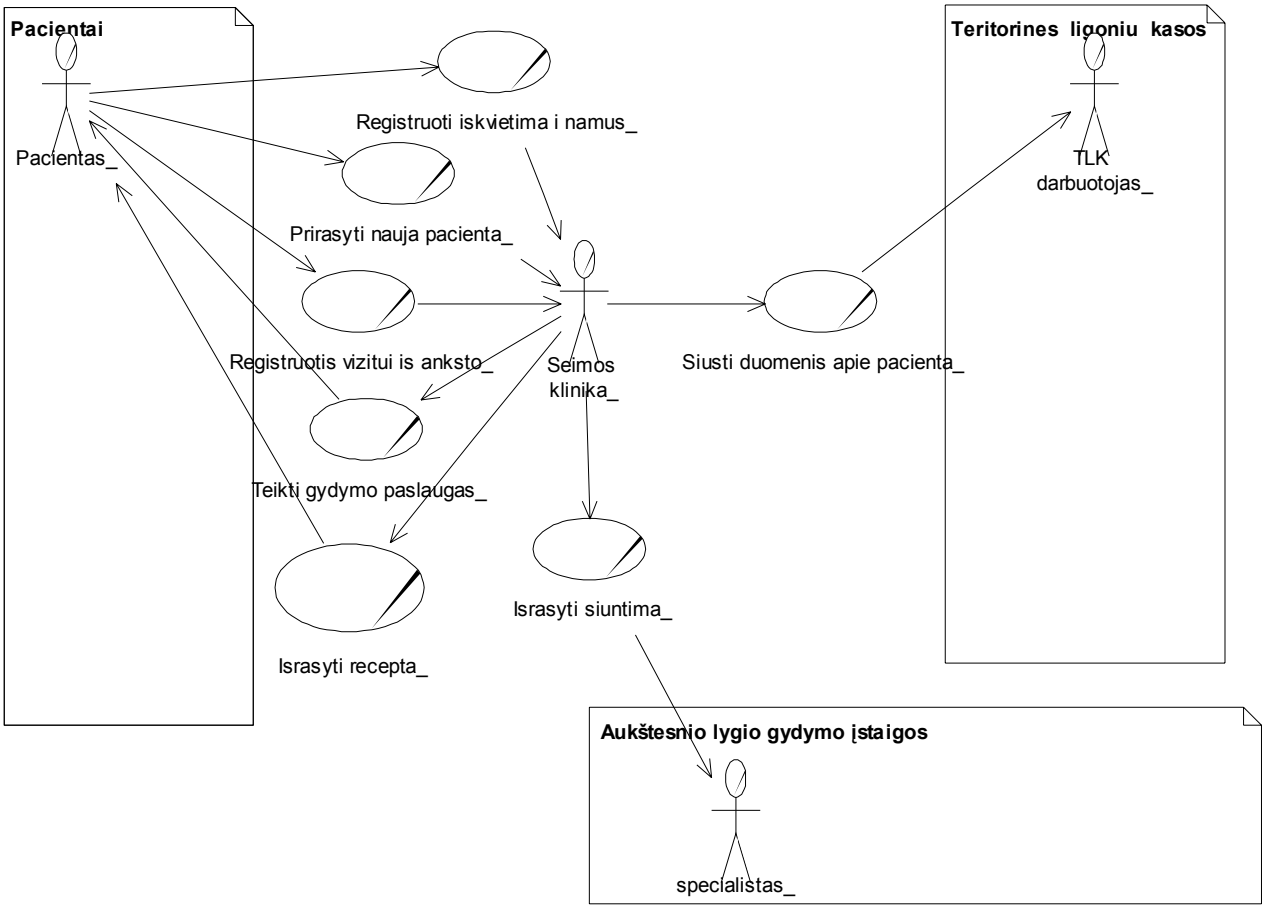
Kol kas nėra sukurto viešai prieinamo programinio paketo, skirto modeliuoti MERODE metodika, kuris ženkliai palengvintų projektavimo bei programinio kodo generavimo iš modelių procesą. Analizės priemone pasirinktas Rational Rose programinis paketas [6], nes jis palaiko UML 2.0, taip pat su juo galima projektuoti MERODE modelius.

2.3 Šeimos klinikos veiklos analizė

2.3.1 Esamos situacijos šeimos klinikoje analizė

2.3.1.1 Organizacijos veiklos sąveikų modelis

Šeimos klinikos veiklos sąveikų modelis pavaizduotas Pav. 4.



Pav. 4. Šeimos klinikos sąveikų modelis.

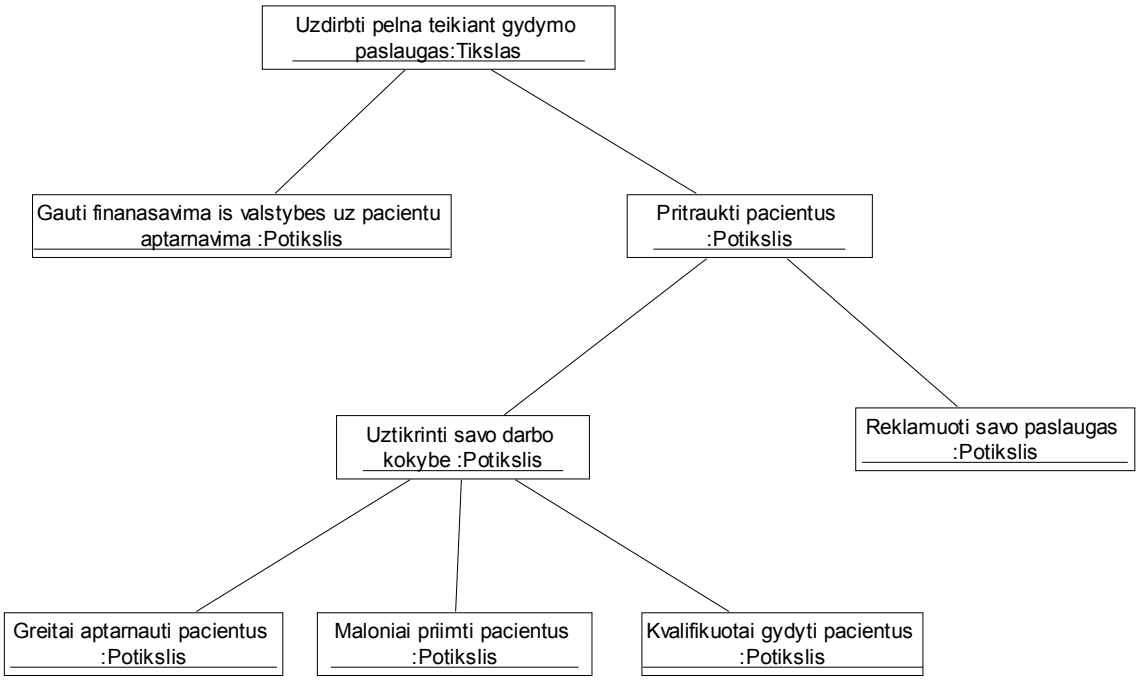
Šeimos klinika (pav. 1) sąveikauja su: pacientais, Teritorinių Ligonių Kasų (TLK) darbuotojais bei kitų stacionarų specialistais ir vykdo tokias funkcijas:

- Prirašyti naują pacientą – pacientas pateikia duomenis apie save ir prašymą būti priregistruotu šeimos klinikoje.

- Registruotis vizitui iš anksto – poliklinikoj prirašytas pacientas registruojasi vizitui pas gydytoją.
- Registruoti iškvietimą į namus – pacientas paskambina į polikliniką, arba kitaip su ja susisiekia, ir užsisako savo gydytojo vizitą į namus.
- Teikti gydymo paslaugas – šeimos klinikoje nustatoma paciento sveikatos būklė ir jei reikia paskiriamas atitinkamas gydymas.
- Išrašyti receptą – pacientui išrašomi receptai vaistams.
- Siųsti duomenis apie pacientą – šeimos klinika siunčia duomenis apie pacientus į Teritorines Ligonių Kasas.
- Išrašyti siuntimą – išrašomi siuntimai į aukštesnio lygio gydymo įstaigą – pas specialistus.

2.3.1.2 Veiklos tikslų modelis

Šeimos klinika yra organizacija, turinti savo tikslus. Šeimos klinikos tikslų hierarchija pavaizduota Pav. 5.

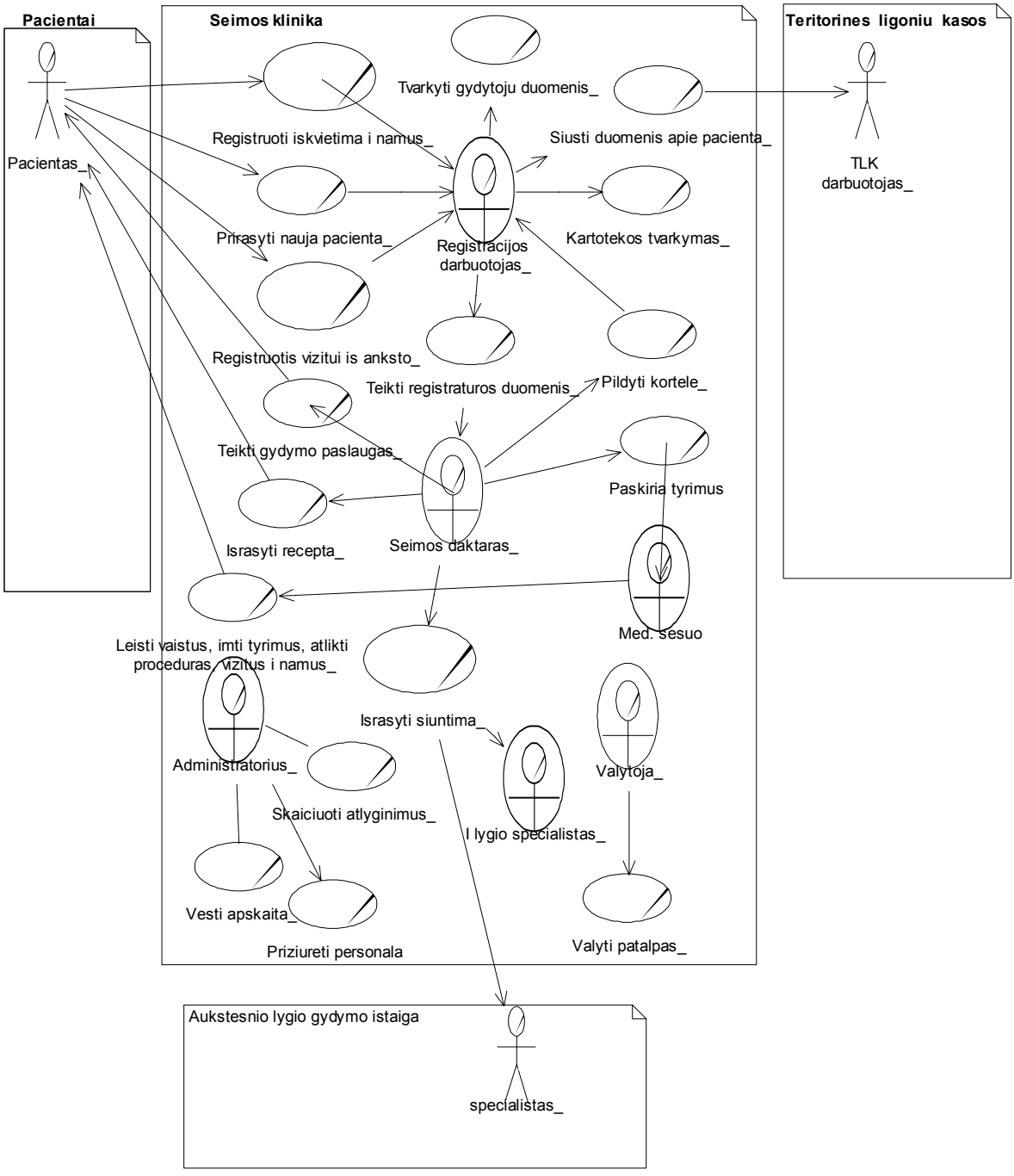


Pav. 5. Šeimos klinikos tikslų modelis.

Šeimos klinikos kompiuterizavimas turėtų padėti įgyvendinti žemiausios hierarchijos tikslus: turėtų pagreitėti aptarnavimas, klinika atrodytų pažangesnė, taip pat pakeltų gydymo kvalifikuotumą, sumažinti žmonių daromų klaidų tikimybę.

2.3.1.3 Veiklos panaudojimo atvejų modelis

Pav. 6 pateikta šeimos klinikos vidinė veikla.



Pav. 6. Šeimos klinikos sąveikų detalizuotas modelis.

Teikti gydymo paslaugas panaudojimo atvejis apima šiuos panaudojimo atvejus: *Paskirti skiepą*, *Paskiepyti*, *Išrašyti siuntimą*, *Peržiūrėti išrašą iš stacionaro*, *Išrašyti nedarbingumo lapelį*, *Paskirti tyrimą*.

Kompiuterizuoti pasirinktas šeimos gydytojo ir registracijos darbuotojo darbas, nes šie darbuotojai tarpusavy susiję, keičiasi duomenų srautais.

Administratorius jau turi savo buhalterinės ir administracinės veiklos kompiuterinę sistemą.

2.3.1.4 Veiklos objektų modelis

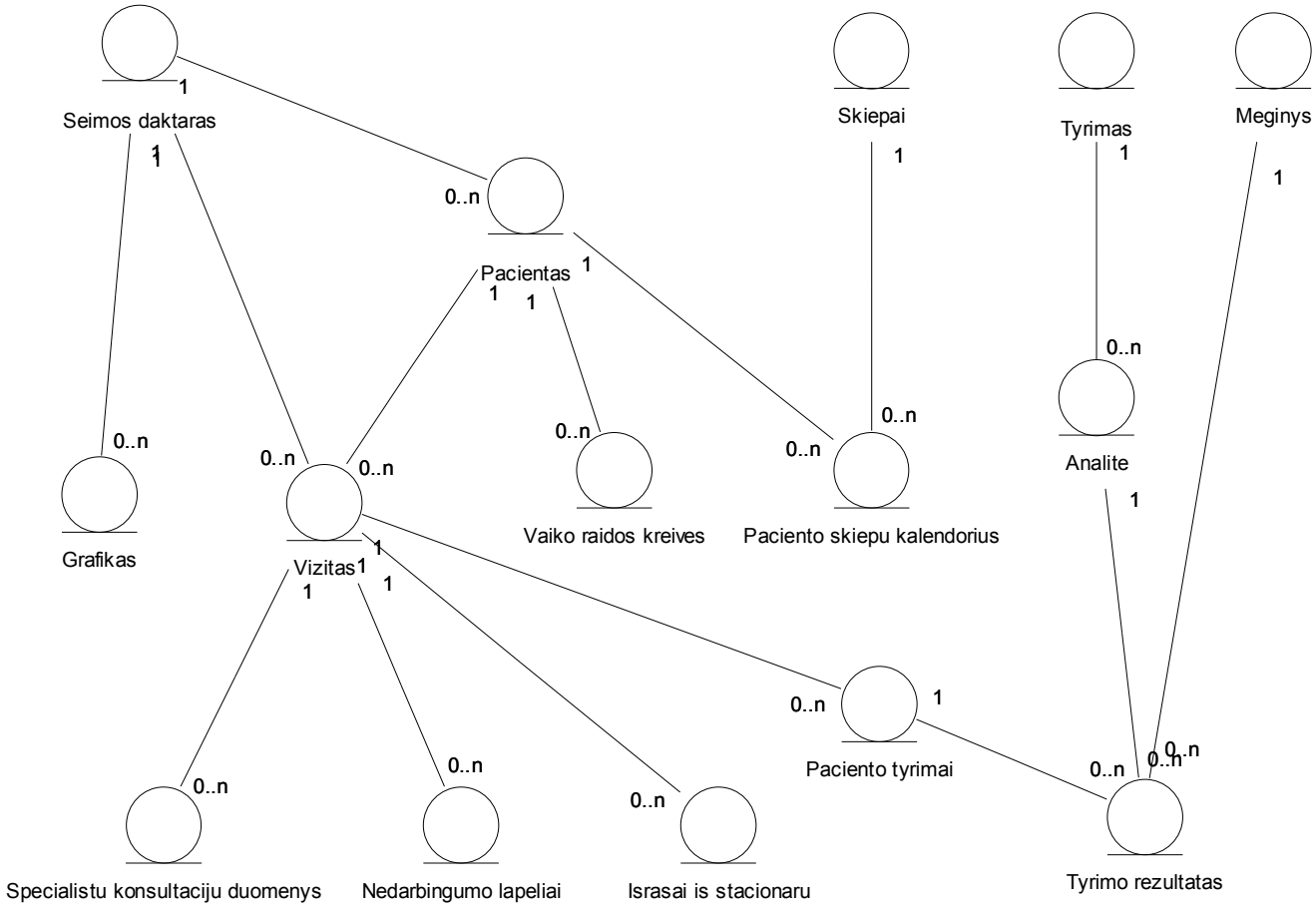
Dalykinės srities aprašas:

Gydytojas įdarbinamas ir dirba tol, kol išeina iš darbo arba būna atleistas. Gydytojui sudaromas darbo grafikas. Pacientas gali būti prirašomas vienam šeimos gydytojui. Vizitas, tai vieno paciento apsilankymas pas vieną gydytoją. Pacientas gali iš anksto užsiregistruoti vizitui pas gydytoją. Vizito laiką ir datą vėliau galima keisti. Pacientas gali ateiti pas gydytoją ir neprisiregistravęs iš anksto. Jis gali prisiregistruoti ir ateiti pas tą gydytoją daug kartų. Gydytojas gali priimti daug pacientų, bet skirtingu laiku. Vizito metu gydytojas gali išrašyti siuntimą specialistui. Siuntimas galioja tam tikrą laiko tarpą, per kurį pacientas pas specialistą gali apsilankyti kelis kartus. Specialistas šeimos gydytojui surašo apžiūros išvadas. Taip pat, jei pacientui tenka gulėtis stacionare (ligoninėje), iš ten šeimos gydytojas gauna išrašo iš stacionaro lapelį, kuriame surašyta anamnezė ir diagnozė. Vizito metu pacientui gali būti išrašomas nedarbingumo lapelis. Jis galioja tam tikrą laiko tarpą. Nedarbingumo lapelis gali būti pratęstas šeimos gydytojo. Pacientams skiriami skiepai. Skiepų kalendorius žymi kada koks skiepas turi būti paskiepytas. Pacientams vaikams vedamos vaiko kreivės, parodančios vaiko vystymosi eigą. Pacientui vizito metu gali būti paskirti tyrimai. Vieną tyrimą sudaro viena arba kelios analitės, kurių kiekis išmatuojamas pateiktame mėginyje. Tyrimo rezultatus peržiūri gydytojas ir daro diagnozes.

Organizacijos veiklos modeliavimas pradedamas nuo objektų tipų identifikavimo. Remdamiesi dalykinės srities aprašu, identifikuojami tokie objektų tipai: pacientas, gydytojas, vizitas, skiepas, skiepų kalendorius, gydytojo darbo grafikas, vaiko kreivės, siuntimas pas specialistą, išrašas iš stacionaro, nedarbingumo lapelis, tyrimas, paskirtas tyrimas, analitė, mėginys, tyrimo rezultatas.. Identifikavus objektų tipus turi būti specifikuotos priklausomybės tarp tų tipų. MERODE metodika šiam tikslui naudoja gyvavimo priklausomybių grafą. Ši priemonė labai panaši į objektų klasių modelį tik turi

papildomas priemones gyvavimo priklausomybei specifikuoti. Kadangi UML klasių modelis neturi specialių priemonių gyvavimo priklausomybei tarp objektų pavaizduoti, tai šiam tikslui bus panaudota klasių išdėstymo tvarka diagramoje, t. y. jei klases B gyvavimas priklauso nuo klasės A gyvavimo, tai klasė B atžvilgiu klasės A diagramoje turi būti pavaizduota žemiau. Nagrinėjamame pavyzdyje objekto Skiepų kalendorius gyvavimas priklauso nuo objektų Skiepas ir Pacientas buvimo. Jei nebūtų bent vieno iš jų, objektas Skiepų kalendorius neegzistuotų.

Šeimos klinikos veikloje dalyvauja objektai, pavaizduoti Pav. 7. Objektai išdėstyti pagal jų priklausomybę nuo vienas kito, kaip siūlo MERODE metodika.



Pav. 7. Šeimos klinikos kompiuterizavimo IS veiklos objektų modelis.

Veikloje dalyvauja trys aktoriai: registracijos darbuotojas, šeimos daktaras ir pacientas.

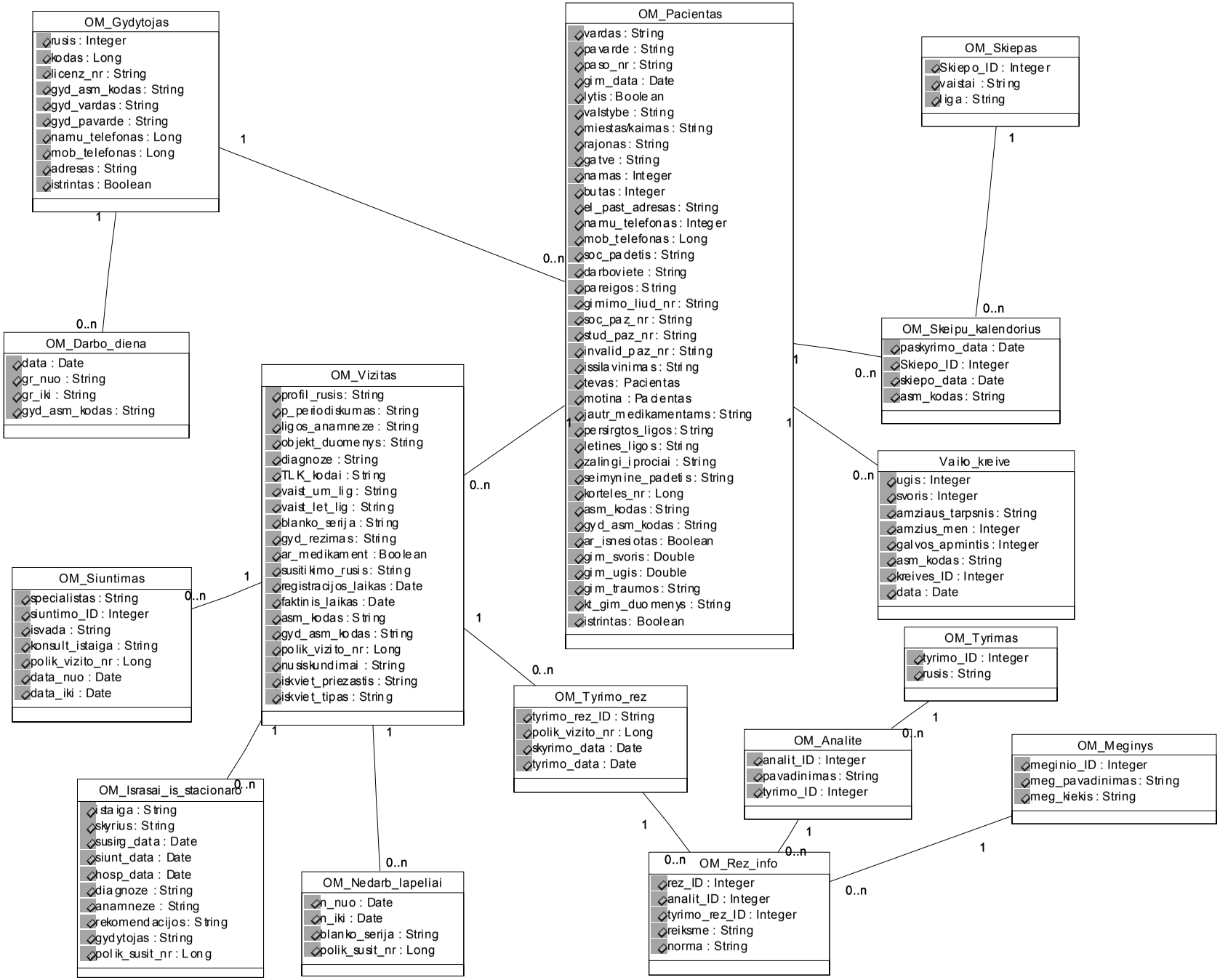
Pacientas pateikia duomenis apie save registracijos darbuotojui ir šis juos registruoja. Registracijos darbuotojas taip pat atsakingas ir už naujų gydytojų registravimą, jo duomenų priežiūrą, darbo grafiko tvarkymą.

Registracijos darbuotojas taip pat vykdo išankstinę registraciją pacientui užsisakius vizitą pas gydytoją. Išankstinė registracija priklauso nuo gydytojo, pas kurį priregistruotas pacientas, darbo grafiko.

Atėjus pacientui sutartu laiku įvyksta vizitas pas šeimos gydytoją, kuris taip pat registruojamas. Pacientas gali ateiti ir be išankstinės registracijos. Jis bus priimtas jei daktaras turės laiko.

Pagal MERODE, pirmiausiai sukuriamas aukščiausiai esantis objektas. Pav. 4 pavaizduotoje sistemoje pirmiausiai gimsta *Šeimos daktaro* objektas. *Paciento* objektą galima sukurti tik jau esant sukurtam *Šeimos daktaro* objektui – pacientą būtina prirašyti kuriam nors šeimos gydytojui. Objektas *Grafikas* taip pat priklauso nuo *Šeimos daktaro* objekto – tai konkretaus šeimos gydytojo darbo grafikas. Analogiškai nuo objekto *Pacientas* priklauso objektai *Vaiko raidos kreivės* ir *Paciento skiepų kalendorius*. Objekto *Vizitas* egzistavimas priklauso nuo dviejų objektų: *Šeimos daktaras* ir *Pacientas*. *Specialistų konsultacijų duomenys*, *Nedarbingumo lapeliai* ir *Paciento tyrimai* objektai gali būti sukurti tik egzistuojant *Vizito* objektui. Taip pat neįvedus *Tyrimo*, *Analitės* bei *Mėginio* negalima užregistruoti *Tyrimo rezultatų*.

Pav. 8 pavaizduotas objektų gyvavimo priklausomybių grafas, kuriame objektai aprašyti detaliau nei Pav. 7 paveiksle.



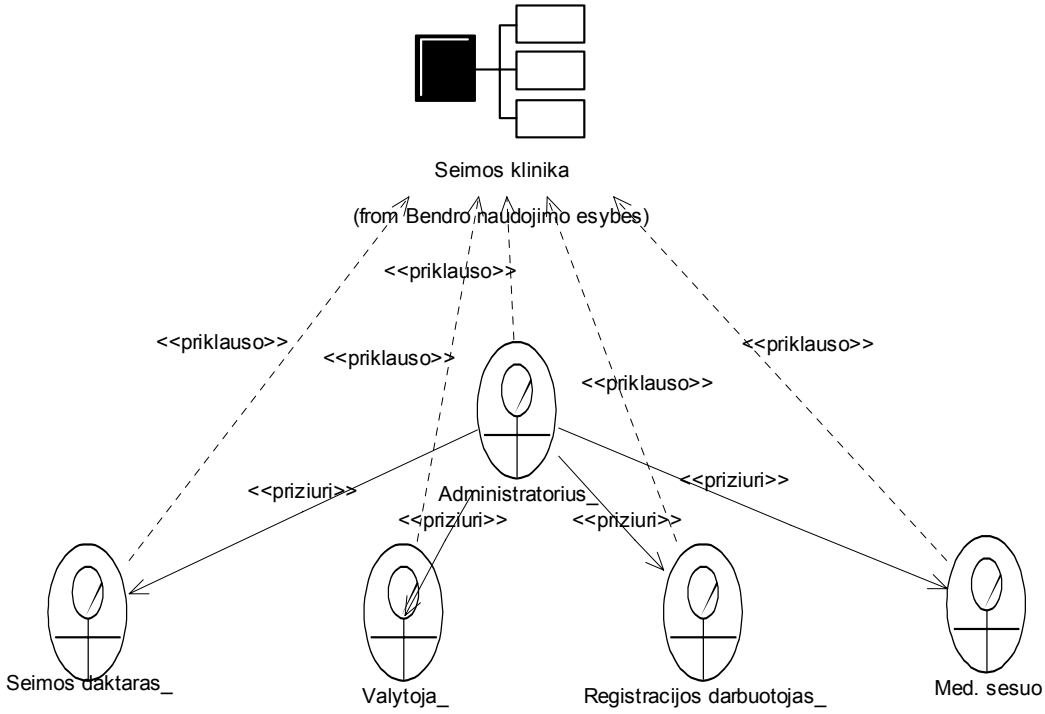
Pav. 8. Objektų gyvavimo priklausomybių grafas.

Reikia paminėti, kad objektų gyvavimo priklausomybių grafas turi tenkinti tokias sąlygas:

- objektų tipo gyvavimas negali priklausyti nuo jo paties,
- gyvavimo priklausomybių grafas yra necikliškas.

2.3.1.5 Organizacinės struktūros modelis

Šeimos klinikos organizacinė struktūra pateikta Pav. 9.



Pav. 9. Šeimos klinikos organizacinė struktūra.

Šeimos kiniką paprastai sudaro administratorius, gydytojai, registracijos darbuotojas, med. seserys, dar būna valytoja.

Administratorius - prižiūri visos klinikos darbą. Jis priima į darbą ir atleidžia darbuotojus, tvarko ūkinius ir finansinius reikalus, sprendžia konfliktus.

Šeimos gydytojas - priiminėja pacientus vizitui, lankosi pas pacientus namuose, nustatinėja ligas, skiria tyrimus, išrašo siuntimus, nedarbingumo lapelius ir t.t.

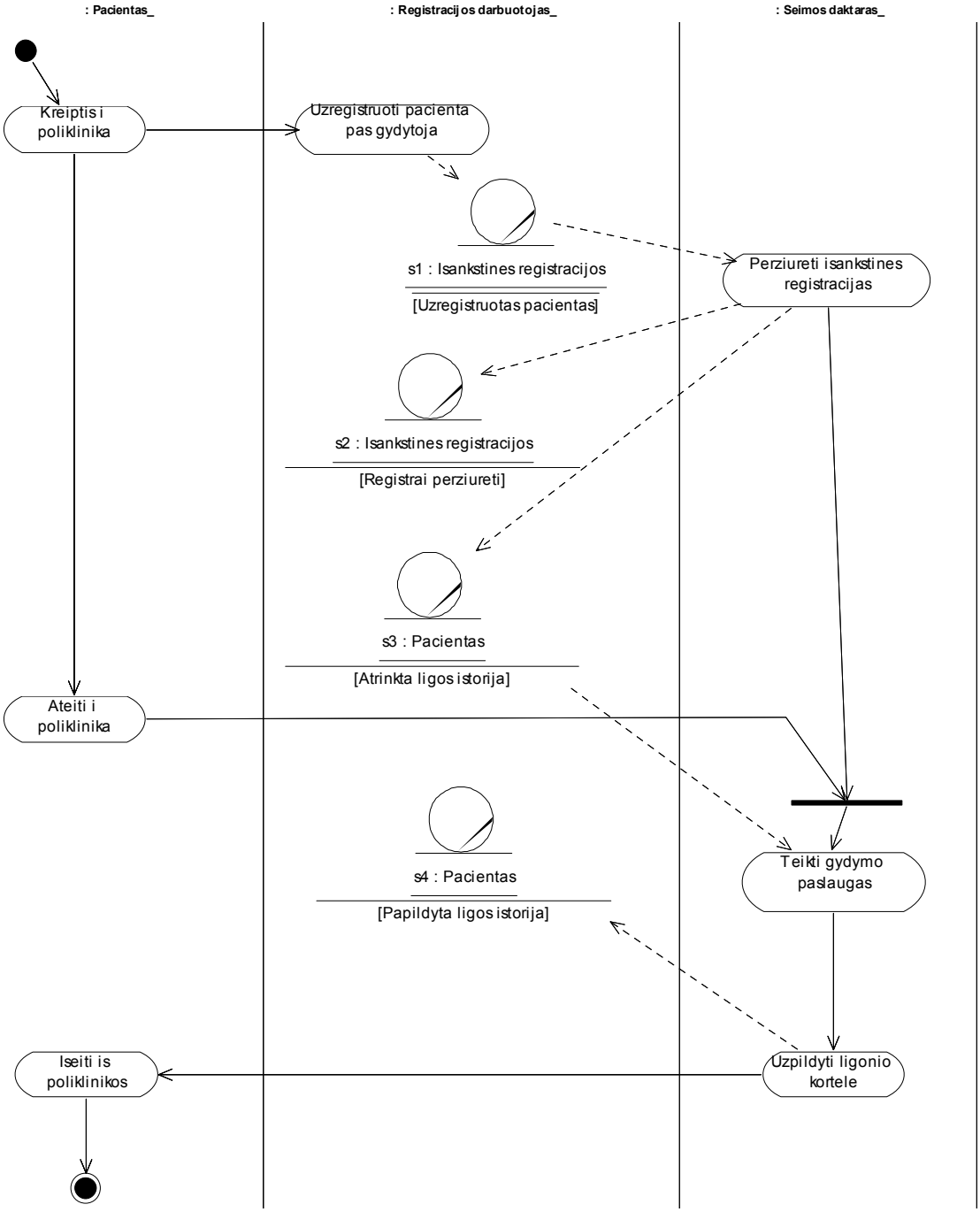
Registracijos darbuotojas – registruoja pacientus, vykdo išankstines registracijas, registruoja naujus gydytojus, jų darbo grafikus.

Med. sesuo – leidžia vaistus, ima tyrimus, atlieka procedūras, lankosi pas pacientus į namus.

Valytoja – valo ir prižiūri klinikos patalpas.

2.3.1.6 Procesų veiklos diagramos

Pav. 10 pavaizduota vizito pas gydytoją veiklos diagrama.



Pav. 10. Vizito pas gydytoją veiklos diagrama.

Išankstinės registracijos – registrai, kuriuose registruojamas pacientas, data ir laikas, kada jis norėtų atvykti vizito pas savo šeimos daktarą.

Į Teikti gydymo paslaugas veiksmą įeina: tyrimų paskyrimas, skiepų paskyrimas ir skiepijimas, siuntimų pas specialistus išrašymas, išrašų iš stacionarų registravimas bei nedarbingumo lapelių išrašymas.

2.3.1.7 Objektų – įvykių lentelė

Svarbus žingsnis modeliuojant organizacijos veiklą – įvykių tipų, kuriuose dalyvauja objektai ir kuriuos turės registruoti kuriamoji sistema, identifikavimas. Tam tikro objekto gyvavimo cikle gali būti daug įvykių, kurie jį sukuria, keičia, užbaigia jo gyvavimą. Kiekvieno objekto gyvavimo cikle turi būti bent du įvykiai: vienas, kuris objektą sukuria, ir vienas, kuris užbaigia jo gyvavimą. Įvykių tipai yra susiejami su objektų tipais. Šiam tikslui yra naudojama objektų-įvykių lentelė. UML neturi tokios priemonės. Todėl bus naudojama tokia lentelė, kokią ją siūlo MERODE metodika (Lentelė 1).

Lentelės struktūra yra tokia: kiekvienam įvykio tipui skiriama eilutė, o kiekvienam objektų tipui – stulpelis. Objektų-įvykių lentelė parodo: 1) kokiuose įvykiuose dalyvauja tam tikro tipo objektai ir 2) kokią įtaką padaro įvykis objektui. Šiems dalykams atskleisti lentelės langeliai užpildomi pagal tokias taisykles:

- jei įvykis sukuria tam tikrą objektą, tai įvykio tipo eilutės ir objekto tipo stulpelio sankirtoje įrašoma raidė C,
- jei įvykis užbaigia tam tikro objekto gyvavimą, tai įvykio tipo eilutės ir objekto tipo stulpelio sankirtoje įrašoma raidė E,
- jei įvykis pakeičia tam tikro objekto savybes (būseną), tai įvykio tipo eilutės ir objekto tipo stulpelio sankirtoje įrašoma raidė M.

Tušti langeliai reiškia, kad objektas nedalyvauja atitinkamuose įvykiuose. Pildant objektų-įvykių lentelę, taip pat laikomas tokių taisyklių:

- *Propagavimo* taisyklė. Jei objekto B gyvavimas priklauso nuo objekto A, tai įvykių, kuriuose dalyvauja objektas B, aibė yra objekto A įvykių aibės poaibis.
- *Įtraukimo* taisyklė. Jei priklausomo objekto B stulpelyje tam tikrose eilutėse įrašyta raidė *C*, tai objekto A, nuo kurio priklauso objektas B, toje pačioje eilutėje turi būti įrašyta arba raidė *C* , arba raidė *M*. Jei priklausomo objekto B stulpelyje tam tikrose eilutėse įrašyta raidė *E*, tai objekto A, nuo kurio priklauso objektas B, toje pačioje eilutėje turi būti įrašyta arba raidė *E* , arba raidė *M*. Jei priklausomo objekto B stulpelyje tam tikrose eilutėse įrašyta raidė M, tai objekto A, nuo kurio priklauso objektas B, toje pačioje eilutėje turi būti įrašyta raidė *M*.
- *Kontrakto* taisyklė. Jei du objektai dalyvauja tame pačiame įvykyje, tai tame įvykyje turi dalyvauti ir bendras priklausomas objektas.

Lentelė 1

Objektų-įvykių lentelė

	Pacientas	Vaiko kreivė	Skiepas	Skiepų kalendorius	Gydytojas	Darbo diena	Vizitas	Siuntimas	Nedarb. lapelis	Išrašas iš stacionaro	Tyrimas	Tyrimo rezult.	Mėginys	Analitė	Rez. info
Prirasyti	C				M										
Idarbinti					C										
Israsyti	E				M										
Iseiti_atleisti					E										
Keisti_pac_duomenis	M				M										
Keisti_gyd_duomenis					M										
Registruoti_vizitui	M				M		C								
Keisti_reg_data	M				M		M								
Ateiti_neprisiregistravus	M				M		C								
Nepasirodyti	M				M		E								
Atvykti_I_vizita	M				M		M								
Baigtis_vizitui	M				M		E								
Ismatuoti_vaika	M	C			M										
Ivesti_nauja_skiepa			C												
Istrinti_skiepa			E												
Paskirti_skiepa	M		M	C	M										
Paskiepyti	M		M	E	M										
Sudaryti_darbo_grafika					M	C									
Israsyti_siuntima	M				M		M	C							
Israsyti_nedarb_lapeli	M				M		M		C						
Ivesti_israsa_is_stacionaro	M				M		M			C					
Ivesti_tyrima											C				
Istrinti_tyrima											E				
Ivesti_megini													C		
Istrinti_megini													E		
Ivesti_analite											M			C	

Istrinti_analite											M			E	
Ivesti_analites_rezultata	M				M		M				M	M	M	M	M
Paskirti_tyrima	M				M		M				M	C	M	M	C
Perziureti_tyrimus	M				M		M				M	E	M	M	E
Apsilankyti_pas_specialista	M				M		M	M							
Baigtis_siuntimo_laikui	M				M		M	E							
Pratesti_nedarb_lapeli	M				M		M		M						
Baigtis_nedarb_lap_laikui	M				M		M		E						

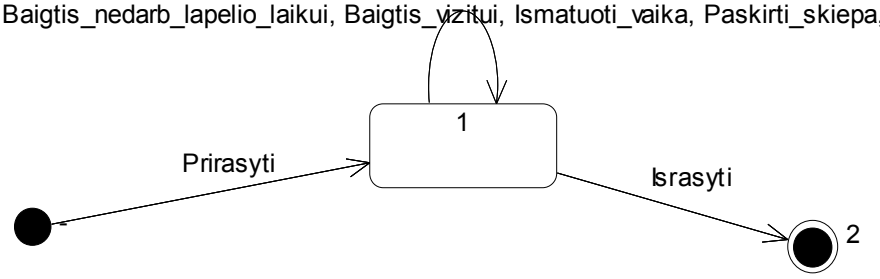
Objektai Vaiko kreivė, Darbo diena ir Išrašas iš stacionaro yra informaciniai – jie neturi kitiems objektams būdingo gyvavimo ciklo. Informaciniai objektai sukuriami saugoti tam tikrą informaciją.

2.3.1.8 Įvykių nuoseklumo apribojimai

Įvykiai objekto gyvenime nevyksta atsitiktine tvarka. Jie turi tam tikrą seką. Pavyzdžiui, pacientas pirmiau užsiregistruoja vizitui, o vėliau į vizitą atvyksta. Atvirkščias variantas būtų netikslingas. Įvykių sekai specifikuoti MERODE metodika siūlo naudoti keletą priemonių: JSD-diagramas, reguliarias išraiškas arba baigtinių būsenų mašinas. UML kalboje objektų būsenų kaitai modeliuoti naudojamos būsenų diagramos, kurios atitinka baigtinių būsenų mašiną. Šiomis diagramomis ir pavaizduoti identifikuotų objektų tipų gyvavimo ciklai. O iš objektų būsenų mašinų gaunami objektų būsenų perėjimai (kiekvienam objektų tipui iš būsenų kaitos modelio sudaromos būsenų kaitos lentelės).

Pacientas:

Keisti_pac_duomenis, Registruoti_vizitui, Keisti_reg_data, Ateiti_neprisiregistravus, Nepasirodyti, Atvykti_i_vizita, Israsyti_siuntima, Israsyti_nedarb_lapeli, Ivesti_israsa_is_stacionaro, Paskirti_tyrima, Ivesti_analites_rez, Perziureti_rez, Apsilankyti_pas_spec, Baigtis_siuntimo_laikui, Pratesti_nedarb_lapeli, Baigtis_nedarb_lapelio_laikui, Baigtis_vizitui, Ismatuoti_vaika, Paskirti_skiepa, Paskiepyti



Pav. 11. Paciento būsenų mašina

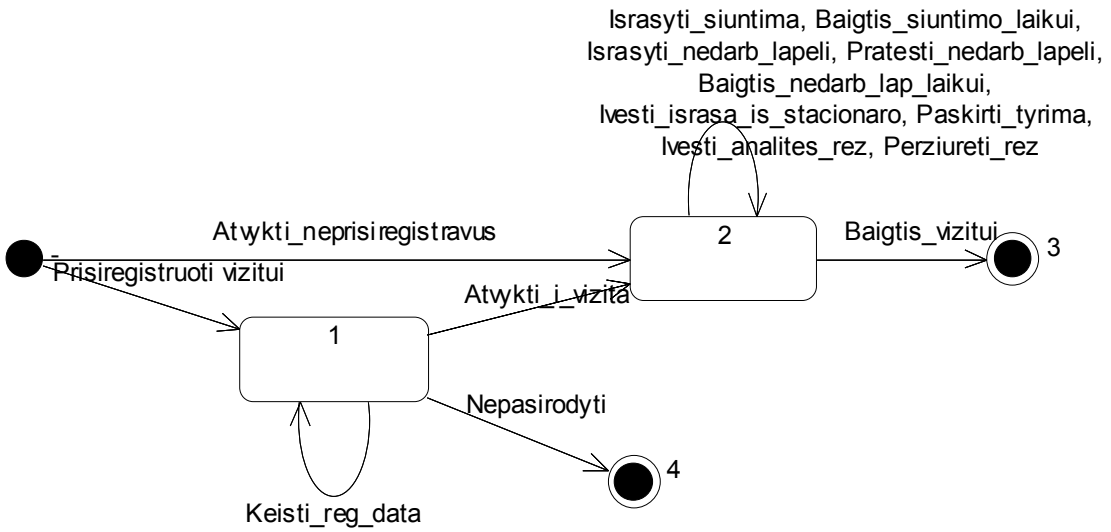
Lentelė 2

Paciento būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Prirasyti	-	1
Keisti_pac_duomenis	1	1
Registruoti_vizitui	1	1
Keisti_reg_data	1	1
Ateiti_neprisiregistravus	1	1
Nepasirodyti	1	1
Atvykti_i_vizita	1	1
Israsyti_siuntima	1	1
Israsyti_nedarb_lapeli	1	1
Ivesti_israsa_is_stacionaro	1	1
Paskirti_tyrima	1	1
Ivesti_analites_rez	1	1
Perziureti_rez	1	1
Apsilankyti_pas_spec	1	1
Baigtis_siuntimo_laikui	1	1
Pratesti_nedarb_lapeli	1	1
Baigtis_nedarb_lap_laikui	1	1
Baigtis_vizitui	1	1

Paskirti_skiepa	1	1
Paskiepyti	1	1
Ismatuoti_vaika	1	1
Israsyti	1	2(F)

Vizitas:



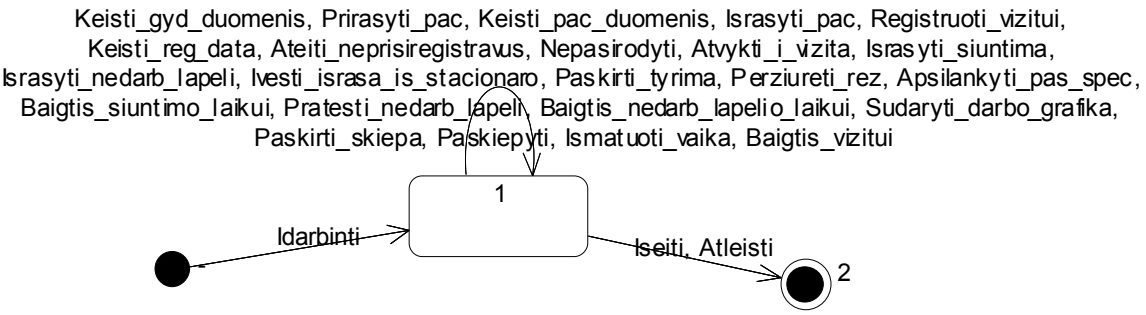
Pav. 12. Vizito būsenų mašina

Lentelė 3

Vizito būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Atvykti_neprisiregistravus	-	2
Registruotis_vizitui	-	1
Keisti_reg_data	1	1
Atvykti_i_vizita	1	2
Israsyti_siuntima	2	2
Israsyti_nedarb_lapeli	2	2
Ivesti_israsa_is_stacionaro	2	2
Paskirti_tyrima	2	2
Ivesti_analites_rez	2	2
Perziureti_rez	2	2
Baigtis_siuntimo_laikui	2	2
Pratesti_nedarb_lapeli	2	2
Baigtis_nedarb_lap_laikui	2	2
Baigtis_vizitui	2	3(F)
Nepasirodyti	1	4(F)

Gydytojas:



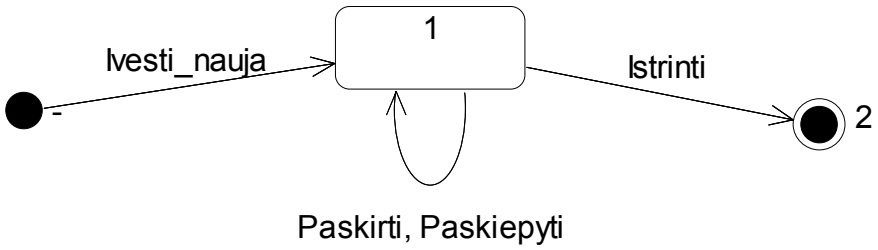
Pav. 13. Gydytojo būsenų mašina

Lentelė 4

Gydytojo būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Įdarbinti	-	1
Keisti_gyd_duomenis	1	1
Prirasyti_pac	1	1
Keisti_pac_duomenis	1	1
Isregistruoti_pac	1	1
Registruoti_vizitui	1	1
Keisti_reg_data	1	1
Ateiti_neprisiregistravus	1	1
Nepasirodyti	1	1
Atvykti_i_vizita	1	1
Baigtis_vizitui	1	1
Israsyti_siuntima	1	1
Israsyti_nedarb_lapeli	1	1
Ivesti_israsa_is_stacionaro	1	1
Paskirti_tyrima	1	1
Ivesti_analites_rez	1	1
Perziureti_rez	1	1
Apsilankyti_pas_spec	1	1
Baigtis_siuntimo_laikui	1	1
Pratesti_nedarb_lapeli	1	1
Baigtis_nedarb_lap_laikui	1	1
Sudaryti_darbo_grafika	1	1
Paskirti_skiepa	1	1
Paskiepyti	1	1
Ismatuoti_vaika	1	1
Iseiti_atleisti	1	2(F)

Skiepas:



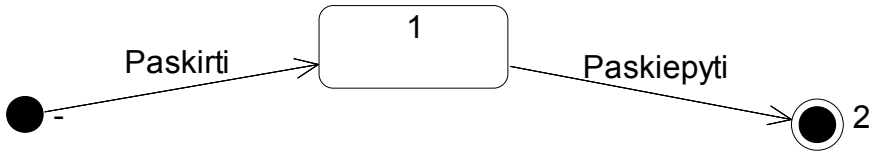
Pav. 14. Skiepo būsenų mašina

Lentelė 5

Skiepo būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Ivesti_nauja_skiepa	-	1
Paskirti	1	1
Paskiepyti	1	1
Istrinti	1	2(F)

Skiepų kalendorius:



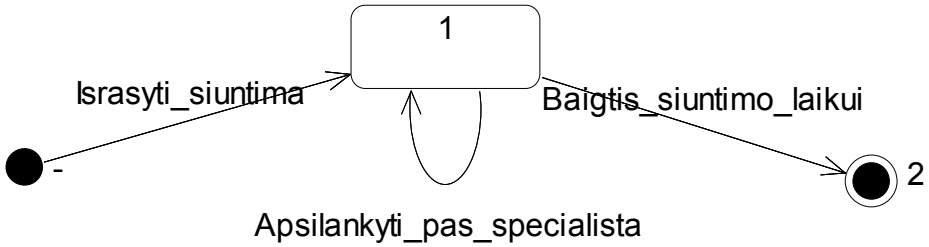
Pav. 15. Skiepų kalendoriaus būsenų mašina

Lentelė 6

Skiepų kalendoriaus būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Paskirti	-	1
Paskiepyti	1	2(F)

Siuntimas:



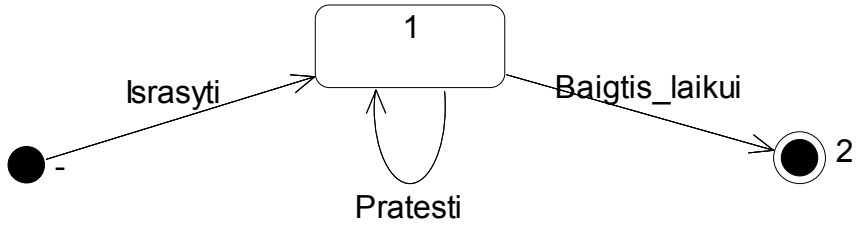
Pav. 16. Siuntimo būsenų mašina

Lentelė 7

Siuntimo būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Israsyti_siuntima	-	1
Apsilankyti_pas_spec	1	1
Baigtis_siuntimo_laikui	1	2(F)

Nedarbingumo lapelis:



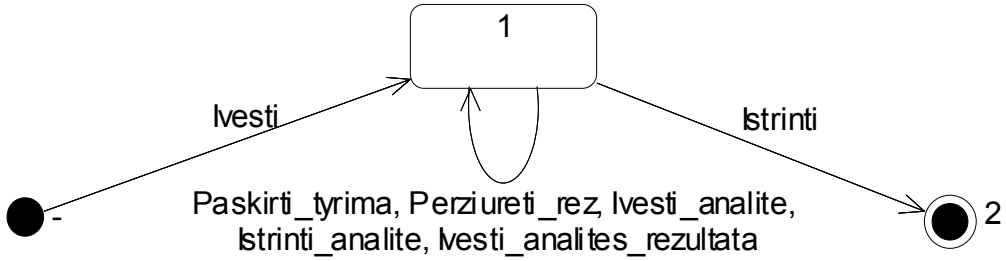
Pav. 17. Nedarbingumo lapelio būsenų mašina

Lentelė 8

Nedarbingumo lapelio būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Israsyti	-	1
Pratesti	1	1
Baigtis_lap_laikui	1	2(F)

Tyrimas:



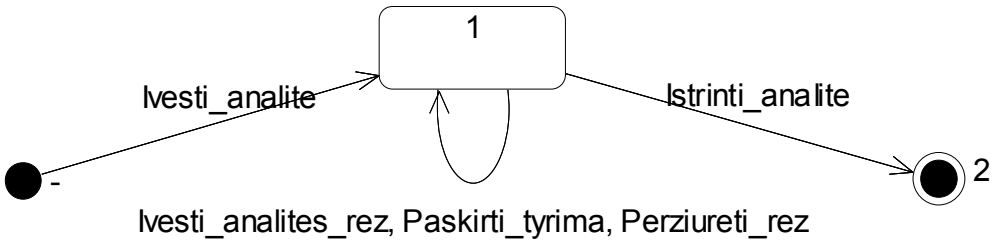
Pav. 18. Tyrimo būsenų mašina

Lentelė 9

Tyrimo būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Ivesti	-	1
Ivesti_analite	1	1
Istrinti_analite	1	1
Ivesti_analites_rez	1	1
Paskirti	1	1
Ivesti_rez	1	1
Istrinti	1	2(F)

Analitė:



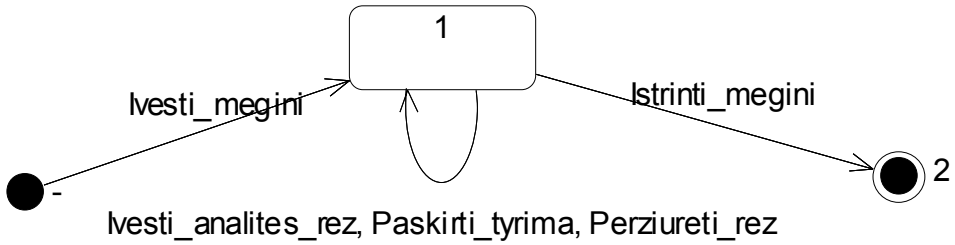
Pav. 19. Analitės būsenų mašina

Lentelė 10

Analitės būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Ivesti_analite	-	1
Ivesti_analites_rez	1	1
Paskirti_tyrima	1	1
Ivesti_tyr_rez	1	1
Istrinti_analite	1	2(F)

Mėginys:



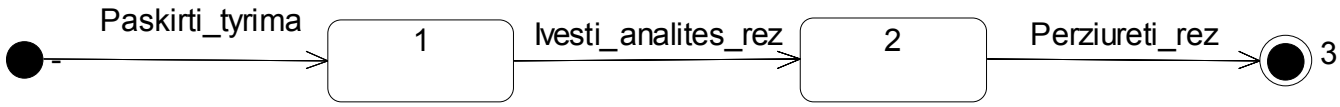
Pav. 20. Mėginio būsenų mašina

Lentelė 11

Mėginio būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Ivesti_megini	-	1
Ivesti_analites_rez	1	1
Paskirti_tyrima	1	1
Ivesti_tyr_rez	1	1
Istrinti_megini	1	2(F)

Tyrimo rezultatas:



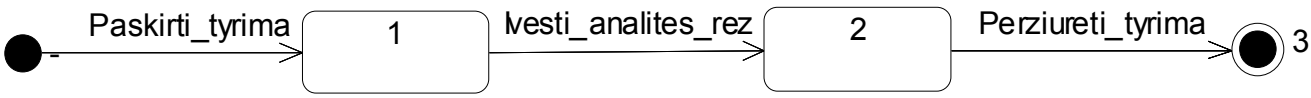
Pav. 21. Tyrimo rezultato būsenų mašina

Lentelė 12

Tyrimo rezultato būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Paskirti_tyrima	-	1
Ivesti_analites_rez	1	2
Perziureti_rez	2	3(F)

Rez_info



Pav. 22. Analitės rezultato būsenų mašina

Lentelė 13

Analitės rezultato būsenų kaitos lentelė

Įvykio tipas	**Iš būsenos**	**Į būseną**
Ivesti_analites_rez	-	1
Paskirti_tyrima	1	2
Perziureti_rez	2	3(F)

2.3.2 Egzistuojančių sistemų sprendimų problemai spręsti lyginamoji analizė

Lietuvoje labiausiai žinomos yra dvi informacinės sistemos, skirtos klinikų vidaus darbui kompiuterizuoti: UAB "SK Impeks MEDICINOS DIAGNOSTIKOS CENTRAS" informacinė sistema „MedIS“ [4] bei UAB “InKOMPas” informacinė sistema „MEDinfo“ [8]. Lentelė 14 lentelėje pateiktas jų atitikimo užsakovo reikalavimams palyginimas.

Lentelė 14

Sistemų „MedIS“ ir „MedInfo“ palyginimas

Programos ypatybės	MedIS	MedInfo
*Ligos istorijos vedimas	+	+
*Pacientų sąrašo pildymas	+	+
*Išankstinė registracija	+	+
*Vaistų paieška	+	-
*Receptų išrašymas	+	-
*Skiepų kalendorius	-	+
*Tyrimai	+	+
Automatinis tyrimų analizavimas	+	-
*Siuntimai	+	-
*Pacientų duomenų paieška	+	+
*Duomenų analizė	+	+
Pacientui suteiktų paslaugų apskaita	+	+
Nuolaidų administravimas	+	+
Paciento mokėjimų apskaita	+	+
*Gydytojų darbo grafikų sudarymas	+	+
Gydytojų bei slaugytojų darbo užmokesčio pagal faktiškai atliktą darbą apskaičiavimas	+	-
Sandėlio apskaita	-	+
*Suderinamumas su TLK programa „Sveidra“	Numatoma naujoje versijoje	+
*Pacientų registracija internetu	Numatoma naujoje versijoje	-

Žvaigždute pažymėti užsakovo reikalavimai.

UAB "SK Impeks MEDICINOS DIAGNOSTIKOS CENTRAS" informacinė sistema „MedIS“ skirta didesnėm įstaigoms, nes ji siūlo stambų tyrimų modelį, kuris mažoms šeimos klinikoms nereikalingas. Be to, „MedIS“ yra brangesnė sistema už “InKOMPas” informacinę sistemą „MEDinfo“. Pastaroji naudoja nemokamą duomenų serverį MySQL Server.

Sistemos „MEDinfo“ pagrindinis trūkumas yra tai, kad joje nėra vaistų paieškos ir receptų išrašymo mechanizmo.

2.4 Projekto tikslas ir jo pagrindimas, kokybės kriterijų apibrėžimas

Projekto tikslas yra išbandyti MERODE galimybes realios sistemos kūrime. MERODE metodu modeliuojamai sistemai būdinga griežta modelių patikra bei programinio išeities teksto pilnumas.

Nauja sistema turėtų tenkinti užsakovo reikalavimus, pateiktus Lentelė 14-oje lentelėje, būti lengvai suprantama ir nebrangi. Tiriant Lietuvos rinką, rasti du produktai, beveik visiškai įgyvendinantys vartotojo reikalavimus. Užsakovas, pasidomėjęs „MedIS“ ir „MEDinfo“ sistemomis, nusprendė, kad priimtinesnė būtų „MEDinfo“, nes ji turi mažiau perteklinių funkcijų bei vykdo tokią svarbią funkciją kaip skiepų kalendorius. Informacinė sistema „MedIS“ taip pat turi norimų privalumų: vaistų paiešką bei receptų išrašymą. Užsakovas negavo demonstracinių programų versijų, dėl to negalėjo sistemų išbandyti praktiškai.

2.5 Projektavimo metodų, priemonių parinkimas

Projektavimui pasirinkti MERODE ir UML metodai. MERODE metodas siūlo griežtos patikros ir atestavimo galimybę, gauti klasių aprašai kur kas detalesni nei generuojant komerciniais CASE įrankiais, tačiau funkcionalumas, vartotojo sąsajos ir sistemos diagramos paprasčiau realizuojamos taikant UML.

2.6 Realizavimo priemonių pasirinkimas

2.6.1 Duomenų bazių valdymo sistemos pasirinkimas

Duomenų bazės valdymo sistemos pasirinkimui naudotas [5] šaltinis. Reikia įvertinti duomenų bazių charakteristikas bei kainą. Duomenų bazės valdymo sistema turi būti patikima, nes bus laikomi svarbūs duomenys, o taip pat pigi, nes ją naudos nedidelės įmonės – šeimos klinikos. Serverio tipo duomenų bazių valdymo sistemų palyginimas pateiktas Lentelė 15 lentelėje.

Lentelė 15

Duomenų bazių valdymo sistemų palyginimas

	Oracle 9i	MS SQL Server 2000 PS2	Sybase ASE	DB2 Universal Database 7.2
Daug konfigūravimo	+	-	+	-
Turtingi valdymo įrankiai	+	+	+	+
Vieta HDD	>1.5 GB	95-270 MB	>42 MB	205 MB
Grafinė aplinka	+	+	+	+
Vartotojai su skirtingom teisėm	+	+	+	+
Suteikti atskiras teises lentelei ar stulpeliui	-	-	+	-
Indeksai	+	+	+	+
Pagalbiniai vedliai	+	+	+/-	+
Perspėja iškilus problemai	+	-	-	-
Parodo sistemos charakteristikas	Real-time	+	+	-
DB dydžio planavimas	+/-	+	-	-
Java palaikymas	+	-	+	+
.NET integracija	-	+	-	-
Pritaikyta Web servisams	-	+	-	+
XML palaikymas	+	+	+	+
Įrankių paprastumas / suprantamumas	+	+	-	-
Priklauso nuo Windows Cluster serviso	-	+	+	-
Turi užklausų analizatorių	+	+	+	+
Suderinta su kito tipo informacijos saugyklom	+	+	+	+
Saugomos procedūros	+	+	+	+
Užklausų saugojimas (cache)	-	-	-	-
Trigeriai	+	+	+	+
Tarpplatforminis bendravimas	+	+	+	+
Transakcijos	+	+	+	+
Duomenų dubliavimas	+	+	+	+
Operacijų vykdymo stabilumas	+	+	-	+
Operacijų vykdymo greitumas	+	+	-	+
DB patikimumas	+	+	+	+
Kaina	$40000 per CPU	$19999 per CPU	$3995 per server	$20000 per CPU

Pasirinkta Oracle duomenų bazių valdymo sistema, nes ji priimtina kokybės atžvilgiu ir jau įdiegta daugelyje šeimos klinikų.

2.6.2 Programavimo kalbos pasirinkimas

Pasirinktas Web sprendimas, nes užsakovas pageidauja, kad kai kurias operacijas galima būtų atlikti internetu (pvz., išankstinė registracija, iškvietimų į namus peržiūra ir pildymas).

Microsoft Inc. sukūrė naują technologiją skirtą Web sprendimams kurti – ASP.NET. Ši technologija pranašesnė už ASP šiais aspektais ([2] ir [1] šaltiniai):

- *Suderinamumas*

ASP.NET sukurti puslapiai veiks šalia ASP sukurtųjų. Nereikėtų jaudintis, kad senesnės versijos programų nebepalaikys naujesnė IIS versija.

- *Kompiliuojamas programos kodas*

ASP.NET galima programuoti kompiliuojamomis kalbomis, tokiomis kaip Visual Basic, C++ arba C#. Net VBScript ir JavaScript kalbos dabar yra kompiliuojamos.

- *COM objektai*

Naudojant ASP, COM sukeldavo tokių problemų kaip komponentų registravimas, serverių perkrovinėjimas. Dabar yra NGWS (Next Generation Windows Services) ir galima komponentų failus tiesiog nukopijuoti kur reikia ir jie veiks.

- *XML konfigūracija*

NGWS platforma, visa metabazė ir konfigūracijos informacija saugoma XML failuose, todėl nebereikia kaitalioti IIS nustatymų. XML failus galima pakeisti ir patalpinti juos į serverį būnant bet kur, net nenaudojant nuotolinio valdymo.

- *Jėga ir lankstumas*

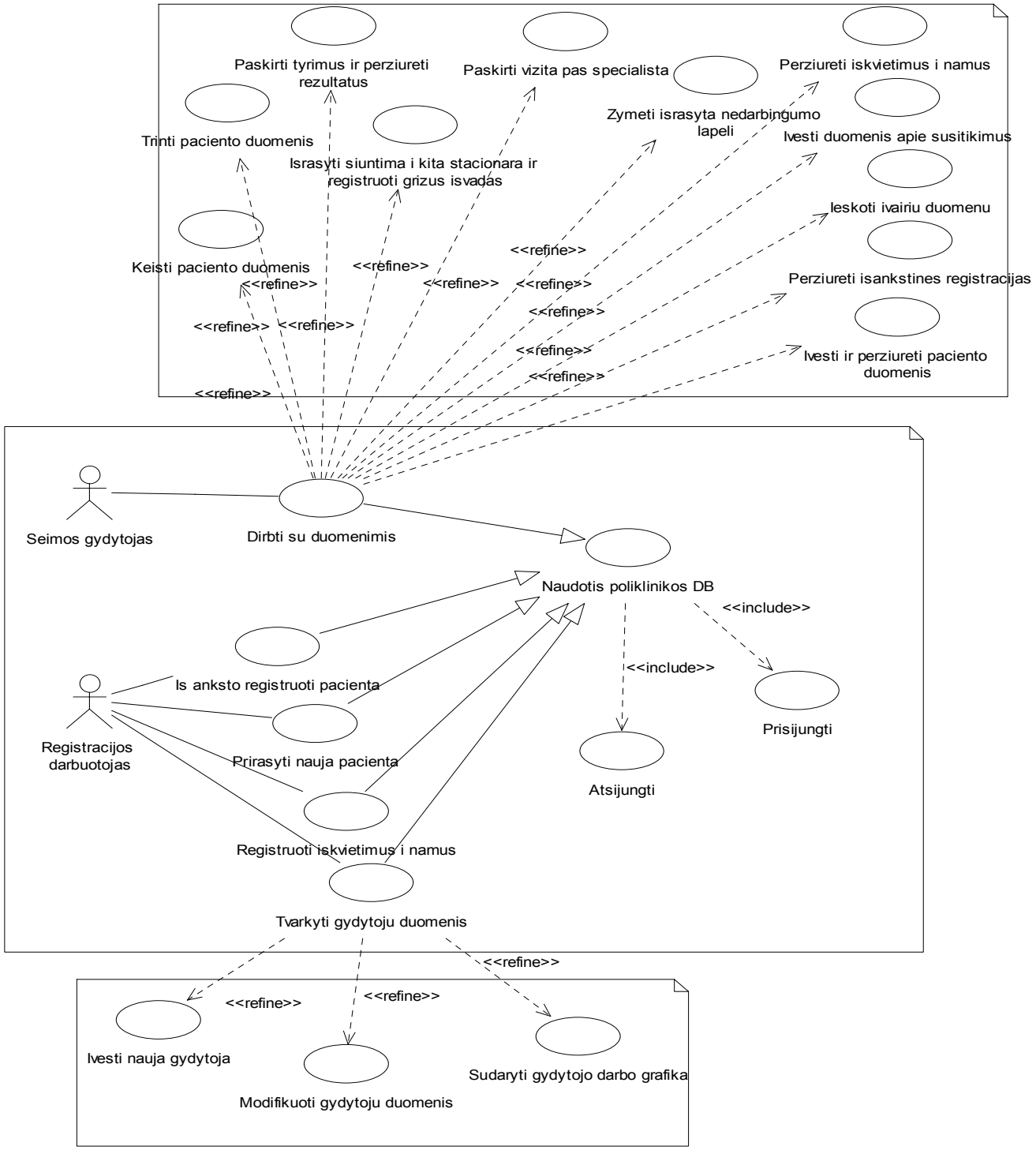
ASP.NET programoms būdinga platformos jėga ir lankstumas. .NET platformos klasių biblioteka, pranešimų apsikeitimų ir duomenų prieigos sprendimai prieinami per internetą. Programa gali būti kuriama galingu Visual Studio įrankiu.

- *Saugumas*

Integruotas Windows autentiškumo patvirtinimas (authentification), konfigūracija kiekvienai programai atskirai užtikrina programos saugumą.

3.1 Šeimos klinikos informacinės sistemos reikalavimų modelis

Pav. 23 pateikiami panaudojimo atvejai šeimos gydytojui ir registracijos darbuotojui.



Pav. 23. Šeimos klinikos kompiuterizavimo sistemos detalizuotas kompiuterizuojamų panaudojimo atvejų modelis

Per magistriniam darbui skirtą laiką nebus spėta visus panaudojimo atvejus realizuoti. Todėl pateikiamos specifikacijos tik tų panaudojimų atvejų, kurie spėti realizuoti.

Panaudojimo atvejų specifikacijos:

Lentelė 16

Panaudojimo atvejis	Prirašyti naują pacientą
Aktorius	Registracijos darbuotojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Pacientas pateikia prašymą būti prirašytas toje klinikoje
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Suvedami paciento duomenys į formą 2. Paspaudžiamas patvirtinimo mygtukas	2.1. Sistema tikrina, ar užpildyti būtinieji laukai 2.2. Sistema įveda paciento duomenis į duomenų bazę 2.3. Sistema praneša apie sėkmingą paciento duomenų įvedimą
Po sąlyga	Pacientas gali būti užregistruotas išankstine registracija pas gydytoją, gydytojas gali pildyti susitikimo su pacientu duomenis, vykdyti paiešką pagal registracinius pacientų duomenis
Alternatyvos (nesėkmės atvejai)	2.1.a Neužpildyti visi būtinieji laukai Sistema išmes pranešimą apie neužpildytus būtinuosius laukus ir grįžta į 1 žingsnį 2.1.b Asmens kodo ilgis ne 11 simbolių Sistema išmes pranešimą apie neužpildytus būtinuosius laukus ir grįžta į 1 žingsnį
Vykdymo variantai	
Veiklos taisyklės	Vartotojas turi įvesti būtinuosius duomenis ir būtinai teisingai įvesti asmens kodą
Specialūs reikalavimai (nefunkciniai)	Duomenys turi saugiai pasiekti duomenų bazę

Įvesti naują gydytoją – analogiškai Lentelė 16 lentelėje pateiktam naujo paciento prirašymui.

Lentelė 17

Panaudojimo atvejis	Iš anksto registruoti pacientą
Aktorius	Registracijos darbuotojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Pacientas turi būti priregistruotas informacinėje sistemoje
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pasirenkama data kalendoriuje 2. Pasirenkamas gydytojas pas kurį nori užsiregistruoti pacientas 3. Pasirenkamas optimaliausias pacientui laikas 4. Įrašomas pacientas 5. Patvirtinamas pasirinkimas įrašymo mygtuku	2.1. Sistema išveda galimas priėmimo valandas pagal gydytojo darbo grafiką 5.1. Sistema įveda duomenis apie išankstinę registraciją į duomenų bazę 5.2. Sistema praneša apie sėkmingai įrašytą išankstinę registraciją
Po sąlyga	Gydytojas galės peržiūrėti išankstines registracijas ir pasiruošti paciento priėmimui

Alternatyvos (nesėkmės atvejai)	2.a Gydytojas pasirinktą dieną nedirba arba neįvestas jo darbo grafikas Sistema informuoja apie neįvestą gydytojo darbo grafiką
Vykdymo variantai	Norimą datą galima pasirinkti kalendoriuje arba po kalendoriumi esančiais išsiskleidžiančiais sąrašais
Veiklos taisyklės	
Specialūs reikalavimai (nefunkciniai)	

Lentelė 18

Panaudojimo atvejis	Registruoti iškvietimus į namus
Aktorius	Registracijos darbuotojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Pacientas, norintis užregistruoti iškvietimą į namus, turi būti įrašytas į informacinės sistemos duomenų bazę
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Iškvietimo į namus duomenys suvedimas į formą 2. Patvirtinimo mygtuko paspaudimas	2.1. Sistema įveda duomenis apie iškvietimą į namus į duomenų bazę 2.2. Sistema praneša apie sėkmingą duomenų įvedimą
Po sąlyga	Iškvietimai į namus gali būti peržiūrimi šeimos gydytojų
Alternatyvos (nesėkmės atvejai)	
Vykdymo variantai	
Veiklos taisyklės	Reikia pasirinkti pacientą, kuris užsisako iškvietimą. Iškvietimas bus priskirtas gydytojui, pas kurį prisirašęs užsisakantis pacientas
Specialūs reikalavimai (nefunkciniai)	

Lentelė 19

Panaudojimo atvejis	Keisti gydytojo duomenis
Aktorius	Registracijos darbuotojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Gydytojas, kurio duomenis norime keisti, turi būti įvestas į duomenų bazę
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pasirenkamas gydytojas, kurio duomenis norime redaguoti 2. Redaguojami duomenys 3. Patvirtinimo mygtuko paspaudimas	1.1. Sistema išveda pasirinkto gydytojo duomenis į formą 3.1. Sistema patikrina ar įvesti būtinieji duomenys 3.2. Sistema įveda duomenis į duomenų bazę 3.3. Sistema praneša apie sėkmingą duomenų įvedimą
Po sąlyga	Galima peržiūrėti pakeistus gydytojo duomenis
Alternatyvos (nesėkmės atvejai)	3.1.a Neužpildyti visi būtinieji laukai Sistema išmes pranešimą apie neužpildytus būtinuosius laukus ir grįžta į 1 žingsnį 3.1.b Asmens kodo ilgis ne 11 simbolių Sistema išmes pranešimą apie neužpildytus būtinuosius laukus ir grįžta į 1 žingsnį
Vykdymo variantai	
Veiklos taisyklės	
Specialūs reikalavimai (nefunkciniai)	Vartotojas turi įvesti būtinuosius duomenis

Lentelė 20

Panaudojimo atvejis	Sudaryti gydytojo darbo grafiką
Aktorius	Registracijos darbuotojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Gydytojas turi būti įvestas į IS duomenų bazę
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pasirinkti datą kalendoriuje 2. Pasirinkti gydytojo, kuriam sudaromas grafikas pavardę 3. Įvedamos darbo valandos 4. Paspaudžiamas patvirtinimo mygtukas	4.1. Sistema patikrina ar įvestos darbo valandos 4.2. Sistema patikrina ar įvestos įmanomos darbo valandos 4.3. Sistema įveda duomenis į IS duomenų bazę 4.4. Sistema praneša apie sėkmingą duomenų įvedimą
Po sąlyga	Galima vykdyti išankstinę pacientų registraciją pas gydytoją toms dienoms, kurioms įvestas darbo grafikas
Alternatyvos (nesėkmės atvejai)	4.1.a Neįvestos darbo valandos Sistema praneša apie neužpildytus laukus Sistema grįžta į 1 žingsnį 4.2.a Įvestos neįmanomos darbo valandos Sistema praneša apie neužpildytus laukus Sistema grįžta į 1 žingsnį
Vykdymo variantai	Norimą datą galima pasirinkti kalendoriuje arba po kalendoriumi esančiais išsiskleidžiančiais sąrašais
Veiklos taisyklės	
Specialūs reikalavimai (nefunkciniai)	Vartotojas turi teisingai įvesti gydytojo darbo valandas

Lentelė 21

Panaudojimo atvejis	Peržiūrėti išankstines registracijas
Aktorius	Šeimos gydytojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Šeimos gydytojas turi būti įvestas į sistemos duomenų bazę
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pasirenkama data kalendoriuje 2. Pasirenkamas gydytojas kurio išankstinė registracija peržiūrima	2.1. Sistema išveda galimas priėmimo valandas pagal gydytojo darbo grafiką 2.2. Sistema išveda pacientų pavardes, užregistruotus galimomis priėmimo valandomis
Po sąlyga	Gydytojas gali eiti tiesiai į susitikimo su pacientu langą
Alternatyvos (nesėkmės atvejai)	
Vykdymo variantai	Norimą datą galima pasirinkti kalendoriuje arba po kalendoriumi esančiais išsiskleidžiančiais sąrašais
Veiklos taisyklės	
Specialūs reikalavimai (nefunkciniai)	

Peržiūrėti iškvietimus į namus– analogiška išankstinės registracijos peržiūrai (žr. Lentelė 21), tik nereikia rinktis datos – rodomi esamos dienos iškvietimai.

Keisti paciento duomenis – analogiška panaudojimo atvejui „Keisti gydytojo duomenis“ (žr. Lentelė 19).

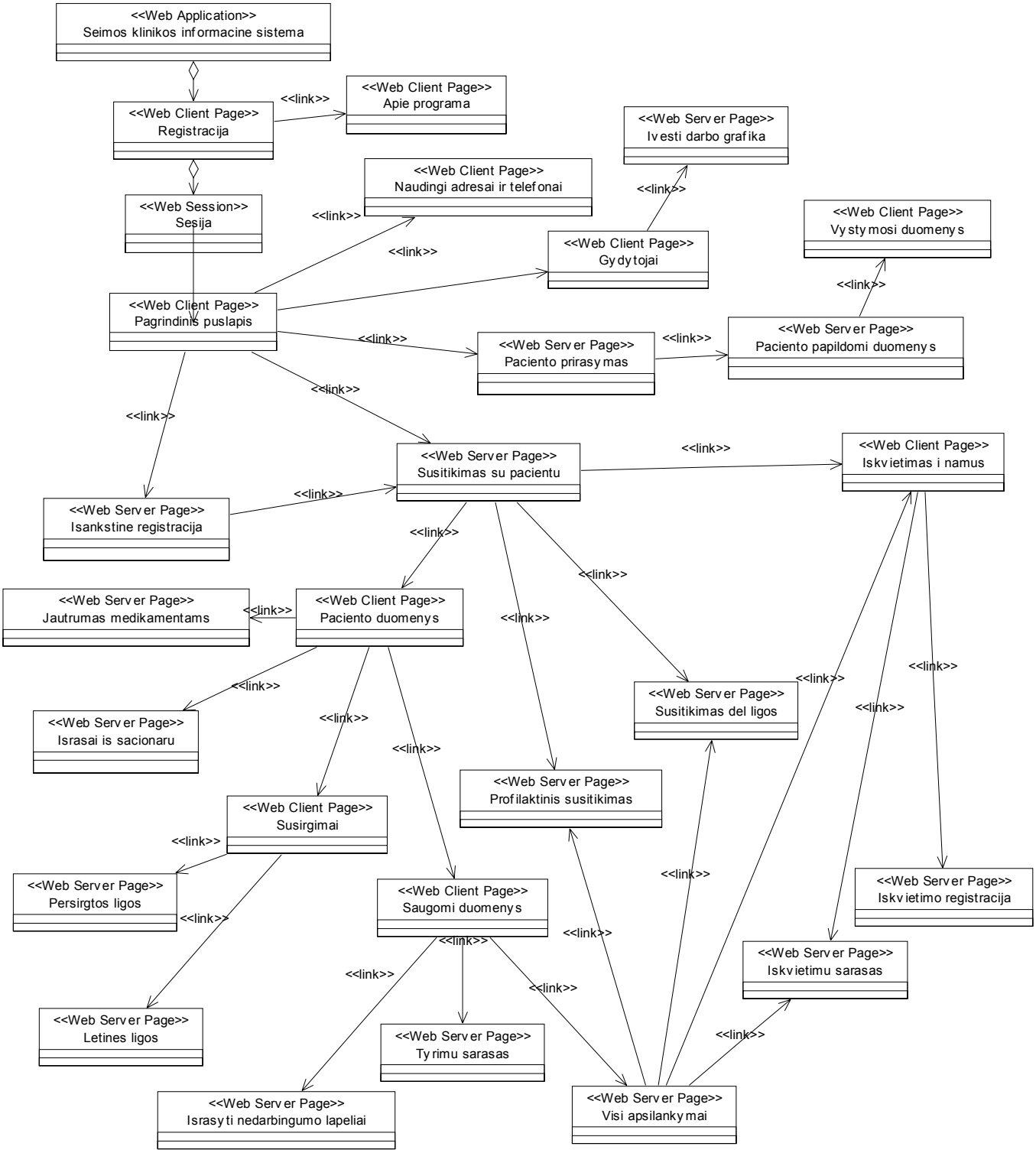
Lentelė 22

Panaudojimo atvejis	Ieškoti įvairių duomenų
Aktorius	Šeimos gydytojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Reikia žinoti bent vieną ieškomo paciento (ar pacientų) savybę
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Užpildomas formos laukas (ar keli laukai), pagal kurį vyks paieška 2. Spaudžiamas paieškos mygtukas	2.1. Sistema tikrina ar užpildytas bent vienas laukas 2.2. Sistema ieško pacientų, kurių atitinkami duomenys sutampa su pateiktais duomenimis 2.3. Sistema pateikia rastų pacientų sąrašą nuorodų pavidalu
Po sąlyga	Nuspaudus pateiktą pasirinkto paciento nuorodą, atidaromas paciento duomenų redagavimo langas
Alternatyvos (nesėkmės atvejai)	2.1.a Neužpildytas nė vienas formos laukas Sistema grįžta į 1 žingsnį 2.2.a Sistema neranda pacientų, kurių duomenys atitiktų ieškomus duomenis Sistema praneša apie nerastus duomenis Sistema grįžta į 1 žingsnį
Vykdymo variantai	Vienu metu galima užpildyti vieną lauką, o galima ir kelis
Veiklos taisyklės	
Specialūs reikalavimai (nefunkciniai)	Turi būti užpildytas bent vienas formos laukas

Lentelė 23

Panaudojimo atvejis	Žymėti išrašytą nedarbingumo lapelį
Aktorius	Šeimos gydytojas
Sistema	Šeimos klinikos informacinė sistema
Prieš sąlyga	Pacientas, kuriam išrašomas nedarbingumo lapelis, turi būti įvestas į IS duomenų bazę
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Užpildoma forma 2. Spaudžiamas įvedimo mygtukas	2.1. Sistema tikrina ar įvesti visi laukai 2.2. Sistema įrašo duomenis į duomenų bazę 2.3. Sistema praneša apie sėkmingai įvestus duomenis
Po sąlyga	Išrašytus nedarbingumo lapelius galima peržiūrėti papildomų duomenų lange
Alternatyvos (nesėkmės atvejai)	2.1.a Neužpildyti visi formos laukai Sistema išveda pranešimą apie neužpildytus laukus Sistema grįžta į 1 žingsnį
Vykdymo variantai	
Veiklos taisyklės	
Specialūs reikalavimai (nefunkciniai)	Turi būti užpildyti visi formos laukai

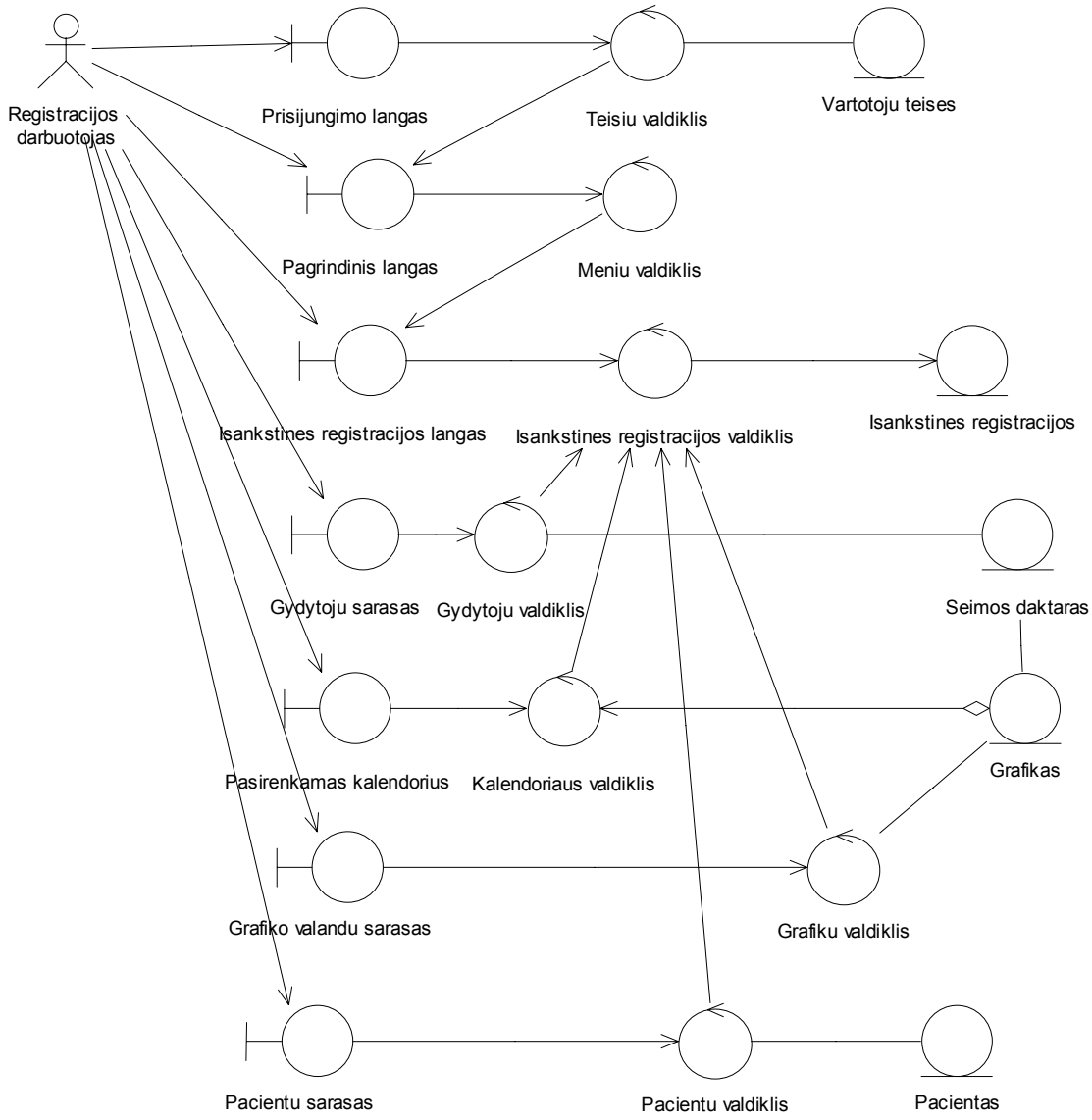
Sistemos navigacinė schema pateikta Pav. 24. Joje matome bendrą internetinių puslapių išsidėstymą ir tarpusavio susietumą.



Pav. 24. Sistemos navigacinė schema.

3.2 Analizės klasių diagramos

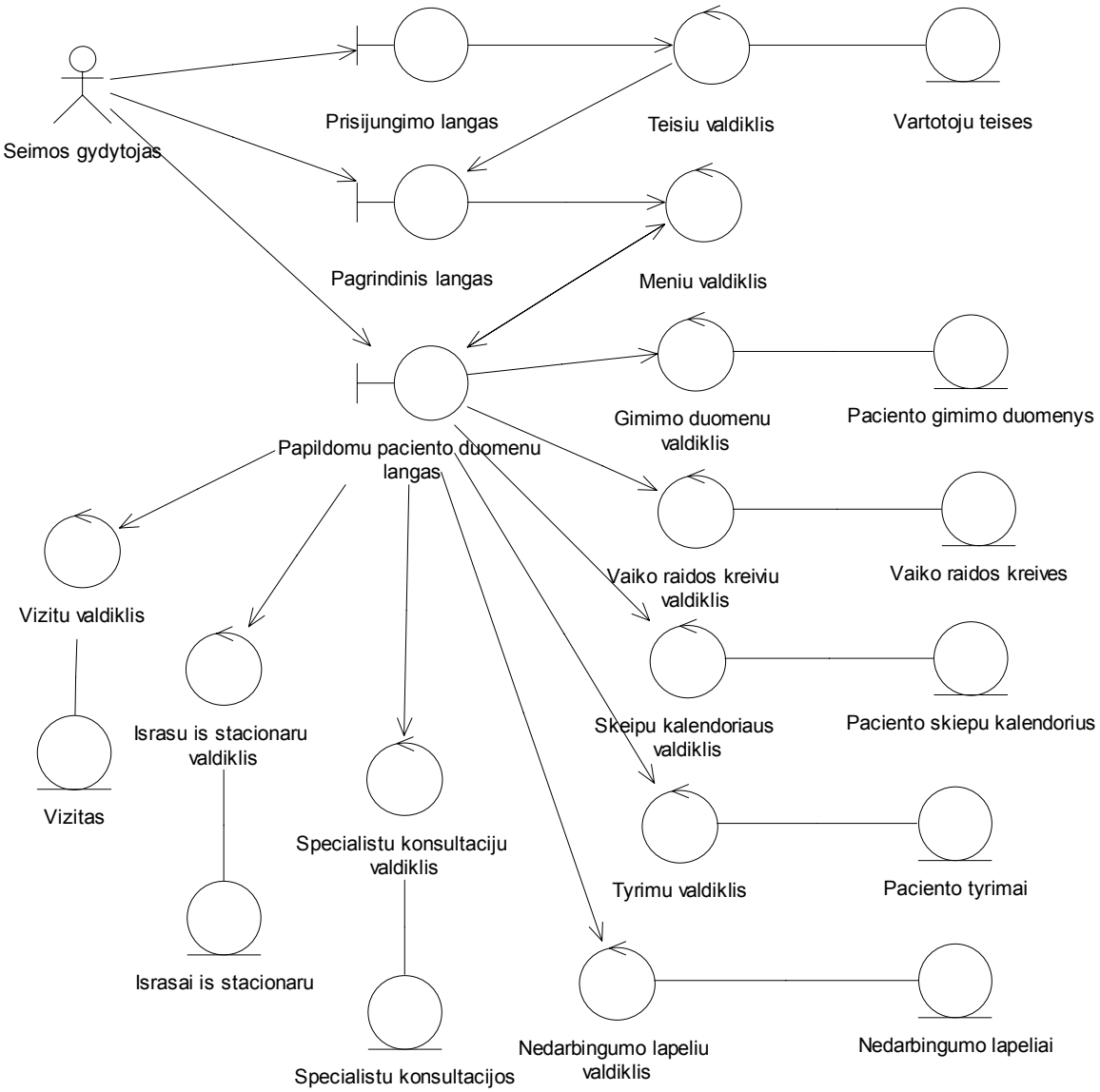
Pav. 25 pavaizduotas išankstinės registracijos analizės klasių diagrama. Joje parodyta kaip registracijos darbuotojas, naudodamasis informacine sistema, registruoja pacientus vizitui pas gydytojus.



Pav. 25. Išankstinės registracijos analizės klasių diagrama

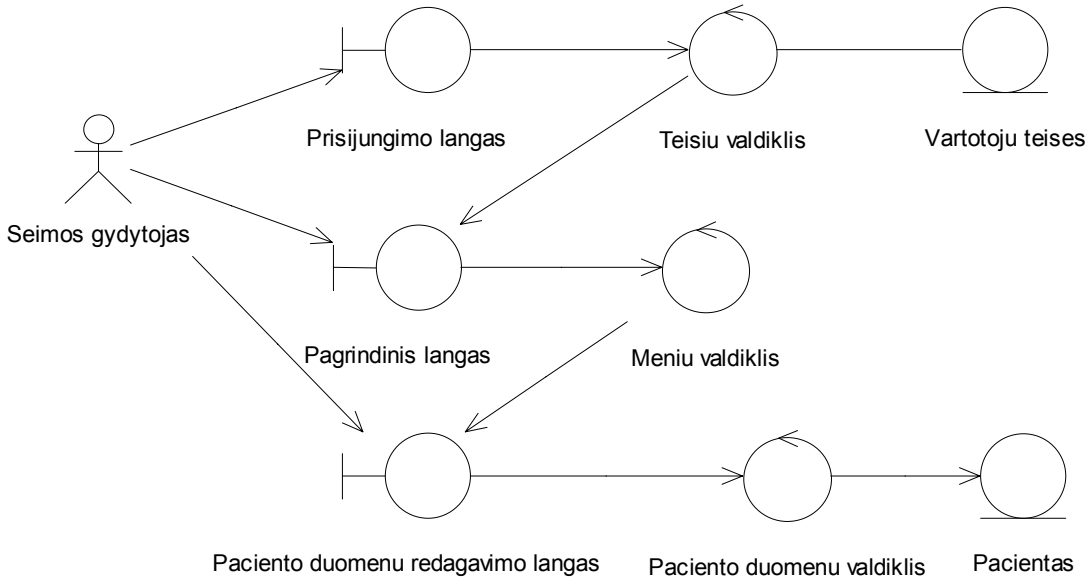
Kalendoriaus valdiklis dar naudojamas gydytojo darbo grafiko sudarymui.

Pav. 26 pavaizduotas šeimos gydytojo paciento duomenų valdymas. Registracijos darbuotojas atsakingas tik už registracinius duomenis, o gydytojas įvedinėja visus likusius. Per papildomų paciento duomenų langą šeimos gydytojas gali įvesti paciento gimimo duomenis, vaiko raidos kreives, skiepų kalendorių, peržiūrėti paskirtus tyrimus ir jų rezultatus, išrašytus nedarbingumo lapelius ir kitus duomenis, sukauptus vizito metu.



Pav. 26. Paciento duomenų įvedimo ir peržiūrėjimo analizės klasių diagrama.

Šeimos gydytojas taip pat gali redaguoti registracijos darbuotojo įvestus registracinius duomenis jei būtų įsivėlusi klaida. Pav. 27 pavaizduota paciento registracinių duomenų redagavimo analizės klasių diagrama.



Pav. 27. Paciento registracinių duomenų redagavimo analizės klasių diagrama.

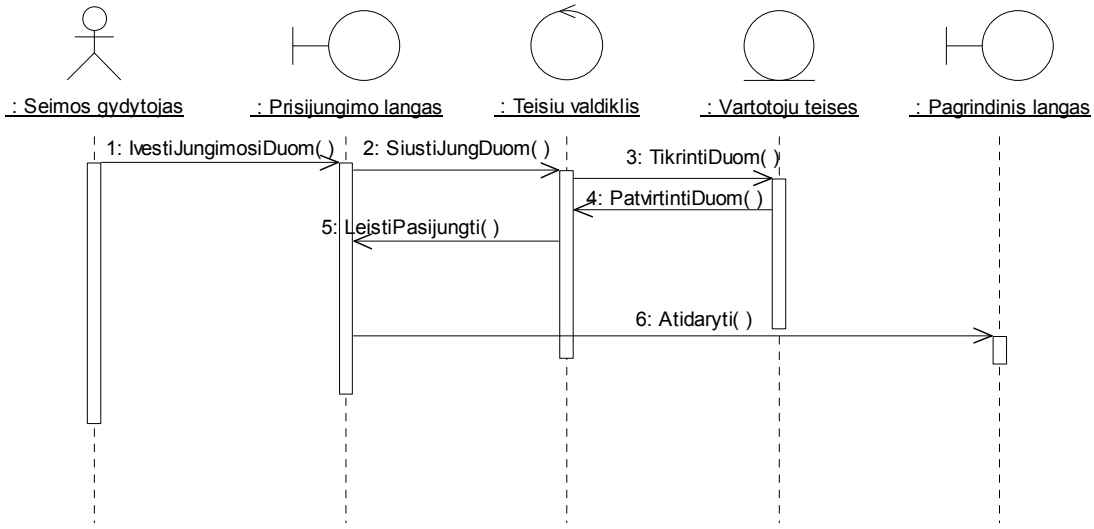
Duomenų apie susitikimą įvedimas, paciento duomenų paieška, išankstinių registracijų bei iškvietimų į namus peržiūros vysta analogiškai, tik šeimos gydytojas pasirenka kitus sistemos puslapius.

Paciento duomenų registravimas, naujo gydytojo prirašymas, gydytojų duomenų modifikavimas, iškvietimų į namus registravimas vyksta analogiškai, tik jį atlieka registracijos darbuotojas ir pasirenkami kiti internetiniai puslapiai.

3.3 Sistemos projektas

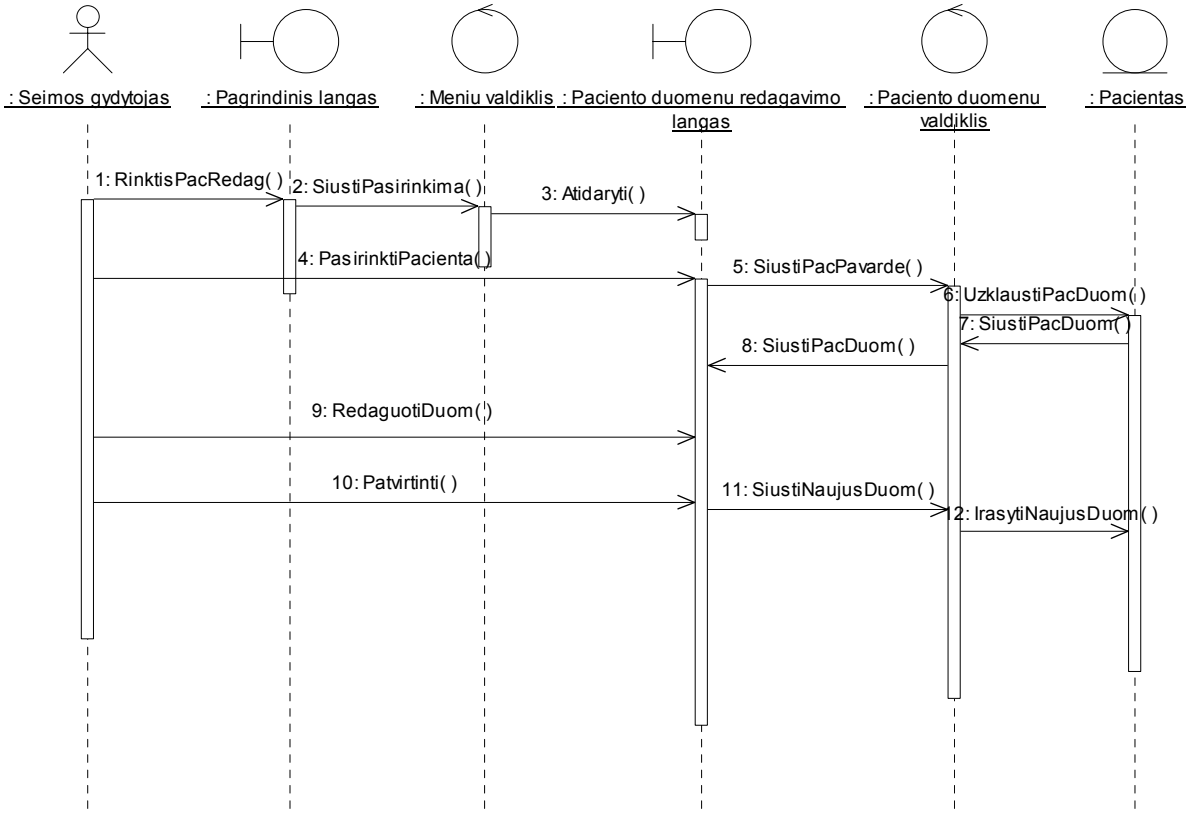
3.3.1 Panaudojimo atvejų sekų (arba bendradarbiavimo) diagramos

Pav. 28 pavaizduotas šeimos gydytojo pasijungimas prie sistemos.



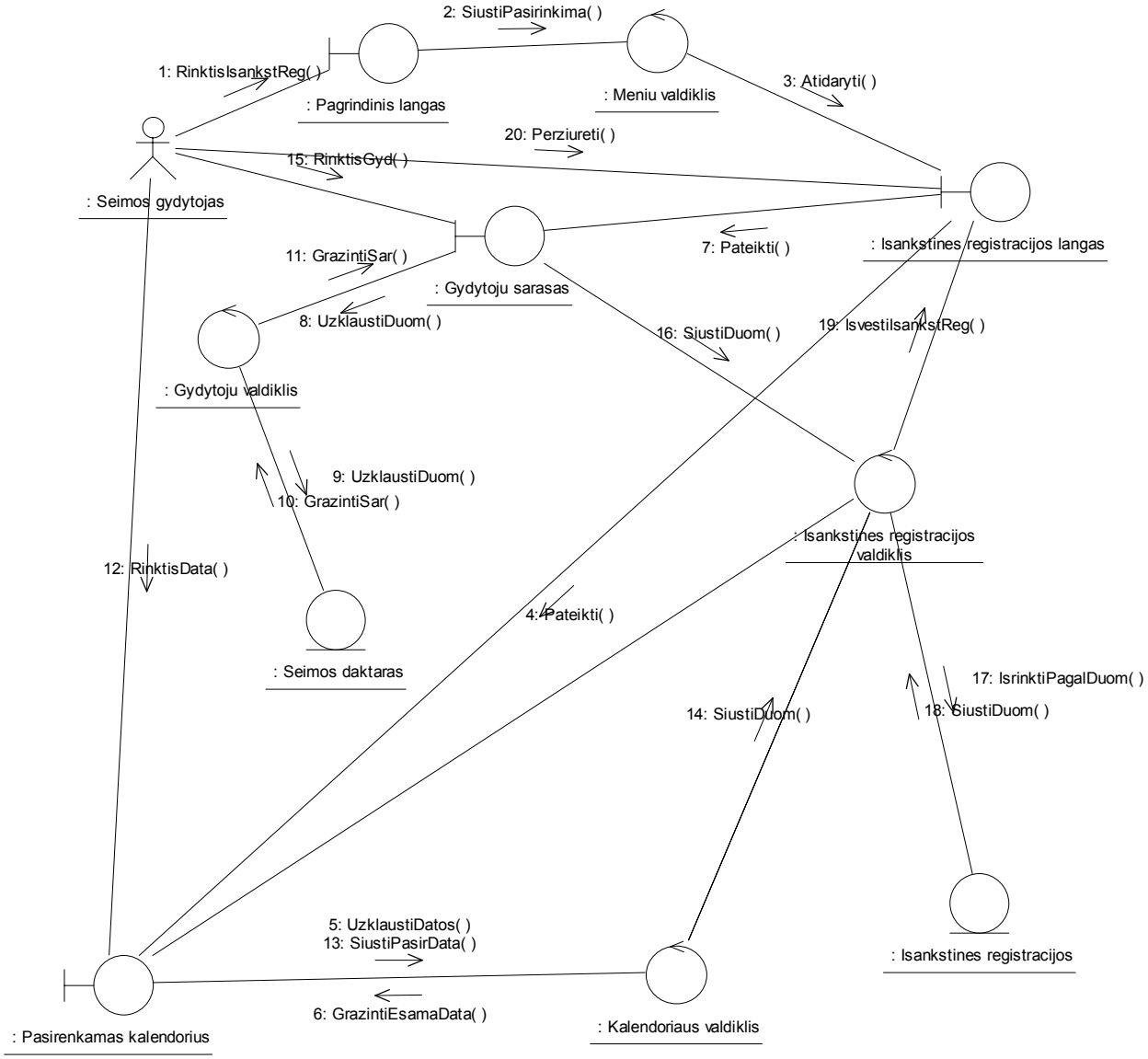
Pav. 28. Prisijungimo prie sistemos panaudojimo atvejo sekų diagrama.

Analogiškai gali pasijungti ir registracijos darbuotojas:



Pav. 29. Paciento duomenų redagavimo sekų diagrama.

Pav. 30 pavaizduota išankstinės registracijos peržiūros panaudojimo atvejis. Jo metu šeimos gydytojas išankstinės registracijos puslapyje esančiame kalendoriuje pasirenka norimą peržiūrėti datą, savo pavardę ir peržiūri užregistruotas išankstines registracijas.



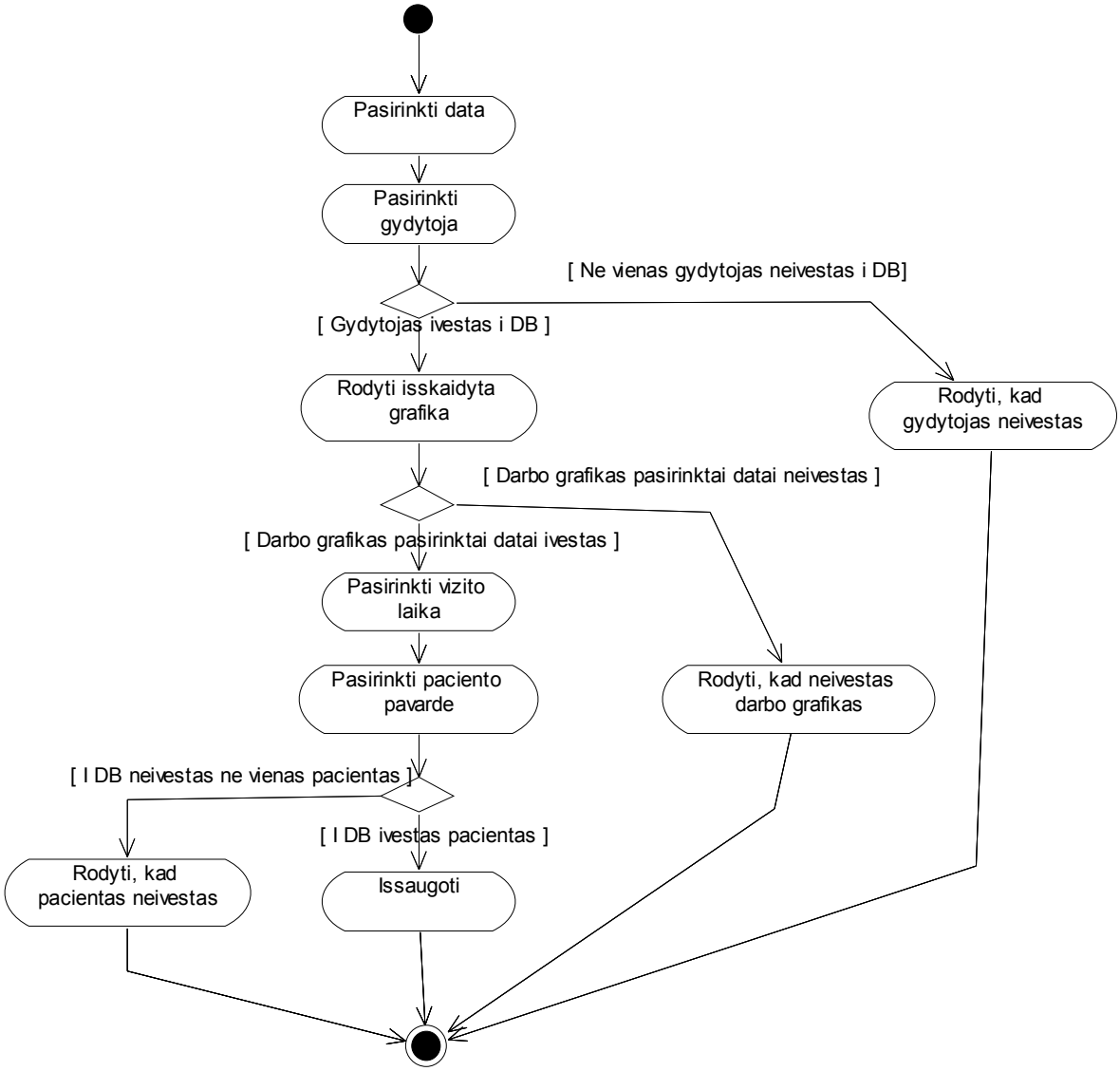
Pav. 30. Išankstinės registracijos peržiūros panaudojimo atvejo bendradarbiavimo diagrama

Paciento duomenų registravimas, naujo gydytojo prirašymas, gydytojų duomenų modifikavimas, iškvietimų į namus registravimas vyksta analogiškai, tik jį atlieka registracijos darbuotojas ir pasirenkami kiti internetiniai puslapiai.

Klasių modelį atitinka 9.2 priede pavaizduotas objektų gyvavimo priklausomybių grafas, kuriame pavaizduoti objektai su metodais, išdėstyti pagal MERODE siūlomą metodiką. Objektai *Vaiko kreivė*, *Darbo diena* ir *Išrašas iš stacionaro* yra informaciniai – jie neturi kitiems objektams būdingo gyvavimo ciklo, dėl to neturi ir metodų. Informaciniai objektai sukuriami saugoti tam tikrą informaciją.

3.3.2 Projekto elementų specifikacijos

Išankstinės registracijos puslapio veikimą galim pavaizduoti algoritmu (žr. Pav. 31). Valdikliai atsižvelgia į tai, ar prašomi duomenys įvesti į duomenų bazę ir jei neįvesti išveda atitinkamą pranešimą.

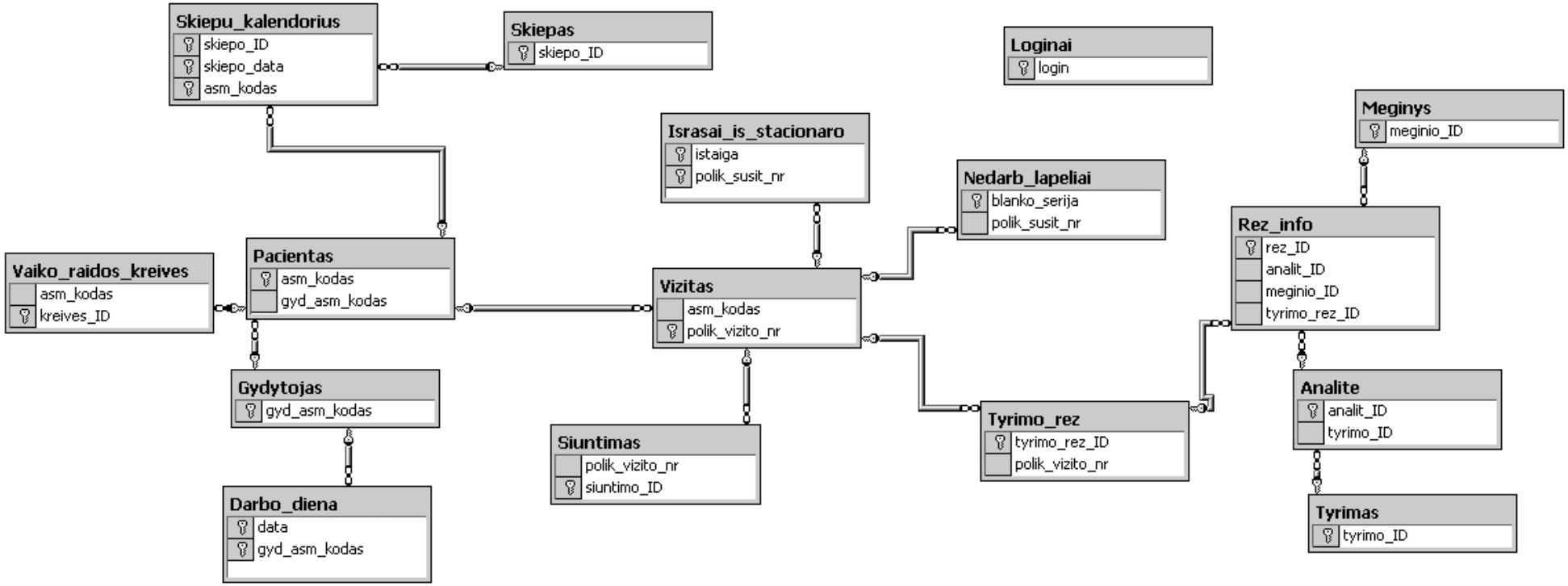


Pav. 31. Išankstinės registracijos algoritmas, aprašytas veiklos diagrama.

Kad apsaugoti sistemą nuo galimų klaidų, reikia apriboti vartotojų laisvę įvedant duomenis į duomenų bazę. Tam tikslui ir naudojami formų laukų ribojimai. Formos yra pagrindinė vartotojo sąsajos dalis, kuri sukuria sąsajos išvaizdą ir tuo pat metu kontroliuoja vartotojo veiksmus.

3.3.3 Duomenų bazės modelis

Duomenų bazės modelis sudarytas pagal objektų gyvavimo priklausomybių grafą (žr. Pav. 8). Duomenų bazės modelis, kuriame rodomi ne visų stulpelių vardai, o tik raktai ir jų pavadinimai, pavaizduotas Pav. 32. Pilnas duomenų bazės modelis pavaizduotas 9.3 priede.



Pav. 32. Duomenų bazės loginė schema

Duomenų bazės lentelių aprašymas pateiktas 24 lentelėje.

Lentelė 24

Duomenų bazės lentelių aprašai

Lentelė	Apibūdinimas
Pacientas	Saugomi duomenys apie pacientus
Vaiko_raidos_kreives	Saugomos vaiko kreivės, nusakančios paciento vystymąsi
Skiepas	Saugoma informacija apie skiepus, kuriais gali būti paskiepyti pacientai
Skiepu_kalendorius	Duomenys apie skiepus, kuriais skiepijamas pacientas
Gydytojas	Duomenys apie gydytojus, dirbančius šeimos klinikoje
Grafikas	Saugomas gydytojų darbo grafikas
Loginai	Saugomi vartotojų, galinčių prisijungti prie sistemos, prisijungimo duomenys
Vizitas	Duomenys apie paciento vizitą pas gydytoją
Israsai_is_stacionaro	Duomenys apie išrašus iš aukštesnio lygio gydymo įstaigų, kuriuos pateikia pacientas
Tyrimai	Duomenys apie galimus tyrimus
Meginys	Duomenys apie tyrimo metu imamus mėginius
Analite	Duomenys apie tyrimų analites
Rez_info	Duomenys apie pacientams paskirtų tyrimų rezultatus
Tyrimo_rez	Duomenys apie tyrimų paskyrimą pacientams
Nedarb_lapeliai	Informacija apie išrašytus nedarbingumo lapelius
Siuntimas	Duomenis apie paskyrimus specialisto konsultacijoms bei konsultacijų rezultatai

Kiekvieną lentelė atskirai su išvardintais atributais, atributų tipais, ilgiais, būtinumais ir aprašymais išvardinta žemiau pateiktose lentelėse.

Lentelė 25

Duomenų bazės lentelės Pacientas atributai ir jų aprašymai

Pacientas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
vardas	char	15	Taip	Paciento vardas
pavarde	char	20	Taip	Paciento pavardė
paso_nr	char	8	Ne	Paciento paso numeris
gim_data	date	4	Taip	Paciento gimimo data
lytis	number	1	Taip	Paciento lytis
valstybe	char	20	Ne	Valstybė, kurioje gyvena pacientas
el_past_adresas	char	40	Ne	Elektroninio pašto adresas
namu_telefonas	number	9	Ne	Paciento namų telefono numeris
mob_telefonas	number	9	Ne	Mobilaus telefono numeris
soc_padetis	char	25	Ne	Socialinė padėtis
darboviete	char	35	Ne	Paciento darbovietės pavadinimas
pareigos	char	35	Ne	Paciento pareigos darbovietėje

gimimo_liud_nr	char	10	Ne	Gimimo liudijimo numeris
soc_paz_nr	char	10	Ne	Socialinio pažymėjimo numeris
stud_paz_nr	char	6	Ne	Studento pažymėjimo numeris
invalid_paz_nr	char	10	Ne	Invalido pažymėjimo numeris
issilavinimas	char	10	Ne	Paciento išsilavinimas
tevas	char	11	Ne	Paciento tėvo asmens kodas
jautr_medikamentams	char	16	Ne	Medikamentai, kuriems jautrus pacientas
persirgtos_ligos	char	16	Ne	Paciento persirgtos ligos
letines_ligos	char	16	Ne	Lėtinės ligos, kuriomis serga pacientas
narkotikai	char	16	Ne	Narkotikai, kuriuos vartoja pacientas
seimynine_padetis	char	11	Ne	Paciento šeimyninė padėtis
korteles_nr	number	8	Ne	Pacientui paskirtos ligoninės kortelės numeris
asm_kodas	char	11	Taip	Paciento asmens kodas
gyd_asm_kodas	char	11	Ne	Gydytojo, pas kurį prirašytas pacientas, asmens kodas
motina	char	11	Ne	Paciento motinos asmens kodas
miestas	char	15	Ne	Miestas, kuriame gyvena pacientas
gatve	char	20	Ne	Gatvė, kurioje gyvena pacientas, pavadinimas
namas	number	4	Ne	Namo, kuriame gyvena pacientas, numeris
butas	number	4	Ne	Buto, kuriame gyvena pacientas, numeris
ar_isnesiotas	number	1	Ne	Žymė ar pacientas gimė išnešiotas
svoris	float	8	Ne	Kūdikio svoris
ugis	float	8	Ne	Kūdikio ūgis
traumos	char	16	Ne	Gimdymo metu kūdikio patirtos traumos
kt_duomenys	char	16	Ne	Kiti gimimo duomenys
ar_ruko	number	1	Ne	Žymė ar pacientas rūko
ar_alkoh	number	1	Ne	Žymė ar pacientas naudoja alkoholį
ar_narkotikai	number	1	Ne	Žymė ar narkotikai vartoja narkotikus
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 26

Duomenų bazės lentelės Skiepas atributai ir jų aprašymai

Skiepas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
skiepo_id	number	4	Taip	Skiepo identifikavimo numeris
vaistai	char	20	Taip	Vaistai, iš kurių sudarytas skiepas
liga	char	20	Ne	Liga, nuo kurios skiepijama
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 27

Duomenų bazės lentelės Skiepu_kalendorius atributai ir jų aprašymai

Skiepu_kalendorius				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
paskyrimo_data	date		Taip	Data, kada pacientui paskirtas skiepas
skiepo_id	char	20	Taip	Skiepo identifikavimo numeris
skiepo_data	date		Ne	Data, kada pacientas paskiepytas
asm_kodas	char	11	Ne	Paciento asmens kodas
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 28

Duomenų bazės lentelės Vaiko_raidos_kreives atributai ir jų aprašymai

Vaiko_raidos_kreives				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
asm_kodas	char	11	Ne	Paciento asmens kodas
kreives_ID	number	4	Taip	Kreivės identifikavimo numeris
amziaus_tarpsnis	char	15	Ne	Vaiko amžiaus tarpsnis, kuriam piešta kreivė
amzius_men	char	15	Ne	Vaiko amžiaus mėnesiais, kada piešta kreivė
ugis	float	8	Ne	Vaiko ūgis
svoris	float	8	Ne	Vaiko svoris
galvos_apimtis	float	8	Ne	Vaiko galvos apimtis
apziuros_data	date		Taip	Data, kada vaikas išmatuotas
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 29

Duomenų bazės lentelės Login atributai ir jų aprašymai

Login				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
login	char	10	Taip	Sistemos vartotojo prisijungimo vardas
pass	char	10	Ne	Sistemos vartotojo prisijungimo slaptažodis

Lentelė 30

Duomenų bazės lentelės Gydytojas atributai ir jų aprašymai

Gydytojas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
rusis	number	2	Taip	Gydytojo rūšis
kodas	number	8	Taip	Gydytojo kodas
licenz_nr	char	10	Taip	Gydytojo licenzijos numeris
gyd_asm_kodas	char	11	Taip	Gydytojo asmens kodas
gyd_vardas	char	15	Taip	Gydytojo vardas
gyd_pavarde	char	20	Taip	Gydytojo pavardė
namu_telefonas	number	9	Ne	Namų telefono numeris
mob_telefonas	number	9	Ne	Mobilaus telefono numeris
adresas	char	50	Ne	Valstybė, kurioje gyvena gydytojas
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 31

Duomenų bazės lentelės Grafikas atributai ir jų aprašymai

Grafikas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
data	smalldatetime	4	Taip	Darbo dienos data
gr_nuo	char	5	Taip	Laikas, nuo kurio dirba gydytojas
gr_iki	char	5	Taip	Laikas, iki kurio dirba gydytojas
gyd_asm_kodas	char	11	Taip	Gydytojo asmens kodas
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 32

Duomenų bazės lentelės Vizitas atributai ir jų aprašymai

Vizitas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
p_periodiskumas	char	20	Ne	Profilaktinio susitikimo periodiškumas
ligos_anamneze	char	16	Ne	Paciento ligos anamnezė
objekt_duomenys	char	16	Ne	Ligos objektyvūs duomenys
diagnoze	char	16	Ne	Nustatyta diagnozė
TLK_kodai	char	16	Ne	Nustatytų ligų TLK kodai
vaist_um_lig	char	16	Ne	Vaistai ir priemonės ūminėms ligoms gydyti
vaist_let_lig	char	16	Ne	Vaistai ir priemonės lėtinėms ligoms gydyti
blanko_serija	char	10	Ne	Išrašyto nedarbingumo lapelio blanko serija
gyd_rezimas	char	20	Ne	Paskirtas gydymo režimas
ar_medikament	bit	1	Ne	Žymė ar paskirtas gydymas medikamentinis
susitikimo_rusis	char	20	Ne	Vizito pas daktarą pobūdis: profilaktinis susitikimas, vizitas dėl ligos arba iškvietimas į namus
registracijos_laikas	datetime	8	Taip	Paciento užsiregistruoto vizito pas gydytoją data ir laikas
data	datetime	8	Ne	Paciento faktinio vizito pas gydytoją data
asm_kodas	char	11	Ne	Paciento, atvykusio į vizitą, asmens kodas
gyd_asm_kodas	char	11	Taip	Gydytojas, su kuriuo vyksta susitikimas, asmens kodas
polik_vizito_nr	number	9	Taip	Susitikimo numeris poliklinikos atžvilgiu
nusiskundimai	char	16	Ne	Paciento nusiskundimai sveikata
iskviet_priezastis	char	16	Ne	Iškvietimo į namus priežastis
iskviet_tipas	char	20	Ne	Iškvietimo į namus tipas
kaina	float	8	Ne	Vizito kaina
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 33

Duomenų bazės lentelės Nedarb_lapeliai atributai ir jų aprašymai

Nedarb_lapeliai				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
n_nuo	date	4	Taip	Data, nuo kurios išrašytas nedarbingumo lapelis
n_iki	date	4	Taip	Data, iki kurios išrašytas nedarbingumo lapelis
blanko_serija	char	10	Taip	Nedarbingumo lapelio blanko serija
polik_vizito_nr	number	9	Taip	Susitikimo numeris poliklinikos atžvilgiu, kurio metu išrašytas nedarbingumo lapelis
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 34

Duomenų bazės lentelės Tyrimas atributai ir jų aprašymai

Tyrimas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
tyrimo_ID	number	9	Taip	Tyrimo identifikavimo numeris
tyrimo_rusis	char	10	Ne	Tyrimo rūšis
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 35

Duomenų bazės lentelės Meginys atributai ir jų aprašymai

Meginys				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
meginio_ID	number	9	Taip	Mėginio identifikavimo numeris
meg_pavadinimas	char	16	Taip	Mėginio pavadinimas
meg_kiekis	char	15	Taip	Mėginio imamas kiekis
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 36

Duomenų bazės lentelės Analite atributai ir jų aprašymai

Analite				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
analites_ID	number	9	Taip	Analitės identifikavimo numeris
tyrimo_ID	number	9	Taip	Tyrimo identifikavimo numeris
pavadinimas	char	15	Taip	Analitės pavadinimas
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 37

Duomenų bazės lentelės Rez_info atributai ir jų aprašymai

Rez_info				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
rez_ID	number	9	Taip	Tyrimo rezultato identifikavimo numeris
tyrimo_ID	number	9	Taip	Tyrimo identifikavimo numeris
analites_ID	number	9	Taip	Analitės identifikavimo numeris
meginio_ID	number	9	Taip	Mėginio identifikavimo numeris
reiksme	char	15	Taip	Tyrimo rezultato reikšmė
norma	char	15	Ne	Tyrimo rezultato norma
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 38

Duomenų bazės lentelės Tyrimo_rez atributai ir jų aprašymai

Tyrimo_rez				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
tyrimo_rez_ID	number	9	Taip	Tyrimo identifikavimo numeris
polik_vizito_nr	number	9	Taip	Susitikimo numeris poliklinikos atžvilgiu, kurio metu paskirti tyrimai
skyrimo_data	date	4	Taip	Tyrimo paskyrimo data
tyrimo_data	date	4	Ne	Tyrimo atlikimo data
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

Lentelė 39

Duomenų bazės lentelės Siuntimas atributai ir jų aprašymai

Siuntimas				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
specialistas	char	20	Taip	Specialisto, pas kurį paskirta konsultacija, pavardė
isvada	char	16	Ne	Specialisto konsultacijos išvada
konsult_istaiga	char	30	Taip	Įstaiga, kurioje dirba konsultuojantis specialistas
polik_susit_nr	number	9	Taip	Susitikimo numeris poliklinikos atžvilgiu, kurio metu paskirta specialisto konsultacija
data_nuo	date	4	Taip	Data, nuo kurios galioja siuntimas
data_iki	date	4	Taip	Data, iki kurios galioja siuntimas
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

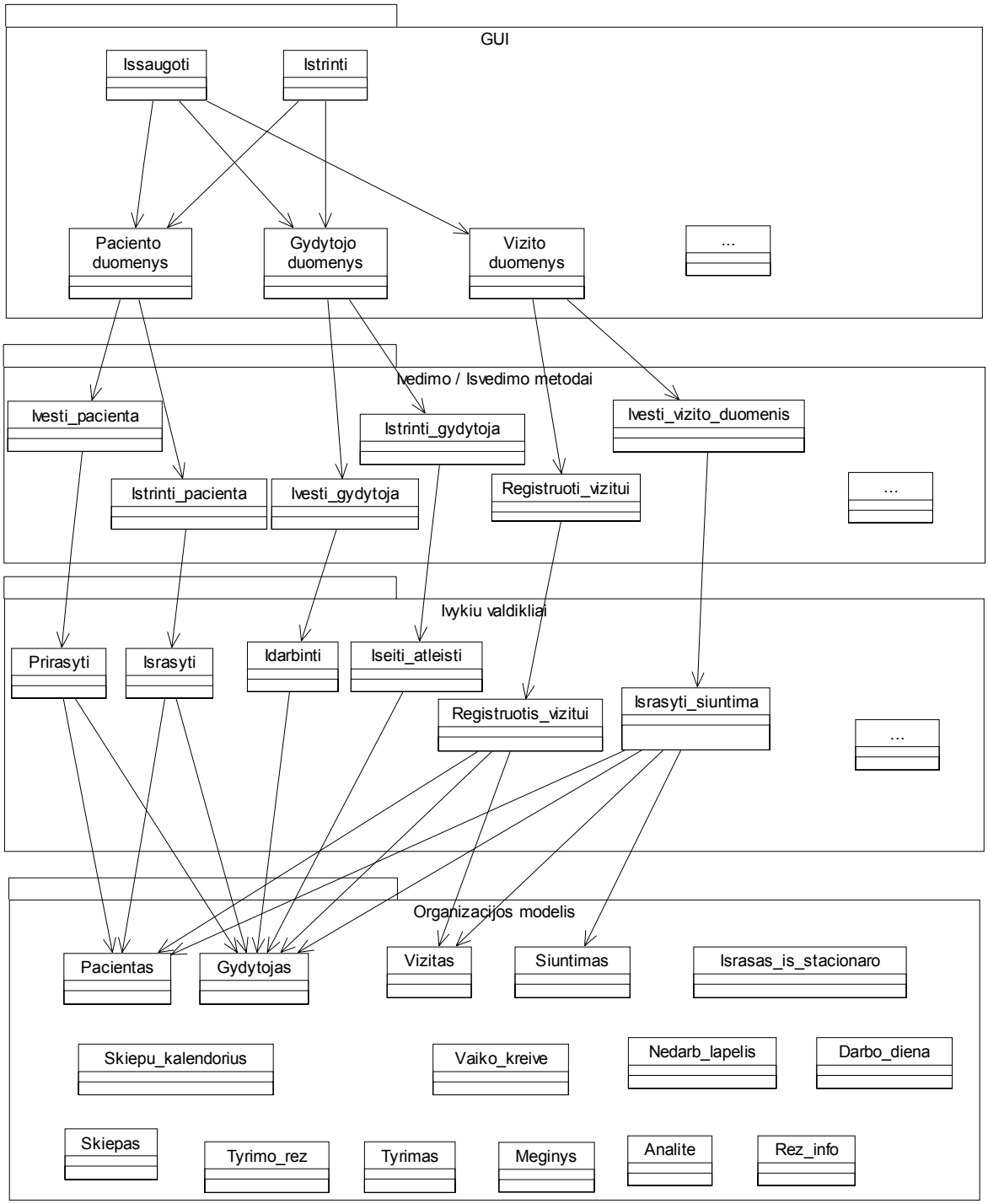
Lentelė 40

Duomenų bazės lentelės Israsai_is_stacionaro atributai ir jų aprašymai

Israsai_is_stacionaro				
Pavadinimas	**Tipas**	**Ilgis**	**Ar būtinas**	**Apibūdinimas**
istaiga	char	30	Taip	Stacionaro pavadinimas, kuriame buvo pacientas
skyrius	char	30	Taip	Stacionaro skyrius, kuriame išrašytas išrašas
susirg_data	date	4	Ne	Paciento susirgimo data
siunt_data	date	4	Taip	Siuntimo išrašymo data
hosp_data	date	4	Ne	Hospitalizavimo data
diagnoze	char	16	Ne	Stacionare nustatyta diagnozė
anamneze	char	16	Ne	Ligos anamnezė
rekomendacijos	char	16	Ne	Rekomendacijos pacientui
gydytojas	char	35	Ne	Gydytojas, išrašęs išrašą
israso_ID	number	9	Taip	Išrašo identifikavimo numeris
polik_susit_nr	number	9	Taip	Susitikimo numeris, kurio metu išrašytas siuntimas į stacionarą
busena	number	1	Ne	Objekto gyvavimo ciklo būsena

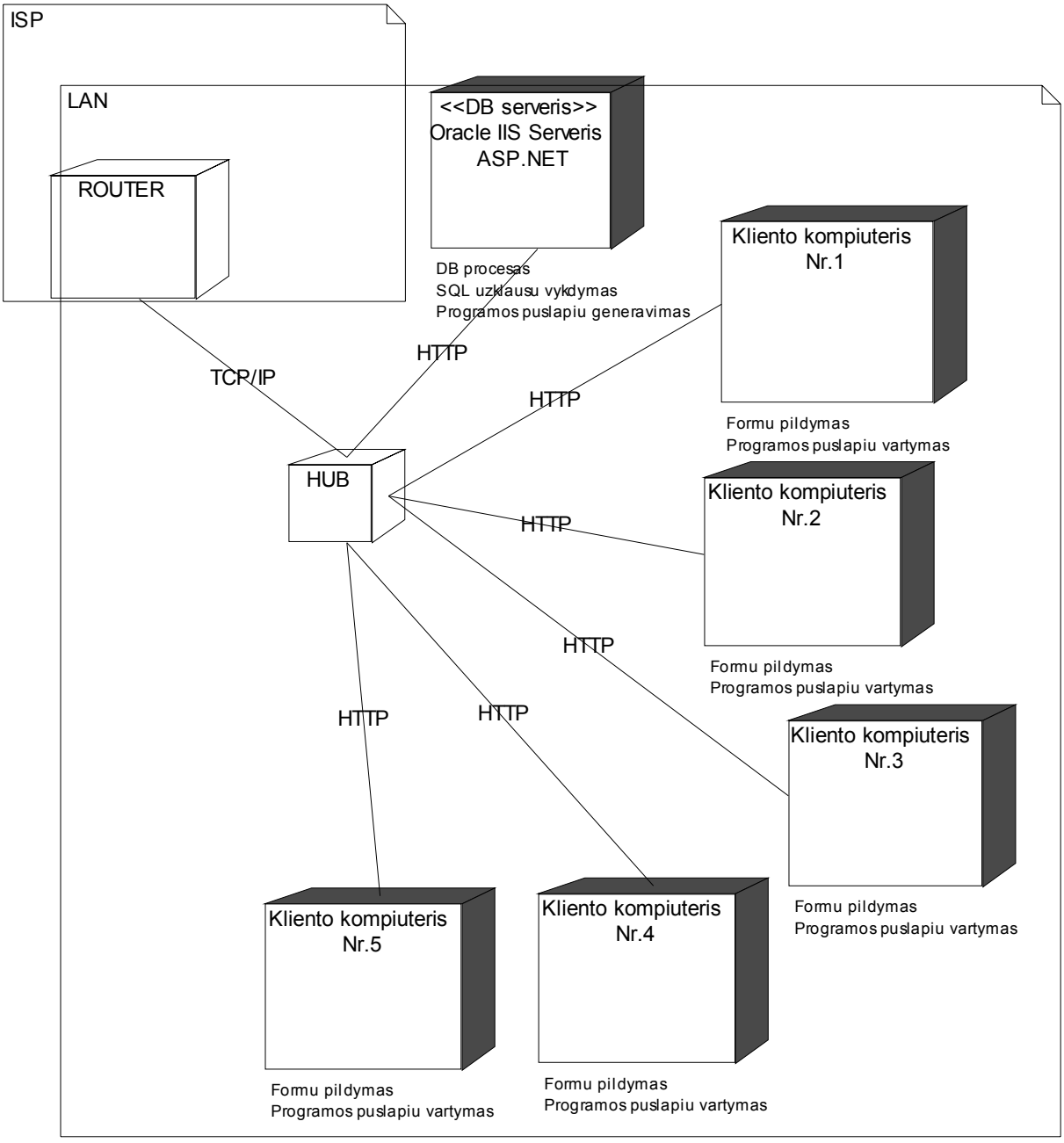
3.3.4 Realizacijos modelis

Realizacijai panaudotas sistemos realizacijos klasių modelis, siūlomas MERODE modeliavimo metodo, pavaizduotas Pav. 33 paveikslėlyje. Organizacijos veiklos sluoksnio objektai tarpusavyje nebendrauja siųsdami žinutes, o į juos kreipiasi įvykių valdikliai, taip įrašydami, keisdami ar nuskaitydami iš objektų informaciją. Vartotojo sąsajos ir įvykių valdiklio tarpininkai – įvedimo ir išvedimo metodai, išskiriami į atskirą sluoksnį.



Pav. 33. Realizacijos klasių modelis

Optimaliausias sistemos panaudojimo variantas – kai duomenų bazė ir taikomoji programa yra viename serveryje, prie kurio tinklo technologijomis būtų pajungti klientiniai kompiuteriai, kuriais naudotųsi poliklinikos darbuotojai (žr. Pav. 34).



Pav. 34. Šeimos klinikos kompiuterizavimo informacinės sistemos įdiegimo diagrama.

Visas dalis – duomenų, vartotojo ir kliento paslaugas – galima sudėti ir į vieną kompiuterį, o galima ir išskirstyti kiekvieną dalį į atskirą (pvz. duomenų bazė duomenų serveryje, informacinė sistema programų serveryje, o klientiniai kompiuteriai prie programų serverio jungiasi per tinklą).

4.1 Programos išeitinio teksto generavimas

Remiantis MERODE metodikos pasiūlytais modeliais: objektų priklausomybės grafu, objektų-įvykių lentele, būsenų diagramomis ir būsenų kaitos lentelėmis, galima suformuoti išeitinį programos tekstą. Remiantis MERODE, galima suformuoti organizacijos modelio objektų bei įvykių valdiklių klases. Organizacijos modelio objektų klasės formuojamos pagal 41 lentelėje pavaizduotą modelį.

Lentelė 41

Klasės aprašo šablonas

Class OBJEKTO_TIPO_PAVADINIMAS
Paveldimumas
// paveldimų objektų sąrašas
Atributai
// nuorodos į tėvinius objektų tipus
// nuorodos į priklausomus objektų tipus
// būsenos skaitiklis
// papildomi atributai
Invariantai
// atributų konstantos
// unikalumo konstantos
Metodai
// atomiškų įvykių tipų metodai

Automatizavus klasės aprašo gavimą, gautume žemiau pateiktą išeitinį klasės *Siuntimas* aprašą:

```
using System;
namespace Enterprise_Model
{
        /// <summary>
        /// Summary description for Siuntimas.
        /// </summary>
        public class Siuntimas : Objects
        {
                //Inherit
                //Attributes
                //master object types
                Vizitas ref_vizitas;

                //dependent object types
                //additional attributes
                int siuntimo_ID;
                string specialistas;
                string konsult_istaiga;
                long polik_vizito_nr;
                DateTime data_nuo;
                DateTime data_iki;
```

```
                //methods
                public Siuntimas(int in_siuntimo_ID, string in_specialistas,
                string in_konsult_istaiga, long in_polik_vizito_nr, DateTime in_data_nuo,
                DateTime in_data_iki)
                {
                        siuntimo_ID = in_siuntimo_ID;
                        specialistas = in_specialistas;
                        konsult_istaiga = in_konsult_istaiga;
                        polik_vizito_nr = in_polik_vizito_nr;
                        data_nuo = in_data_nuo;
                        data_iki = in_data_iki;
                }

                public void Israsyti_siuntima()
                {
                        state = 1;
                }

                public void Apsilankyti_pas_spec()
                {
                        if (state == 1)
                        state = 1;
                }

                public void Baigtis_siuntimo_laikui()
                {
                        if (state ==1)
                        state = 2;
                }
        }
}
```

Sudėtingesnės klasės *Gydytojas* išeitinis aprašas pateiktas 9.4 priede.

Automatizavus įvykių valdiklio gavimą, gautume žemiau pateiktą išeitinį klasės *Idarbinti* aprašą:

```
using System;

namespace EventHandlers
{
        /// <summary>
        /// Summary description for Idarbinti.
        /// </summary>
        public class Idarbinti : EventHandler
        {
                //attributes
                Enterprise_Model.Gydytojas gyd;

                //methods
                override protected void Validate()
                {
                        OK = true;
                        if ((OK == true) && (con.YraGyd(gyd) == true))
                        {
                                OK = false;
                                reason = "Gydytojas jau ivestas";
                        }
                        check = true;
                }

                override protected void Vykdyti()
                {
                                con.Idarbinti_db(gyd);
                }
        }
}
```

Sudėtingesnio įvykių valdiklio *Paskirti_tyrima* išeitinis aprašas pateiktas 9.5 priede.

Pilkame fone esantis tekstas parašytas programuotojo.

Jei egzistuotų automatinis išeitinio programos teksto generatorius pagal MERODE metodo modelius, programuotojui reikėtų įrašyti tik keletą eilučių.

Palyginus MERODE metodu gaunamą su Rational Rose įrankiu iš klasių modelio sugeneruotu programos išeitiniu tekstu [3], matosi, kad MERODE generuotas tekstas kur kas pilnesnis (Lentelė 42).

Lentelė 42

MERODE metodu ir Rational Rose įrankiu gautų programos išeitinių tekstų struktūrų palyginimas

	Rational Rose	**MERODE**
Klasės antraštė	+	+
Nuorodos į tėvines klases	-	+
Nuorodos į vaikines klases	+	+
Metodų deklaracijos	+	-
Metodai konstruktoriai	-	+
Metodų antraštės	+	+
Esamos būsenos patikrinimas	-	+
Būsenos pakeitimas	-	+
Įvykių valdiklio klasė	-	+

4.2 Eksperimentinė šeimos klinikos informacinė sistema

Šeimos klinikos informacinė sistema yra eksperimentinė sistema, skirta ištirti MERODE metodikos taikymą didelei sistemai kurti.

Organizacijos modelio ir įvykių valdiklių sluoksnius būtų galima sugeneruoti, naudojantis MERODE modeliais, o grafinės sąsajos ir įvedimo/išvedimo metodų sluoksnius savo nuožiūra kuria programuotojas/projektuotojas.

Šiai informacinei sistemai kurti pasirinkta internetinė vartotojo sąsaja, kas lemia platų duomenų prieinamumą. Pav. 35 paveiksle pavaizduotas paciento registracijos langas.

Pav. 35. Paciento registracijos langas

Pav. 36 paveiksle pavaizduotas išankstinės registracijos langas.

Pav. 36. Išankstinės registracijos langas

5 IŠVADOS

Tyrimo metu buvo sukurtos dvi tos pačios programos realizacijos, grįstos įprastu objektiniu ir MERODE pasiūlytu objektiniu modeliavimu. Modeliuojant MERODE, modeliuotojui paliekama mažiau laisvės, nes MERODE specifikacija griežtesnė, kas sumažina klaidų programoje tikimybę. MERODE metodu modeliuojant ir realizuojant eksperimentinę informacinę sistemą, prieita prie išvadų:

1. Jei klasių ir įvykių valdiklių išeitinio programos teksto generavimas pagal MERODE modelius būtų automatizuotas, būtų sutaupoma programos kūrimo laiko.
2. Kuriant MERODE metodu sumodeliuotą informacinę sistemą projektuotojui-programuotojui dirbti paprasčiau, nes sistemos struktūra ir klasės labiau specifikuotos.
3. Tiksliai sumodeliavus ir sugeneravus programos išeitinį tekstą, būtų išvengta daug klaidų, nes programos teksto kūrimas būtų automatizuotas ir apsaugotas nuo žmogiškų klaidų.
4. Dėl automatizavimo įrankio nebuvimo programuotojui reikia pačiam rašyti daug programos išeities teksto.

6 LITERATŪRA

Literatūros šaltiniai:

[1] Aerohost Web Systems. What is ASPX and Microsoft .NET?, žinių bazė, komercinis produktas. [žiūrėta 2003-12-07]. Prieiga per internetą: http://aerohost.com/DotNet-aspx.htm

[2] ASP 101. Chris Payne, ASP+ versus ASP, straipsnis. [žiūrėta 2003-12-07]. Prieiga per internetą: http://www.asp101.com/articles/chris/aspvsnet/default.asp

[3] Dūdonytė, B. MERODE metodika ir jos pritaikymo galimybės// Informacinė visuomenė ir universitetinės studijos: 9-osios tarpuniversitetinės magistrantų ir doktorantų konferencijos pranešimų medžiaga [Kaunas, 2004 m. balandžio 15 d.]. Kaunas, 2004, p. 179-186.

[4] Medicinos diagnostikos centras. Medicininė informacinė sistema. Komercinis produktas. [žiūrėta 2003-11-27]. Prieiga per internetą: http://www.medcentras.lt/medis.htm

[5] PC Magazine. The independent guide to technology. Database, recent reviews. Žurnalas. [žiūrėta 2003-10-21]. Prieiga per internetą: http://www.pcmag.com/category2/0,4148,4177,00.asp

[6] Rational Rose Developer. Features and benefits, žinių bazė (komercinis produktas). [žiūrėta 2003-12-05]. Prieiga per internetą: http://www-306.ibm.com/software/awdtools/developer/rose/features/

[7] Snoeck, M. Object-Oriented Enterprise Modeling with MERODE. Leuven University Press, 1999.

[8] UAB InKompas. Medicinos įstaigų kompiuterizavimas. Programos medicinos statistikams. Poliklinikos statistikos programa. Komercinis produktas. [žiūrėta 2003-11-20]. Prieiga per internetą: http://www.inkompas.lt

7 TERMINŲ IR SANTRAUPŲ ŽODYNAS

MERODE – (Model-based Existence-dependency Relationship Object-oriented Development) Modeliu grįstas egzistavimo priklausomybių sąryšio objektiškai orientuotas programų vystymas.

UML – (Unified Modelling Language) Modeliavimo kalba, naudojama objektiškai orientuotame projektavime.

CASE – (Computer Aided Software Engineering) kompiuterinės programinės priemonės, skirtos projektavimui palengvinti.

DBVS – duomenų bazės valdymo sistema.

IIS – (Internet Information Services) interneto informacijos paslaugos.

ASP – (Active Sever Page) aktyvių serverio puslapių technologija.

XML – (Extended Markup Language) išplėstinė formalizuotų tekstų kūrimo kalba.

COM – (common object model) bendras objektų modelis.

JSD-diagrama – (Jackson‘s System Development) diagrama, sukurta Jackson pasiūlytu sistemų kūrimo standartu.

8 SANTRAUKA ANGLŲ KALBA

Application of MERODE methodology to information systems

Summary

An object-oriented methodology MERODE is analyzed in this paper. Main aim of the article is to demonstrate practical application of MERODE methodology for creating computerized information systems, analyzing advantages and disadvantages of the new methodology. MERODE models are being combined with UML models. MERODE is used for modeling objects and events models that concludes classes and methods in the program code. UML models ar being used for analysing organisation aims, use cases that need to be computerized. The program code, recieved from MERODE models is more extensive than generated from UML models.

9 PRIEDAI

9.1 Kompiuterinės sistemos „MEDinfo“ aprašymas

UAB “InKOMPas”
Tel. faks. (8-5) 2 101 201
Tel. (8-5) 2 101 202
Mob. tel. (8-686) 13479
El. paštas admin@inkompas.lt

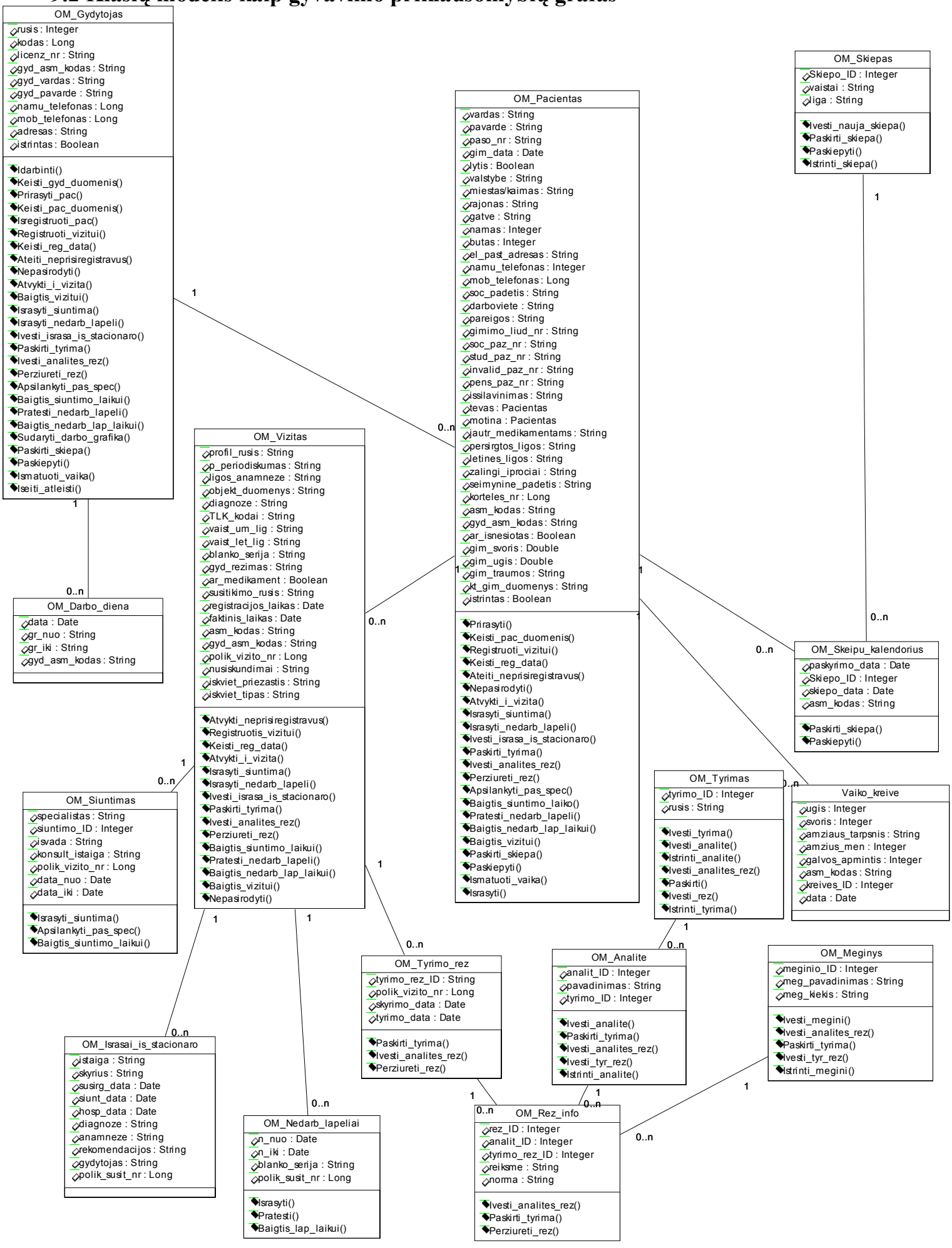
Kompiuterinė sistema “MEDinfo”

Tai medicinos įstaigos pacientų bei paslaugų apskaitos sistema, kurioje pabandėme suderinti vadybos dalykus ir medicininės informacijos kaupimą. Su sistema gali dirbti ar naudotis suvestais duomenimis praktiškai visos įstaigos grandys. Registratūra gali naudotis registracijų žurnalu ir patogiu būdu formuoti pacientų eiles bet kuriai pasirinktai dienai. Gydytojas gali vesti, spausdinti ir peržiūrėti ligos istorijos įrašus, diagnozes, tyrimų rezultatus, formuoti paciento gydymo planus ir t.t.. Administracija turi galimybes gauti pačią įvairiausią statistinę informaciją apie įstaigos grandžių darbą, paslaugų statistikas, gydytojų darbo statistiką, filtruoti pacientus pagal pačius įvairiausius kriterijus (atliktas paslaugas, apsilankymų dažnumą, paslaugų suteikimo datas ir t.t.), vesti korespondenciją su pacientais. Buhalterija gali pasinaudoti įvairiomis finansinėmis ataskaitomis ir suvestinėmis, stebėti pacientų įsiskolinimų ir apmokėjimų eigą, stebėti draudimo bendrovių ar kitų įstaigų, kurios įsipareigoja apmokėti už paslaugas, mokėjimų eigą ir šių mokėjimų statistiką. Taip pat buhalterija gali naudotis medžiagų sandėlio apskaitos moduliu, kurio pagalba gali vesti medžiagų pirkimus, stebėti sandėlio užpildymą, medžiagų nurašymą ir spausdinti medžiagų nurašymo aktus. Kasa veda paciento mokėjimų už paslaugas apskaitą, spausdina sąskaitas ir finansines suvestines.
Greta viso to, buvo stengiamasi, kad vartotojo darbas išliktų kiek įmanoma paprastesnis, darbiniai programos langai aprūpinti vienoda vartotojo sąsaja, stiliumi, įvairiomis duomenų paieškos ir funkcijomis, pagalbos langais. Vartotojas, padirbėjęs su kuria nors viena funkcija, nesunkiai galės įsisavinti ir kitas. Programa sukonstruota taip, kad vartotojas jaustųsi tikru duomenų šeimininku.
Pabrėžtinas ir ataskaitų formavimo būdas, leidžiantis filtruoti, grupuoti duomenis pagal pačius įvairiausius požymius ir atspausdinti ataskaitas, atsakančias į sudėtingiausius klausimus.
Realizuota keletas naudingų administravimo funkcijų. Tai duomenų atsarginių kopijų ir duomenų atstatymo funkcija, prisijungimo prie duomenų bazės parametrų nustatymo funkcija (reikalinga, kai prie duomenų bazės reikia prisijungti per kompiuterių tinklą), asmens sveikatos kortelės numerio formato nustatymo funkcija ir darbo su registracijų žurnalu parametrų nustatymas. Pastarosios dvi funkcijos leidžia programą nustatyti pagal konkrečius įstaigos poreikius.
Kompiuterinė sistema yra nuolatinio vystymo ir atnaujinimo kelyje. Ji nuolat papildoma naujomis funkcijomis ir galimybėmis. Taip pat paliekama galimybė priderinti ar įvesti naujas funkcijas pagal konkrečius kliento poreikius.

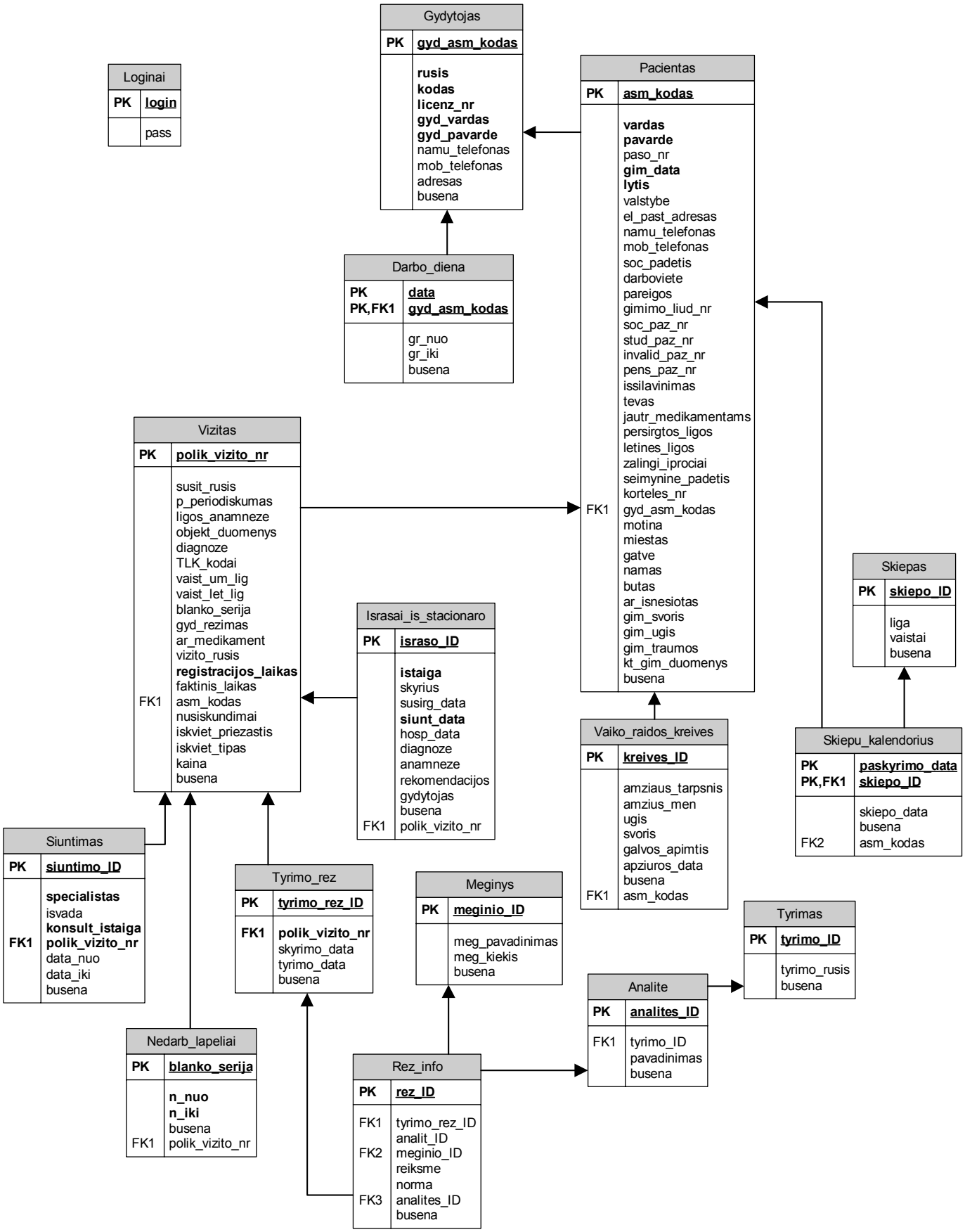
Pagrindinės funkcijos bei galimybės

1. **Pacientų sąrašo pildymas** ir pacientų paieška bei filtravimas pagal įvairius kriterijus.
2. **Pacientų registracijos žurnalas**. Kabinetų ir gydytojų užimtumo ir eilių formavimas. Išankstinė registracija į kabinetą arba pas gydytoją. Registruojama naudojant gydytojų ir kabinetų darbo grafikus.
3. **Asmens sveikatos istorijos pildymas** naudojantis formų šablonais, kuriuos pats gydytojas iš anksto paruošia. Programa automatizuoja paciento, gydytojo ir kitų duomenų įvedimą.
4. **Pacientui atliktų tyrimų rezultatų (tame tarpe ir laboratorinių tyrimų) kaupimas, spausdinimas bei analizė.**
5. **Paciento gydymo planų ir grafikų sudarymas (pvz. kūdikių ir vaikų skiepijimo grafiko sudarymas)**. Programa suranda ir nufiltruoja pacientus, kuriems tam tikrame laikotarpyje gydytojas paskyrė kokias nors procedūras ar darbus. Grafiko sudarymo palengvinimui galima naudoti iš anksto paruoštus darbų šablonus.
6. **Pacientui suteiktų paslaugų apskaita.**
7. **Nuolaidų administravimas.** Pacientui suteiktų nuolaidų ir išorinių siuntėjų apskaita.
8. **Paciento mokėjimų ir įsiskolinimų apskaita**. Sąskaitų, finansinių suvestinių spausdinimas.
9. **Draudimo bendrovių, bei kitų organizacijų, kurios įsipareigojo apmokėti už pacientui suteiktas paslaugas, mokėjimų ir įsiskolinimų apskaita**.
10. **Korespondencijos su pacientais vedimas**. Pagal įvairiausius kriterijus galima paruošti pacientų sąrašus ir atspausdinti adresus ar laiškus pagal iš anksto sukurtus šablonus. Programa automatiškai įterpia paciento duomenis.
11. **Medžiagų sandėlio apskaitos modulis**. Galima sekti medžiagų likučius, spausdinti užsakymus tiekėjams, įvesti ir nurašyti medžiagas. Paruošti nurašytų , sandėlyje esančių ir pradėtų naudoti medžiagų ataskaitas.
12. **Ataskaitos**. Paslaugų, finansinės, mokėjimų, pacientų, įmonių ir kt. ataskaitos. Duomenis galima nufiltruoti ir rūšiuoti pagal įvairiausius kriterijus.
13. **Duomenų paieška.** Visi pagrindiniai programos langai (pacientų, paslaugų, mokėjimų) turi sudėtingą duomenų filtravimo mechanizmą, kuris leidžia greitai surasti norimus duomenis pagal įvairiausias kriterijų kombinacijas.
14. **Administravimo priemonės**. Galimybė užduoti asmens sveikatos istorijos numerio formatą. Sukurti archyvinę duomenų kopiją ir atkurti duomenų bazę iš archyvo. Keisti prisijungimo prie duomenų bazės parametrus.

9.2 Klasių modelis kaip gyvavimo priklausomybių grafas



9.3 Loginė duomenų bazės schema



9.4 Išeitinis klasės *Gydytojas* aprašas

```
using System;

namespace Enterprise_Model
{
	/// <summary>
	/// Summary description for Gydytojas.
	/// </summary>
	public class Gydytojas : Objects
	{
		//Inherit

		//Attributes
		//master object types
		//dependent object types
		Pacientas ref_pacientas;
		Darbo_diena ref_darbo_diena;

		//additional attributes
		long rusis;
		long kodas;
		string licenc_nr;
		string gyd_asm_kodas;
		string vardas;
		string pavarde;
		long namu_tel;
		long mob_tel;
		string valstybe;
		string miestas_kaimas;
		string rajonas;
		string gatve;
		int namas;
		int butas;

		//methods
		public Gydytojas(long in_rusis, long in_kodas, string in_licenc_nr,
		string in_gyd_asm_kodas, string in_vardas, string in_pavarde,
		long in_namu_tel, long in_mob_tel, string in_valstybe, string in_miestas_kaimas,
		string in_rajonas, string in_gatve, int in_namas, int in_butas)
		{
			rusis = in_rusis;
			kodas = in_kodas;
			licenc_nr = in_licenc_nr;
			gyd_asm_kodas = in_gyd_asm_kodas;
			vardas = in_vardas;
			pavarde = in_pavarde;
			namu_tel = in_namu_tel;
			mob_tel = in_mob_tel;
			valstybe = in_valstybe;
			miestas_kaimas = in_miestas_kaimas;
			rajonas = in_rajonas;
			gatve = in_gatve;
			namas = in_namas;
			butas = in_butas;
		}

		public void Idarbinti()
		{
			state = 1;
		}

		public void Keisti_gyd_duom(Gydytojas gyd)
		{
			if (state == 1)
			{
				rusis = gyd.rusis;
				kodas = gyd.kodas;
```

```
                licenc_nr = gyd.licenc_nr;
                gyd_asm_kodas = gyd.gyd_asm_kodas;
                vardas = gyd.vardas;
                pavarde = gyd.pavarde;
                namu_tel = gyd.namu_tel;
                mob_tel = gyd.mob_tel;
                valstybe = gyd.valstybe;
                miestas_kaimas = gyd.miestas_kaimas;
                rajonas = gyd.rajonas;
                gatve = gyd.gatve;
                namas = gyd.namas;
                butas = gyd.butas;
                state = 1;
        }
}

public void Prirasyti_pac()
{
        if (state == 1)
                state = 1;
}

public void Keisti_pac_duomenis()
{
        if (state == 1)
                state = 1;
}

public void Isregistruoti_pac()
{
        if (state == 1)
                state = 1;
}

public void Registruoti_vizitui()
{
        if (state == 1)
                state = 1;
}

public void Keisti_reg_data()
{
        if (state == 1)
                state = 1;
}

public void Ateiti_neprisiregistravus()
{
        if (state == 1)
                state = 1;
}

public void Nepasirodyti()
{
        if (state == 1)
                state = 1;
}

public void Atvykti_i_vizita()
{
        if (state == 1)
                state = 1;
}

public void Baigtis_vizitui()
{
        if (state == 1)
                state = 1;
}
```

```
public void Israsyti_siuntima()
{
	if (state == 1)
		state = 1;
}

public void Israsyti_nedarb_lapeli()
{
	if (state == 1)
		state = 1;
}

public void Ivesti_israsa_is_stacionaro()
{
	if (state == 1)
		state = 1;
}

public void Paskirti_tyrima()
{
	if (state == 1)
		state = 1;
}

public void Ivesti_analites_rez()
{
	if (state == 1)
		state = 1;
}

public void Perziureti_rez()
{
	if (state == 1)
		state = 1;
}

public void Apsilankyti_pas_spec()
{
	if (state == 1)
		state = 1;
}

public void Baigtis_siuntimo_laikui()
{
	if (state == 1)
		state = 1;
}

public void Pratesti_nedarb_lapeli()
{
	if (state == 1)
		state = 1;
}

public void Baigtis_nedarb_lap_laikui()
{
	if (state == 1)
		state = 1;
}

public void Sudaryti_darbo_grafika()
{
	if (state == 1)
		state = 1;
}

public void Paskirti_skiepa()
{
```

```
                    if (state == 1)
                        state = 1;
                }

                public void Paskiepyti()
                {
                    if (state == 1)
                        state = 1;
                }

                public void Ismatuoti_vaika()
                {
                    if (state == 1)
                        state = 1;
                }

                public void Baigtis_darbo_dienai()
                {
                    if (state == 1)
                        state = 1;
                }

                public void Iseiti_atleisti()
                {
                    if (state == 1)
                        state = 2;
                }

        }
}
```

9.5 Išeitinis įvykių valdiklio *Paskirti_tyrima* aprašas

```
using System;

namespace EventHandlers
{
	/// <summary>
	/// Summary description for Paskirti_tyrima.
	/// </summary>
	public class Paskirti_tyrima : EventHandler
	{
		//attributes
		Enterprise_Model.Tyrimas tyrimas;
		Enterprise_Model.Analite analite;
		Enterprise_Model.Meginys meg;
		Enterprise_Model.Vizitas vizitas;
		Enterprise_Model.Pacientas pac;
		Enterprise_Model.Gydytojas gyd;
		Enterprise_Model.Tyrimo_rez tyr_rez;

		//methods
		public void Validate()
		{
			OK = true;
			if ((OK == true) & (analite.state != 1))
			{
				OK = false;
				reason = "Analite netinkamoj busenoj";
			}

			if ((OK == true) & (meg.state != 1))
			{
				OK = false;
				reason = "Meginys netinkamoj busenoj";
			}

			if ((OK == true) & (tyrimas.state != 1))
			{
				OK = false;
				reason = "Tyrimas netinkamoj busenoj";
			}

			if ((OK == true) & (gyd.state != 1))
			{
				OK = false;
				reason = "Gydytojas netinkamoj busenoj";
			}

			if ((OK == true) & (pac.state != 1))
			{
				OK = false;
				reason = "Pacientas netinkamoj busenoj";
			}

			if ((OK == true) & (vizitas.state != 2))
			{
				OK = false;
				reason = "Vizitas netinkamoje busenoje";
			}
			check = true;
		}

		public void Run()
		{

			Validate();
			if (OK & check)
			{
```

```
                                con.Paskirti_tyrima_rez_db(tyr_rez, analite, tyrimas, meg, vizitas, pac, gyd);
                                tyr_rez.Paskirti_tyrima();
                                meg.Paskirti();
                                analite.Paskirti();
                                tyrimas.Paskirti();
                                vizitas.Paskirti_tyrima();
                                pac.Paskirti_tyrima();
                                gyd.Paskirti_tyrima();

                        }
                }
        }
}
```

9.6 Straipsnis „MERODE metodika ir jos pritaikymo galimybės“

MERODE METODIKA IR JOS PRITAIKYMO GALIMYBĖS

Birutė Dūdonytė

Kauno technologijos universitetas,Informacijos sistemų katedra, Studentų g. 50

(Vadovė: Rita Butkienė)

Straipsnyje nagrinėjama objektinė metodika MERODE, skirta informacinėms sistemoms kurti. Pagrindinis straipsnio tikslas – pademonstruoti MERODE metodikos praktinio pritaikymo galimybes kuriant kompiuterizuotas informacines sistemas.

1 Įvadas

Objektiškai orientuotos modeliavimo kalbos, tokios kaip UML, siūlo daug modelių, aprašančių informacines sistemas įvairiais požiūriais. Komerciniai modeliavimo įrankiai (tokie kaip Rational Rose, MagicDraw, MS Visio, Describe ir pan.) negali automatiškai patikrinti UML išreikštų modelių suderinamumo tarpusavyje dėl to, kad UML neturi formalių taisyklių, nusakančių modelių tarpusavio sąryšius [2]. Vieno modelio principo alternatyva [3] siūlo modelių suderinamumo tikrinimą. Šis požiūris pagrįstas vienu modeliu, kuriam sukuriami skirtingi požiūriai, kurių tarpusavio suderinamumas gali būti patikrintas automatiškai. Toks ir yra Belgijos mokslininkų sukurtas MERODE modeliavimo metodas.

2 MERODE motyvacija ir apžvalga

2.1 Sprendžiamos problemos

Ilgą laiką programinės įrangos modeliavimo metodai buvo orientuoti arba vien į objektus, arba vien į įvykius. Tačiau tikrovėje procesai ir duomenys tarpusavyje tampriai susiję objekto sąvokoje. Be to, dauguma metodų sistemos specifikacijai naudoja natūralios kalbos ir pusiau formalaus grafinio žymėjimo mišinį. Tokie žymėjimai gali būti naudojami neformalioms analizėms, bet griežta patikra ir atestavimas yra beveik neįmanomi. Dar viena problema kyla dėl to, kad programinės įrangos realizavimas įtakoja daugelį modeliavimo technologijų. Orientacija į objektus pirmiausia atsirado programavime. Tie patys principai buvo pritaikyti modeliavime, idėjos neperžiūrėjus iš esmės. Analizė turėtų tik aprašyti dalykinę sritį, bet neįtakoti galimų problemos sprendimo variantų. Objektiškai orientuotos sistemos paprastai aprašomos kaip tarpusavyje bendraujančių objektų rinkinys. Augant projekto apimčiai, objektų modeliai yra linkę išsiplėsti iki sudėtingų tarpusavy bendraujančių tinklų. Nepaisant daugelio objektiškai orientuotos sistemos teikiamų pranašumų, gauta didelė sistema sunkiai valdoma.

MERODE buvo sukurtas siekiant išspręsti šias problemas: savybių, garantuojančių specifikacijos kokybę, trūkumą; objektiškai orientuotų analizės priemonių polinkį į realizaciją; aukšto lygio agregavimo priemonių trūkumą taikant objektiškai orientuotą metodiką [1]. Tai įtakojo pagrindines MERODE ypatybes: modeliu pagrįstą sistemos kūrimo metodiką, abstraktesnes objektų sąveikos sąvokas, formalų visų priemonių aprašymą.

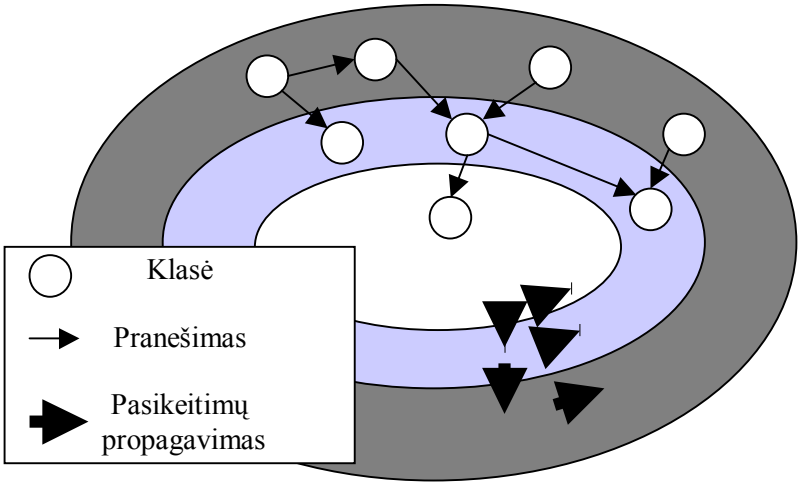
2.2 Modeliu grįstas sistemos kūrimas

Modeliu grįstas projektavimas vadovaujasi idėja, kad informacinė sistema sudėtinga tiek, kiek sudėtinga projektuojama dalykinė sritis [1]. Informacinių sistemų projektuotojai, teikiantys pirmenybę realaus pasaulio modelio, o ne norimos sistemos funkcionalumo projektavimui, geriau susitvarko su projektuojamos sistemos sudėtingumu.

Analizės modelis – tai abstrakcija to, ką sistema turi daryti, o ne to, kaip ji tai turi daryti. Objekto modelis, sukurtas analizės fazėje apibrėžia objekto klases, jų bruožus, metodus ir tarpusavio sąveiką. Bendrai paėmus, objektinės metodikos nesuskirsto specifikuotų objektų klasių į kategorijas. Tačiau šis suskirstymas yra reikalingas, jei norime valdyti ir palaikyti specifikaciją (ir realizaciją).

Norint sistemiškai suskirstyti objektų klases, reikia atsižvelgti į tam tikras specifikacijos savybes bei jos prigimtį. Pavyzdžiui, interfeiso reikalavimai keičiasi daug dažniau nei veiklos modelis. Geriau objektų klases sugrupuoti į sluoksnius, atsižvelgiant į jų kategoriją, nei turėti vientisą objektų rinkinį. Vidinio sluoksnio objektai stabiliausi, o išoriniai – labiau linkę keistis. Klasės gali naudotis tik arčiau vidaus, bet ne arčiau išorės nuo jų esančių klasių paslaugomis. Tai užtikrintų, kad išoriniuose sluoksniuose vykdomi pakeitimai nepaveiktų arba mažai paveiktų vidines klases. Į vidinį sluoksnį įdėjus stabilias klases, jos gali išlikti nepakitusios keičiant klases, esančias išoriniam sluoksnyje (žr. pav. 1).

Norint gauti tokį sluoksniuotą objektų rinkinį, specifikacija taip pat turėtų būti sluoksniuota.



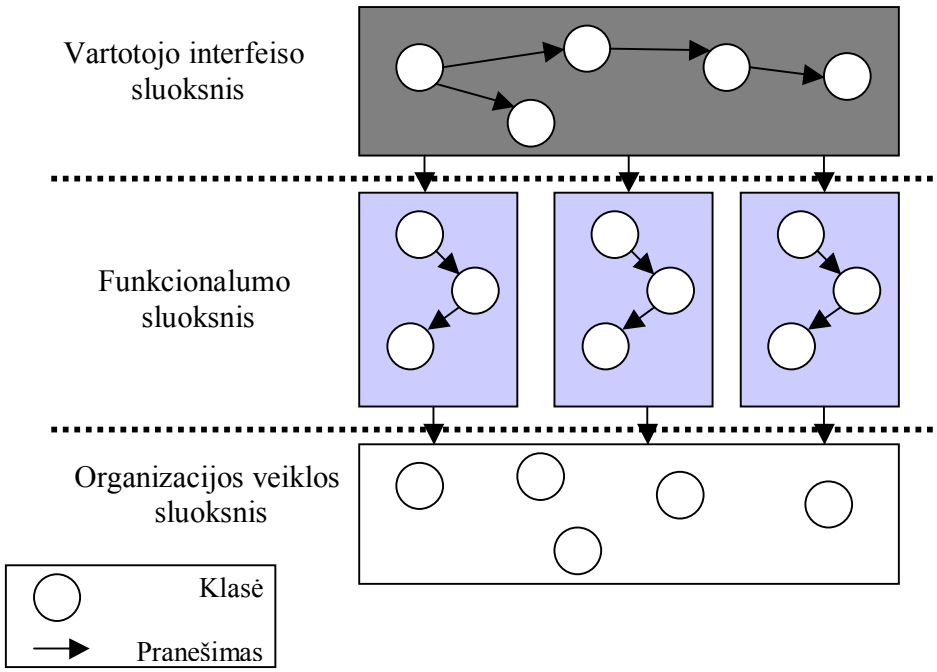
1 pav. Objektai sluoksniuose.

2.3 Specifikacijos sluoksniavimas

MERODE metodu kuriamos sistemos turi natūralią sluoksnių struktūrą grupuojant specifikacijas pagal objekto aspektus [1]. Kai kurios specifikacijos kyla iš esminių verslo reikalavimų. Kitos specifikacijos paprastai susijusios su esama informacine sistema ir kyla iš daugiau administracinio tipo funkcijų, tokių kaip duomenų įvedimas, ataskaitų generavimas ir pan. Pirmojo tipo specifikacijos sudaro organizacijos veiklos sluoksnį, apie jį esantis funkcionalumo sluoksnis sudarytas iš antro tipo specifikacijos. Jis teikia duomenų įvedimo ir išvedimo paslaugas.

Dauguma dabartinių programinės įrangos projektavimo metodų neatskiria organizacijos veiklos nuo funkcionalumo. Jie paprastai veiklos objektui priskiria tiek veiklos atributus ir paprogrames, tiek įvedimo ir išvedimo procedūras. MERODE toks grupavimas negalimas specifikavimo stadijoje. Jis gali atsirasti tik realizavimo stadijoje.

Ant funkcionalumo sluoksnio dedamas vartotojo sąsajos sluoksnis suklijuoja individualias funkcionalumo sluoksnio funkcijas tarpusavy ir leidžia vartotojui sukelti, įsiterpti ir pratęsti funkcijų veikimą. Gauname labai lanksčią sluoksniuotą programos architektūrą (žr. pav. 2). Pastebėkite pav. 2, kad organizacijos veiklos sluoksnio objektai tarpusavyje nebendrauja siųsdami žinutes. Tai parodo MERODE ypatingą požiūrį į objektų sąveiką: organizacijos objektai sąveikauja tarpusavyje tik drauge įtraukti į veiklos įvykius.



2 pav. Informacinės sistemos architektūra pagal MERODE

Šios architektūros nereikėtų maišyti su trijų sluoksnių architektūra paskirstytoms sistemoms. Trijų sluoksnių architektūroje atskiriama vartotojo sąsaja, logika ir duomenys. MERODE architektūroje funkcionalumo sluoksnyje yra tiek programos logika, tiek duomenys ir organizacijos veiklos sluoksnis turi tiek organizacijos objektų duomenis, tiek veiklos logiką.

2.4 Modeliavimas taikant MERODE metodiką

MERODE metodikoje išskiriami du pagrindiniai sistemos kūrimo etapai: specifikavimas ir realizavimas, o specifikavimo etape išskiriamos tokios trys fazės:

1. Organizacijos veiklos modeliavimas,
2. Funkcionalumo modeliavimas,
3. Interfeiso modeliavimas.

Toliau detaliau bus pristatyta organizacijos veiklos modeliavimo fazė, nes joje naudojamos priemonės išskiria MERODE metodiką iš kitų.

MERODE organizacijos veiklos modeliui sudaryti naudojamos trijų tipų priemonės: objektų tipų gyvavimo priklausomybių grafas, objektų-įvykių lentelė ir objektų elgsenos schemos. Gyvavimo priklausomybės grafo dėka užtikrinamas suderinamumas tarp dalinių organizacijos veiklą aprašančių modelių ir tokiu būdu galima pasiekti geresnę specifikacijos kokybę nei atskirai tikrinant sintaksinį atitikimą tarp dalinių schemų. MERODE metodikoje specifikacijos kokybę nusako specifikacijos vidinė darna (skirtingi modeliai modeliuoja skirtingus sistemos aspektus, bet turi persidengiančios semantikos) ir korektiškumas. Griežta ir nedviprasmiškai apibrėžta MERODE modeliuose naudojamų sąvokų sintaksė ir semantika (šiam tikslui panaudota procesų algebra) sudaro sąlygas formaliai patikrinti specifikacijos vidinį suderinamumą bei korektiškumą.

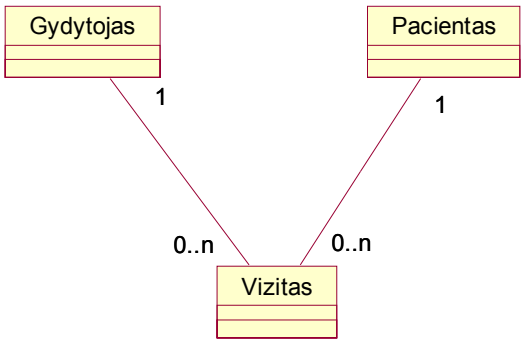
Kitame poskyryje iliustruosime organizacijos veiklos modeliavimo taikant MERODE metodiką ypatybes. Organizacijos veiklos modeliui sudaryti bus naudojama UML.

2.5 MERODE metodiką iliustruojantis pavyzdys

Metodikai iliustruoti pasirinkta dalykinė sritis – paciento apsilankymo pas gydytoją registravimas. Dalykinės srities aprašas būtų toks:

> *Vizitas, tai vieno paciento apsilankymas pas vieną gydytoją. Pacientas gali iš anksto užsiregistruoti vizitui pas gydytoją. Vizito laiką ir datą vėliau galima keisti. Pacientas gali ateiti pas gydytoją ir neprisiregistravęs iš anksto. Jis gali prisiregistruoti ir ateiti pas tą patį bei kitus gydytojus daug kartų. Gydytojas gali priimti daug pacientų, bet skirtingu laiku.*

Organizacijos veiklos modeliavimas pradedamas nuo objektų tipų identifikavimo. Remdamiesi dalykinės srities aprašu, identifikuojami tokie objektų tipai: pacientas, gydytojas ir vizitas. Identifikavus objektų tipus turi būti specifikuotos priklausomybės tarp tų tipų. MERODE metodika šiam tikslui naudoja gyvavimo priklausomybių grafą. Ši priemonė labai panaši į objektų klasių modelį tik turi papildomas priemones gyvavimo priklausomybei specifikuoti. Kadangi UML klasių modelis neturi specialių priemonių gyvavimo priklausomybei tarp objektų pavaizduoti, tai šiam tikslui bus panaudota klasių išdėstymo tvarka diagramoje, t. y. jei klases B gyvavimas priklauso nuo klasės A gyvavimo, tai klasė B atžvilgiu klasės A diagramoje turi būti pavaizduota žemiau. Nagrinėjamame pavyzdyje objekto *Vizitas* gyvavimas priklauso nuo objektų *Gydytojas* ir *Pacientas* buvimo. Jei nebūtų bent vieno iš jų, objektas *Vizitas* neegzistuotų (3 pav.).



3 pav. Klasių modelis kaip gyvavimo priklausomybių grafas.

Reikia paminėti, kad objektų gyvavimo priklausomybių grafas turi tenkinti tokias sąlygas:

- objektų tipo gyvavimas negali priklausyti nuo jo paties,
- gyvavimo priklausomybių grafas yra necikliškas.

Antras svarbus žingsnis modeliuojant organizacijos veiklą – įvykių tipų, kuriuose dalyvauja objektai ir kuriuos turės registruoti kuriamoji sistema, identifikavimas. Tam tikro objekto gyvavimo cikle gali būti daug įvykių, kurie jį sukuria, keičia, užbaigia jo gyvavimą. Kiekvieno objekto gyvavimo cikle turi būti bent du įvykiai: vienas, kuris objektą sukuria, ir vienas, kuris užbaigia jo gyvavimą. Įvykių tipai yra susiejami su objektų tipais. Šiam tikslui yra naudojama

objektų-įvykių lentelė. UML neturi tokios priemonės. Todėl bus naudojama tokia lentelė, kokią ją siūlo MERODE metodika (1 lentelė).

1 lentelė. Objektų-įvykių lentelė.

	Pacientas	Gydytojas	Vizitas
Prirašyti	C		
Įdarbinti		C	
Išrašyti	E		
Išeiti, atleisti		E	
Keisti paciento duomenis	M		
Keisti gydytojo duomenis		M	
Prisiregistruoti vizitui	M	M	C
Keisti registracijos datą	M	M	M
Ateiti neprisiregistravus	M	M	C
Nepasirodyti prisiregistravus	M	M	E
Atvykti į vizitą	M	M	E

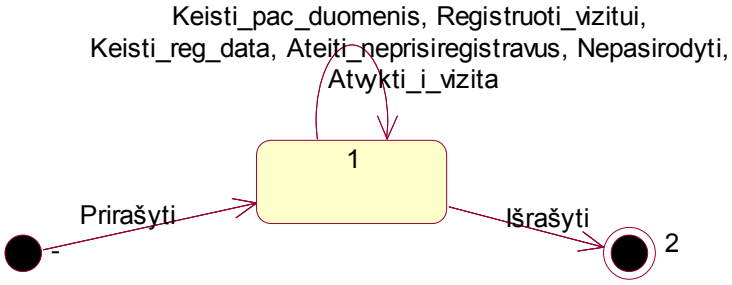
Lentelės struktūra yra tokia: kiekvienam įvykio tipui skiriama eilutė, o kiekvienam objektų tipui – stulpelis. Objektų-įvykių lentelė parodo: 1) kokiuose įvykiuose dalyvauja tam tikro tipo objektai ir 2) kokią įtaką padaro įvykis objektui. Šiems dalykams atskleisti lentelės langeliai užpildomi pagal tokias taisykles:

- jei įvykis sukuria tam tikrą objektą, tai įvykio tipo eilutės ir objekto tipo stulpelio sankirtoje įrašoma raidė C,
- jei įvykis užbaigia tam tikro objekto gyvavimą, tai įvykio tipo eilutės ir objekto tipo stulpelio sankirtoje įrašoma raidė E,
- jei įvykis pakeičia tam tikro objekto savybes (būseną), tai įvykio tipo eilutės ir objekto tipo stulpelio sankirtoje įrašoma raidė M.

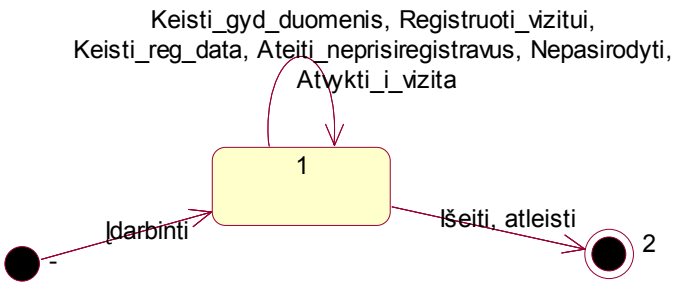
Tušti langeliai reiškia, kad objektas nedalyvauja atitinkamuose įvykiuose. Pildant objektų-įvykių lentelę, taip pat laikomas tokių taisyklių:

- *Propogavimo* taisyklė. Jei objekto B gyvavimas priklauso nuo objekto A, tai įvykių, kuriuose dalyvauja objektas B, aibė yra objekto A įvykių aibės poaibis.
- *Įtraukimo* taisyklė. Jei priklausomo objekto B stulpelyje tam tikrose eilutėse įrašyta raidė *C*, tai objekto A, nuo kurio priklauso objektas B, toje pačioje eilutėje turi būti įrašyta arba raidė *C* , arba raidė *M*. Jei priklausomo objekto B stulpelyje tam tikrose eilutėse įrašyta raidė *E*, tai objekto A, nuo kurio priklauso objektas B, toje pačioje eilutėje turi būti įrašyta arba raidė *E* , arba raidė *M*. Jei priklausomo objekto B stulpelyje tam tikrose eilutėse įrašyta raidė M, tai objekto A, nuo kurio priklauso objektas B, toje pačioje eilutėje turi būti įrašyta raidė *M*.
- *Kontrakto* taisyklė. Jei du objektai dalyvauja tame pačiame įvykyje, tai tame įvykyje turi dalyvauti ir bendras priklausomas objektas.

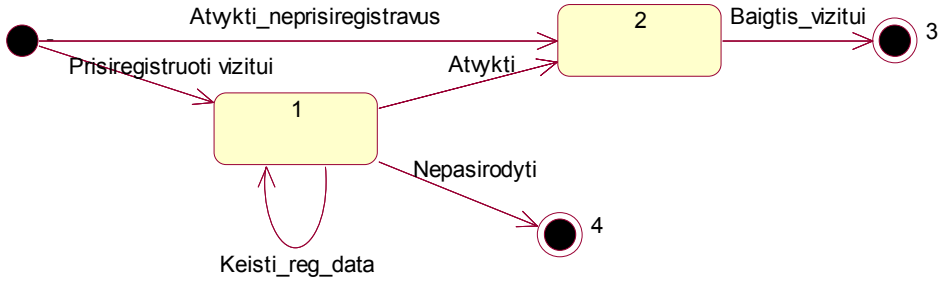
Įvykiai objekto gyvenime nevyksta atsitiktine tvarka. Jie turi tam tikrą seką. Pavyzdžiui, pacientas pirmiau užsiregistruoja vizitui, o vėliau į vizitą atvyksta. Atvirkščias variantas būtų netikslingas. Įvykių sekai specifikuoti MERODE metodika siūlo naudoti keletą priemonių: JSD-diagramas, reguliarias išraiškas arba baigtinių būsenų mašinas. UML kalboje objektų būsenų kaitai modeliuoti naudojamos būsenų diagramos, kurios atitinka baigtinių būsenų mašiną. Šiomis diagramomis pavyzdyje ir pavaizduoti identifikuotų objektų tipų gyvavimo ciklai (4, 5, 6 pav.)



4 pav. Objekto *Pacientas* baigtinių būsenų diagrama.



5 pav. Objekto *Gydytojas* baigtinių būsenų diagrama.



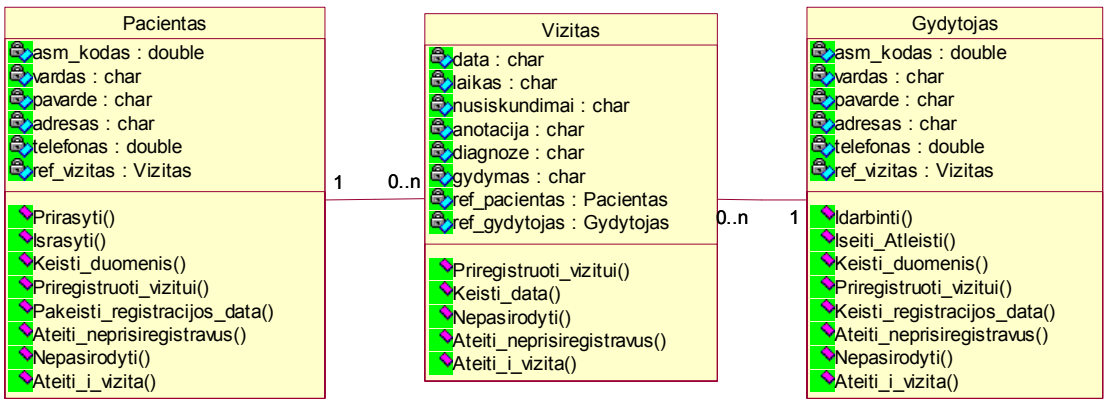
6 pav. Objekto *Vizitas* baigtinių būsenų diagrama.

3 Klasių aprašų generavimas

Organizacijos veiklos modelis sudaromas tam, kad suprasti dalykinę sritį, išsiaiškinti kuriamai sistemai keliamus reikalavimus. Objektinis organizacijos veiklos modelis gali būti panaudotas kuriamos sistemos klasių aprašams generuoti. Šių aprašų turinys priklauso nuo priemonių, kuriomis buvo panaudotos modeliui specifikuoti. Palyginsime klasių aprašus, kurie buvo gauti CASE įrankio *Rational Rose* priemonėmis ir pritaikius MERODE metodiką. Klasėms aprašyti pasirinkta C++ programavimo kalba.

3.1 Klasės aprašo generavimas Rational Rose įrankiu

Klasės aprašui sugeneruoti klasių diagramą (3 pav.) papildoma atributais ir metodais (7 pav.) taip kaip reikalauja MERODE metodika. Klasės atributai specifikuojami pagal 8 paveiksle pateiktą šabloną. Kiekvienam įvykių tipui, kuriame dalyvauja objektų tipas, atitinkamoje klasėje specifikuojamas metodas.



7 pav. Klasių diagrama.

CASE įrankio *Rational Rose* priemonės, pasirinkus C++ programavimo kalbą, klasės *Pacientas* specifikacijos pagrindu sugeneruoja žemiau pateiktą aprašą..

Pacientas.h failas:

```
#ifndef PACIENTAS_H_HEADER_INCLUDED_BFB059EB
#define PACIENTAS_H_HEADER_INCLUDED_BFB059EB
#include "Vizitas.h"

//##ModelId=404F9762031D
class Pacientas
{
  public:
    //##ModelId=404F97F70050
    Prirasyti();

    //##ModelId=404F981B019D
    Israsyti();

    //##ModelId=404F98250010
    Keisti_duomenis();

    //##ModelId=404F982F030E
    Priregistruoti_vizitui();

    //##ModelId=404F983C0399
    Pakeisti_registracijos_data();

    //##ModelId=404F986601F5
    Ateiti_neprisiregistravus();

    //##ModelId=404F98700109
    Nepasirodyti();

    //##ModelId=404F987D0338
    Ateiti_i_vizita();
  private:
    //##ModelId=405108CE01F5
    double asm_kodas;

    //##ModelId=405108DB0031
    char vardas;

    //##ModelId=405108DE02DF
    char pavarde;

    //##ModelId=405108E5004A
    char adresas;

    //##ModelId=405108EA0083
    double telefonas;
    //##ModelId=405C53D302D2
    Vizitas ref_vizitas;

};

#endif /* PACIENTAS_H_HEADER_INCLUDED_BFB059EB
*/
```

Pacientas.cpp failas:

```
#include "Pacientas.h"

//##ModelId=404F97F70050
Pacientas::Prirasyti()
{
}
//##ModelId=404F981B019D
Pacientas::Israsyti()
{
}
//##ModelId=404F98250010
Pacientas::Keisti_duomenis()
{
}
//##ModelId=404F982F030E
Pacientas::Priregistruoti_vizitui()
{
}
//##ModelId=404F983C0399
Pacientas::Pakeisti_registracijos_data()
{
}
//##ModelId=404F986601F5
Pacientas::Ateiti_neprisiregistravus()
{
}
//##ModelId=404F98700109
Pacientas::Nepasirodyti()
{
}
//##ModelId=404F987D0338
Pacientas::Ateiti_i_vizita()
{
}
```

Reikia pastebėti, kad nors klasei *Pacientas* ir buvo sudaryta būsenų diagrama, generatorius šios specifikacijos nepanaudoja. Aprašo tekstas nėra labai informatyvus. Be to, *Rational Rose* įrankis įterpė savo atributų bei metodų numeraciją komentarų pavidalu.

Class OBJEKTO_TIPO_PAVADINIMAS
Paveldimumas
// paveldimų objektų sąrašas
Atributai
// nuorodos į tėvinius objektų tipus
// nuorodos į priklausomus objektų tipus
// būsenos skaitiklis
// papildomi atributai
Invariantai
// atributų konstantos
// unikalumo konstantos
Metodai
// atomiškų įvykių tipų metodai

Pav. 8. Klasės aprašo šablonas

3.2 Klasės aprašo gavimas MERODE metodu

Didžiajai klasės aprašo daliai gauti MERODE metodu naudojami visi organizacijos veiklą aprašantys modeliai: objektų tipų gyvavimo priklausomybių grafas, objektų-įvykių lentelė bei objektų būsenų kaitos modeliai. Iš objektų tipų gyvavimo priklausomybių grafo gaunamos visos nuorodas į tėvinius ir priklausomus objektų tipus. Iš objektų-įvykių lentelės gaunami metodų pavadinimai atitinkamiems įvykiams. O iš objektų būsenų mašinų gaunami objektų būsenų perėjimai (kiekvienam objektų tipui iš būsenų kaitos modelio sudaromos būsenų kaitos lentelės, 2 lentelė).

2 lentelė. Paciento būsenų perėjimai.

Įvykis	Iš būsenos	Į būseną
Prirašyti	-	1
Išrašyti	1	2(F)
Keisti duomenis	1	1
Priregistruoti vizitui iš anksto	1	1
Pakeisti registracijos datą	1	1
Ateiti neprisiregistravus	1	1
Nepasirodyti prisiregistravusiam	1	1
Atvykti į vizitą	1	1

Automatizavus klasės aprašo gavimą, gautume žemiau pateiktą išeitinį klasės *Pacientas* aprašo tekstą:

```
class Pacientas
{
public:
  ///references to dependent object types
  std::list<Vizitas> ref_vizitas;
    // state indicators
  int state;
  // additional atributes
  int asm_kodas;
  std::string vardas;
  std::string pavarde;
  std::string adresas;
  int telefonas;
public:
  Pacientas(
int input_asm_kodas, std::string input_vardas
std::string input_pavarde, std::string
input_adresas, int input_telefonas)
   : asm_kodas( input_asm_kodas),
     vardas( input_vardas),
        pavarde( input_pavarde),
     adresas( inut_adresas),
     telefonas( input_telefonas),
     state( 1)
  {
    //invariant
    if( !Unique( asm_kodas)) throw "not unique";
    if( pavarde == "")) throw "empty";
  };
  ~Pacientas(void);

  bool Israsyti()
  {
    if( state != 1) return false;

    std::vector<Vizitas*>::iterator vizitas;
    for( vizitas = ref_vizitas.begin(); vizitas
!= ref_vizitas.end();; vizitas++)
    {
      if( !((*vizitas)->state == 3 || (*vizitas)-
>state == 4)) return false;
    }
    state = 2;
    return true;
  }
  void Keisti_pac_duomenis(
int input_asm_kodas, std::string input_vardas
std::string input_pavarde, std::string
input_adresas, int input_telefonas)
   : asm_kodas( input_asm_kodas),
     vardas( input_vardas),
        pavarde( input_pavarde),
     adresas( inut_adresas),
     telefonas( input_telefonas),
     state( 1)
  {
    //invariant
    if( !Unique( asm_kodas)) throw "not unique";
    if( pavarde == "")) throw "empty";

  };

void Priregistruoti_vizitui()
{
       if( state == 1)
              {
              state = 1;
              }
}

void Pakeisti_data()
{
       if( state == 1)
              {
              state = 1;
              }
}

void Ateiti_neprisiregistravus()
{
       if( state == 1)
              {
              state = 1;
              }
}

void Nepasirodyti()
{
       if( state == 1)
              {
              state = 1;
              }
}

void Ateiti_i_vizita( Vizitas* l)
{
       if( state == 1)
              {
              state = 1;
              }
       }
};
```

Gautas klasės aprašas yra pilnesnis, nei gautas įrankio *Rational Rose* priemonėmis. Tekstas, esantis pilkame fone, įrašytas programuotojo. Visa kita pagal MERODE būtų galima sugeneruoti automatiškai. Reikia pastebėti, kad keletas metodų, pavyzdžiui Nepasirodyti nieko neatlieka (nepakeičia objekto būsenos). Tačiau tokių metodų įtraukimas į aprašą yra pagrįstas, nes gali būti tokių metodų kurie nepakeičia objekto būsenos tačiau pakeičia tam tikrų atributų reikšmes. Atributų reikšmių pakeitimas yra nesugeneruojamas, o įrašomas paties programuotojo. Be to, tušti metodai bet kada gali būti pašalinti.

4 Išvados

MERODE - tai formali modeliavimo metodika, leidžianti patikrinti organizacijos veiklos modelių tarpusavio suderinamumą. MERODE metodu gauti klasių aprašai kur kas detalesnis už esamų komercinių CASE įrankių generuojamą tekstą. MERODE – nesudėtinga ir naudinga priemonė programuotojui. Tačiau norint ją taikyti efektyviai reikalingi CASE įrankiai. Taikyti šią metodiką su esamais komerciniais CASE įrankiais yra sudėtinga, nes juose nerealizuotos visos priemonės, kurių reikia šiai metodikai, pavyzdžiui, nėra galimybės sudaryti objektų-įvykių lentelę. Ateityje planuojama toliau tirti MERODE metodikos pritaikymo galimybes informacinėms sistemoms kurti.

Literatūros sąrašas

[1] IBM. Rational Software. URL: http://www-306.ibm.com/software/rational/ (žiūrėta 2004.03).

[2] MagicDraw. Product info. URL: http://www.magicdraw.com/ (žiūrėta 2004.03).

[3] Microsoft Office Online. Visio. URL: http://office.microsoft.com/home/office.aspx?assetid=FX01085798 (žiūrėta 2004.03).

[4] Embarcadero Technologies. Describe. URL: http://www.embarcadero.com/products/describe/ (žiūrėta 2004.03).

[5] P.A. Muller. Instant UML. *Wrox Press Ltd*, 1997.

[6] R.Paige, J.Ostroff. The Single Model Principle. Journal of Object Technology, vol. 1, no. 5, lapkritis-gruodis 2002, pp. 63-81. URL: http://www.jot.fm/issues/issue_2002_11/column6 .

[7] M.Snoeck, G.Dedene, M.Verhelst, A.M.Depuydt. Object-Oriented Enterprise Modeling with MERODE. Leuven University Press, 1999.

MERODE methodology and its application possibilities

An object-oriented methodology MERODE is analyzed in this paper. Main aim of the article is to demonstrate practical application of MERODE methodology for creating computerized information systems.